POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Dimostrazioni a conoscenza zero

Relatori
Prof. Carlo Sanna
Dott. Andrea Gangemi

Candidato
Giuseppe Piazza

Marzo 2026

Indice

Sinonimi e abbreviazioni

Input: x, informazione in ingresso

Witness: segreto, testimone

Proof: prova, dimostrazione

Statement: affermazione

Proof of Knowledge: prova della conoscenza

Prover: dimostratore

Verifier: verificatore

Completeness: completezza

Soundness: correttezza

Zero-knowledge: conoscenza zero

Random Oracle Model: modello oracolo casuale

Non-Interactive Zero Knowledge: NIZK, conoscenza zero non interattiva

SHVZK: special honest-verifier zero-knowledge, onesto verificatore a conoscenza zero speciale

ZKP: zero-knowledge proof, dimostrazione a conoscenza zero

QAP: quadratic arithmetic program

POW: proof of work

POS: proof of stake

Abstract

Il presente lavoro di tesi vuole indagare un tema di crescente rilevanza negli ultimi anni nell'ambito della sicurezza informatica: le dimostrazioni a conoscenza zero (Zero-Knowledge Proofs, ZKP). In un contesto digitale in cui la protezione dei dati e la tutela della privacy assumono un ruolo sempre più centrale, le ZKP rappresentano uno strumento crittografico innovativo che consente di dimostrare la veridicità di un'informazione senza rivelarne il contenuto. Tale principio, introdotto nel 1985 da Goldwasser, Micali e Rackoff, ha rivoluzionato il modo di concepire la sicurezza, trovando applicazione in molteplici ambiti come l'autenticazione, le firme digitali e, più recentemente, le blockchain e le criptovalute.
Le dimostrazioni a conoscenza zero permettono infatti di garantire riservatezza, integrità e trasparenza nei sistemi distribuiti, conciliando esigenze di verifica e protezione dei dati sensibili. In particolare, nell'ambito della blockchain, esse offrono soluzioni efficaci per assicurare l'anonimato verificabile delle transazioni e la validazione dei dati senza intermediari. L'integrazione di tali tecniche ha portato alla nascita di protocolli avanzati come ZK-SNARK e Bulletproofs, che migliorano le prestazioni e la sicurezza delle reti decentralizzate.
In questo contesto si colloca il lavoro di tesi, che ha analizzato in modo approfondito i fondamenti teorici delle dimostrazioni a conoscenza zero, con particolare attenzione ai Σ-protocols e ai modelli interattivi e non interattivi. È stata studiata la trasformazione di una prova interattiva in una non interattiva mediante la trasformata di Fiat–Shamir, esaminandone le proprietà di completezza, correttezza e conoscenza zero. Successivamente, il lavoro ha approfondito il ruolo delle ZKP all'interno delle tecnologie blockchain, evidenziando come esse consentano di aumentare la sicurezza e la scalabilità dei sistemi distribuiti.
Infine, sono stati analizzati nel dettaglio i protocolli ZK-SNARK e Bulletproofs, mettendone in luce il funzionamento, i requisiti computazionali, le differenze strutturali e le principali applicazioni pratiche, come l'offuscamento delle transazioni in Bitcoin e Monero. I risultati dello studio hanno confermato come le dimostrazioni a conoscenza zero costituiscano un elemento chiave per l'evoluzione della sicurezza digitale, offrendo nuove prospettive per la creazione di sistemi crittografici efficienti, trasparenti e rispettosi della privacy.

Abstract (Inglese)

This thesis aims to investigate a topic of growing importance in recent years within the field of cybersecurity: Zero-Knowledge Proofs (ZKPs). In a digital environment where data protection and privacy are increasingly critical, ZKPs represent an innovative cryptographic tool that allows one party to prove the validity of a statement without revealing any additional information about it. This concept, first introduced in 1985 by Goldwasser, Micali, and Rackoff, has profoundly changed the understanding of security, finding applications in multiple areas such as authentication, digital signatures, and, more recently, blockchain technologies and cryptocurrencies.

Zero-Knowledge Proofs ensure confidentiality, integrity, and transparency in distributed systems by reconciling the need for verifiability with the protection of sensitive data. In particular, within the blockchain context, they provide effective solutions to achieve verifiable anonymity in transactions and data validation without intermediaries. The integration of such techniques has led to the development of advanced protocols like ZK-SNARKs and Bulletproofs, which improve the efficiency and security of decentralized networks.

In this framework, the present work provides an in-depth analysis of the theoretical foundations of Zero-Knowledge Proofs, with a particular focus on Σ-protocols and both interactive and non-interactive models. The study examines the transformation of an interactive proof into a non-interactive one through the Fiat–Shamir heuristic, analyzing its main properties of completeness, soundness, and zero-knowledge. The thesis then explores the role of ZKPs within blockchain systems, highlighting how these mechanisms can enhance security, scalability, and transparency in distributed architectures.

Finally, the research focuses on the ZK-SNARK and Bulletproofs protocols, detailing their structure, computational requirements, and practical implementations, such as transaction obfuscation in Bitcoin and Monero. The results confirm that Zero-Knowledge Proofs represent a cornerstone for the future of digital security, offering new perspectives for the development of efficient, transparent, and privacy-preserving cryptographic systems.

Introduzione

Negli ultimi decenni, l'evoluzione della crittografia ha trasformato radicalmente il modo in cui vengono concepiti la sicurezza informatica e lo scambio di informazioni digitali. L'aumento esponenziale delle transazioni elettroniche, la diffusione della blockchain e l'affermarsi delle criptovalute hanno reso evidente la necessità di protocolli sempre più robusti, in grado di garantire autenticità, integrità e riservatezza dei dati senza compromettere la trasparenza dei sistemi. In questo scenario si inseriscono le dimostrazioni a conoscenza zero (Zero-Knowledge Proofs, ZKP), strumenti crittografici che permettono di dimostrare la veridicità di un'informazione senza rivelarne il contenuto.
L'idea alla base delle ZKP, introdotta nel 1985 da Shafi Goldwasser, Silvio Micali e Charles Rackoff [1], rappresenta una delle più importanti innovazioni nel campo della teoria della complessità e della sicurezza informatica. Essa consente a un "dimostratore" (prover) di convincere un "verificatore" (verifier) del possesso di una certa conoscenza o del fatto che una data affermazione sia vera, senza rivelare alcun dettaglio aggiuntivo oltre alla veridicità stessa della dichiarazione. Questo paradigma, apparentemente controintuitivo, ha trovato applicazione in numerosi ambiti: dai protocolli di autenticazione alle firme digitali, dalle criptovalute alle applicazioni decentralizzate basate su blockchain.
Nel contesto attuale, caratterizzato da una crescente attenzione verso la privacy dei dati e la sicurezza delle transazioni digitali, le dimostrazioni a conoscenza zero si configurano come una tecnologia strategica per il futuro della sicurezza informatica. In particolare, esse offrono una soluzione efficace al problema dell'anonimato verificabile, ovvero la possibilità di garantire l'autenticità di una transazione o di un'identità digitale senza compromettere la riservatezza delle informazioni personali. Questo principio è alla base di nuove forme di criptovalute e protocolli blockchain di nuova generazione, come Zcash, Monero, o i più recenti sistemi basati su ZK-SNARK e Bulletproofs, che permettono di validare transazioni in modo sicuro e trasparente, minimizzando al contempo l'esposizione dei dati sensibili.

Obiettivo della tesi

L'obiettivo principale di questa tesi è analizzare in modo sistematico e approfondito i meccanismi delle dimostrazioni a conoscenza zero, evidenziandone i fondamenti teorici, le principali implementazioni e le applicazioni pratiche nel contesto della blockchain e delle criptovalute. In particolare, il lavoro mira a:

- esaminare i principi alla base dei Σ-protocols e dei modelli interattivi e non interattivi delle prove a conoscenza zero;
- illustrare il processo di trasformazione di una prova interattiva in una non interattiva, mediante la trasformata di Fiat–Shamir, e analizzare le proprietà di completezza, correttezza e conoscenza zero che caratterizzano tali protocolli;
- approfondire il ruolo della blockchain come piattaforma applicativa per i sistemi ZKP, esplorandone le potenzialità in termini di sicurezza, decentralizzazione e trasparenza;
- studiare due delle più importanti implementazioni moderne di ZKP — ZK-SNARK e Bulletproofs — evidenziandone differenze strutturali, vantaggi, limiti e casi d'uso reali.

Attraverso questo percorso, la tesi intende fornire una visione organica delle dimostrazioni a conoscenza zero, sottolineando come esse costituiscano un punto di incontro tra ricerca teorica e applicazioni pratiche nel dominio della crittografia moderna.

Struttura della tesi

L'elaborato in questione, a seguito di questa introduzione iniziale, presenta nel capitolo 1 una trattazione teorica delle dimostrazioni a conoscenza zero, illustrandone i principi fondamentali, le origini storiche e l'evoluzione fino ai protocolli moderni. In tale capitolo vengono analizzate le principali proprietà che caratterizzano queste prove — completeness, soundness e zero-knowledge — e vengono approfonditi i modelli computazionali di riferimento, come il Random Oracle Model. Inoltre, viene dedicata particolare attenzione ai Σ-protocols, ai loro requisiti di sicurezza e ai protocolli derivati, tra cui il protocollo di Schnorr, che costituisce una base teorica essenziale per numerose applicazioni crittografiche.
Successivamente, nel capitolo 2, l'attenzione si sposta sulla blockchain, contesto applicativo nel quale le dimostrazioni a conoscenza zero trovano un impiego particolarmente efficace. Vengono introdotti i concetti di distributed ledger e di algoritmi di consenso, come il Proof of Work e il Proof of Stake, insieme alle caratteristiche strutturali e funzionali della tecnologia blockchain. Il capitolo analizza inoltre le principali problematiche legate a questa tecnologia, tra cui la scalabilità, la sicurezza e l'impatto ambientale, e presenta come caso di studio esemplificativo il sistema Bitcoin, basato su un registro distribuito e immutabile.
Nel capitolo 3 vengono trattati i ZK-SNARKs (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge), una delle implementazioni più avanzate delle prove a conoscenza zero non interattive. In questa parte si illustrano il funzionamento dei Quadratic Arithmetic Programs (QAP) e dei Rank-1 Constraint Systems (R1CS), che rappresentano la base matematica del protocollo, e viene mostrato come questi strumenti consentano di verificare in modo compatto e sicuro la correttezza di un calcolo complesso, con applicazioni dirette nell'ambito della blockchain.
Il capitolo 4 è dedicato ai Bulletproofs, un protocollo più recente e innovativo che coniuga efficienza computazionale e trasparenza, eliminando la necessità di un trusted setup. Il capitolo approfondisce i principi su cui si fonda tale protocollo, le tecniche di range proof e le principali applicazioni pratiche, con particolare riferimento alla blockchain Monero, nella quale i Bulletproofs sono impiegati per garantire la riservatezza e l'anonimato delle transazioni.
Infine, nel capitolo 5 vengono presentate le conclusioni del lavoro, in cui si riassumono le principali considerazioni emerse e si delineano le prospettive future di ricerca. In particolare, si discute la possibilità di integrare le dimostrazioni a conoscenza zero con altre tecnologie emergenti, come la crittografia omomorfica e le blockchain di nuova generazione, al fine di sviluppare soluzioni sempre più efficienti, sicure e rispettose della privacy, applicabili in contesti finanziari, industriali e istituzionali.

Capitolo 1

1 Dimostrazione a conoscenza zero

In crittografia, una dimostrazione a conoscenza zero o un protocollo a conoscenza zero è un metodo interattivo utilizzato da un soggetto, ovvero il dimostratore/prover, per dimostrare a un altro soggetto, ovvero il verificatore/verifier, che una determinata affermazione è vera, senza rivelare nient'altro oltre alla veridicità della stessa. L'essenza delle prove a conoscenza zero è che sarebbe banale dimostrare di possedere la conoscenza di determinate informazioni semplicemente rivelandole; la sfida consiste di fatto nel provare tale possesso senza rivelare le informazioni stesse o ulteriori informazioni.
Se la prova di una dichiarazione richiede che il dimostratore possieda alcune informazioni segrete, il verificatore non sarà in grado di provare la dichiarazione a nessun altro senza possedere le tali informazioni. L'affermazione da provare deve includere che il dimostratore dichiari di possedere tale conoscenza, ma senza includere o trasmettere la conoscenza stessa nella affermazione. In caso contrario, l'affermazione non verrebbe dimostrata a conoscenza zero. Si parla dunque di **dimostrazione a conoscenza zero** solamente quando è il prover a possedere le informazioni segrete.
Si parla di **dimostrazione a conoscenza zero interattiva** quando vi è l'interazione tra una parte che dimostra la propria conoscenza e una controparte che convalida la dimostrazione. L'input interattivo si svolge solitamente tramite una o più sfide (challenge) tali da permettere al dimostratore di convincere il verificatore che la propria affermazione è vera, ovvero che il dimostratore possiede la conoscenza dichiarata.
Si parla di **dimostrazioni a conoscenza zero non interattive** quando la validità della dimostrazione si basa su ipotesi computazionali, tipicamente le ipotesi di una funzione di Hash crittografica ideale.
Tipicamente, il protocollo viene avviato dal prover e terminato dal verificatore, che alla fine accetta o rifiuta la prova del prover. Viene chiamata *trascrizione* dell'esecuzione del protocollo la sequenza dei messaggi scambiati dalle due parti [2].
Una dimostrazione a conoscenza zero deve soddisfare tre proprietà:

- ✓ **Completezza (completeness)**: se il protocollo è gestito da un dimostratore onesto e da un verificatore onesto, il verificatore accetta sempre la prova;
- ✓ **Correttezza (soundness)**: se l'affermazione è falsa, nessun dimostratore disonesto potrà convincere un verificatore onesto che essa è vera (la probabilità che un dimostratore disonesto convinca un verificatore onesto è detta "cheating probability");
- ✓ **Conoscenza zero (zero-knowledge)**: se l'affermazione è vera, il verificatore non apprende nient'altro sull'input privato del prover onesto dall'esecuzione del protocollo, a parte il fatto che l'affermazione dichiarata è vera [3].

Le prime due sono proprietà di sistemi di dimostrazione interattivi più generali. La terza è proprio ciò che rende la dimostrazione a conoscenza zero.
Le dimostrazioni a conoscenza zero non sono dimostrazioni in senso matematico perché vi è sempre una piccola probabilità che un dimostratore disonesto riesca a convincere un verificatore di un'affermazione falsa. Di fatto le dimostrazioni a conoscenza zero sono "dimostrazioni"

probabilistiche piuttosto che deterministiche. Tuttavia, esistono tecniche per ridurre questa probabilità a valori trascurabilmente piccoli.

1.1 Cenni di storia

Le **dimostrazioni a conoscenza zero** furono concepite per la prima volta nel 1985 da Shafi Goldwasser, Silvio Micali e Charles Rackoff nel loro articolo "The Knowledge Complexity of Interactive Proof-Systems" [4]. Questo documento ha introdotto la gerarchia "**IP**" (**I**nteractive **P**olynomial time) dei sistemi a dimostrazione interattiva e ha concepito il concetto di "complessità conoscitiva", una misura della quantità di conoscenza sulla dimostrazione trasferita dal dimostratore al verificatore.
Ha anche fornito la prima dimostrazione a conoscenza zero per un problema concreto, quello di decidere i **non residui quadratici mod m.** Di seguito ad un articolo di László Babai e Shlomo Moran, sono stati inventati i sistemi di prova interattivi, per i quali tutti e cinque gli autori hanno vinto il primo Premio Gödel nel 1993, citando testualmente:

"Di particolare interesse è il caso dove questa conoscenza addizionale è essenzialmente 0 e mostriamo che è possibile dimostrare interattivamente che un numero è non residuo quadratico mod "m" rilasciando conoscenza addizionale 0. Ciò è sorprendente giacché non c'è algoritmo efficiente capace di decidere circa i residui modulo "m" quando la fattorizzazione di "m" non è data. Inoltre, tutte le dimostrazioni "NP" per il problema esibiscono la fattorizzazione prima di "m". Questo indica che aggiungere interazione al processo di dimostrazione può decrementare la quantità d'informazione che deve essere comunicata per dimostrare un teorema." [2]

Nel 1985 Babai introdusse i giochi di **Arthur-Merlin**, i quali sono un sistema a dimostrazione interattiva in cui il Verifier (Arthur) è un computer probabilistico di velocità polinomiale che dispone di un generatore di numeri casuale, mentre il Prover (Merlin) è un oracolo con infinito potere computazionale. A differenza del sistema proposto da Goldwasser, Micali e Rackoff (GMR) con la classe **IP** in cui il Prover può solamente vedere i messaggi che produce il Verifier (concetto di *Private Coin*), nei giochi di Arthur-Merlin i lanci di moneta sono pubblici (*Public Coin)*, di fatto il Prover ha accesso a tutte le scelte randomiche effettuate dal Verifier [5].
Nel 1988 Babai e Moran, nel loro lavoro "Arthur-Merlin Games: A Randomized Proof System and a Hierarchy of Complexity Classes" [6], attenzionarono le classi di complessità **AM** (Arthur-Merlin) e **MA** (Merlin-Arthur).
In AM, come detto precedentemente, le monete sono mantenute pubbliche, di conseguenza il Prover ha accesso a tutte le scelte randomiche effettuate dal Verifier. Goldwasser e Sipser [1] dimostrarono che il concetto di moneta privata, punto di forza per IP, non garantiva la sicurezza dei protocolli, in quanto un Prover di un protocollo Arthur-Merlin a monete pubbliche può, grazie alla sua potenza di calcolo e con sole due interazioni in più, giungere alle stesse conclusioni di un protocollo a moneta privata. In formula:

- $IP(t(n)) \subseteq AM(t(n)+2)$. (dove $t(n)$ è limitato in modo polinomiale)

La comunicazione in AM è definita come $AM[k]$, dove k è una costante che rappresenta il numero di messaggi scambiati. Per definizione $AM = AM[2]$, in quanto:

- Arthur inizia la conversazione con il lancio delle monete ed invia il risultato a Merlin;
- Merlin risponde con una presunta prova;
- Arthur, infine, effettua la verifica (non invia il risultato a Merlin).

Nota: Se $k \geq 2$ è pari, la comunicazione inizia sempre da Arthur, viceversa inizierà da Merlin.

La classe di complessità MA è un caso particolare in quanto nella comunicazione viene scambiato un solo messaggio, Merlin invia un certificato ad Arthur il quale, tramite un calcolo probabilistico in tempo polinomiale, deciderà se accettarlo o meno.
Si può infine dimostrare come MA è contenuta in AM. Si consideri il caso $AM[3]$:

- Merlin invia un certificato ad Arthur;
- Arthur, dopo averlo ricevuto, può effettuare il lancio delle monete richiesto e inviarle a Merlin, ignorando la sua futura risposta.

1.2 Dimostrazioni a conoscenza zero interattive

Un sistema a dimostrazione a conoscenza zero interattiva è un automa[1] che modella e definisce lo scambio di messaggi fra due parti: **Prover** e **Verifier**. Il Verifier è una macchina di Turing[2] probabilistica onesta e con capacità computazionali deboli (limitate ad una velocità polinomiale) il cui obbiettivo è quello di verificare se una determinata affermazione v nota ad entrambe le parti (detta anche stringa o parola), solitamente matematica, è valida. Un'affermazione è valida se si può dimostrare in tempo polinomiale tramite una dimostrazione (proof) che risulta appartenente ad un linguaggio L.
Il Prover è, invece, una macchina con infinito potere computazionale il cui obiettivo è riuscire a convincere il Verifier, tramite le sue risposte, che l'input v appartiene al linguaggio. Egli è capace di risolvere determinati tipi di problemi, ad esempio problemi decisionali o problemi di funzione:

- Un problema decisionale è rappresentato come un set di numeri naturali o stringhe. Un'istanza di tale problema è un numero naturale o una stringa. La soluzione di tale istanza è "si" (oppure "1") se il numero è contenuto all'interno del set, o "no" (oppure "0") altrimenti.
- Un problema di funzione è rappresentato da una funzione f $(N \rightarrow N)$. Un'istanza del problema è un input v di f e la soluzione `e il valore $f(v)$.

Il Prover potrebbe, però, agire in disonestà, per cui il Verifier dovrà porre una serie di challenge in successione, con l'obbiettivo di verificare se egli dispone di una qualche informazione segreta (detta comunemente witness) che gli permette di comprovare la veridicità di tale

[1] Un automa è un sistema dinamico discreto (nella scansione del tempo e nella descrizione del suo stato) e tempo-invariante (il sistema si comporta alla stessa maniera indipendentemente dall'istante di tempo in cui agisce). Un esempio classico di automa è la macchina di Turing.
[2] Una macchina di Turing è un automa costituito da un sistema di regole ben definite. In altre parole, è un modello astratto che definisce una macchina in grado di eseguire algoritmi, il cui scopo `e valutare i linguaggi e risolvere funzioni matematiche.

affermazione. In questo modo il Verifier può asserire, con un certo grado di sicurezza, che l'input appartiene al linguaggio e quindi il Prover è onesto.

1.3 Prove di conoscenza

Una prova di conoscenza (**Proof of Knowledge**), è un tipo speciale di dimostrazione a conoscenza zero interattiva in cui "il Prover *conosce* un segreto" ed egli deve riuscire a convincere il Verifier di conoscerlo; mentre, nelle dimostrazioni a conoscenza zero interattive descritte precedentemente, l'obiettivo del Prover era quello di dimostrare che un input v appartenesse ad un linguaggio L. Per questo tipo speciale di prove, la proprietà di correttezza (nota anche come correttezza speciale) è definita tramite un **Knowledge Extractor** (estrattore di conoscenza). Un Knowledge Extractor è uno speciale tipo di Verifier computazionalmente forte e volutamente non onesto, utilizzato per dimostrare la proprietà di correttezza di una prova di conoscenza [7].
Un classico esempio di dimostrazione di **Proof of Knowledge** sta nel mostrare che il Prover conosce un isomorfismo tra due grafi. Un grafo è un insieme finito di elementi detti nodi o vertici che possono essere collegati tra loro da linee chiamate archi o spigoli (edges). Più formalmente, si dice grafo una coppia ordinata $G = (V, E)$ di insiemi, con V insieme dei nodi ed E insieme degli archi. Ad esempio, il grafo definito da V = {a, b, c, d} e E = {{a, b}, {b, c}, {c, d}, {a, c}, {a, d}} può essere disegnato come un poligono quadrato con una delle due diagonali. Due grafi sono chiamati isomorfi se c'è una rietichettatura dei vertici che converte un grafo nell'altro.
Nell'esempio precedente, si può trasformare il quadrato con la diagonale $a - c$ nello stesso quadrato con l'altra diagonale $b - d$ permutando ciclicamente i vertici $a \mapsto b \mapsto c \mapsto d \mapsto a$. Trovare un isomorfismo tra due grandi grafici casuali è un problema alquanto difficile, pertanto ha senso dimostrare la conoscenza di un tale isomorfismo senza rivelarlo. Assumiamo quindi che un dimostratore P e un verificatore V conoscano due grafi isomorfi G_0 *e* G_1, e P conosca anche un isomorfismo del grafo $\phi : G_0 \to G_1$. Assumiamo, per semplicità, che entrambi i grafi abbiano lo stesso insieme di vertici, e quindi anche l'isomorfismo è una permutazione di questo insieme.
Per avviare il protocollo, P sceglie una permutazione casuale ψ_1 dei vertici e definisce il grafo $G_2 = \psi_1(G_1)$, che è chiaramente isomorfo a G_1. P definisce anche l'isomorfismo $\psi_0 = \psi_1 \circ \phi$, ovvero un isomorfismo tra G_0e G_2. Di conseguenza accade che:

- P invia il grafo G_2 a V , ad esempio, fornendo una lista o una descrizione dei suoi archi.
- Successivamente, V sceglie un bit casuale $c \in \{0,1\}$ e lo invia a P come challenge.
- P risponde alla sfida inviando a V l'isomorfismo ψ_c.
- V accetterà la dimostrazione che P conosce un isomorfismo tra G_0e G_1 se ψ_c è biiettiva e $G_2 = \psi_c(G_c)$.

Il protocollo è banalmente corretto, poiché un verificatore onesto accetterà sempre la prova fornita da un dimostratore onesto.
Considerando adesso un dimostratore P (possibilmente disonesto) se esso è accettato da un verificatore onesto V con probabilità maggiore di $1/2$, allora c'è una probabilità non trascurabile che P possa rispondere correttamente alle due sfide $c = 0$ e $c = 1$ per lo stesso

grafo G_2. Pertanto, dalle due risposte ψ_0 e ψ_1, P stesso può calcolare l'isomorfismo $\phi = {\psi_1}^{-1} \circ \psi_0$.
Per la proprietà a conoscenza zero, esiste un simulatore che può ottenere da un verificatore (possibilmente disonesto) una trascrizione indistinguibile da quella ottenuta da una reale esecuzione del protocollo tra un dimostratore onesto e lo stesso verificatore. Infatti:

- Il simulatore (che non conosce l'isomorfismo $\phi : G_0 \to G_1$), sceglie un bit casuale $c \in \{0, 1\}$.
- Il simulatore sceglie le permutazioni casuali ψ_0 e ψ_1 dell'insieme dei vertici e definisce $G_2 = \psi_c(G_c)$.
- Successivamente, il simulatore avvia l'esecuzione del protocollo inviando G_2 a V.

Se la sfida data da V è diversa da c, esso interrompe l'esecuzione del protocollo e ricomincia scegliendo un nuovo bit casuale. In caso contrario, il simulatore emette la trascrizione $(G2, c, \psi_c)$.
L'errore di correttezza di $1/2$ può essere ridotto arbitrariamente ripetendo il protocollo molte volte indipendenti. Il verificatore accetterà la prova aggregata solo se tutte le singole esecuzioni sono accettate [3].

1.4 Sigma Protocol

I protocolli Sigma sono una speciale classe di **Proof of Knowledge** (prova di conoscenza), in cui il Prover avvia il protocollo con un messaggio a chiamato commitment, quindi il Verifier invia un secondo messaggio c chiamato sfida ed infine viene inviato un ultimo messaggio r, chiamato risposta, dal Prover. La trascrizione del protocollo è quindi la tupla (a, c, r).
Un **Σ-protocol** deve necessariamente soddisfare 3 proprietà: completezza (**completeness**), correttezza speciale (**special soundness**), e verificatore onesto a conoscenza zero speciale (**special honest-verifier zero-knowledgeness**). Definiamo $R = \{(v; w)\} \subseteq V \, x \, W$ una relazione binaria, dove $v \in V$ denota l'input/istanza di un qualche problema computazionale ed è comune sia al prover che al verifier, mentre $w \in W$ denota un segreto/witness (soluzione di tale istanza).
Definiamo $L_R = \{ v \in V : \exists_{w \in W} \ (v; w) \in R \}$ il linguaggio corrispondente alla relazione R[3].

[3] Il linguaggio L_R sarà solitamente un linguaggio NP (non-deterministic polynomial time), quindi la relazione R sarà una relazione NP.

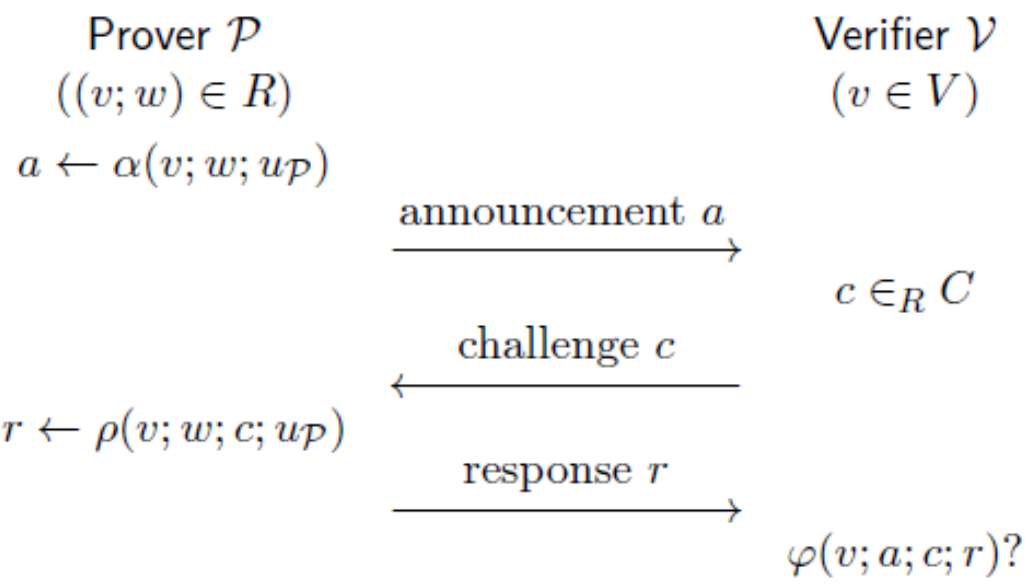

Figura 1 Sigma-Protocol per la relazione R. *[8]*

La conversazione $(\mathbf{a}; \mathbf{c}; \mathbf{r})$ è accettata se $\boldsymbol{\varphi}(\mathbf{v}; \mathbf{a}; \mathbf{c}; \mathbf{r})$ è valida. Un **Σ-protocol** per la relazione R è un protocollo basato su 3 round tra un prover P e un verificatore V, con schema riportato nella figura, che soddisfi le tre proprietà.

1.4.1 Proprietà Sigma Protocol

Le proprietà implicano che un sigma-protocol per la relazione R è sempre un sistema di dimostrazione interattiva per il linguaggio L_R con probabilità di errore 2^{-t} [9].
Nei successivi paragrafi verranno approfondite le 3 proprietà fondamentali.

1.4.2 Proprietà di Completeness

Definizione di Completezza (Completeness). Definiamo $(V, P)_{v,w}$ come la conversazione tra un Prover onesto e un Verifier onesto, con input comune v e un segreto w che conosce solamente il Prover. Se un'affermazione è vera, allora il Verifier ne accetta sempre la dimostrazione. Si parla in questo caso di *Perfect Completeness*.
In formula: $\forall\, (v; w) \in R, L_R[\, P(v; w) \leftrightarrow V(v) = accept] = 1$.
Se è invece presente un certo margine di errore dovuto, ad esempio, ad algoritmi probabilistici, si parla di *normale completezza*.
Qualsiasi **Σ-protocol** soddisferà la proprietà di Completezza [10].

1.4.3 Proprietà di Special Soundness

Definizione di Correttezza speciale (Special soundness). Un sigma protocol soddisfa la proprietà di Special Soundness se esiste un algoritmo ε (*estrattore*) p.p.t. (**probabilistic polynomial-time**[4]), il quale, dato ogni $v \in V$ e ogni coppia di conversazioni/trascrizioni $(a; c; r)$ e $(a; c'; r')$ con $c \neq c'$, calcola sempre un segreto w che soddisfi la condizione $(v; w) \in R$.

La Correttezza speciale è tipicamente ottenuta tramite l'ausilio di un estrattore di conoscenza costituito da un algoritmo che estrae il segreto da due trascrizioni accettanti $(a; c; r)$ e $(a; c'; r')$, con un commitment comune ma sfide diverse. Queste due trascrizioni possono essere ottenute dall'estrattore e una volta che il Prover ha emesso il commitment a, può interromperne l'esecuzione. Successivamente il Prover, tramite un processo di Fork, viene diviso in due istanze parallele, e l'esecuzione viene ripresa, fornendo due sfide casuali c e c' alle due istanze del Prover. Infine, ogni istanza, invia la sua risposta corrispondente, r o r'. Assumiamo adesso che il Verifier onesto prenda una sfida a caso da un insieme di γ elementi, dove γ è una funzione superpolinomiale. Si può dimostrare che se il Prover è accettato con una probabilità $\frac{1}{\gamma} + \alpha$ con α non trascurabile, allora la probabilità che la procedura di fork ottenga due trascrizioni accettanti con lo stesso commitment e sfide diverse è anch'essa non trascurabile [3].

<u>Commenti avanzati:</u> Correttezza della conoscenza (Knowlodge soundness)
Abbiamo definito come i protocolli Sigma sono una speciale classe di Prove di conoscenza.
Si può affermare che un protocollo è una Prova di conoscenza con errore di conoscenza ϵ, se esiste un estrattore tempo polinomiale atteso ε' tale che per ogni w e ogni Prover P:

$$L_R\big[\, P(v; w) \in R : x \leftarrow \varepsilon'^{P^*}(w)\big] \;\geq\; L_R\,[\,\langle P^*, V\rangle(v) = 1\,] - \epsilon$$

Nella notazione ε'^{P^*}, ε' è un algoritmo che ottiene l'accesso alla "black-box" dell'algoritmo P^*.
Se un protocollo Sigma soddisfa il requisito di correttezza della conoscenza, allora è anche una prova di conoscenza con errore di conoscenza al massimo di $1/|C|$.
Vediamo come trasformare un estrattore ε, che soddisfa il requisito di correttezza della conoscenza in un estrattore di conoscenza ε' come segue:
ε' esegue il Prover per ottenere il commitment a, gli invia una sfida casuale c e riceve una risposta r. Se $(a; c; r)$ non è una trascrizione accettante, l'estrattore ricomincia il protocollo. Altrimenti, se $(a; c; r)$ è una trascrizione accettante, l'estrattore riavvolge il Prover al suo stato dopo aver inviato il commitment a, gli invia una nuova sfida c' e ottiene una nuova risposta r'. Se anche la seconda trascrizione è accettata e $c \neq c'$, allora l'estrattore ε esegue ε' su $(a; c; r)$ e $(a; c'; r')$ per calcolare un segreto w che soddisfi la condizione $(v; w) \in R$.
Se la seconda trascrizione non è accettata, viene eseguito nuovamente il protocollo con un'altra sfida [11].

[4] Nella teoria della complessità, PP è la classe di problemi decisionali risolvibili da una macchina probabilistica di Turing in tempo polinomiale, con una probabilità di errore inferiore a 1/2 per tutte le istanze.

L'importanza di questa proprietà sta nel dimostrare la conoscenza di un segreto, ed è compito dell'estrattore riuscire ad estrarre in tempo polinomiale dal Prover, dato il suo codice, una prova valida.

1.4.4 Proprietà di Special Honest-verifier Zero-knowledge

Definizione di Onesto verificatore a conoscenza zero speciale (Special honest-verifier zero-knowledgeness). Esiste un algoritmo S (*simulatore SIM*) p.p.t., il quale, dato qualsiasi $v \in L_R$ e qualsiasi sfida $c \in C$, genera conversazioni $(a; c; r)$ tra due P e V onesti con la stessa distribuzione di probabilità delle conversazioni con un input comune v e una challenge c, dove P usa un qualsiasi segreto w che soddisfi la condizione $(v; w) \in R$. Inoltre, dato un qualsiasi $v \in V/L_R$, il simulatore S è richiesto per produrre trascrizioni accettanti arbitrarie $(a; c; r)$, per ogni sfida con $c \in C$. Se C è costituito da un singolo elemento, allora il **Σ-protocol** viene considerato "***semplice*** (***trivial***)".

Quest'ultima proprietà è solitamente ottenuta da un Simulatore che prima sceglie una sfida casuale c, e successivamente calcola valori adeguati per il commitment a e la risposta r al fine di produrre una trascrizione accettante con la stessa distribuzione di probabilità della trascrizione della reale esecuzione del protocollo. La peculiarità di questa proprietà è che il Simulatore non è a conoscenza del segreto w , ma conosce solamente l'input v e la sfida c.
La proprietà di SHVZK è essenziale quando si parla di sicurezza nello scambio di messaggi tra le parti, in quanto il Simulatore può costruire il primo e l'ultimo messaggio scambiato nel protocollo al fine di non compromettere la prova stessa.

In che modo implica il concetto di Zero-knowledge nelle dimostrazioni?

Con Zero-knowledge si ha la certezza che il Verificatore, dalla dimostrazione, non impari nient'altro che la verità dell'affermazione. La soluzione proposta da Goldwasser, Micali e Rackoff, afferma che un sistema di dimostrazioni viene definito ZK (Zero-knowledge) se, per tutti i Verificatori, esiste un terzo algoritmo ovvero un Simulatore, che, avendo a disposizione determinate informazioni che non siano quelle segrete del Prover, può generare una prova che è indistinguibile dalla prova generata dal Prover. Di fatto, un Simulatore genera una prova valida con una distribuzione identica a quella reale senza utilizzare il segreto del Prover, così facendo il Verifier non può ottenere informazioni aggiuntive dalla prova reale se non la verità dell'affermazione stessa.
Lemma 1. Le proprietà dei **Σ-protocol** sono invarianti rispetto ad una composizione in parallelo, ad esempio ripetere per R un **Σ-protocol** due volte in parallelo, produce un nuovo protocollo sigma con lunghezza della sfida pari a 2t.
Lemma 2. Se esiste un **Σ-protocol** per R, allora la lunghezza della sfida sarà pari a t.
Dimostrazione.
Sia t' la lunghezza della sfida per un dato protocollo P. Qualsiasi lunghezza t inferiore a t' può essere implementata come segue:

- Il Prover invia il primo messaggio a;
- Il Verifier, una volta aver ricevuto il messaggio, invia una stringa casuale di t-bit c;

- Il Prover aggiunge $t' - t$ zeri alla stringa c ricevuta, creando così c' e calcola la risposta r in funzione di c' e la invia al Verifier;
- Il Verifier controlla la risposta r ricevuta in funzione della nuova sfida c'.

In conclusione qualsiasi lunghezza $t > t'$ può essere implementata ripetendo il protocollo P j volte in parallelo, in modo tale che $jt' \geq t$.

1.5 Funzioni di Hash

Una funzione $H : \{0,1\}^* \rightarrow \{0,1\}^k$, la quale mappa una stringa di bit a lunghezza arbitraria in una a lunghezza di bit fissa k, con $k > 0$, è definita **funzione di Hash**. Le funzioni di hash informalmente, prendono come input dei dati e producono come output una stringa univoca di bit. A parità di input, la funzione hash riproduce sempre la stessa stringa di bit. L'output di una funzione di hash è spesso chiamato digest o hash.
Una funzione di hash viene chiamata funzione di hash crittografica se soddisfa tre proprietà di sicurezza:

- **Resistenza alla pre-immagine**. Garantisce che, dato un valore di hash h, deve essere difficile risalire ad un messaggio m con $hash(m) = h$. Nessuno deve riuscire ad invertire la funzione di hash per recuperare l'input dato un output (concetto di funzione unidirezionale).
- **Resistenza alla seconda pre-immagine**. Dato un input m_1 e il digest calcolato tramite l'hashing[5], deve essere difficile trovare un secondo input m_2 , tale che $hash(m_1) = hash(m_2)$
- **Resistenza alle collisioni**. Garantisce che, dati due input m_1 e m_2, deve essere difficile trovare lo stesso hash, $hash(m_1) = hash(m_2)$.

Queste proprietà implicano che un eventuale attacco non permetta di rimpiazzare o modificare un messaggio senza conseguenze sul risultante valore di hash. Pertanto, se due stringhe hanno lo stesso digest, si può essere fiduciosi nel pensare che siano identiche. In particolare, la resistenza alla seconda pre-immagine dovrebbe impedire ad utente malevolo di elaborare un messaggio con lo stesso hash di un messaggio che l'utente stesso non può controllare. Invece la resistenza alla collisione, impedisce all'attaccante di creare due messaggi distinti con lo stesso hash. Nella pratica, una funzione hash è solo una resistenza alla seconda pre-immagine; motivo per cui è considerata insicura e, di conseguenza, non raccomandata per applicazioni reali. D'altra parte, la resistenza alle collisioni risulta insufficiente per molti usi pratici, e di conseguenza una funzione di hash crittografica deve essere il più possibile simile a una funzione casuale spesso chiamata: **Oracolo Casuale.**
Esempi pratici di funzioni di hash crittografiche sono MD5, SHA-1, SHA-256, rispettivamente con lunghezze di output $k = 128$, $k = 160$, $k = 256$.
Affinché una funzione di hash possa garantire tutte e tre le proprietà di sicurezza menzionate in precedenza, deve fornire almeno 128 bit di sicurezza contro tutti e tre gli attacchi. L'attacco più semplice è solitamente trovare collisioni dovute al limite del compleanno (attacco del compleanno) [12].

[5] L'hashing è il processo di conversione di una determinata chiave in un altro valore, espresso sotto forma di codice alfanumerico. La funzione di hash viene utilizzata per generare un nuovo valore secondo un algoritmo matematico e unidirezionale, ovvero senza possibilità di riconvertire l'hash nella chiave originale.

Paradosso del compleanno

Il paradosso afferma che, in un gruppo di 23 persone, vi è il 50% di probabilità che almeno due di essi compiano gli anni lo stesso giorno. Questo paradosso, grazie ai suoi principi matematici, viene spesso sfruttato per gli attacchi crittografici (l'attacco del compleanno) per la crittanalisi degli algoritmi di cifratura. Lo scopo dell'attacco, data una funzione f , è quello di trovare 2 numeri x_1 e x_2 tali che $f(x_1) = f(x_2)$. Se tale coppia di valori in input x_1 e x_2 producono lo stesso output, allora tale coppia è chiamata *collisione.*

1.6 Modello oracolo causale

In crittografia, un modello Oracolo Casuale (random oracle model, ROM) viene inteso come una funzione di hash crittografica, che quando interrogato si comporterà come una scatola nera contenente una funzione casuale $\mathcal{H}:\{0,1\}^* \rightarrow \{0,1\}^k$. Se l'oracolo viene interrogato su un valore di input x , esso restituirà un valore di output pari a $\mathcal{H}(x)$, pertanto se l'oracolo viene interrogato più volte sullo stesso input, restituirà sempre lo stesso output essendo un funzione di hash.
Durante l'esecuzione del protocollo, l'oracolo casuale può essere interrogato sia dal Prover che dal Verifier:

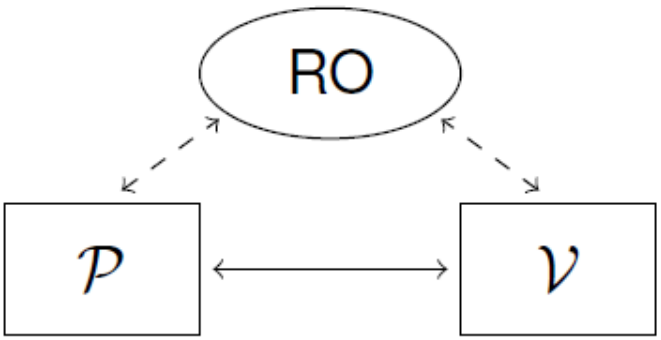

Figura 2 Oracolo Casuale. *[13]*

Il modello dell'Oracolo Casuale fu introdotto nel 1993 da Bellare e Rogaway in "*ACM Conference on Computer and Communications Security"* [14], per fornire una misura di confidenza nella robustezza della funzione hash progettata. Nella realtà, quando si progetta una funzione hash h, si testa la sicurezza di h nel modello dell'oracolo casuale per avere una evidenza della sua sicurezza.

Si può dunque sintetizzare il modello come segue:

- L'oracolo sceglie in modo casuale una funzione hash in $F^{X,Y}$;

- Gli utenti (sia Prover che Verifier) non hanno a disposizione una formula o un algoritmo che permetta loro di calcolare il valore di h in x, possono solamente chiedere all'oracolo il valore $h(x)$;
- I meccanismi interni all'oracolo non sono noti, si parla dunque di scatola nera o black-box;
- Le richieste sono private: nessuno è a conoscenza di quale sia la richiesta effettuata da un utente, nè tantomeno se essa è stata fatta;
- Le risposte date dall'oracolo sono coerenti. Se in seguito alla richiesta x l'oracolo restituisce $h(x)$, questo valore viene memorizzato in una tabella. Successivamente, se viene fatta la stessa richiesta x anche da utenti diversi, l'oracolo recupera $h(x)$ dalla tabella senza calcolarlo nuovamente.

Pertanto, l'oracolo appare come una tabella di numeri casuali. Così per un utente che avanza all'oracolo una richiesta x , i valori assumibili da $h(x)$ sono tutti equiprobabili e indipendenti [15].

1.7 Protocollo di identificazione di Schnorr

Il protocollo (o schema) di identificazione di Schnorr introdotto nel 1990 viene preso spesso come riferimento, in quanto esso non è solamente un **Σ-protocol** Public-Coin a conoscenza zero, ma è una prova di conoscenza a conoscenza zero.
Costruiremo una ZKP in modo incrementale da dei protocolli per mostrare come Peggy (Prover) può dimostrare di conoscere w senza rivelarlo. Il modo tipico per affrontare questo tipo di problema in crittografia è "nascondere" il valore con una certa casualità (ad esempio crittografandolo) tramite un metodo algebrico. Una prima soluzione è quella di aggiungere un valore k al segreto, generato casualmente: $s = k + w$.
Peggy può quindi inviare a Victor il segreto nascosto s insieme al valore casuale k. Quello che Victor sa è che il segreto w si nasconde nell'esponente di g perché conosce $Y = g^w$.
Per essere certo che Peggy conosca davvero il segreto, Victor può controllare se ciò che gli ha dato corrisponde effettivamente a ciò che sa. Di conseguenza Victor controlla che questi due numeri siano uguali:

- $g^s = g^{k+w}$
- $Y \times g^k = g^w \times g^k = g^{w+k}$

L'idea è che solo qualcuno, che conosce il segreto w, potrebbe aver costruito un segreto s che soddisfi questa equazione. E come tale, è una **prova di conoscenza**. Questo sistema ZKP così descritto non è sicuro. Infatti, poiché l'equazione che nasconde il segreto w ha solo un'incognita (w stessa), Victor può semplicemente invertire l'equazione per recuperare il segreto: $w = s - k$**.**
Per risolvere questo problema, Peggy può nascondere il valore casuale k stesso. Questa volta, deve nascondere il valore casuale nell'esponente (invece di aggiungerlo a un altro valore casuale) per assicurarsi che l'equazione di Victor funzioni ancora: $R = g^k$**.**
In questo modo, Victor non apprende il valore k e, quindi, non può recuperare il segreto w. Eppure, ha ancora abbastanza informazioni per verificare che Peggy conosca w. Victor deve semplicemente verificare che:

- $g^s = g^{k+w} = g^k \times g^w$ sia uguale a $Y \times R = g^w \times g^k$

Vi è però un ultimo problema con questo schema: Peggy può barare. Può far credere a Victor di conoscere w, senza conoscere veramente w invertendo il passaggio in cui calcola la sua dimostrazione. Prima genera un valore casuale s e poi calcola il valore R in base ad s:

- $R = g^s \times Y^{-1}$

Victor quindi calcola $Y \times R = Y \times g^s \times Y^{-1}$, che in effetti corrisponde a g^s.

NOTA: Il metodo di Peggy di usare un inverso per calcolare un valore è utilizzato in molti attacchi in crittografia. Per far sì che il protocollo ZKP funzioni adeguatamente, Victor deve assicurarsi che Peggy calcoli s da R e non l'inverso. Per fare ciò, Victor rende **interattivo** il protocollo:

1) Peggy deve eseguire il "commit" del valore casuale k in modo che non possa cambiarlo in seguito.
2) Dopo aver ricevuto il valore k da Peggy, Victor introduce nel protocollo un valore casuale $\boldsymbol{c}$ (chiamato **challenge**) e lo invia a Peggy.
3) Peggy può quindi calcolare il suo "valore nascosto" in base al valore casuale k e alla sfida c.

Poiché Peggy non può eseguire l'ultimo passaggio senza la sfida c di Victor, e Victor non gliela invierà senza aver ricevuto il valore k da Peggy, quest'ultima è costretta a calcolare s basandosi su k. Il protocollo ottenuto è spesso indicato come **protocollo di identificazione di Schnorr**.

NOTA: Il protocollo di identificazione di Schnorr funziona nel modello **HVZK** (**Honest verifier Zero-knowledge**): se il Verifier (Victor) agisce in modo disonesto e non sceglie una challenge casuale, potrebbe ricavare informazioni sul segreto w [12].

1.8 Dimostrazioni a conoscenza zero non interattive

A seguito della prima pubblicazione riguardo la conoscenza-zero venne rilasciata, la dimostrazione matematica dell'esistenza di prove a conoscenza-zero non interattive da Manuel Blum, Silvio Micali e Paul Feldman nel loro lavoro "Non-Interactive Zero-Knowledge and Its Applications" [16]. Nel 1991 inoltre, sempre Blum e Micali assieme ad altri due crittografi italiani, Alfredo De Santis e Giuseppe Persiano, ne parlarono più approfonditamente estendendone i concetti in "Non-Interactive Zero-Knowledge" [17].
I protocolli non interattivi sono più robusti rispetto a quelli interattivi, principalmente perché non dipendono da ipotesi sulle proprietà della rete di comunicazione, e quindi in un protocollo interattivo bisogna occuparsi dei ritardi di comunicazione e dell'ordine di consegna dei diversi messaggi generati dal protocollo. Nella maggior parte dei **Σ-protocol**, l'ordine dei messaggi è fondamentale per la sicurezza. Infatti, se la sfida si generasse prima del commitment, la prova non fornirebbe alcuna garanzia che l'affermazione provata sia effettivamente valida. I protocolli non interattivi a conoscenza zero possono essere basati su diversi modelli di sicurezza [3].
Il topic discusso negli articoli sopracitati è la possibilità di disfarsi dell'interazione ciclica fra le parti, della scelta randomica della challenge del Verifier e consentire la verifica del segreto del Prover con un singolo messaggio. Tutto questo è possibile se viene scambiata, prima di

comunicare, una stringa corta casuale comune (Common Reference String, **CRS**) da utilizzare come chiave. Ciò si traduce in linguaggio matematico col concetto di funzione one-way (o unidirezionale), ed in crittografia col concetto di funzione di Hash. Quest'ultima è sufficiente per ottenere una conoscenza zero computazionale. Per permettere lo scambio della chiave è necessaria, tuttavia, una fase iniziale di comunicazione, detta di "trusted setup", in cui un "trusted party" la genera pubblicamente. Un'alternativa al modello CRS è utilizzare un modello a Oracolo Random[6] [7].

1.8.1 Trasformata di Fiat-Shamir

La Trasformata (o euristica) di Fiat-Shamir è una tecnica proposta da Fiat e Shamir per la conversione di un **Σ-protocol** interattivo in uno non-interattivo, ideale per un modello a Oracolo Random. Affinché il metodo funzioni, la prova interattiva originale deve godere della proprietà di essere Public Coin (moneta pubblica), ovvero il Prover ha accesso a tutte le scelte randomiche effettuate dal Verifier durante il protocollo di prova. L'euristica è stata originariamente presentata senza una prova di sicurezza; in seguito, Pointcheval e Stern hanno dimostrato la sua sicurezza contro gli attacchi di messaggi scelti nel modello a Oracolo Random, cioè supponendo che esistano oracoli casuali.
Nel caso in cui non esistano oracoli casuali, l'euristica Fiat-Shamir è stata definita insicura da Shafi Goldwasser e Yael Tauman Kalai. Inoltre se la prova interattiva viene utilizzata come strumento di identificazione, la versione non interattiva può essere utilizzata direttamente come firma digitale utilizzando il messaggio come parte dell'input dell'oracolo casuale [18].
Si osserva ora più formalmente il processo che consente tale conversione.

1.8.2 Come trasformare una dimostrazione a conoscenza zero interattiva in una non-interattiva

Partendo dall'idea di base attenzionata nei paragrafi precedenti, un **Σ-protocol** interattivo si evolve in 3 round: commitment, challenge e response.

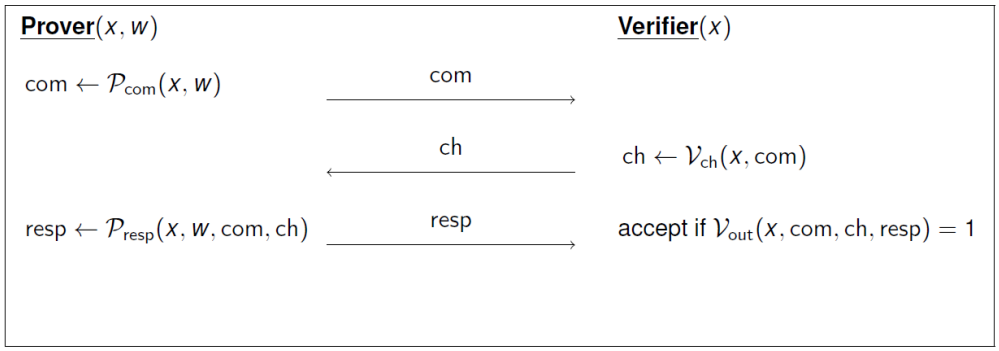

Figura 3 Trasformata Fiat-Shamir, Sigma-Protocol interattivo. *[13]*

[6] Un Oracolo Random è una funzione matematica che associa ad ogni possibile domanda una risposta veramente random scelta all'interno del suo dominio di output. Essi sono utilizzati nelle dimostrazioni crittografiche quando non sono note implementazioni concrete di funzioni con proprietà matematiche richieste dal problema.

Al fine di rafforzare la causalità tra i messaggi di commitment e di challenge, quest'ultimo, in un protocollo non interattivo, verrà calcolato come risultato di una funzione Hash sull'input delle informazioni pubbliche e del commitment.
Il setup e la generazione del commitment e della risposta avviene allo stesso modo del protocollo interattivo, eccetto per lo spazio utilizzato per la challenge che risiedeva nell'insieme $\{0,1\}$, mentre adesso vi è uno spazio maggiore che permette di controllare l'errore di correttezza con la funzione di hash $H : \{0,1\}^* \rightarrow S_{ch}$, dove S_{ch} è lo spazio di output di V_{ch}.

- Il Prover inizia il protocollo calcolando il commitment partendo dall'input x e il segreto w.
- Successivamente il Prover calcola la challenge, la quale non viene più generata dal Verifier, utilizzando una funzione di Hash che prende come parametri in ingresso l'input e il commitment. La funzione di Hash è un modello ad Oracolo Casuale che si comporta come una scatola nera quando esso verrà interrogato. L'output della funzione di Hash per ogni dato input è un valore casuale distribuito uniformemente. Pertanto, la sfida viene scelta randomicamente, come nel **Σ-protocol** interattivo.
- Il Prover inoltre calcola la risposta e la invia al Verifier, contestualmente al commitment e alla challenge.
- Infine il Verifier, prima di accettare o meno la risposta inviata dal Prover, effettua un controllo sulla sfida ricevuta, il risultato dovrà essere identico a quello del Prover in quanto per definizione una funzione di Hash, e in questo caso un Oracolo Casuale, restituirà sempre lo stesso output se l'input non varia.

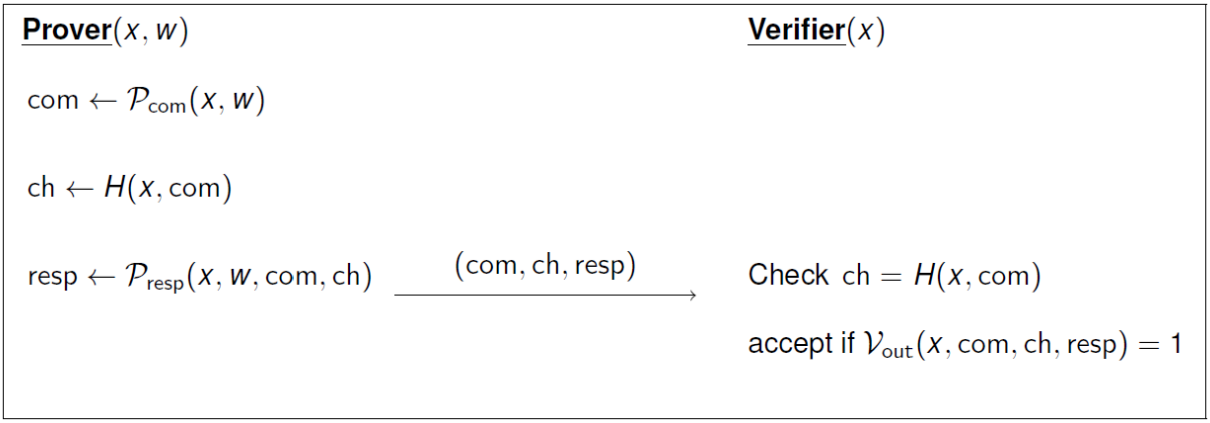

Figura 4 Trasformata Fiat-Shamir, Sigma-Protocol non interattivo. *[13]*

Intuitivamente, l'uso di una funzione di Hash modellata come Oracolo Casuale garantisce che un Prover non possa barare modificando il calcolo del commitment a posteriori, poiché modificandolo significherebbe cambiare la sfida. Questo metodo produce nella pratica argomenti **NIKZ** altamente efficienti.
Nel calcolo dell'Hash possono essere inserite alcune informazioni extra, come ad esempio un messaggio **M**, trasformando così una Proof of Knowledge in uno schema di firma digitale. Inserendo il messaggio nella funzione di Hash, la nuova challenge sarà calcolata come $ch = H(M, x, com)$.
La chiave privata di firma è il segreto/witness, la chiave pubblica è l'input ed infine la firma è la dimostrazione non interattiva. La proprietà di correttezza della dimostrazione a conoscenza zero garantisce l'autenticità della firma.

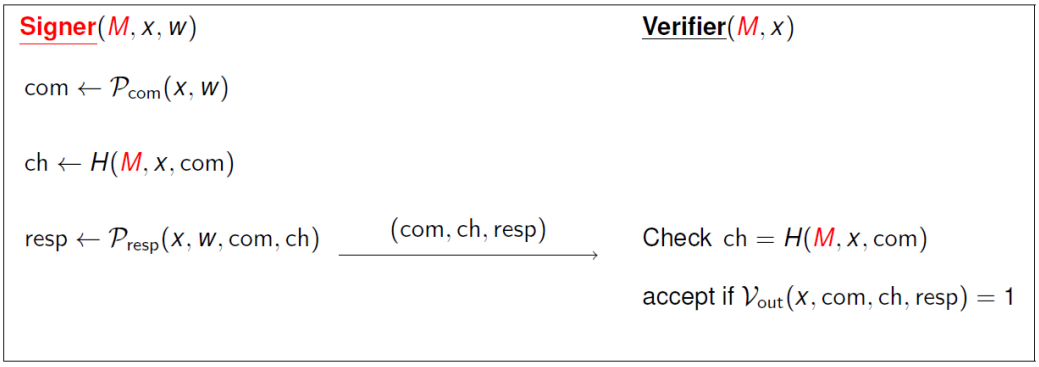

Figura 5 Trasformata Fiat-Shamir, Schema di Firma Digitale. *[13]*

1.9 Esempi dimostrativi di Sigma Protocol

In questa sezione verranno mostrati alcuni esempi di implementazione di Sigma Protocol basandosi su composizioni **AND-Proof** e **OR-Proof.** Nello specifico, definiamo meccanismi per provare la conoscenza di più segreti/testimoni indipendenti (composizione AND) e per provare la conoscenza di un segreto su un insieme (composizione OR).

1.9.1 AND-Proof

Date due relazioni $R^0 = \{(x_0; w_0)\}$ e $R^1 = \{(x_1; w_1)\}$, si ottiene un **Σ-protocol** per la relazione $R^\wedge := \{(x_0, x_1;\ w_0, w_1) \mid (x_0; w_0) \in R^0 \wedge (x_1;\ w_1) \in R^1\}$ eseguendo un **Σ-protocol** per R^0 e un **Σ-protocol** per R^1 in parallelo, utilizzando una sfida comune (supponendo che entrambi i protocolli utilizzino lo stesso spazio di sfida) [19].

1. Il primo algoritmo $P_1(x, w)$ del Prover procede come segue:
   (1) L'algoritmo analizza $(w_0, w_1) := w$ e $(x_0, x_1) := x$.
   (2) Calcola $(a_0, st_0) \leftarrow {P_1}^0(x_0, w_0)$ e $(a_1, st_1) \leftarrow {P_1}^1(x_1, w_1)$.
   (3) L'output dell'algoritmo sarà $(a, st) := ((a_0, a_1), (st_0, st_1))$.
2. Il secondo algoritmo $P_2(x, w, c, st)$ del Prover procede come segue:
   (1) Verifica che $c \in \mathbb{Z}_p$ e se così non fosse interrompe il protocollo.
   (2) Analizza $(st_0, st_1) := st$.
   (3) Calcola $r_0 \leftarrow {P_2}^0(x_0, w_0, c, st_0)$ e $r_1 \leftarrow {P_2}^1(x_1, w_1, c, st_1)$.
   (4) L'output sarà $r := (r_0, r_1)$.
3. L'algoritmo del Verifier $V(x, a, c, r)$ procede come segue:
   (1) Analizza $(r_0, r_1) := r$.
   (2) L'output dell'algoritmo sarà $V^0(x_0, a_0, c, r_0) \wedge V^1(x_1, a_1, c, r_1)$.
4. Il Simulatore $Sim\ (x, c)$ lavora come segue:
   (1) Analizza $(x_0, x_1) := x$.
   (2) Sceglie $c \leftarrow\!\!\shortmid\ \mathbb{Z}_p$.
   (3) Calcola $(x_0, c, r_0) \leftarrow Sim_0(x_0, c)$ e $(x_1, c, r_1) \leftarrow Sim_1(x_1, c)$.
   (4) Infine l'algoritmo genererà tale output $(a, c, r) := ((x_0, x_1), c, (r_0, r_1))$.

Le due singole istanze del **Σ-protocol** vengono eseguite in parallelo, in modo che il Prover generi e invii i due commitment al Verifier. Quindi, quest'ultimo invia una singola sfida

comune e il Prover risponde con le due risposte calcolate dai due singoli **Σ-protocol**. Il Verifier accetta la prova se entrambe le trascrizioni accettanti, corrispondenti ad entrambi i singoli **Σ-protocol**, sono verificate. L'unica restrizione dei due **Σ-protocol** è che entrambi condividono la stessa serie di possibili sfide.
L'estrattore e i simulatori a conoscenza zero sono costruiti in modo semplice, combinando quelli dei singoli **Σ-protocol**. Con ciò si intende che da due trascrizioni accettanti ogni estrattore è in grado di recuperare il segreto corrispondente. Allo stesso modo, da una sfida casuale e risposte generate casualmente, i simulatori possono calcolare i valori adeguati dei commitment che si traducono in trascrizioni simulate distribuite in modo identico. Di conseguenza, una singola sfida può essere utilizzata per dimostrare due affermazioni contemporaneamente [3].

1.9.2 OR-Proof

Tale dimostrazione, tramite una costruzione di base di un **Σ-protocol**, consente ad un Prover di dimostrare che conosce un segreto w, dati due input x_0 , x_1, tale che $(x_0\,; w) \in R \vee (x_1\,; w) \in R$, senza rivelare quale sia il caso trattato.
Pertanto assumiamo che:

- ✓ sia dato un **Σ-protocol** $\mathcal{P}$ per R;
- ✓ $(x_0\,;\ x_1\,)$ siano input comuni a P,V, e che w sia un il segreto che conosciuto solamente da P;
- ✓ $(x_b\,; w) \in R$, dove $b = 0\ \vee\ 1$.

L'idea è quella di far completare al Prover due istanze del protocollo $\mathcal{P}$, rispetto sia a x_0 che a x_1 . Per x_b può eseguire il protocollo realmente, mentre per x_{1-b} dovrà far uso di un simulatore S. Più precisamente, si consideri il seguente protocollo, che chiameremo $\mathcal{P}^{\vee}$:

1. P calcola il primo messaggio a_b in $\mathcal{P}$, utilizzando x_b , w come input;
   P sceglie in modo casuale la challenge c_{1-b} ed esegue il simulatore S sull'input x, c_{1-b} tale che l'output sia $(a_{1-b}, c_{1-b}, r_{1-b})$.
2. V sceglie una stringa casuale s di t-bit e la invia a P.
3. P setta $c_b = s \oplus c_{1-b}$ e calcola la risposta r_b in $\mathcal{P}$ per la challenge c_b utilizzando come parametri in input x_b, a_b, c_b, w , ed invia c_0, r_0, c_1, r_1 a V.
4. V controlla che $s = c_0 \oplus c_1$ e che le conversazioni $(a_0, c_0, r_0), (a_1, c_1, r_1)$ sono conversazioni accettanti in $\mathcal{P}$, rispettivamente su input x_0 e x_1 [20].

<u>*Teorema*</u>. Sia: $R^{\vee} = \{((x_0, x_1), w) \mid (x_0, w)\ \in R \vee (x_1, w)\ \in R\}$, allora il protocollo $\mathcal{P}^{\vee}$ sopra descritto è un **Σ-protocol** per $R^{\vee}$. Inoltre per ogni Verifier V^*, la probabilità di distribuzione delle conversazioni tra P e V^*, dove w è tale che $(x_b\,; w) \in R$, è indipendente da b .

<u>*Dimostrazione*</u>. È chiaro che il protocollo presenta la canonica forma dello scambio di messaggi in 3 mosse. Per verificare la correttezza, siano date due conversazioni accettanti $(a_0, a_1, s,\ c_0, c_1, r_0,\ r_1), (a_0, a_1, s',\ c_0', c_1', r_0',\ r_1')$ con s $\neq$ s′, allora per qualsiasi $n = 0\ \vee\ 1$, $c_n \neq c_n'$, e successivamente da $(a_n, c_n, r_n), (a_n, c_n', r_n')$, è possibile calcolare il segreto w tale che $(x_n, w) \in R$, grazie alla correttezza speciale del **Σ-protocol** $\mathcal{P}$.

Per la proprietà di SHVZK, dato $s = c_0 \oplus c_1$ con c_0, c_1 casualmente, si esegue il simulatore S su entrambi gli input $(x_0, c_0), (x_1, c_1)$.
Infine assumiamo di avere un arbitrario Verifier V^*, e si osservi che la distribuzione delle conversazioni tra il Prover e il Verifier può essere analizzata come segue:

- hanno la forma $(a_0, a_1, s, c_0, c_1, r_0, r_1)$, dove a_0, a_1 sono esattamente distribuiti come avrebbe scelto un Prover onesto (grazie alla **perfect honest verifier zero-knowledge** del protocollo $\mathcal{P}$);
- s ha qualunque distribuzione che V^* emette come output, dato x_0, x_1, a_0, a_1. Inoltre c_0, c_1 sono casuali, $s = c_0 \oplus c_1$;
- infine r_0 ha qualunque distribuzione che un Prover onesto emette come output, in quanto l'input era x_0 e la prima parte della conversazione era a_0, c_0 .

Conclusioni simili potevano essere ottenute considerando il caso con risposta r_1. Questo è triviale in seguito alla proprietà di perfect honest verifier zero-knowledge per r_{1-b}, e di conseguenza la distribuzione non dipende da b.
Da quest'ultima affermazione si evince che il protocollo $\mathcal{P}^V$ può essere definito come "witness indistinguishable" (WI), ovvero come testimone indistinguibile perché vi sono diversi valori di w che un Prover può conoscere che gli consentirebbero di completare il protocollo con successo. Ma purtroppo non vi è un modo per il quale si possa affermare, dalle conversazioni, quale dei possibili valori egli conosca [20].

Capitolo 2

2 Blockchain: il futuro della Sicurezza Digitale e delle Transazioni

La blockchain può essere descritta come un registro pubblico e distribuito che archivia tutti gli eventi digitali conclusi e condivisi tra i membri della rete. Ogni transazione viene confermata attraverso il consenso della maggior parte dei partecipanti, e una volta registrati, i dati non possono essere modificati o eliminati. Questo sistema offre un registro chiaro e trasparente di ogni singola transazione. Bitcoin è il caso più conosciuto di utilizzo della blockchain, ma non è un esempio isolato. Questa tecnologia ha creato un mercato multimiliardario di transazioni non regolamentate, portando con sé sfide regolamentari significative riguardanti i governi e le istituzioni finanziarie. Nonostante queste sfide, i benefici della blockchain superano le difficoltà normative e centralizzate. Un'applicazione importante della blockchain è rappresentata dagli smart contracts o contratti intelligenti, i quali sono strumenti informatici progettati per automatizzare l'esecuzione di accordi tra le parti. Inoltre, la "proprietà intelligente" utilizza la blockchain per gestire il controllo di beni e risorse che possono riguardare sia beni fisici (come automobili e case) sia beni intangibili (come le azioni aziendali) [21].
Nel contesto attuale, le criptovalute, e in particolare Bitcoin, sono diventate un argomento di grande interesse sia nell'industria che nel mondo accademico. Bitcoin ha raggiunto un mercato di 10 miliardi di dollari nel 2016 grazie alla sua struttura di archiviazione dei dati che consente transazioni senza intermediari. La blockchain, introdotta nel 2008 e operativa dal 2009, funziona come un registro pubblico in cui le transazioni vengono registrate in una serie di blocchi che si accumulano nel tempo. Le sue caratteristiche principali, come decentralizzazione, persistenza e anonimato, contribuiscono a ridurre i costi e migliorare l'efficienza.
Tuttavia, la blockchain deve affrontare alcune sfide tecniche, come la scalabilità. La dimensione del blocco di Bitcoin è attualmente limitata a 1 MB e un blocco viene estratto circa ogni dieci minuti, limitando la rete a circa 7 transazioni al secondo, insufficiente per il trading ad alta frequenza. Aumentare le dimensioni dei blocchi richiederebbe più spazio di archiviazione e rallenterebbe la propagazione nella rete, rischiando di portare alla centralizzazione. Altri problemi includono il mining egoistico, che può ridurre la sicurezza della rete e causare biforcazioni, e la possibile perdita di privacy, anche se le transazioni sono effettuate con chiavi pubbliche e private. Inoltre gli algoritmi di consenso, come il Proof of Work e il Proof of Stake, hanno problemi come l'alto consumo energetico e la concentrazione della ricchezza [22].

2.1 Glossario pratico

Questa tecnologia rappresenta una delle innovazioni più significative dell'era digitale, rivoluzionando la gestione delle informazioni, delle transazioni e dei processi economici su scala globale. Tuttavia, per comprendere appieno il suo potenziale trasformativo, è fondamentale acquisire una solida padronanza della terminologia specifica che la caratterizza. Questo glossario pratico si propone di fornire una panoramica esaustiva e approfondita dei concetti chiave legati alla blockchain, spiegando in maniera dettagliata e rigorosa i termini e i principi fondamentali che definiscono questo ambito in continua evoluzione.

- **Blockchain**. Una blockchain è un registro digitale peer-to-peer delle transazioni, che può essere distribuito pubblicamente o privatamente tra tutti gli utenti, rendendolo decentralizzato e distribuito. Questa tecnologia impiega la crittografia e un meccanismo di consenso per verificare le transazioni, assicurandone la legittimità, prevenendo la doppia spesa e consentendo transazioni di alto valore in un contesto privo di fiducia. La blockchain garantisce trasparenza e elimina la necessità di intermediari o amministratori esterni.
- **Distributed ledger technology** (DLT). Anche se spesso viene usata come sinonimo di blockchain, la DLT si riferisce in modo più specifico all'aspetto del registro distribuito e decentralizzato presente nella tecnologia blockchain. Con la DLT, un registro può essere gestito, protetto e autenticato facendo affidamento su una rete decentralizzata di computer, piuttosto che su un'unica autorità centrale. Di conseguenza, il registro viene conservato e gestito da molteplici individui o organizzazioni (distribuito), senza che esista una copia principale o master.
- **Proof of Work** (PoW). È uno dei due principali meccanismi di consenso utilizzati per validare le transazioni su una blockchain. Nel sistema Proof of Work, i partecipanti alla rete, chiamati miner, competono per aggiungere il prossimo blocco di transazioni alla blockchain risolvendo un complesso puzzle crittografico. Questo processo non solo convalida le transazioni precedenti, ma consente ai miner di guadagnare commissioni di transazione come ricompensa per il loro lavoro.
- **Proof of Stake** (PoS). In questo sistema, i partecipanti alla rete, chiamati validatori, "bloccano" una certa quantità di monete digitali come garanzia. La probabilità che un validatore venga scelto per confermare un blocco è direttamente proporzionale alla quantità di monete che ha investito nella rete. Di conseguenza maggiore è la quota di monete che un validatore ha "messo in gioco", maggiori sono le sue possibilità di verificare e aggiungere nuovi blocchi alla blockchain.
- **Mining**. È il processo attraverso il quale gli utenti, conosciuti come miner, verificano e convalidano le transazioni su una blockchain che utilizza il meccanismo di consenso Proof of Work. Questo processo implica la risoluzione di complessi problemi crittografici per aggiungere nuovi blocchi alla catena e garantire l'integrità e la sicurezza della rete.
- **Valuta Virtuale**. Si tratta di una forma digitale di valore che può essere scambiata online e utilizzata come unità di conto o riserva di valore. La valuta virtuale può presentarsi sotto forma di monete digitali o token, emessi da organizzazioni virtuali (come The DAO, acronimo di ***"Decentralized Autonomous Organization"***, ovvero ***"Organizzazione Autonoma Decentralizzata"***). Questi strumenti possono conferire diritti specifici, come la possibilità di rivendere il token o ricevere un rimborso. A differenza delle valute fiat, che sono ufficialmente riconosciute e garantite da un governo ma non sono supportate da beni fisici come l'oro (come il dollaro americano), la valuta virtuale non è legata a un'autorità centrale e non ha valore intrinseco supportato da beni fisici.
- **Criptovaluta**. È una forma di valuta virtuale che utilizza la crittografia per garantire la sicurezza delle transazioni, piuttosto che affidarsi a un'autorità centrale di controllo. Esempi noti di criptovalute includono Bitcoin o Ethereum. Queste valute digitali fungono da unità di valore per facilitare le transazioni all'interno della rete blockchain su cui operano.

- **Token**. È una forma di criptovaluta sviluppata su una blockchain che offre una varietà di funzioni oltre a, o in alternativa a, quella di semplice valuta, sia all'interno della piattaforma che al di fuori di essa. Sia i token che le altre valute virtuali possono potenzialmente aumentare di valore, soprattutto se cresce la domanda per le applicazioni e le funzionalità specifiche legate a un determinato token o valuta virtuale [23].

2.2 Architettura della Blockchain

La blockchain è descritta come un database distribuito, condiviso e accettato tra una rete peer-to-peer[7]. È composta da una catena di blocchi collegati, ognuno dei quali contiene transazioni con timestamp, protette da crittografia a chiave pubblica e validate dalla comunità della rete. Una volta che un dato è aggiunto alla blockchain, diventa immutabile, trasformando così la blockchain in un registro permanente delle attività passate. Quando un blocco si riempie, i nodi eseguono contemporaneamente il Proof of-Work. Questi calcoli sono essenziali per garantire la sicurezza del sistema, poiché obbligano i nodi a consumare risorse di calcolo, evitando l'inclusione di transazioni fraudolente. Il primo nodo che risolve il puzzle Proof-of-Work diffonde sia la soluzione che il blocco di transazioni agli altri nodi, che verificano velocemente la correttezza del blocco. Una volta che il 51% della potenza computazionale della rete approva il blocco, si procede alla registrazione delle nuove transazioni all'interno di un nuovo blocco, estendendo così la catena in modo sicuro [24].
Ogni blocco nella blockchain si collega direttamente al blocco precedente attraverso un riferimento, che è essenzialmente un valore hash del blocco antecedente, chiamato "blocco padre". Il primo blocco della blockchain, noto come "blocco genesi", è l'unico a non avere un blocco padre. Un blocco si compone di due parti principali: l'intestazione (header) e il corpo (body), come illustrato nella figura.

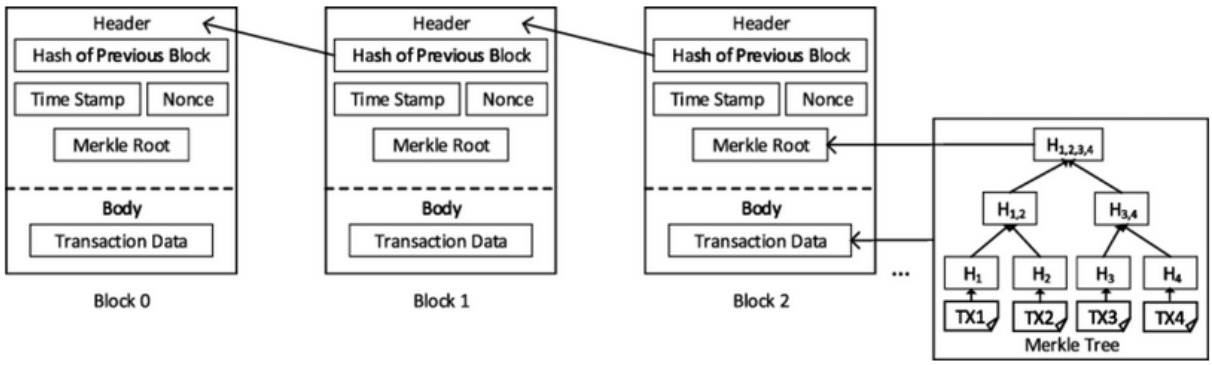

***Figura 6 Struttura blocchi della Blockchain** [25]*

La struttura del blocco header è così composta:

- **Versione del blocco**: Indica quale insieme di regole di convalida devono essere seguite per quel particolare blocco. In pratica, consente di aggiornare le regole di consenso man mano che la blockchain si evolve, senza compromettere i blocchi precedenti;
- **Hash del blocco padre**: Un valore hash a 256 bit che fa riferimento al blocco precedente nella catena. Questo collegamento cripta le informazioni del blocco precedente e

[7] Peer-to-peer(p2p) o rete paritetica: architettura di rete informatica in cui I nodi sono tra loro paritetici, potendosi comportare sia da client che da server.

garantisce la continuità e l'integrità della blockchain, poiché ogni blocco dipende dal precedente;

- **Hash radice dell'albero Merkle**: Questo è il valore hash che rappresenta tutte le transazioni incluse nel blocco. L'albero di Merkle è una struttura che consente di raggruppare le transazioni e assicurare che i dati non siano stati alterati. L'hash radice è un singolo valore che riassume tutte le transazioni nel blocco in modo efficiente;
- **Timestamp**: Il timestamp rappresenta l'ora esatta in cui il blocco è stato creato, espresso in secondi a partire dal 1° gennaio 1970 alle 00:00 UTC (nota come Epoch Time). Questo valore è importante per ordinare cronologicamente i blocchi;
- **nBits**: Rappresenta il target di difficoltà attuale per il calcolo dell'hash del blocco, espresso in un formato compatto. Questo valore determina quanto è difficile trovare un hash che soddisfi i requisiti di validazione del blocco, influenzando così il processo di mining;
- **Nonce**: È un campo di 4 byte che solitamente inizia da 0 e aumenta progressivamente a ogni tentativo di calcolo dell'hash del blocco. I miner modificano il valore del nonce ripetutamente per cercare di trovare un hash valido che soddisfi le regole di difficoltà definite dal target (nBits).

Il body del blocco invece è composto da un contatore di transazioni e dalle transazioni stesse. Il numero massimo di transazioni che un blocco può contenere dipende dalla dimensione del blocco e dalla dimensione di ciascuna transazione. La blockchain utilizza un meccanismo di crittografia asimmetrica per convalidare l'autenticità delle transazioni [26]. In un ambiente non affidabile, viene utilizzata una firma digitale basata sulla crittografia asimmetrica.
La blockchain affronta il problema della doppia spesa grazie alla crittografia a chiave pubblica, in cui ogni utente possiede una chiave privata e condivide la chiave pubblica con gli altri partecipanti. Il concetto centrale è un database distribuito che contiene registrazioni di transazioni verificate dal consenso della maggior parte dei partecipanti, rendendo impossibile che transazioni fraudolente superino la verifica collettiva. Una volta che un record è creato e accettato, non può più essere modificato. Questo processo consente la creazione di un timestamp elettronico comune, affidabile per tutti i partecipanti, e permette di verificare facilmente l'origine e l'autenticità delle informazioni, senza la necessità di un intermediario esterno per convalidare i dati.

2.2.1 Caratteristiche

La blockchain può essere suddivisa in tre principali categorie: **pubblica**, **privata** e **consortile**, ognuna delle quali presenta caratteristiche e applicazioni specifiche, ma tutte condividono alcuni principi fondamentali.
La **decentralizzazione** è uno degli aspetti centrali della blockchain. Nei sistemi tradizionali, ogni transazione deve essere convalidata attraverso un'agenzia centrale di fiducia, come una banca, il che comporta costi e colli di bottiglia sui server centrali. Al contrario, nelle reti blockchain, le transazioni possono avvenire direttamente tra nodi peer to peer (P2P), senza la necessità di un'autorità centrale, riducendo i costi di sviluppo e operativi e migliorando l'efficienza del sistema. Un esempio di **blockchain pubblica** è Bitcoin, di cui parleremo nei paragrafi successivi, dove ogni nodo può partecipare al processo di consenso e le transazioni sono visibili a tutti, garantendo una totale trasparenza.

La **persistenza** è un'altra caratteristica chiave: le transazioni sono memorizzate in blocchi distribuiti su tutta la rete e, una volta confermate, è quasi impossibile alterarle. Questo garantisce un alto livello di sicurezza, poiché qualsiasi tentativo di falsificazione viene facilmente rilevato dagli altri nodi della rete. Tuttavia, la **blockchain consortile** o privata, pur garantendo una maggiore efficienza grazie al minor numero di validatori, può essere vulnerabile a manomissioni se la maggioranza del consorzio o l'organizzazione dominante decide di alterare la catena.
Un altro aspetto importante è l'**anonimato**. Gli utenti della blockchain possono interagire utilizzando indirizzi generati, senza rivelare la propria identità, e possono creare più indirizzi per aumentare la privacy. Tuttavia, questa non è una protezione completa, poiché la privacy può essere compromessa in determinate circostanze, come accade nelle blockchain pubbliche.
Infine, l'**auditabilità** della blockchain ne migliora la tracciabilità e la trasparenza. Poiché ogni transazione è registrata con un timestamp e validata da vari nodi, è possibile verificare e tracciare tutte le transazioni precedenti. Questo è particolarmente evidente nella blockchain di Bitcoin, dove ogni transazione può essere tracciata in modo iterativo fino a quelle precedenti, garantendo un elevato livello di trasparenza dei dati [27].
Questi aspetti distintivi non solo facilitano la creazione di nuove soluzioni tecnologiche, ma contribuiscono anche a migliorare l'efficienza e la sicurezza nei vari ambiti di applicazione della blockchain. Con l'evoluzione continua della tecnologia e l'espansione dei suoi usi, la blockchain rappresenta una delle innovazioni più promettenti del nostro tempo, capace di trasformare profondamente numerosi settori e processi tradizionali.

2.2.2 Applicazioni

La blockchain, grazie alle sue caratteristiche, trova applicazione in una vasta gamma di settori. Esploriamo come questa tecnologia stia cambiando il mondo e quali sono le principali aree di applicazione.

- Quando si pensa alla blockchain, la prima applicazione che viene in mente è spesso quella delle criptovalute. Bitcoin ed Ethereum sono esempi emblematici di come la blockchain possa rivoluzionare il sistema monetario tradizionale. Queste criptovalute utilizzano la blockchain per gestire le transazioni in modo sicuro e trasparente, senza la necessità di intermediari come le banche. Inoltre, il mondo delle criptovalute ha aperto la strada a nuovi modelli di investimento e di finanziamento, come le Initial Coin Offerings (ICO) e le soluzioni di Finanza Decentralizzata (DeFi);
- un'altra applicazione affascinante della blockchain è rappresentata dai contratti intelligenti, o smart contracts. Questi sono essenzialmente programmi che si auto-eseguono quando vengono soddisfatte determinate condizioni. Ad esempio si può pensare ad un contratto che automatizza il pagamento di una polizza assicurativa non appena viene confermato un ritardo di volo: il processo è completamente automatizzato, riducendo il rischio di errore umano e velocizzando le operazioni. I contratti intelligenti trovano applicazione in numerosi ambiti, dall'assicurazione al settore immobiliare, rendendo i processi più efficienti e meno suscettibili a manipolazioni;
- nel mondo della gestione della supply chain, la blockchain sta portando un cambiamento significativo. Grazie alla sua capacità di registrare ogni fase della catena di fornitura in modo immutabile, le aziende possono ora tracciare il percorso dei prodotti dalla produzione al consumatore finale. Questo non solo aumenta la trasparenza, ma aiuta

anche a prevenire le frodi e a garantire la qualità dei prodotti. Ad esempio, nel settore alimentare, la blockchain permette di monitorare e verificare l'origine degli alimenti, offrendo ai consumatori una maggiore fiducia nella qualità e nella sicurezza dei prodotti che acquistano;

- nel settore sanitario, la blockchain promette di migliorare la gestione dei dati dei pazienti. Le informazioni mediche possono essere condivise tra istituzioni e professionisti della salute in modo sicuro e autorizzato, migliorando l'efficienza e garantendo una maggiore privacy.

La blockchain sta aprendo nuove possibilità in molti settori, offrendo soluzioni che migliorano la sicurezza, la trasparenza e l'efficienza. Anche se ci sono ancora delle sfide da affrontare, come la scalabilità e la regolamentazione, le sue applicazioni continuano a espandersi, promettendo di cambiare profondamente il modo in cui gestiamo le informazioni e le transazioni nella nostra vita quotidiana.

2.2.3 Sfide e limitazioni

La blockchain, pur offrendo innovazioni e opportunità straordinarie, non è esente da sfide e limitazioni. Comprendere queste problematiche è essenziale per valutare il suo potenziale e le aree in cui è necessario ulteriore sviluppo.
Uno dei principali problemi della blockchain è la scalabilità. La blockchain offre un sistema decentralizzato che elimina la necessità di intermediari, migliorando l'efficienza e riducendo i costi nelle transazioni. Tuttavia, con l'aumento del numero di transazioni, la rete blockchain può diventare congestionata, portando a tempi di conferma più lunghi e a costi di transazione più elevati durante i periodi di alta domanda.
Un altro aspetto cruciale riguarda la sicurezza e le vulnerabilità. La blockchain è progettata per essere altamente sicura grazie alla sua struttura decentralizzata e all'uso della crittografia. Le transazioni sono registrate in blocchi immutabili, che rendono difficile la manipolazione dei dati e garantiscono un alto livello di integrità. Tuttavia, nonostante questa sicurezza intrinseca, la blockchain non è immune da vulnerabilità. Attacchi come quelli del "51%", in cui un gruppo di miner acquisisce il controllo della maggior parte della potenza di calcolo della rete, possono minacciare la sicurezza della blockchain. Inoltre, i contratti intelligenti, se non scritti correttamente, possono contenere errori di codifica che possono essere sfruttati da malintenzionati. Sebbene sia teoricamente possibile, prendendo come esempio la blockchain di Bitcoin, essa non è mai stata vittima di un attacco del 51% andato a buon fine. Con l'espansione del network, la sua sicurezza aumenta, rendendo improbabile che i miner scelgano di investire ingenti risorse per attaccare la rete, poiché le ricompense derivanti da un comportamento onesto sono più vantaggiose. Inoltre, un attacco del 51% riuscirebbe a manipolare solo le transazioni recenti per un breve periodo, poiché i blocchi sono legati da prove crittografiche, e modificare i blocchi più vecchi richiederebbe una potenza computazionale inimmaginabile [28].
Il consumo energetico rappresenta un'altra importante sfida. La blockchain, in particolare nei sistemi basati su algoritmi di consenso come il Proof of Work, può essere estremamente dispendiosa in termini di energia. L'estrazione di criptovalute richiede una considerevole quantità di potenza di calcolo, con conseguenti alti consumi energetici. Questo non solo ha implicazioni ambientali, ma comporta anche costi operativi elevati per i miner e per le reti che utilizzano queste tecnologie. Di fatto alcuni progetti stanno esplorando alternative più

ecologiche, come il Proof of Stake, che promettono di ridurre il consumo energetico.
Infine, la regolamentazione è un altro aspetto cruciale da considerare. La blockchain offre un alto livello di trasparenza e integrità dei dati, il che può facilitare la conformità a normative e regolamenti in vari settori. Tuttavia, la sua natura decentralizzata e spesso anonima può complicare l'applicazione delle normative esistenti, creando incertezze legali. Le autorità di regolamentazione stanno cercando di adattare le leggi per affrontare queste sfide, ma la mancanza di un quadro normativo chiaro può ostacolare l'adozione su larga scala.

In definitiva, la blockchain non è solo una tecnologia del presente, ma un pilastro fondamentale per lo sviluppo delle future infrastrutture digitali. Come società, sarà cruciale trovare il giusto equilibrio tra innovazione, regolamentazione e sostenibilità, per sfruttare appieno i benefici di questa trasformazione tecnologica.

2.3 Bitcoin: l'alba delle Crypto

Verso la fine del 2008, un individuo o un gruppo di persone sotto lo pseudonimo "Satoshi Nakamoto", di cui l'identità reale è ancora sconosciuta, pubblicò online un documento intitolato "Bitcoin: A Peer-to-Peer Electronic Cash System" [29]. Questo whitepaper segnò l'inizio dell'era delle criptovalute e della tecnologia blockchain, che, ha il potenziale per rivoluzionare lo scambio di informazioni come non accadeva dai tempi della nascita di internet.
Negli ultimi decenni, il commercio elettronico è cresciuto esponenzialmente, facendo affidamento principalmente su istituzioni finanziarie che fungono da terze parti fidate per elaborare i pagamenti elettronici. Tuttavia, questo modello presenta diverse problematiche. Le transazioni completamente irreversibili non sono possibili poiché le istituzioni finanziarie devono intervenire per risolvere le dispute, aumentando così i costi di transazione e limitando le transazioni di piccola entità. Inoltre, il bisogno di fiducia si diffonde, costringendo i commercianti a richiedere più informazioni dai clienti e accettare una certa percentuale di frodi come inevitabile.

2.3.1 Le transazioni in Bitcoin: l'uso della crittografia e dell'hashing

Bitcoin propone un sistema di pagamento elettronico basato su prova crittografica, consentendo transazioni dirette tra le parti senza la necessità di intermediari fidati. Il cuore del sistema Bitcoin è la gestione delle transazioni attraverso una catena di firme digitali. Ogni proprietario di Bitcoin trasferisce la proprietà della moneta firmando digitalmente l'hash della transazione precedente e la chiave pubblica del successivo proprietario. Questo crea una catena continua di proprietà che può essere verificata dal destinatario, assicurando che l'importo inviato non sia già stato speso in una transazione precedente.
Il proprietario avvia il trasferimento della moneta al successivo firmando digitalmente un hash della transazione precedente, crittografandolo. La crittografia utilizzata da Bitcoin impiega due chiavi matematicamente correlate: una chiave pubblica e una chiave privata.
La prima è necessaria per crittografare la transazione insieme alla chiave privata del proprietario per creare la firma digitale; è importante sottolineare che l'indirizzo Bitcoin riportato in una transazione non corrisponde direttamente alla chiave pubblica del destinatario. Esso viene

generato a partire dalla chiave pubblica tramite un processo di hashing (SHA-256 seguito da RIPEMD-160) e successiva codifica in formato Base58Check. L'indirizzo così ottenuto identifica in modo univoco il destinatario all'interno della rete e viene associato alla "fine" della moneta nel momento del trasferimento.

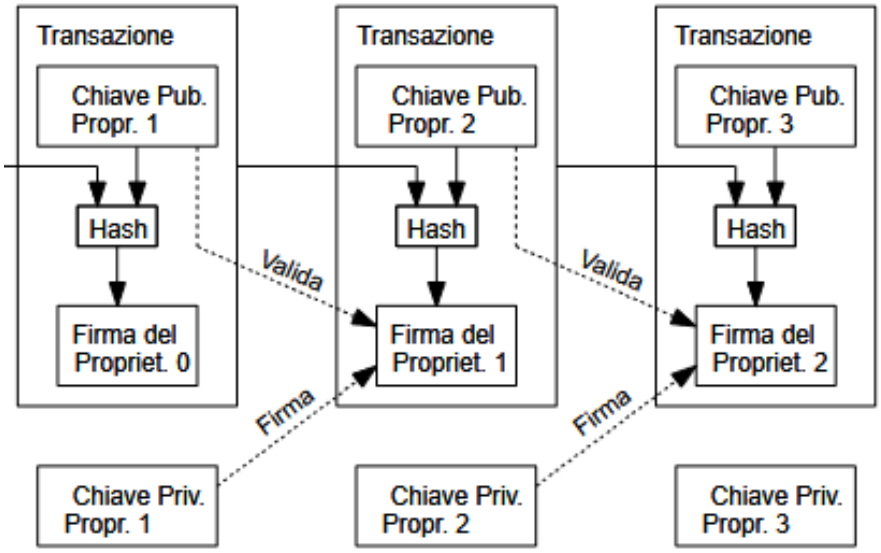

Figura 7 Transazioni nella rete di Bitcoin [29]

Ma come può il destinatario, ovvero il successivo proprietario, assicurarsi che l'importo inviatogli non sia stato già speso in una transazione precedente?
L'unico modo per garantire questo è che la rete raggiunga un consenso su tutte le transazioni precedenti e sull'ordine in cui sono state effettuate. L'ordine valido di tutte le transazioni nella rete deve essere annunciato pubblicamente affinché tutti possano sapere cosa è valido. Ogni destinatario di una transazione desidera una prova che, al momento della ricezione, la maggior parte della rete concordi sul fatto che lui sia stato il primo a ricevere quella transazione e che nessun altro destinatario l'abbia ricevuta in precedenza [30].
La rete Bitcoin opera su un sistema distribuito di computer. Tutti i processi della rete vengono eseguiti simultaneamente su centinaia e migliaia di computer, detti nodi, situati in vari paesi in tutto il mondo. Maggiore è il numero di computer nella rete, maggiore è il numero di copie dei record, rendendo il sistema ancora più sicuro. Per dimostrare l'ordine in cui le transazioni sono state generate, la rete Bitcoin utilizza un "*timestamp server distribuito*" che aggiunge una marcatura temporale all'hash di un blocco simultaneamente su tutti i nodi della rete. Il timestamp fornisce la prova che i dati devono necessariamente esistere a quel momento e ogni timestamp include il timestamp precedente nel suo hash. In questo modo, si forma una catena in cui ogni nuovo timestamp rafforza quelli precedenti.

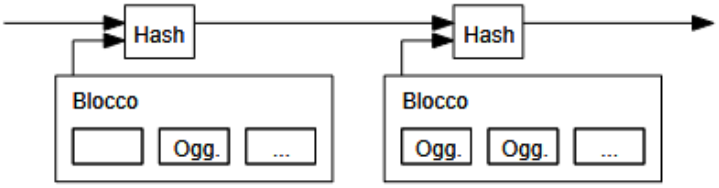

Figura 8 Aggiunta dell'Hash nel blocco [29]

La rete è decentralizzata e peer-to-peer, permettendo così ai nodi di unirsi e lasciare la rete a piacimento. I nodi trasmettono le transazioni su base “best effort” e accettano la catena di blocchi più lunga come prova della sequenza cronologica degli eventi. Questo approccio decentralizzato garantisce la resilienza della rete contro attacchi e fallimenti singoli, poiché non esiste un punto di controllo centrale.

2.3.2 Protocollo di consenso: il Proof of Work

Per mantenere l’integrità del registro delle transazioni, Bitcoin utilizza un protocollo di consenso chiamato **Proof of Work**. Questo sistema richiede che i nodi della rete risolvano complessi problemi matematici per aggiungere nuovi blocchi alla blockchain. Il Proof of Work non solo garantisce la sicurezza del sistema, ma rende anche estremamente difficile per qualsiasi attaccante modificare le transazioni passate senza rifare il lavoro computazionale di tutti i blocchi successivi. Questo meccanismo assicura che la catena di blocchi più lunga, rappresentante la maggioranza della potenza di calcolo onesta, sia considerata valida.
In termini generali, il "Proof of Work" consiste nella risoluzione di un compito moderatamente difficile da parte di un utente sul proprio computer. Questa attività deve soddisfare determinati requisiti predefiniti ed è intrinsecamente complessa da eseguire. Inizialmente, il protocollo Proof of Work fu ideato per ridurre l’invio di e-mail di spam richiedendo al mittente di completare alcune piccole attività (“lavoro”) prima di poter inviare un’e-mail.
Nella rete Bitcoin, questo compito si è evoluto nella risoluzione di un puzzle crittografico. Un certo numero di transazioni viene raggruppato in un blocco, che contiene vari dati:

- un indice;
- un timestamp;
- un elenco delle transazioni;
- una prova;
- l’hash del blocco precedente;
- informazioni di controllo;
- Un campo chiamato nonce (numero usato una sola volta), il quale viene aggiunto alla fine di questo blocco per eseguirne l’hashing.

I nodi della rete Bitcoin, chiamati anche *miners* o "*minatori*", iniziano a esaminare, testare e scartare milioni di nonce ogni secondo per trovare quello che soddisfi l’obiettivo fissato dalla rete al momento della creazione del blocco. Continuano questo "lavoro" finché non trovano un valore che dia all’hash del blocco il livello di difficoltà richiesto: un inizio con un certo numero di bit zero.

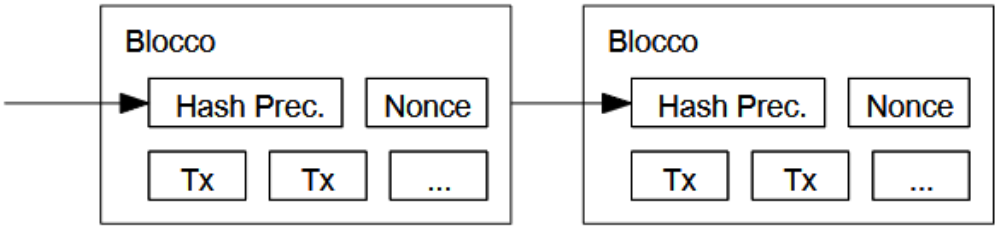

***Figura 9 Concatenazione dei blocchi di Bitcoin dopo l’applicazione del PoW** [29]*

Per incentivare i nodi a sostenere la rete, Bitcoin introduce un meccanismo di ricompensa. Ogni volta che un nodo risolve un problema di Proof of Work e aggiunge un nuovo blocco alla

blockchain, viene ricompensato con una certa quantità di nuovi Bitcoin. Questo sistema non solo incoraggia la partecipazione alla rete, ma fornisce anche un metodo per distribuire inizialmente le monete. Col tempo, la ricompensa per blocco diminuisce e i nodi saranno incentivati principalmente dalle commissioni di transazione. Poiché per frodare la rete sarebbe necessaria un'enorme potenza di calcolo, è più probabile che i nodi rimangano onesti. Questo perché investire la potenza di calcolo nel mining e nella generazione di nuove monete è più redditizio rispetto a investire risorse per cercare di ottenere il controllo della rete. Di fatto la probabilità che un utente malintenzionato raggiunga il lavoro dei nodi onesti diminuisce in modo esponenziale mano a mano che vengono aggiunti blocchi successivi.

2.3.3 RGB++ e Bitcoin: Strategie avanzate per l'offuscamento delle transazioni

Il protocollo RGB, non legato al significato comune di Red-Green-Blue, ma bensì a "Really Good Bitcoin", rappresenta un avanzamento significativo nell'ecosistema delle blockchain, in particolare per quanto riguarda l'espansione delle capacità della blockchain di Bitcoin. RGB è concepito come un protocollo di livello superiore che consente la creazione e la gestione di asset digitali, quali token, NFT e smart contracts, senza sovraccaricare la blockchain principale. Lo si può intendere come un sistema di smart contracts che utilizza la validazione lato client. Funziona come un secondo livello (layer 2) su Bitcoin, integrandosi con la rete Lightning Network[8]. La validazione lato client, che attenzioneremo in seguito, implica che tutte le informazioni e le regole relative al "contratto intelligente" vengono gestite al di fuori della blockchain principale di Bitcoin. Questa soluzione sfrutta un'architettura off-chain che migliora sia la privacy che l'efficienza delle transazioni.
Il protocollo RGB è nato da una serie di contributi di diversi autori susseguitisi nel tempo. È stato ispirato dai primi lavori di Peter Todd sulla validazione lato client e sui sigilli monouso. Giacomo Zucco, nel 2016, diede una prima idea per RGB il quale è stato implementato nel 2017. Successivamente, nel 2019, il Dr. Maxim Orlovsky, insieme a Giacomo Zucco, hanno creato l'Associazione "Lightning Network and Bitcoin Protocol" (LNP/BP), e il Dr. Orlovsky è diventato il lead designer e principale contributore del protocollo RGB [31].
Esso si basa su tre principi fondamentali: decentralizzazione, privacy e scalabilità.

- La decentralizzazione viene garantita dal fatto che le operazioni vengono eseguite principalmente fuori dalla catena principale, riducendo l'impatto sulla blockchain di Bitcoin. Questo consente di mantenere la natura distribuita della rete, senza la necessità di affidarsi a intermediari centralizzati.
- La privacy è migliorata attraverso l'utilizzo di tecniche crittografiche avanzate, che consentono l'offuscamento delle informazioni relative alle transazioni. Gli utenti possono trasferire asset senza rivelare dettagli sensibili, proteggendo così la loro identità e le loro transazioni da occhi indiscreti.
- La scalabilità è ottenuta grazie alla gestione off-chain delle operazioni, che diminuisce il carico sulla blockchain e aumenta l'efficienza complessiva del sistema. Ciò consente

[8] Il Lightning Network è una rete di micropagamenti off-chain per Bitcoin che consente transazioni rapide e a basso costo mediante canali di pagamento bidirezionali. I nodi aprono canali su blockchain e aggiornano il bilancio localmente fino alla chiusura del canale, riducendo il carico on-chain e aumentando la scalabilità. L'uso di contratti Hashed Time-Locked (HTLC) garantisce la sicurezza nei pagamenti multi-hop.

di supportare un numero maggiore di transazioni senza compromettere le prestazioni della rete.

Vediamo come, tramite RGB++, vengono introdotte nuove funzionalità avanzate per l'offuscamento delle transazioni, migliorando significativamente la privacy degli utenti. Una delle caratteristiche principali è la confidenzialità degli asset: le informazioni sulla proprietà e la quantità degli asset trasferiti non sono immediatamente visibili sulla blockchain pubblica. Questo è reso possibile tramite tecniche crittografiche come le zero-knowledge proofs (ZKPs), trattate nel primo capitolo, che permettono di dimostrare la validità di una transazione senza rivelare dettagli specifici. Le ZKPs consentono di verificare l'integrità delle transazioni e la conformità alle regole del protocollo senza esporre dati sensibili. L'architettura off-chain del protocollo è una componente chiave per l'offuscamento delle transazioni. Le transazioni off-chain infatti vengono registrate solo dai partecipanti diretti, mantenendo così un alto livello di riservatezza.

Ciò significa che le informazioni dettagliate sulle transazioni non vengono pubblicate sulla blockchain pubblica, rendendo molto più difficile osservare e tracciare i movimenti degli asset. Le firme sovrapposte rappresentano un ulteriore livello di sicurezza, garantendo che solo le parti coinvolte possano accedere ai dettagli completi della transazione. Questo viene ottenuto combinando diverse tecniche crittografiche per mascherare le informazioni specifiche della transazione. Ad esempio, una transazione può essere firmata utilizzando chiavi pubbliche che non rivelano l'identità dei partecipanti fino a quando non viene effettuata una verifica crittografica da parte delle parti autorizzate. Questo approccio non solo migliora la privacy, ma aumenta anche la sicurezza complessiva del sistema, poiché riduce il rischio di attacchi mirati basati su informazioni pubblicamente disponibili. Gli utenti possono quindi operare in un ambiente più sicuro, sapendo che le loro transazioni non sono facilmente accessibili a terzi.

Validazione lato client nel protocollo RGB++

Uno degli aspetti più innovativi del protocollo RGB++ è l'adozione della validazione lato client, un concetto introdotto da Peter Todd. In questo modello, la responsabilità di verificare la validità delle transazioni viene delegata ai nodi client anziché ai nodi della rete principale. Questo approccio riduce significativamente il carico computazionale sulla blockchain principale, migliorando la scalabilità del sistema. Nel contesto di RGB++, la validazione lato client opera in modo che:

- ogni utente o nodo coinvolto in una transazione verifichi autonomamente la sua validità;
- i client raccolgono tutte le informazioni necessarie e utilizzano algoritmi crittografici per garantire il rispetto delle regole del protocollo, riducendo così la latenza e i costi delle transazioni.
- le transazioni, oltre a memorizzare i dati off-chain, vengono associate ad un set **UTXO** (**Unspent Transaction Output**) mediante l'utilizzo di sigilli monouso, che bloccano gli output delle transazioni Bitcoin come misura di sicurezza aggiuntiva. Questi sigilli impediscono a due parti diverse di fornire versioni contrastanti di dati che dovrebbero essere identici.

Alla base della validazione delle transazioni in Bitcoin e nei protocolli derivati come RGB++ c'è il concetto di UTXO. Ogni transazione Bitcoin è composta da input e output. Gli output di una transazione, quando non sono ancora stati spesi, diventano UTXO, e rappresentano il saldo disponibile per futuri utilizzi da parte del proprietario delle chiavi associate. La validazione di una nuova transazione richiede che i client controllino che gli UTXO utilizzati come input non

siano stati precedentemente spesi (double-spending) e che siano sufficienti a coprire l'importo della transazione [32].

Come già accennato, il protocollo RGB++ viene utilizzato per:

- ✓ la gestione dei biglietti di eventi sportivi, spettacoli teatrali, conferenze e altri tipi di eventi dove è necessario emettere e gestire biglietti digitali;
- ✓ la creazione di token di accesso che conferiscono diritti di membership a club esclusivi, abbonamenti digitali o accesso a contenuti speciali;
- ✓ la distribuzione di NFT (Non-Fungible Tokens), gli artisti possono creare e distribuire NFT utilizzando RGB++, beneficiando della sicurezza e decentralizzazione di Bitcoin, con l'aggiunta di privacy per i collezionisti.

Esaminiamo un esempio pratico di come il protocollo RGB++ possa essere impiegato per creare e trasferire un asset digitale con offuscamento delle transazioni, applicando questo concetto alla gestione di biglietti digitali per un evento.

Scenario: Vendita e trasferimento di biglietti digitali per un concerto.

1) Creazione dei biglietti digitali, ovvero gli Asset RGB.
    - Un'organizzazione di eventi decide di creare un asset RGB per vendere biglietti digitali per un concerto utilizzando il protocollo RGB++ su Bitcoin. Questo asset specifica il numero di biglietti disponibili (ad esempio, 1000 biglietti), il prezzo di ciascun biglietto, e altri dettagli come la data e il luogo dell'evento;
    - Questi biglietti digitali sono associati a un UTXO specifico sulla blockchain di Bitcoin ed il collegamento tra l'UTXO e i biglietti è registrato off-chain, nei metadati RGB.

2) Acquisto di un biglietto.
    - Un utente desidera acquistare un biglietto per il concerto recandosi sul sito web dell'organizzatore dell'evento e seleziona il biglietto;
    - Effettua il pagamento in Bitcoin tramite una transazione regolare, che include un output associato all'UTXO che rappresenta il biglietto digitale;
    - L'organizzatore invia i metadati del biglietto (off-chain) all'utente, attraverso un canale sicuro. Questi metadati includono informazioni come il numero di serie del biglietto e la prova crittografica (firma digitale).

3) Trasferimento del biglietto.
    - Supponiamo adesso che l'utente non possa più partecipare al concerto e voglia vendere il biglietto ad un suo amico;
    - Il trasferimento avviene creando una nuova transazione Bitcoin che punta all'UTXO del biglietto. Questa transazione è pubblica sulla blockchain, ma non rivela alcun dettaglio specifico sul biglietto o sul destinatario finale;
    - L'utente trasmette i metadati aggiornati del biglietto (off-chain) al suo amico, che ora diventa il nuovo proprietario del biglietto.

4) Verifica ed accesso al concerto.

- Il giorno del concerto, il nuovo proprietario si reca all'evento e presenta il biglietto digitale;
- Il sistema dell'organizzatore verifica il biglietto utilizzando i metadati off-chain. Questo include controllare la validità dell'UTXO associato e confermare che il biglietto non sia stato utilizzato in precedenza;
- Una volta verificato, il biglietto è considerato valido, e il proprietario può accedere al concerto.

5) Offuscamento e privacy.
   - Durante tutto il processo, le transazioni Bitcoin utilizzate per trasferire i biglietti appaiono come normali transazioni, senza rivelare nulla sul fatto che stiano effettivamente trasferendo biglietti digitali;
   - Gli osservatori della blockchain vedono solo transazioni Bitcoin standard e non hanno accesso ai metadati RGB che contengono le informazioni dettagliate sui biglietti.

L'utilizzo del protocollo RGB++ su Bitcoin consente di creare, trasferire e gestire asset digitali con un elevato grado di privacy e offuscamento. L'esempio precedente dimostra come questa tecnologia possa essere applicata nella vita quotidiana, assicurando la sicurezza delle transazioni e la tutela della privacy degli utenti.

Tra i principali vantaggi possiamo includere:
- ✓ **Elevata privacy** → le tecniche di offuscamento e la gestione off-chain delle transazioni garantiscono che solo le parti coinvolte possano conoscere i dettagli delle transazioni, proteggendo così gli utenti da potenziali minacce;
- ✓ **Scalabilità** → è notevolmente aumentata grazie alla riduzione del carico sulla blockchain di Bitcoin;
- ✓ **Sicurezza** → poiché RGB++ utilizza la blockchain di Bitcoin come base;
- ✓ **Interoperabilità** → la possibilità di combinare RGB++ con il Lightning Network consente transazioni rapide e a basso costo.

Tra i principali svantaggi possiamo includere:
- ✓ **Complessità tecnica** → l'implementazione e l'uso di RGB++ richiedono competenze tecniche avanzate, il che può rappresentare un ostacolo significativo per l'adozione su larga scala;
- ✓ **Limitata Adozione** → essendo una tecnologia relativamente nuova, RGB++ potrebbe non essere ancora ampiamente supportato da wallet, exchange e altre piattaforme

In conclusione possiamo affermare come RGB++ rappresenti una tecnologia potente e innovativa che sfrutta la sicurezza e la decentralizzazione della blockchain di Bitcoin per offrire un livello avanzato di privacy e controllo nella gestione degli asset digitali. Tuttavia, la sua adozione su larga scala è ancora ostacolata da una complessità tecnica significativa e da una limitata infrastruttura di supporto [33].

Capitolo 3

3 ZK-SNARK

Un protocollo zk-SNARK è un protocollo NIZK che combina la non interattività a meccanismi atti ad ottenere prestazioni elevate. Esso è acronimo di "Zero-Knowledge succinct non-interactive argument of knowledge". Nel dettaglio:

- **ZK –** *Zero Knowledge*: il segreto *w* non viaggia nel canale di comunicazione, nulla è rivelato se non la verità dell'affermazione stessa.
- **S –** *Succint* (sintentico): la dimensione della prova è piccola rispetto all'affermazione o al segreto, e la verifica è al più sub-lineare (sempre in riferimento al segreto).
- **N –** *Non-interactive*: la comunicazione avviene in un unico messaggio, è possibile scrivere e memorizzare una prova, senza la necessità di avere cicli di domande/risposte. Essa viene calcolata OFFchain, senza alcuna interazione tra il prover e il verifier, e solo successivamente pubblicata su una blockchain, e tutti possono verificarla.
- **AR –** *Argument* (o "computationally sound proof"): la proprietà di correttezza è computazionalmente sicura (computational soundness), ma dimostratori con potenza computazionale sufficiente possono convincere il verificatore di un'affermazione errata
- **K –** *of Knowledge-validity*: non è possibile per il prover costruire una prova valida senza avere la conoscenza del segreto w; di conseguenza, per ogni prover, esiste un Knowledge Extractor in grado di ricavare il segreto per l'affermazione**.**

A seguito di questa definizione è necessario analizzare più approfonditamente la proprietà di correttezza. Essa, come la proprietà di conoscenza-zero, può essere definita sotto tre diversi livelli di sicurezza, a seconda delle assunzioni effettuate per l'esistenza del Knowledge Extractor:

- Correttezza **perfetta** (*perfect soundness*): nessun Prover, con capacità computazionali illimitate, è in grado di ingannare il Verifier.
- Correttezza **statistica** (*statistic soundness*): la probabilità che un Prover con capacità computazionali illimitate riesca ad ingannare il Verifier è statisticamente trascurabile.
- Correttezza **algoritmica** o **computazionale** (*computational soundness*): un Prover con capacità computazionali limitate non sono in grado di ingannare il Verifier.

Il termine "Argument" indica quindi una prova la cui sicurezza dal punto di vista della correttezza dipende da assunzioni computazionali. Ciò si contrappone al termine "Proof", nel quale la correttezza è statistica [7].
Un protocollo zk-SNARK è descritto da tre algoritmi:

- **Gen**, è l'algoritmo di configurazione il quale genera una stringa crs usata successivamente nello step di verifica; inoltre genera alcune chiavi di verifica vrs. Solitamente questo algoritmo viene inizializzato da un "trusted party" ovvero da una parte terza fidata.
- **Prove**, è l'algoritmo di dimostrazione che prende in input il crs, l'affermazione u, il segreto corrispondente w, e restituisce come output la dimostrazione π.
- **Verify**, è l'algoritmo che prende in input la chiave di verifica vrs, l'affermazione u e

la dimostrazione π, e ritorna come output il valore 1 se accettata, oppure 0 se rifiutata.

È bene evidenziare che non tutti i protocolli SNARKs necessitano di una pre-elaborazione da parte di un trusted party. Protocolli che ricorrono ad una parte fidata:

- Marlin [34], un sistema di prova universale efficiente per il verificatore basato su AHP;
- Sonic [35], utile nelle applicazioni che utilizzano verifiche in batch;
- Libra [36] e Plonk [37], utilizzati mediante un argomento di permutazione.

Protocolli che non utilizzano una parte fidata (transparent SNARKs):

- Aurora [38], basato su Oracle Proofs interattivo;
- STARK [39], Fractal [40], usato nei modelli ad oracolo casuale;
- Spartan [41];
- Halo [42];
- Hyrax [43].

Gli SNARKS devono soddisfare alcune proprietà di sicurezza che proteggono simultaneamente sia il Prover dalla divulgazione del segreto, che il Verifier da una prova contraffatta. Definiamo di seguito le nozioni di sicurezza necessarie per definire uno zk-SNARK.

Definizione zk-SNARK

In questa sezione vengono fornite le definizioni formali per la nozione di SNARK. Sia $R := R_\lambda$ una relazione binaria per un linguaggio L_R NP definita nel capitolo 1.4. Definiamo la coppia $(u, w) \in R$, corrispondenti rispettivamente all'affermazione e al segreto.
Definiamo:

- $G(1^\lambda, R) \rightarrow (crs, vrs, td)$, è l'algoritmo di configurazione il quale riceve in ingresso dei parametri di sicurezza $\lambda \in \mathbb{N}$ e la relazione R, e restituisce come output una stringa crs, delle chiavi di verifica vrs e una chiave di decrittazione td, detta anche *trapdoor*, associata alla *crs* che consente l'estrazione del segreto da una prova valida;
- $P(crs, u, w) \rightarrow \pi$, è l'algoritmo del Prover che prende in input la stringa crs, l'affermazione u e il segreto w, restituendo in output diversi argomenti π;
- $Ver(crs, u, \pi) \rightarrow b$, è l'algoritmo del Verifier che prende in input l'affermazione u insieme ad un argomento π e la stringa crs. Restituisce un output di $b = 1$ se la dimostrazione è accettata, $b = 0$ altrimenti;
- $Sim(crs, td, u) \rightarrow \pi$, il Simulatore prende in input la stringa crs, la *trapdoor* td e l'affermazione u insieme alla dimostrazione π, restituendo come output un argomento π.

Un argomento non interattivo $\Pi = (G, P, Ver, Sim)$ per R è uno zk-*SNARK* se soddisfa:

- ✓ **Completezza**. Data un'affermazione vera per la relazione R, un dimostratore onesto P con un segreto valido è in grado di convincere un verificatore Ver. Più formalmente, per ogni $\lambda \in \mathbb{N}$ e per ogni $(u, w) \in R$:

$$P_r\left[Ver(crs, u, \pi) = 1 \mid (crs, td) \leftarrow G\left(1^\lambda, R\right), \pi \leftarrow P(crs, u, w)\right] = 1$$

- ✓ **Correttezza della conoscenza**. La nozione di correttezza della conoscenza implica che esiste un estrattore che può calcolare un segreto ogni volta che l'avversario produce un argomento valido. L'estrattore ha il pieno accesso alle informazioni, comprese eventuali monete casuali. Più formalmente, per ogni avversario PPT $\mathcal{A}$ esista un estrattore PPT $\mathcal{E}_{\mathcal{A}}$ tale che:

$$P_r\left[Ver(crs,u,\pi)=1 \wedge (u,w)\notin R \mid (crs,td)\leftarrow G\left(1^{\lambda},R\right),\left((u,\pi);w\right)\leftarrow \mathcal{A}\parallel \mathcal{E}_{\mathcal{A}}(crs)\right]=negl$$

- ✓ **Succinctness (brevità)**. Un argomento non interattivo in cui il verificatore viene eseguito in tempo polinomiale in $\lambda+|u|$ e la dimensione della prova è polinomiale in λ è chiamato Snark di pre-elaborazione.
- ✓ **Conoscenza zero statistica**. Un argomento è a conoscenza zero se non divulga nessuna informazione oltre alla verità dell'affermazione. Più formalmente, se per ogni $\lambda\in\mathbb{N}$, per ogni $(u,w)\in R$ e per ogni avversario PPT $\mathcal{A}$, le seguenti due distribuzioni sono statisticamente vicine:

$$D_0=\left\{\pi_0\leftarrow P(crs,u,w):(crs,td)\leftarrow G(1^{\lambda},R)\right\}$$
$$D_1=\left\{\pi_1\leftarrow Sim(crs,td,u):(crs,td)\leftarrow G(1^{\lambda},R)\right\}$$

3.1 Introduzione ai Quadratic Arithmetic Programs

Uno dei principali approcci per le costruzioni e le implementazioni di SNARK hanno come punto di partenza centrale il framework basato sui programmi quadratici introdotto da Gennaro et al [44].
Questo framework comune consente di costruire SNARK per programmi istanziati come circuiti booleani o aritmetici. Questo approccio ha portato a rapidi progressi verso calcoli pratici verificabili. Ad esempio, utilizzando programmi span per circuiti aritmetici (**QSP**), il protocollo Pinocchio [45] fornisce la prova che il calcolo remoto verificato può essere più veloce del calcolo locale. Allo stesso tempo, la loro costruzione è a conoscenza zero, consentendo al server di mantenere privati i valori intermedi e aggiuntivi utilizzati nel calcolo. Le versioni ottimizzate di SNARK basate sull'approccio QAP vengono utilizzate in varie applicazioni pratiche, comprese le criptovalute come Zcash [46], per garantire l'anonimato tramite la proprietà ZK.

3.2 Circuiti aritmetici

I **circuiti aritmetici** costituiscono la base computazionale per molti schemi di **prova a conoscenza zero** (Zero-Knowledge Proofs, ZKP), tra cui gli zk-SNARKs. Nell'ambito delle zk-SNARKs, la computazione che si vuole verificare viene rappresentata come un circuito aritmetico. Questo circuito definisce l'insieme delle operazioni dell'aritmetica di base ($somma\ e\ moltiplicazione$), per risolvere il problema computazionale. Non sono consentiti loop, operatori di confronto ($<,>,\leq,\geq$) e nemmeno il modulo (%) nel dettaglio:

- i **loop** (o cicli iterativi) non sono consentiti perché i circuiti aritmetici, così come i modelli che ne derivano (R1CS e QAP), devono avere una dimensione fissa e predefinita. Ogni passo della computazione deve essere rappresentato da un numero limitato di operazioni aritmetiche. I loop, per loro natura, possono eseguire un numero

variabile di iterazioni, che dipendono dai dati di input o da condizioni esterne;
- gli **operatori di confronto** non sono direttamente implementabili nei circuiti aritmetici per due ragioni principali. La prima, le operazioni vengono eseguite su numeri che appartengono a un campo finito $\mathbb{F}_p$, che non supporta confronti tra numeri nello stesso modo dei numeri reali. I confronti richiedono operazioni logiche binarie, come le comparazioni bit a bit, che non sono facilmente rappresentabili attraverso operazioni aritmetiche. La seconda motivazione riguarda l'assenza di operazioni condizionali in quanto i circuiti aritmetici non possono gestire in modo naturale operazioni condizionali basate sui confronti, come le istruzioni **if-then-else**;
- l'**operatore modulo** non è accettato nei circuiti aritmetici perché richiede la divisione e il calcolo del resto, operazioni che non sono direttamente rappresentabili in termini di addizioni e moltiplicazioni nei campi finiti.

Successivamente, il circuito aritmetico viene trasformato in un sistema di vincoli lineari chiamato **Rank-1 Constraint System (R1CS)**, e da qui in un **Quadratic Arithmetic Program (QAP)**, che è alla base della verifica efficiente delle zk-SNARKs.

3.3 Rank-1 Constraint Systems (R1CS)

I **Rank-1 Constraint Systems (R1CS)** sono una rappresentazione fondamentale per esprimere e verificare le computazioni negli schemi di **zk-SNARKs**. Essi forniscono un meccanismo per modellare le computazioni in forma di vincoli che possono essere verificati in modo efficiente, mantenendo però sicurezza e riservatezza. Nello specifico, gli R1CS permettono di trasformare una computazione aritmetica in un insieme di vincoli lineari che descrivono relazioni tra variabili, garantendo che un verificatore possa confermare la correttezza della computazione senza conoscere gli input effettivi (proprietà di zero-knowledge).
Il cuore della struttura di un R1CS sta nel rappresentare ogni passaggio della computazione come un **vincolo lineare**. In pratica, questo significa che per ogni operazione logico-aritmetica espressa nel circuito originale, corrisponde un vincolo che coinvolge una o più variabili di input e variabili ausiliarie. L'obiettivo è che ogni vincolo sia soddisfatto affinché l'intera computazione possa essere considerata valida. La particolarità di questa rappresentazione è che permette di convertire qualsiasi computazione aritmetica, anche molto complessa, in una forma verificabile da un'entità terza, attraverso il calcolo di vincoli. In dettaglio, un **R1CS** è costituito da un insieme di **triplette di vettori** (a, b, c) e la soluzione del sistema è un **vettore** s. Ogni tripla rappresenta un vincolo che deve essere soddisfatto dalla computazione, e il vettore s contiene sia gli **input pubblici** che i valori **privati** (o ausiliari) utilizzati nella computazione. La validità del sistema è verificata attraverso l'equazione chiave:

$$s.a \cdot s.b - s.c = 0$$

Qui, il simbolo "." rappresenta il prodotto scalare tra il vettore delle soluzioni s e i vettori a, b e c. In altre parole, per verificare la correttezza della computazione, si moltiplicano gli elementi di a e s, e quelli di b e s, sommando i risultati nelle rispettive posizioni. Dopodiché, si calcola il prodotto di questi due risultati parziali e si confronta con il prodotto scalare tra s e c. Se l'uguaglianza è rispettata, significa che il vincolo è soddisfatto e quindi la computazione è corretta. Questa struttura permette di verificare qualsiasi computazione aritmetica con una

complessità computazionale relativamente bassa, poiché tutte le operazioni richieste si riducono a prodotti scalari e moltiplicazioni tra vettori. Inoltre, il sistema è **rank-1** (grado 1), il che significa che ogni vincolo è una relazione bilineare tra le variabili di input, e questo lo rende molto più semplice da calcolare rispetto ad altre rappresentazioni più complesse. In sintesi, gli **R1CS** trasformano la logica dei circuiti aritmetici in vincoli lineari che garantiscono una verifica sicura e privata della computazione, riducendo il problema a un insieme di equazioni bilineari.
Questo approccio è centrale nella costruzione di zk-SNARKs, dove la combinazione di **succinctness**, **non-interactivity** e **zero-knowledge** consente di ottenere schemi crittografici potenti e applicabili a una vasta gamma di scenari, come la privacy nelle transazioni blockchain o l'autenticazione senza divulgazione di informazioni sensibili.

3.4 Definizione di QAP

Un Programma Aritmetico Quadratico **(QAP)** è una rappresentazione matematica avanzata che trova un utilizzo fondamentale nell'ambito dei **zk-SNARKs**, essendo uno strumento chiave per verificare la correttezza di computazioni in modo efficiente e sicuro. La funzione principale del QAP è quella di tradurre una computazione espressa come sistema di vincoli, tipicamente formulato tramite un R1CS, in un formato polinomiale che facilita la costruzione della prova crittografica. Grazie alla rappresentazione polinomiale dei QAP, è possibile utilizzare efficienti tecniche di verifica basate sull'algebra dei campi finiti. Nel processo di zk-SNARKs, il QAP è fondamentale perché fornisce il ponte tra i vincoli lineari definiti nell'R1CS e i metodi di verifica crittografica. Ogni vincolo del sistema è espresso come un polinomio quadratico, e la validità della computazione può essere verificata controllando se un certo polinomio "target" può essere diviso da altri polinomi costruttivi.
Il QAP è costruito a partire da tre insiemi di $m + 1$ polinomi $V = \{v_i(x)\}$, $W = \{w_i(x)\}$, $Y = \{y_i(x)\}$, con $i \in \{0, 1 \dots m\}$ e un polinomio target $t(x)$. Questi elementi combinati consentono di rappresentare i vincoli aritmetici della computazione in forma polinomiale.
Supponiamo che la funzione aritmetica F accetti in input n elementi nel campo finito $\mathbb{F}$[9] e produca n' elementi in output, per un totale di $N = n + n'$ elementi tra input e output. Un'assegnazione $(c_1, \dots, c_N) \in \mathbb{F}^N$ di valori agli input e output di F è considerata **valida**, se e solo se esistono coefficienti ausiliari $(c_{N+1}, \dots, c_m) \in \mathbb{F}$ tali che il polinomio $p(x)$, costruito come combinazione pesata dei polinomi $v_i(x), w_i(x), y_i(x)$, sia divisibile per il polinomio target $t(x)$. Formalmente:

$$p(x) := \left(v_0(x) + \sum_{i=1}^{m} c_i v_i(x)\right) \cdot \left(w_0(x) + \sum_{i=1}^{m} c_i w_i(x)\right) - \left(y_0(x) + \sum_{i=1}^{m} c_i y_i(x)\right)$$

***Equazione (1)** [47].*

e la condizione di divisibilità richiede che:

$$\frac{t(x)}{p(x)}$$

[9] In matematica, un campo è una struttura algebrica composta da un insieme non vuoto e da due operazioni binarie interne, chiamate somma e prodotto. Nell'esempio specifico, il campo $\mathbb{F}$ è un campo finito e grazie alle sue proprietà, ricopre un ruolo importante in diversi algoritmi crittografici.

Il QAP è essenziale per garantire la sicurezza e la privacy nelle zk-SNARKs, poiché fornisce le basi matematiche per costruire prove a conoscenza zero. Le sue principali caratteristiche che contribuiscono a questi obiettivi sono:

1. *Privacy degli Input* → la rappresentazione polinomiale del QAP consente di nascondere gli input della computazione al verificatore. Solo la validità dei vincoli viene verificata, senza che gli input privati vengano rivelati, garantendo così la proprietà di **zero-knowledge**.
2. *Succintezza delle Prove* → il QAP permette di ridurre una computazione complessa a una prova estremamente compatta, che può essere verificata rapidamente. Questo rende gli zk-SNARKs scalabili e applicabili anche in contesti con risorse computazionali limitate.
3. *Inviolabilità dei Vincoli* → la struttura del QAP garantisce che ogni violazione dei vincoli risulti immediatamente evidente nella verifica del polinomio. Questo meccanismo protegge dalla falsificazione delle computazioni.
4. *Efficienza Computazionale* → il test di divisibilità polinomiale, elemento chiave della verifica basata su QAP, è computazionalmente meno oneroso rispetto alla verifica diretta dei vincoli lineari. Questo rende possibile verificare prove anche in ambienti con risorse computazionali limitate.

Grazie alle sue proprietà, il QAP è utilizzato in vari ambiti che richiedono privacy e sicurezza computazionale quali:

- Blockchain e transazioni private, ad esempio in **Zcash** per verificare transazioni senza rivelare dettagli sulle parti coinvolte o sugli importi trasferiti;
- Computazione delegata, in quanto è possibile delegare **calcoli a terze parti** mantenendo la possibilità di verificarne la correttezza senza eseguire l'intera computazione.

3.4.1 Esempio di Vitalik Buterin

L'esempio trattato da Vitalik Buterin nel suo blog [48], vuol dimostrare di conoscere la soluzione di un'equazione cubica: $x^3 + x + 5 = 35$. Ottenere il risultato potrebbe sembrare semplice, ma non tanto da rendere il QAP risultante così scontato e banale.
Premessa, riscriviamo l'equazione proposta sotto forma di codice:

- def $equazione(x)$:
  $y = x^3$
  $return\ y + x + 5$

Il primo step da eseguire è l'operazione di "**flattening**", tale procedura di "appiattimento" consiste nel convertire il codice originale, che può contenere affermazioni ed espressioni arbitrariamente complesse, in una sequenza di affermazioni che possono presentare due forme:

1) $x = y$, dove y può essere una variabile o un numero;
2) $x = y\ (op)\ z$, dove op è un'operazione aritmetica, mentre y e z possono essere

variabili, numeri o sotto espressioni.
Applicando il *flattening* alla nostra equazione, otteniamo:

$$var_1 = x * x$$
$$y = var_1 * x$$
$$var_2 = y + x$$
$$\sim out = var_2 + 5$$

Si può notare come l'output restituito sia perfettamente equivalente a quello del codice originale, ma ora è espresso in termini di operazioni elementari.
Nel secondo step il codice viene convertito in un **sistema di vincoli di rango 1** (**R1CS**). A partire dalle operazioni ottenute tramite flattening, è possibile costruire un sistema R1CS (Rank-1 Constraint System) che ne codifica la correttezza computazionale attraverso un insieme di vincoli bilineari. Ogni operazione elementare individuata dal flattening viene tradotta in un vincolo bilineare che ne garantisce la correttezza computazionale. Per chiarire come avviene in concreto la costruzione dei coefficienti nei vettori A, B, C consideriamo le principali casistiche di assegnazione generate dal flattening:

Operazione di moltiplicazione → un'istruzione del tipo z = x · y viene convertita in un vincolo R1CS che impone $x \cdot y - z = 0$. Per ottenere ciò, scegliamo A e B con coefficiente 1 nelle posizioni corrispondenti alle variabili x e y rispettivamente, e poniamo un coefficiente 1 per z nel vettore C. In questo modo il prodotto scalare $A \cdot w$ restituisce il valore di x, $B \cdot w$ quello di y, e $C \cdot w$ quello di z, traducendo il vincolo nella forma $(x) \cdot (y) = z$.

Operazione di addizione (o sottrazione) → un'istruzione $z = x + y$ si traduce in un vincolo lineare $x + y - z = 0$, che possiamo rappresentare come $(x + y) \cdot 1 - z = 0$. In termini di vettori, A avrà coefficiente 1 sia sulla variabile x sia sulla variabile y (così che $A \cdot w = x + y$), B avrà coefficiente 1 sulla posizione della variabile fittizia $\sim one$ (in modo che $B \cdot w = 1$), e C avrà coefficiente 1 sulla variabile z (ottenendo $C \cdot w = z$). Il prodotto scalare doppio $(A \cdot w) \cdot (B \cdot w) = z$ risulta quindi $x + y$, che uguagliato a $C \cdot w$ impone $z = x + y$.

Assegnazione di costante → se l'istruzione produce una costante, ad esempio $z = 5$, il vincolo corrispondente deve imporre $5 - z = 0$. Per ottenere ciò in forma R1CS utilizziamo la variabile $\sim one$ impostando $A \cdot w = 5$ (cioè coefficiente 5 sulla variabile costante 1 nel vettore A), $B \cdot w = 1$ (come sopra) e $C \cdot w = z$. In questo modo $(A \cdot w) \cdot (B \cdot w) = 5$ e il vincolo risultante forza z ad assumere il valore 5. Come si nota, in tutti i casi i vettori A, B e C risultano sparsi, cioè nulli in quasi tutte le posizioni tranne quelle delle variabili effettivamente coinvolte nell'operazione in questione.

In figura viene mostrato l'esempio di R1CS:

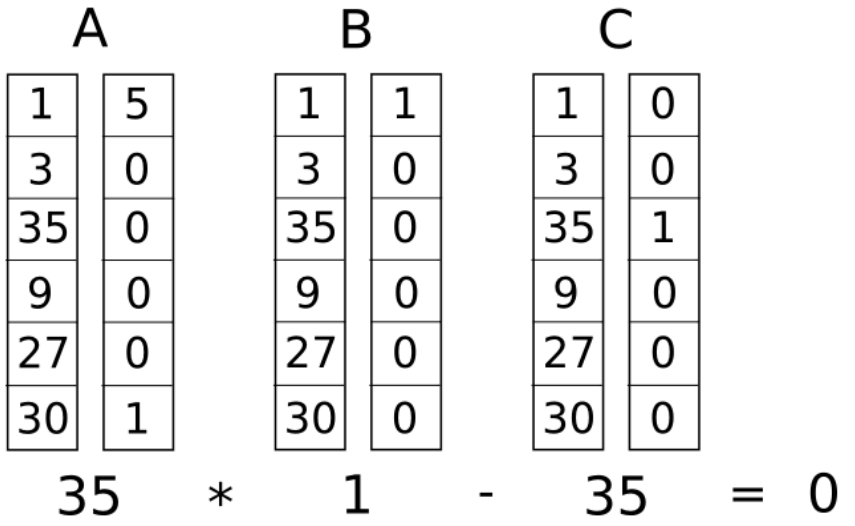

Figura 10 Esempio di R1CS. *[48]*

In formula:

$$[(1 \times 5) + (30 \times 1)] \times [1] - [35 \times 1] = 0$$

Equazione (2) [48]

Non vi sarà un solo vincolo, ma uno per ogni porta logica. Esiste infatti una modalità per convertire una porta logica in una tripla a seconda dell'operazione $(+, -, *, \div)$. La lunghezza di ogni vettore è uguale al numero totale di variabili nel sistema, come primo indice ci sarà una variabile "fittizia" che chiameremo $\sim one$, la quale rappresenta il numero 1, le variabili di input, un'ulteriore variabile fittizia $\sim out$ che rappresenta l'output, e tutte le variabili intermedie variabili (var_1, var_2);
Di seguito la mappatura delle variabili che verranno utilizzate:

- $\sim one$
- x
- $\sim out$
- sym_1
- y
- sym_2

Il vettore risultante sarà costituito dall'assegnazione di tutte le variabili, nell'ordine in cui le andremo ad attenzionare.
Vediamo dunque, per la prima porta $(var_1 = x * x)$, la seguente tripla (a, b, c):

$$a = [0, 1, 0, 0, 0, 0]$$
$$b = [0, 1, 0, 0, 0, 0]$$
$$c = [0, 0, 0, 1, 0, 0]$$

Supponiamo che il vettore risultante contenga nella seconda e quarta posizione rispettivamente i valori 3 e 9, indipendentemente dagli altri valori presenti nel vettore, il prodotto scalare $3 \cdot$

$3 = 9$ soddisferà la soluzione. Considerando il caso in cui nella seconda posizione ci sia il valore -3 e nella quarta il valore 9, la soluzione sarà comunque accettata; così come considerando i valori 7 nella seconda posizione e 49 nella quarta, il prodotto scalare soddisferà la soluzione. Lo scopo di questo primo controllo è quello di verificare la congruenza degli ingressi e delle uscite della sola prima porta.

Per la seconda porta ($y = var_1 * x$) calcoleremo il prodotto scalare in maniera simile a quanto visto precedentemente, con la differenza che qui viene effettuato un check affinché $var_1 * x = y$

$$a = [0, 0, 0, 1, 0, 0]$$
$$b = [0, 1, 0, 0, 0, 0]$$
$$c = [0, 0, 0, 0, 1, 0]$$

Nella quarta e seconda posizione sono presenti i valori 9 e 3, il loro prodotto scalare è uguale al valore presente nella quinta posizione ovvero 27.

Vediamo adesso la terza porta ($var_2 = y + x$):

$$a = [0, 1, 0, 0, 1, 0]$$
$$b = [1, 0, 0, 0, 0, 0]$$
$$c = [0, 0, 0, 0, 0, 1]$$

Lo schema è leggermente differente, si procede sommando il prodotto del primo elemento del vettore risultante per il secondo, e del primo elemento per il quinto; è necessario verificare che la loro somma sia uguale al valore del sesto elemento. Vediamo in formule: $(1 \times 3) + (1 \times 27) = 30$
Si può notare come il primo elemento è sempre 1, di fatto viene applicato un semplice controllo addizionale affinché l'output sia uguale alla somma dei due input.
Infine analizziamo la quarta porta ($\sim out = var_2 + 5$):

$$a = [5, 0, 0, 0, 0, 1]$$
$$b = [1, 0, 0, 0, 0, 0]$$
$$c = [0, 0, 1, 0, 0, 0]$$

Qui, viene eseguito l'ultimo controllo, $\sim out = var_2 + 5$. Il prodotto scalare viene calcolato prendendo il sesto elemento del vettore risultante, sommando cinque volte il primo elemento, il quale vale 1 (come detto in precedenza), e confrontandolo con il terzo elemento, dove viene memorizzata la variabile d'uscita.
Ed ecco il sistema R1CS con quattro vincoli. Il testimone risulta essere l'assegnazione di tutte le variabili, inclusi input, output e variabili interne:

$$[1, 3, 35, 9, 27, 30]$$

Di seguito il sistema R1CS completo:

$$A$$
$$[0, 1, 0, 0, 0, 0]$$
$$[0, 0, 0, 1, 0, 0]$$
$$[0, 1, 0, 0, 1, 0]$$
$$[5, 0, 0, 0, 0, 1]$$

$$B$$
$$[0, 1, 0, 0, 0, 0]$$
$$[0, 1, 0, 0, 0, 0]$$
$$[1, 0, 0, 0, 0, 0]$$
$$[1, 0, 0, 0, 0, 0]$$

$$C$$
$$[0, 0, 0, 1, 0, 0]$$
$$[0, 0, 0, 0, 1, 0]$$
$$[0, 0, 0, 0, 0, 1]$$
$$[0, 0, 1, 0, 0, 0]$$

Il passo successivo sarà quello di convertire l'R1CS nel formato QAP, il quale implementa esattamente la stessa logica utilizzando i polinomi invece dei prodotti scalari. Si passa dunque da quattro gruppi di tre vettori di lunghezza sei, appena visti, a sei gruppi di tre polinomi di terzo grado. I vincoli sono rappresentati dal risultato dei polinomi per ogni coordinata x calcolata. Ad esempio valutando i polinomi in $x = 1$, si otterrà il primo insieme di vettori, in $x = 2$ il secondo e via dicendo.
Per realizzare un esempio pratico, consideriamo un polinomio che passi attraverso i punti $(1,3)$, $(2,2)$ e $(3,4)$, utilizzando l'interpolazione di Lagrange[10].

In formula:

$$L(x) = \sum_{i=0}^{n} y_i \prod_{j \neq i, j=0}^{n} \frac{(x - x_j)}{(x_i - x_j)}$$

Equazione (3) [48]

dove n è il numero di punti meno uno (nel nostro caso 2, avendo 3 punti), x_i e y_i sono le coordinate dei punti attraverso i quali il polinomio deve passare, e il prodotto è calcolato su tutte le j.
Creiamo il primo polinomio il quale presenta un picco in $x = 1$ e sia zero negli altri punti $x = 2$ e $x = 3$.
Il polinomio sarà quindi: $(x - 2)(x - 3)$.

[10] Il metodo di interpolazione di Lagrange è una tecnica matematica adoperata per determinare un polinomio che attraversi precisamente un insieme specificato di punti. Si basa sulla costruzione di un polinomio di grado minore possibile che raggiunga valori predeterminati in punti selezionati. Per ciascuna ascissa x, si formula un polinomio che assume il valore di ordinata y desiderato in quella specifica ascissa e zero nelle altre ascisse di interesse. Il polinomio risultante si ottiene sommando insieme tutti questi polinomi.

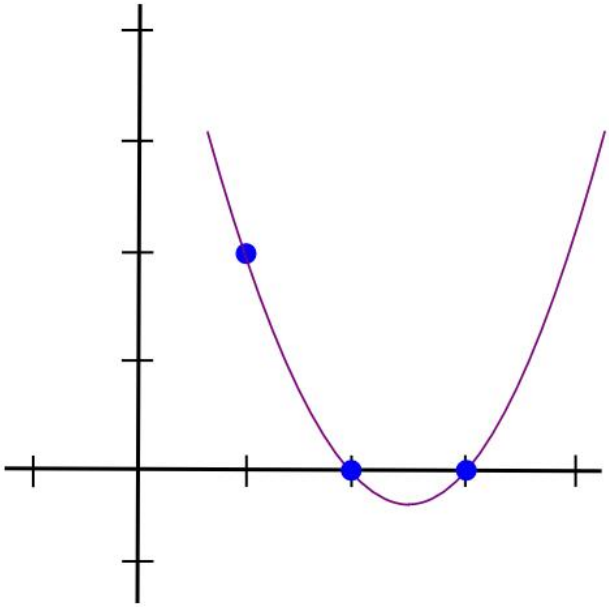

Figura 11 Polinomio con picco in x=1. [48]

Il passo successivo sarà quello di ridimensionarlo affinché l'altezza in $x = 1$ sia corretta. Per il punto $(1,3)$ avremo dunque che:

$$\frac{(x-2)(x-3)\times 3}{(1-2)(1-3)} = 1{,}5x^2 - 7{,}5x + 9$$

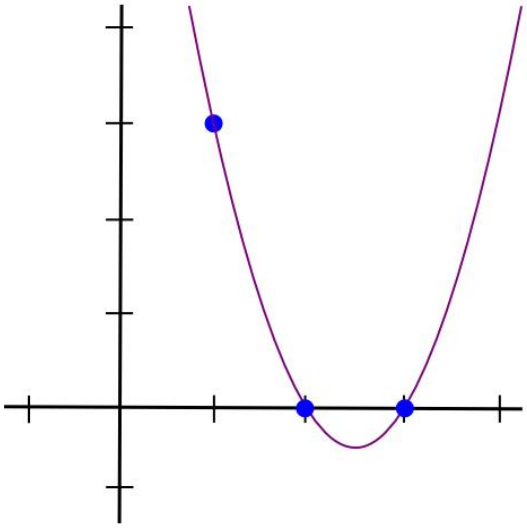

Figura 12 Polinomio ridimensionato [48]

Procederemo in egual modo con gli altri due punti, ottenendo due nuovi polinomi simili ma con picchi rispettivamente in $x = 2$ e $x = 3$.

Per il punto (2,2) avremo che:

$$\frac{(x-1)(x-3)\times 2}{(2-1)(2-3)} = -2x^2 + 8x - 6$$

Per il punto (3,4) avremo che:

$$\frac{(x-1)(x-2)\times 4}{(3-1)(3-2)} = 2x^2 - 6x + 4$$

Infine, sommando i tre polinomi risultanti otterremo il polinomio desiderato che passa per i punti specificati: $1{,}5x^2 - 5{,}5x + 7$

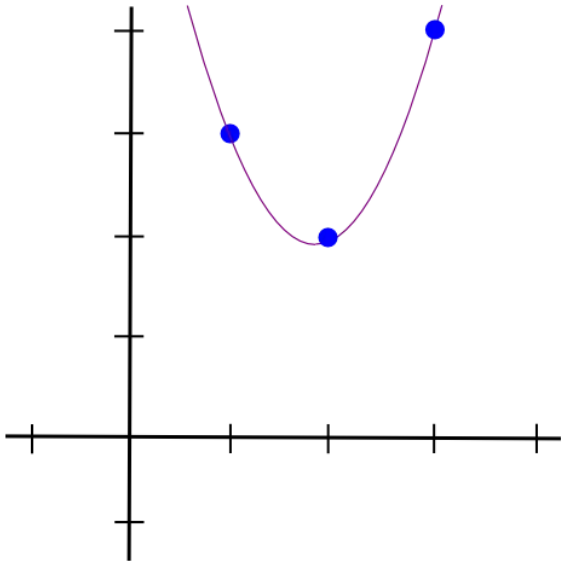

***Figura 13 Polinomio risultante** [48]*

L'algoritmo descritto inizialmente richiede un tempo di esecuzione pari a $O(n^3)$, dove n rappresenta il numero di punti. Ogni punto richiede $O(n^2)$ operazioni per moltiplicare i polinomi. Attuando un'ottimizzazione tramite la Trasformata veloce di Fourier[11] è possibile ridurre il tempo di esecuzione a $O(n^2)$, il quale è fondamentale lavorando con funzioni utilizzate nei zk-SNARKS in cui, nelle implementazioni pratiche, sono presenti migliaia unità logiche.

Utilizziamo adesso l'interpolazione di Lagrange per trasformare il nostro R1CS. Da ogni vettore a, b e c prenderemo il primo valore. Questi valori rappresentano i coefficienti dei vincoli nella forma $a \cdot b = c$.
Per ogni posizione i, si utilizza l'interpolazione di Lagrange per creare un polinomio che attraversi tutti i primi valori dei vettori. Quando il polinomio viene valutato in i, restituisce il primo valore del $i - esimo$ vettore a, b o c. Questo processo viene ripetuto per tutti i valori

[11] La Trasformata Veloce di Fourier (FFT, acronimo di Fast Fourier Transform) è un algoritmo efficiente per calcolare la Trasformata di Fourier discreta (DFT) di una sequenza di dati campionati nel dominio del tempo.

del vettore al fine di generare un insieme di polinomi "interpolanti". Tali polinomi nel contesto di zk-SNARKS sono utilizzati per consentire a un *verifier* di verificare la correttezza delle asserzioni senza dover esporre i dati originali (i vettori a, b e c), mantenendo la privacy dell'input. Di seguito vediamo come vengono calcolati i polinomi risultanti utilizzando i valori dell'R1CS tramite l'interpolazione polinomiale di Lagrange.

- Per la variabile $\sim one$ (indice 0), i valori di A ai quattro vincoli sono $(0,0,0.5)$. Si interpola dunque fra i punti $(1,0)$, $(2,0)$, $(3,0)$, $(4,5)$, dove soltanto nell'ultimo punto si ha che $y = 5$.

$$A(x) = 5\,\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} = \frac{5}{6}\,(x^2-3x+2)(x-3) = \mathbf{0.833x^3 - 5x^2 + 9.166x - 5}$$

- Per la variabile x (indice 1), i valori sono $(1,0,1,0)$. Si interpola tra i punti (1,1), (2,0), (3,1), (4,0), di conseguenza il polinomio finale sarà la somma dei due termini.

$$A_1(x) = 1\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} = -\frac{1}{6}(x^2-5x+6)(x-4) = \mathbf{-0.166x^3 + 1.5x^2 - 4.333x + 4}$$

$$A_1(x) = 1\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} = -\frac{1}{2}(x^2-3x+2)(x-4) = \mathbf{-0.5x^3 + 3.5x^2 - 7x + 4}$$

Sommando i due termini il polinomio sarà uguale a: $= \mathbf{-0.666x^3 + 5.0x^2 - 11.333x + 8}$

- Per la variabile $\sim out$ (indice 2), tutti i coefficienti sono nulli, di conseguenza si avrà il polinomio nullo. $A_2(x) = 0$

- Per la variabile sym_1 (indice 4), i valori sono $(0,1,0,0)$. Si interpola nei punti (1,0), (2,1), (3,0), (4,0).

$$A_3(x) = 1\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} = \frac{1}{2}(x^2-4x+3)(x-4) = \mathbf{0.5x^3 - 4x^2 + 9.5x - 6}$$

- Per la variabile y (indice 5), i valori sono $(0,0,1,0)$. Si interpola nei punti (1,0), (2,0), (3,1), (4,0).

$$A_4(x) = 1\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} = -\frac{1}{2}(x^2-3x+2)(x-4) = \mathbf{-0.5x^3 + 3.5x^2 - 7x + 4}$$

- Per la variabile sym_2 (indice 6), i valori sono $(0,0,0,1)$. Si interpola nei punti (1,0), (2,0), (3,0), (4,1).

$$A_5(x) = 1\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} = \frac{1}{6}(x^2-3x+2)(x-3) = \mathbf{0.166x^3 - x^2 + 1.833x - 1}$$

Analogamente, applicando l'interpolazione di Lagrange, si otterranno anche i valori per i polinomi B e C.

$$polinomio\ A$$
$$[-5.0, 9.166, -5.0, 0.833]$$
$$[8.0, -11.333, 5.0, -0.666]$$
$$[0.0, 0.0, 0.0, 0.0]$$
$$[-6.0, 9.5, -4.0, 0.5]$$
$$[4.0, -7.0, 3.5, -0.5]$$
$$[-1.0, 1.833, -1.0, 0.166]$$

$$polinomo\ B$$
$$[3.0, -5.166, 2.5, -0.333]$$
$$[-2.0, 5.166, -2.5, 0.333]$$
$$[0.0, 0.0, 0.0, 0.0]$$
$$[0.0, 0.0, 0.0, 0.0]$$
$$[0.0, 0.0, 0.0, 0.0]$$
$$[0.0, 0.0, 0.0, 0.0]$$

$$polinomio\ C$$
$$[0.0, 0.0, 0.0, 0.0]$$
$$[0.0, 0.0, 0.0, 0.0]$$
$$[-1.0, 1.833, -1.0, 0.166]$$
$$[4.0, -4.333, 1.5, -0.166]$$
$$[-6.0, 9.5, -4.0, 0.5]$$
$$[4.0, -7.0, 3.5, -0.5]$$

I coefficienti sono salvati in ordine ascendente, il primo polinomio si presenta come: $0.833\ x^3 - 5\ x^2 + 9.166\ x - 5$. Questo insieme di polinomi, con un nuovo polinomio Z che verrà introdotto successivamente, costituisce i parametri per questa particolare istanza di QAP. Valutiamo ora i polinomi in $x = 1$, sommando tutti i coefficienti. Otterremo esattamente lo stesso insieme di tre vettori della prima porta logica che abbiamo creato precedentemente.

A	B	C
0	0	0
1	1	0
0	0	0
0	0	1
0	0	0
0	0	0

Verifica del QAP

La sezione sulla verifica del QAP riguarda il controllo delle condizioni necessarie per dimostrare la correttezza di un programma aritmetico quadratico. Ora è possibile verificare tutti i vincoli contemporaneamente eseguendo il controllo del prodotto scalare sui polinomi, invece di controllare i vincoli individualmente nell'R1CS.

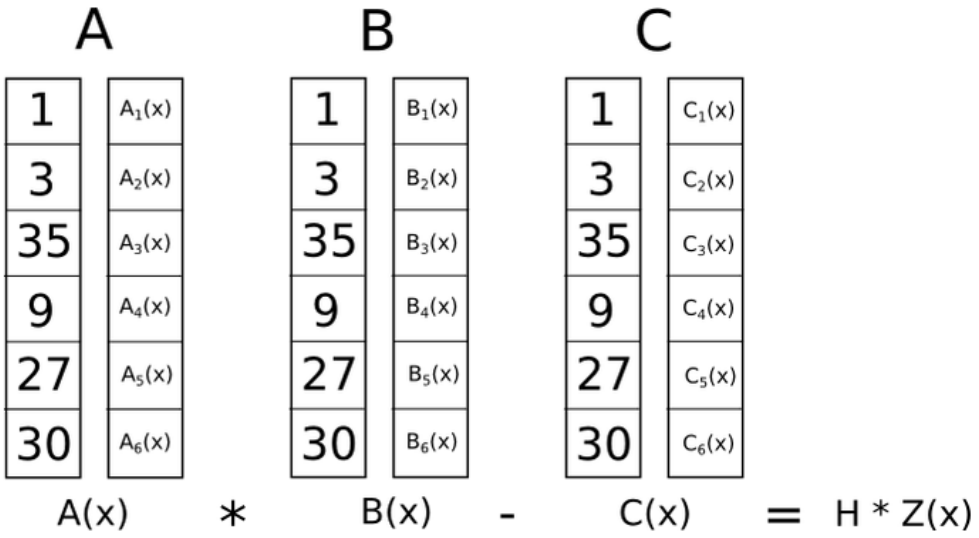

***Figura 14 Esempio di R1CS calcolato con vincoli simultaneamente** [48]*

Poiché il controllo del prodotto scalare in questo caso consiste in una serie di addizioni e moltiplicazioni di polinomi, il risultato sarà anch'esso un polinomio. Se, valutando questo polinomio risultante in ogni coordinata x usata per rappresentare una porta logica, otteniamo zero, allora tutti i controlli sono considerati validi. Al contrario, se almeno in una coordinata otteniamo un valore diverso da zero, significa che i valori in ingresso e in uscita da quella porta logica sono incoerenti. Si noti che il polinomio risultante non deve necessariamente essere zero ovunque; può assumere qualsiasi valore nei punti che non corrispondono a nessuna porta logica, purché il risultato sia zero in tutti i punti che corrispondono a una porta logica.
Per finire, verificheremo la correttezza del QAP sfruttando un principio fondamentale della teoria dei polinomi, il lemma di *Schwartz-Zippel*; invece di valutare il polinomio $t = A.s \cdot B.s - C.s$ in ogni punto corrispondente a una porta, divideremo t per un altro polinomio Z assicurandoci che la divisione non lasci alcun resto. Il lemma di Schwartz-Zippel [49] afferma che, dato un polinomio non nullo f in n variabili su un campo $\mathbb{F}$, di grado totale d, e un insieme finito $S \subset \mathbb{F}$, allora la probabilità che f si annulli in un punto scelto uniformemente a caso da S^n è al più $\frac{d}{|S|}$.
Applicato al nostro caso, ciò implica che se $t \not\equiv 0$, allora la probabilità che $t(r) = 0$ per un $r \in \mathbb{F}$ scelto casualmente è trascurabile, purché $|\mathbb{F}| \gg deg(t)$. Dunque, se $t(r) = 0$, possiamo con elevata probabilità affermare che t è il polinomio nullo, e quindi che la computazione codificata dal QAP è corretta.

Esempio pratico ➞ Definiamo il polinomio minimo $Z = (x-1) \times (x-2) \times (x-3) \times (x-4)$ il quale sarà zero in tutti i punti che corrispondono alle porte logiche.
Eseguiamo il controllo del prodotto scalare con i polinomi sopra menzionati. Per prima cosa, consideriamo i polinomi intermedi:

$$A.s = [43.0, -73.333, 38.5, -5.166]$$
$$B.s = [-3.0, 10.333, -5.0, 0.666]$$
$$C.s = [-41.0, 71.666, -24.5, 2.833]$$

Il risultato del prodotto scalare tra A e B sarà uguale a:

$$A.s \times B.s = [-129.0, 664.332, -1088.277, 808.666, -294{,}777, 51.5, -3.444]$$

Successivamente troviamo il polinomio $t = A.s \cdot B.s - C.s$:

$$t = [-88.0, 592.666, -1063.777, 805.833, -294.777, 51.5, -3.444]$$

Infine eseguiamo la divisione t per Z, dove quest'ultimo è costruito come il prodotto dei fattori $(x - r_i)$, dove $r_i$sono le coordinate x delle porte logiche:

$$Z = [24, -50, 35, -10, 1]$$

Otteniamo dunque un risultato con resto zero:

$$h = \frac{t}{Z} = [-3.666, 17.055, -3.444]$$

In conclusione, abbiamo quindi trovato la soluzione per il QAP. Se tentiamo di falsificare una qualsiasi variabile nella soluzione R1CS da cui stiamo derivando questo QAP, ad esempio impostando l'ultima a 31 invece che a 30, otterremmo un polinomio t che non supera uno dei controlli (in quel caso specifico, il risultato in $x = 3$ sarebbe -1 invece di 0), e inoltre t non sarebbe un multiplo di Z; infatti, dividendo t per Z otterremmo un resto di $[-5.0, 8.833, -4.5, 0.666]$. [48]

3.5 Costruzione di SNARKs tramite QAP

In questa sezione vedremo la costruzione di uno schema SNARK basato su QAP di Parno et al. [45], utilizzando uno strumento di codifica Enc su un campo $\mathbb{F}$ composto dai seguenti algoritmi:

- $K(1^\lambda) \rightarrow (pk, sk)$ è un algoritmo di generazione di chiavi che prende in input alcuni parametri di sicurezza e restituisce come output alcuni stati segreti sk insieme ad alcune informazioni pubbliche pk.
- $Enc(s) \rightarrow z$ è un algoritmo di codifica che associa un elemento s del campo, ad un valore di codifica. In base all'algoritmo di codifica, Enc richiederà l'informazione pubblica pk generata da K o lo stato segreto sk.

Gli algoritmi sopracitati devono soddisfare tre proprietà:

1) *Additivamente omomorfo* → intuitivamente, per l'algebra lineare, $Enc(x + y) =$

$Enc(x) + Enc(y)$.

2) *Rilevamento della radice quadrata* → esiste un algoritmo efficiente che, dato $Enc(a_0), \dots, Enc(a_t)$ e il polinomio quadratico $pp \in \mathbb{F}[x_0, \dots, x_t]$, sia in grado di distinguere se $pp(a_1, \dots, a_t) = 0$.
3) *Verifica dell'immagine* → esiste un algoritmo $ImVer$ che sia in grado di distinguere se un elemento c è una codifica corretta di un elemento del campo ($ImVer(c) \rightarrow 0/1$).

Quando si parla di polinomio quadratico $pp \in \mathbb{F}[x_0, \dots, x_t]$, ci si riferisce a un polinomio definito su un campo $\mathbb{F}$ e che coinvolge le variabili $x_0, \dots, x_t$, dove il grado massimo di qualsiasi termine nel polinomio è 2. Ciò significa che ogni termine del polinomio può essere al massimo di secondo grado rispetto alle variabili. La proprietà importante di interesse spesso è la capacità di valutare se il polinomio si annulla per un particolare insieme di valori delle variabili, ovvero se $pp(a_1, \dots, a_t) = 0$ per alcuni $(a_0, \dots, a_t) \in \mathbb{F}^{t+1}$.

La forma generale di un polinomio quadratico in $t + 1$ variabili può essere espressa come:

$$pp(x_0, \dots, x_t) = \sum_{0 \le i \le j \le t} c_{ij} x_i x_j + \sum_{i=0}^{t} b_i x_i + d$$

Dove:

- c_{ij} sono i coefficienti dei termini quadratici $x_i x_j$ (con $0 \le i \le j \le t$);
- b_i sono i coefficienti dei termini lineari x_i;
- d è il termine costante;
- tutti i coefficienti c_{ij}, b_i, d appartengono al campo $\mathbb{F}$

Snarks from QAP

Analizziamo nello specifico il flusso di esecuzione:

- **Algoritmo di Generazione** $Gen(1^\lambda, C) \rightarrow (crs, vrs)$
  L'algoritmo di setup Gen , prende come input 1^λ[12] e il circuito C con N valori in ingresso/uscita. Esso genera un QAP Q di dimensione m e grado d su un campo $\mathbb{F}$, che verifica C. Definisce $I_{mid} = \{N + 1, \dots m\}$, e successivamente esegue il setup per lo schema di codifica Enc (con lo stato segreto sk, oppure senza, $sk = \perp$). Infine il generatore esegue un campionamento $\alpha, \beta_v, \beta_w, \beta_y, s \leftarrow \mathbb{F}$ tale che $t(s) \neq 0$, e restituisce $(vrs = sk, crs)$, dove crs è:

[12] L'espressione 1^λ non rappresenta l'elevamento a potenza del numero 1, bensì una notazione convenzionale utilizzata nella teoria della complessità e nella crittografia per indicare che il parametro di sicurezza λ è fornito in *codifica unaria*. 1^λ rappresenta una stringa composta da "uni" di lunghezza λ, e serve a specificare che l'input dell'algoritmo ha effettivamente lunghezza λ. Ciò implica che l'algoritmo è dunque polinomiale rispetto a λ stesso, e non rispetto alla lunghezza binaria di λ (che sarebbe $log_2 \lambda$).

$$crs := \begin{pmatrix} Q, Enc, \{Enc(1), Enc(s), \dots, Enc(s^d), Enc(\alpha), Enc(\alpha s), \dots, Enc(\alpha s^d)\}, \\ \{Enc(\beta_v), Enc(\beta_w), Enc(\beta_y)\} \\ \{Enc(\beta_v v_i(s))\}_{i \in I_{mid}}, \{Enc(\beta_w w_i(s))\}_{i \in [m]}, \{Enc(\beta_y y_i(s))\}_{i \in [m]} \end{pmatrix}$$

Come menzionato precedentemente, solo l'algoritmo di generazione conosce tutti i termini necessari per il setup iniziale. Esaminiamo nel dettaglio il ruolo di ciascuno di essi.

- α: è utilizzato per moltiplicare le valutazioni dei polinomi durante la codifica, assicurando che la separazione tra le parti del calcolo sia mantenuta;
- $\beta_v, \beta_w, \beta_y$: sono i coefficienti associati alle componenti del QAP, usati per mescolare le valutazioni $v(x), w(x), y(x)$;
- s: è un elemento casuale utilizzato per valutare i polinomi nei punti specifici e per costruire le codifiche omomorfiche.

Tutti questi valori sono necessari per mantenere la sicurezza dello schema SNARK, in quanto sono tutti segreti. Una volta creati i parametri comuni (crs), questi valori non devono essere rivelati o utilizzati direttamente da nessun altro.
Il generatore utilizza questi valori per calcolare e codificare vari polinomi e valutazioni necessari per il $prover$ e il $verifier$. Una volta che i parametri di setup sono generati, il sistema può funzionare senza la necessità di conoscere questi valori segreti, garantendo che il $prover$ possa generare prove e il $verifier$ possa verificarle senza esporre i segreti critici.

- **Prover** $Prove(crs, u, w) \rightarrow \pi$
  L'algoritmo del Prover $Prove$, sull'affermazione in input $u := (c_1, \dots, c_N)$, calcola un segreto $w := (c_{N+1} \dots c_m)$ e $v_{mid}(x) = \sum_{i \in I_{mid}} c_i v_i(x)$, $v_c(x) \sum_{i \in [m]} c_i v_i(x)$, $w_c(x) \sum_{i \in [m]} c_i w_i(x)$, $y_c(x) \sum_{i \in [m]} c_i y_i(x)$ tale che:

$$t(x)\ divide\ p(x) = v_c(x) w_c(x) - y_c(x)$$

  Successivamente calcola il polinomio $h(x) := \frac{p(x)}{t(x)}$
  Utilizzando la proprietà di additivamente omomorfo dello schema di codifica Enc e i valori crs, il Prover calcola le codifiche delle seguenti valutazioni polinomiali in s:

$$\begin{aligned} H &:= Enc(h(s)), & \hat{H} &:= Enc(\alpha h(s)), \\ V_{mid} &:= Enc(v_{mid}(s)), & \hat{V}_{mid} &:= Enc(\alpha v_{mid}(s)), \\ W &:= Enc(w_c(s)), & \hat{W} &:= Enc(\alpha w_c(s)), \\ Y &:= Enc(y_c(s)), & \hat{Y} &:= Enc(\alpha y_c(s)), \\ B &:= Enc\left(\beta_v v_c(s) + \beta_w w_c(s) + \beta_y y_c(s)\right). \end{aligned}$$

  Dove il polinomio è definito come $v_{mid}(x) := \sum_{i \in I_{mid}} c_i v_i(x)$. Dal momento che i valori di c_i, con $i \in [N]$ corrispondono all'input u (noto anche al Verifier), quest'ultimo può calcolare la parte mancante della combinazione lineare $v_c(x)$ per $\{v_i(x)\}$ e codificarla come segue: $V := Enc(v_c(s))$.
  La dimostrazione risulterà composta dai seguenti elementi $(H, \hat{H}, V_{mid}, \hat{V}_{mid}, W, \hat{W}, Y, \hat{Y}, B)$.

- **Verifier** $Ver(vrs, u, \pi) \rightarrow 0/1$
  Il verifier, dopo aver ricevuto la prova π e l'affermazione u, utilizza la proprietà del rilevamento della radice quadrata dello schema di codifica Enc per verificare che la dimostrazione soddisfi:
  - ✓ *Termini di estraibilità*, $\widehat{H} \stackrel{?}{=} \alpha H, \widehat{V}_{mid} \stackrel{?}{=} \alpha V_{mid}, \widehat{W} \stackrel{?}{=} \alpha W, \ \widehat{Y} \stackrel{?}{=} \alpha Y$.
    I termini appena definiti sono progettati per consentire l'estrazione dei dati utilizzando un'assunzione di conoscenza.
  - ✓ *Controllo di divisibilità*, $H \cdot T \stackrel{?}{=} \mathrm{V} \cdot \mathrm{W} - \mathrm{Y}$ dove $T = Enc(t(s))$, $V := Enc(v_c(s))$ e può essere calcolato usando il crs. Ciò corrisponde al vincolo di divisione polinomiale.
  - ✓ *Controllo span lineare*, $B \stackrel{?}{=} \beta_v V + \beta_w W + \beta_y Y$, quest'ultimo controllo garantisce che i polinomi $v_c(x), w_c(x), y_c(x)$ siano effettivamente combinazioni lineari dell'insieme iniziale di polinomi $\{v_i\}_i, \{w_i\}_i, \{y_i\}_i$ [47].

Uno degli aspetti più rilevanti dello schema Pinocchio [45] è la sua capacità di produrre **prove succinte**, la cui **dimensione** rimane **costante** indipendentemente dalla complessità del circuito aritmetico. Nel lavoro originale, gli autori specificano che la proof ha sempre una lunghezza di 288 byte, un valore che non varia anche in presenza di circuiti con milioni di vincoli, ed è composta da **8 elementi di gruppo** (7 in G_1 e 1 in G_2). Nell'esempio presentato, tale prova è rappresentata da 9 elementi di gruppo codificati su curve ellittiche; tale differenza è dovuta a una scelta implementativa che include un elemento addizionale non sempre conteggiato nella formulazione teorica del protocollo, ma che non altera la proprietà di dimensione costante. Questa proprietà è di particolare importanza in applicazioni pratiche, poiché consente di trasmettere e memorizzare prove di calcoli complessi senza che la loro dimensione aumenti proporzionalmente alla complessità computazionale.
Un ulteriore vantaggio di Pinocchio riguarda il **tempo di verifica**, che nel modello proposto rimane costante al variare della complessità del circuito. Il Verifier esegue sempre un numero fisso di operazioni crittografiche principalmente pairing bilineari e verifiche di equazioni determinate in fase di setup indipendentemente dal numero di vincoli.
In particolare, richiede **12 pairing** organizzati in un numero fisso di equazioni bilineari, sempre uguale per ogni prova, a prescindere dalla dimensione del circuito.

Pinocchio vs lo stato dell'arte di Groth16

Lo stato dell'arte di SNARK per QAP è il celebre risultato di Groth [50] che raggiunge una dimensione di prova di tre elementi di gruppo nel Generic GroupModel, migliorando sensibilmente Pinocchio sotto il profilo della **compattezza** e della **rapidità di verifica**..
In Groth16, la proof è sempre costituita da **3 elementi di gruppo** (2 in G_1 e 1 in G_2) [51], contro gli 8 elementi della formulazione originale di Pinocchio (o i 9 elementi nell'esempio riportato). Questo garantisce una dimensione costante estremamente ridotta tipicamente poche centinaia di byte in qualsiasi scenario applicativo.
Anche il tempo di verifica è costante, ma ulteriormente ottimizzato: il Verifier deve sempre verificare **una sola equazione bilineare** composta da **3 pairing**, indipendentemente dal numero di vincoli. Ciò riduce ulteriormente il carico computazionale, permettendo verifiche ancora più rapide e prevedibili.

A livello di sicurezza sono simili principalmente per il rischio sulla compromissione del Setup in quanto rappresenta un rischio significativo.
Per quanto concerne il loro uso applicativo, Pinocchio è stato utilizzato in molte applicazioni pratiche come la verifica di calcoli cloud, blockchain e privacy-preserving smart contracts. Groth16 è ampiamente utilizzato in applicazioni moderne di blockchain (come Zcash), grazie alla sua efficienza e alla compattezza delle prove.
In conclusione Groth16 rappresenta un miglioramento significativo rispetto a Pinocchio per quanto riguarda la dimensione della prova e l'efficienza della verifica. Tuttavia, entrambi i protocolli richiedono un *trusted setup*, che è una delle principali limitazioni degli SNARKs. La scelta tra i due può dipendere dalle specifiche esigenze di efficienza, dimensione della prova e sicurezza nell'applicazione desiderata.

Difetti delle SNARKs

Le SNARKs, pur essendo molto efficienti, presentano alcune limitazioni quali:

1. **Trusted Setup**: è necessario un setup iniziale affidabile, poiché una sua compromissione potrebbe mettere a rischio l'intera sicurezza del sistema;
2. **Complessità del Setup**: La fase di setup può essere complessa e costosa;
3. **Dipendenza dai Gruppi a Curva Ellittica**: Molti SNARKs richiedono operazioni su curve ellittiche specifiche, che potrebbero non essere supportate uniformemente in tutti i sistemi.
4. **Infrastruttura Critica**: La gestione sicura del trusted setup è cruciale e rappresenta un punto di vulnerabilità.

3.6 Applicazioni pratiche dei SNARK su blockchain

Il primo esempio concreto di impiego di SNARK è nella criptovaluta **Zcash**, che utilizza prove a conoscenza zero per realizzare transazioni completamente shielded, nascondendo mittente, destinatario e importo [52]. Nel protocollo Sapling, Zcash adotta il sistema Groth16 [50], che genera prove estremamente compatte, appena tre elementi di gruppo, e richiede solo poche operazioni di pairing per la verifica. Questo è possibile grazie a una fase iniziale di trusted setup: l'esecuzione di una cerimonia multiparte produce un proving key segreto e un verifying key pubblico. Il proving key contiene elementi casuali ("toxic waste") privati, mentre il verifying key è condiviso in rete. Grazie a questo setup, le prove SNARK possono essere generate e verificate in modo molto efficiente, ma a costo di dover fidarsi che il setup iniziale sia stato condotto correttamente. Per superare questa dipendenza, Electric Coin Company ha sviluppato **Halo 2**, un nuovo costrutto zk-SNARK che elimina completamente il bisogno di trusted setup [53]. Halo 2 sfrutta tecniche di "inner-product argument" e polinomiali IOP per supportare composizione ricorsiva delle prove senza parametri tossici. In questo modo Zcash potrà aggiornare il protocollo, introducendo nuovi asset definiti dall'utente, senza eseguire ulteriori ceremony. Halo 2 si basa anche su assunzioni crittografiche più semplici di Groth16, evitando le pairing complesse su BLS12-381, riducendo la superficie d'attacco e aumentando l'agilità del protocollo. Grazie a queste tecnologie, Zcash continua a offrire privacy elevata, mantenendo la sicurezza e l'integrità delle regole di consenso.

I **zk-Rollup** rappresentano una seconda applicazione pratica dei SNARK, orientata alla scalabilità delle blockchain come Ethereum. In uno zk-rollup, un operatore (sequencer) aggrega fuori catena gruppi di transazioni e ne calcola un nuovo stato globale. Quindi genera una prova di validità (un zk-SNARK) che attesta la correttezza di tutte le transazioni elaborate. Lo stato aggiornato e la validità proof vengono inviati a un contratto intelligente di livello 1 su Ethereum: il contratto aggiorna la root di stato del rollup solo dopo aver verificato la proof ([54] - [55] - [56]). In tal modo, quasi tutte le operazioni computazionali sono scaricate sul livello 2, mentre Ethereum si limita a verificare una prova succinta. Questo meccanismo assicura che ogni transazione rispetti le regole di consenso senza doverla ricalcolare on-chain. Di conseguenza i zk-rollup aumentano drasticamente il throughput e riducono i costi delle fee. Uno studio dimostra che un rollup può processare decine di transazioni complessi al secondo (ben oltre le poche unità di Ethereum) [54]. Nonostante questa efficienza, la sicurezza rimane elevata. Le prove a conoscenza zero conferiscono al rollup la stessa certezza di validità che avremmo se tutte le transazioni fossero eseguite on-chain.

Diversi progetti concretizzano questi vantaggi. Matter Labs ha sviluppato zkSync Era, una soluzione Layer-2 zk-rollup, progettata per essere pienamente compatibile con l'Ethereum Virtual Machine (EVM). Utilizza prove a conoscenza zero di tipo zk-SNARK, permettendo la validazione off-chain delle transazioni mantenendo la sicurezza della mainnet Ethereum. Grazie a ciò, gli smart contract sono scritti in Solidity o Vyper come su Ethereum e l'intero ecosistema Ethereum (client, IDE e librerie) rimane riutilizzabile [57]. Polygon zkEVM e Linea di ConsenSys sono due zk-rollup di tipo 2 (zkEVM) che puntano a offrire un'esperienza di sviluppo molto simile a Ethereum. Entrambi supportano smart contract in Solidity con compatibilità quasi totale con l'EVM e consentono l'uso di tool familiari (Hardhat, Foundry, Remix, MetaMask, Infura, ecc.) senza richiedere compilatori speciali [58]. Linea enfatizza la piena integrazione con l'ecosistema Ethereum esistente (wallet EVM, tooling, nodi, ecc.). La differenza principale rispetto ad altre soluzioni è puramente architetturale: le modifiche interne non penalizzano l'esperienza di sviluppo, che rimane pressoché identica a quella su Ethereum. Analogamente, il rollup Scroll adatta alcuni opcode Ethereum in modo da rientrare nei circuiti zk-SNARK, pur mantenendo la compatibilità con il linguaggio Solidity [59]. Dall'altro lato, StarkNet (StarkWare) adotta un approccio diverso, utilizzando prove zk-STARK (trasparenti, resistenti al quantum) e un linguaggio nativo Cairo anziché l'EVM [60]. Non essendo compatibile nativamente con la EVM, StarkNet richiede toolchain specifiche, ma offre l'assenza totale di trusted setup e una forte sicurezza crittografica.

In tutti questi casi l'uso di SNARK concreti si traduce in benefici pratici chiari: Zcash garantisce privacy completa ai suoi utenti, mentre zk-Rollup e zk-EVM abilitano enormi guadagni di scalabilità e throughput di Ethereum senza rinunciare alla sicurezza della verifica on-chain. Tali soluzioni riflettono i vantaggi empirici dei protocolli SNARK: transazioni private e concise in Zcash e validità crittograficamente garantita con bassa latenza e costi contenuti negli ambienti L2. Tutto ciò le rende applicazioni di riferimento nel panorama blockchain contemporaneo.

Capitolo 4

4 Bulletproofs

Bulletproofs (letteralmente *prove di proiettile*) è stato proposto dall'Applied Cryptography Group (ACG) di Stanford nel dicembre 2017 in un documento accademico con contributi dell'University College di Londra e Blockstream. Bulletproof è un nuovo protocollo di prova a conoscenza zero non interattivo, utilizzato per dimostrare che un valore segreto impegnato (*commitment*) si trovi in un dato intervallo. La dimensione della prova è solo logaritmica nella dimensione del segreto. Essi consentono di dimostrare che un valore impegnato si trova in un intervallo utilizzando solamente $2\,log_2(n) + 9$ elementi di gruppo e di campo, dove n è la lunghezza in bit dell'intervallo. I tempi di generazione e verifica delle prove sono lineari in n.
I bulletproof, a differenza degli SNARKS, non richiedono alcun algoritmo gestito da un trusted party; si basano solamente sull'ipotesi dei logaritmi discreti e sono resi non interattivi utilizzando l'euristica di Fiat-Shamir. Tuttavia, la verifica di un bulletproof richiede più tempo rispetto alla verifica di una prova SNARK.
Essi sono progettati per consentire transazioni riservate (**confidential transactions, CT**) efficienti in Bitcoin e altre criptovalute. Le transazioni riservate furono introdotte da Gregory Maxwell, nascondono l'importo trasferito nella transazione, e contengono una prova crittografica per garantire la loro validità. I bulletproof riducono le dimensioni della prova crittografica da oltre $10kB$ a meno di $1kB$.
Le blockchain per la privacy come Monero, dopo il loro passaggio all'utilizzo dei bulletproofs, hanno registrato una riduzione fino all'80% delle dimensioni e delle commissioni delle transazioni [61].
Inoltre Bulletproof supporta l'aggregazione di prove di intervallo, in modo che una parte possa dimostrare che m *commitments* si trovano in un determinato intervallo fornendo $O(\log(m))$ elementi per la lunghezza di una singola prova. Per aggregare prove da più parti, abilitiamo quest'ultime affinché possano generare una singola prova senza rivelarsi reciprocamente i loro input tramite un semplice protocollo **MPC** (*multi-party computation*) per la costruzione di Bulletproof.

Range proofs

Le Range proofs, o prove di intervallo, sono state le prime forme di convalida degli impegni utilizzate nelle transazioni riservate. Forniscono un modo per dimostrare che un valore segreto, crittografato o "committato", si trova all'interno di un determinato intervallo senza rivelarne l'esatto valore. Ad esempio, con una semplice prova di intervallo, si potrebbe affermare che la defunta regina Elisabetta è nata dopo la Prima Guerra Mondiale e prima della Seconda senza rivelare il suo effettivo anno di nascita.
Le prove di intervallo sono state principalmente adottate da protocolli crittografici che migliorano la privacy e utilizzate per offuscare gli importi delle transazioni su reti come Monero. È interessante notare che Monero ha implementato un sistema di verifica ancora più efficiente chiamato Bulletproofs+. Infatti, dopo una transazione sulla blockchain di Monero, Bulletproofs+ dimostra che il tuo pagamento è un numero positivo senza rivelare l'importo pagato. [62]

Fondamenti teorici dei Bulletproofs

I Bulletproofs sono un protocollo di prove a conoscenza zero non interattivo progettato per dimostrare proprietà di valori "impegnati" (ad esempio, con Pedersen commitment) senza rivelare il valore stesso. In particolare sono nati per ottimizzare le range proofs in transazioni confidenziali. Grazie a questi meccanismi, il prover può convincere il verificatore che un valore segreto rientra in un dato intervallo (ad esempio che un ammontare monetario è compreso tra 0 e N) senza svelarne alcuna informazione aggiuntiva. L'approccio proposto da Bünz et al. nel 2018 descrive Bulletproofs come prove molto brevi e prive di una fase iniziale di «trusted setup» [61], rendendo il protocollo particolarmente adatto a sistemi distribuiti come le blockchain. Ad esempio, nelle Confidential Transactions di Monero l'impiego di Bulletproofs ha permesso di ridurre drasticamente (circa l'80%) la quantità di dati necessari per validare gli importi delle transazioni, a vantaggio di costi di banda e storage molto minori.
Concettualmente, i Bulletproofs funzionano combinando Pedersen commitments con un argomento di prodotto scalare iterativo. In pratica, il prover crea impegni crittografici dei valori segreti (usando Pedersen commitment) e poi dimostra in zero-knowledge, tramite un complesso protocollo basato su operazioni nei gruppi ellittici, che quei valori soddisfano la condizione richiesta (ad esempio che siano in un certo intervallo). La prova interattiva risultante viene resa non interattiva tramite l'euristica di Fiat–Shamir, che sostituisce gli scambi di messaggi con l'uso di hash pubblici. In questo modo il prover può generare una singola prova da inviare al verificatore. La sicurezza del protocollo si basa sulla difficoltà del problema del logaritmo discreto nelle curve ellittiche, analogamente ad altri schemi crittografici, garantendo che il valore rimanga nascosto. Grazie alla trasformazione Fiat–Shamir, la prova non interattiva assume di fatto la forma di una «firma» crittografica sulla validità della proprietà dichiarata, verificabile efficientemente dal verificatore.
Un aspetto fondamentale delle Bulletproofs è l'assenza di qualsiasi trusted setup. Ciò significa che non è richiesta alcuna cerimonia di generazione di parametri segreti condivisi: tutti i parametri pubblici possono essere fissati in modo deterministico o casuale, senza alcun "toxic waste" iniziale. In termini pratici, questo conferisce al protocollo un grado di trasparenza e sicurezza aggiuntivo, poiché non esiste una terza parte affidabile da cui dipendere. In secondo luogo, le prove Bulletproofs sono estremamente compatte: la loro dimensione cresce solo in modo logaritmico rispetto alla precisione del valore controllato. Ad esempio, nella descrizione originaria si mostra che per un valore a n bit bastano circa $2log\_2\ (n) + 9$ elementi di gruppo per dimostrare che il valore rientra nell'intervallo desiderato, contro dimensioni lineari richieste da soluzioni tradizionali. Questa caratteristica rende le prove molto brevi anche per range di valori molto ampi. Inoltre è possibile aggregare più prove di intervallo in una singola prova combinata, aggiungendo solo un lieve sovraccarico logaritmico di dati, il che è particolarmente utile quando si vogliono dimostrare contemporaneamente condizioni su più impegni.

4.1 Applicazioni di Bulletproof

Rimescolamenti verificabili.

Nelle applicazioni dei rimescolamenti verificabili o *verifiable shuffles*, si considerano due elenchi di valori impegnati $x_1, \dots, x_n$ e $y_1, \dots, y_n$. L'obiettivo è dimostrare che la seconda lista è una permutazione della prima. Tale problematica prende il nome di rimescolamento verificabile. Esso ha diverse applicazioni nelle reti miste [63] e nelle *solvency proofs* [64]. Attualmente il rimescolamento più efficiente ha come dimensione $O(\sqrt{n})$; i bulletproof possono essere utilizzati per raggiungere dimensioni di $O(\log n)$.
I due elenchi di valori impegnati sono forniti come input al protocollo del circuito, il quale può implementare un rimescolamento ordinando i due elenchi e verificarne l'uguaglianza. Un circuito di ordinamento può essere implementato utilizzando le moltiplicazioni $O(n \cdot \log(n))$, il che significa che la dimensione della prova sarà soltanto $O(\log(n))$. Grazie all'efficienza concreta di bulletproof, utilizzarlo per applicazioni di mescolamenti verificabili sarebbe molto efficiente nella pratica in quanto costruirne la dimostrazione e verificarla richiederebbe tempo lineare in n [61].

Dimostrazioni NIKZ per gli Smart Contracts

Sulla piattaforma decentralizzata Ethereum, si utilizzano gli smart contracts o contratti intelligenti per consentire transazioni complesse. I contratti intelligenti, come qualsiasi altra transazione blockchain, sono pubblici e non forniscono privacy intrinseca.
Al fine di garantire la privacy, un primo strumento proposto, sono state le dimostrazioni a conoscenza zero non interattive. Tuttavia, questi protocolli sono limitati in quanto la stessa NIZK non è adatta per la verifica da parte di uno smart contract in quanto la comunicazione sulla blockchain con uno smart contract è costosa e il potere computazionale di quest'ultimo è molto limitato. Di contro i protocolli Snarks sembrerebbero i più adatti grazie alle loro prove succinte e verificatori efficienti, ma richiedono una complessa configurazione attendibile da una terza parte fidata.
Bulletproof riesce ad apportare migliorie nei dettagli dei protocolli citati in precedenza; se vi sono più utenti coinvolti contemporaneamente, ad esempio ad un'asta, ogni partecipante deve avere la certezza che tutte le offerte vengano ben formulate. Per ovviare ai lunghi tempi di verifica nei Bulletproofs, gli smart contract possono agire in modo ottimistico e verificare una prova solamente se qualche parte ne contesta la validità. Inoltre può essere ulteriormente migliorato utilizzando un modello di delega arbitrale interattivo [65], precedentemente proposto per altre applicazioni blockchain. In questo modello, il dimostratore fornisce una prova insieme a un impegno succinto. Uno sfidante, il quale non è d'accordo con il calcolo eseguito, si impegna in una ricerca binaria interattiva per trovare il primo punto di divergenza nel calcolo. Lo smart contract può quindi eseguire questo singolo passaggio di calcolo ed individuare la parte che ha fornito una traccia di esecuzione errata.
Di fatto, la proprietà interessante di questo protocollo, è che anche quando una dimostrazione dovesse venire contestata, lo smart contract deve verificare esclusivamente un singolo passaggio di calcolo, ovvero un singolo gate del circuito di verifica.
In combinazione con piccoli Bulletproof, questa esecuzione può generare smart contract più complessi ma che preservano la privacy.

4.2 Offuscamento delle transazioni nella blockchain di Monero

Monero è una criptovaluta che pone un forte accento sulla privacy e sull'anonimato delle transazioni. A differenza di Bitcoin e molte altre criptovalute, Monero implementa una serie di tecnologie avanzate per offuscare le transazioni, rendendo estremamente difficile, se non impossibile, tracciare il mittente, il destinatario e l'importo di una transazione. Questo capitolo esplora i principali meccanismi utilizzati da Monero per garantire l'anonimato e la sicurezza delle transazioni. Gli indirizzi degli utenti che inviano Monero sono protetti tramite **ring signatures**, mentre gli indirizzi degli utenti che ricevono la transazione vengono celati tramite gli **stealth addresses**, e l'importo della transazione viene offuscato utilizzando **RingCT** [66]. Di seguito i principali meccanismi utilizzati da Monero per garantire l'anonimato e la sicurezza delle transazioni.

Firme ad Anello (Ring Signatures)

Le firme ad anello rappresentano una delle tecnologie chiave impiegate da Monero per offuscare l'identità del mittente di una transazione. Quando un utente invia Monero, la firma ad anello mescola le chiavi pubbliche dell'utente con quelle di altri utenti presenti nella rete. Questo processo crea un anello di potenziali firmatari, confondendo così l'osservatore esterno che non può determinare con certezza quale chiave pubblica sia la vera fonte della transazione. Questa tecnica non solo garantisce l'anonimato del mittente, ma aumenta anche la sicurezza complessiva della rete Monero.

Transazioni Confidenziali ad Anello (Ring Confidential Transactions, RingCT)

L'introduzione delle RingCT ha rappresentato un significativo avanzamento nella tecnologia di offuscamento delle transazioni. Mentre le firme ad anello nascondono l'identità del mittente, le RingCT estendono questa protezione nascondendo anche l'importo delle transazioni. Utilizzando prove a conoscenza zero (zero-knowledge proofs), RingCT permette di verificare la validità delle transazioni senza rivelare l'importo effettivo. Questo duplice livello di anonimato, che copre sia il mittente che l'importo, rende Monero estremamente resistente all'analisi forense[13] delle transazioni blockchain.

Indirizzi Stealth (Stealth Addresses)

Gli indirizzi stealth sono utilizzati per garantire la privacy del destinatario di una transazione. Quando un utente riceve Monero, il mittente genera un indirizzo una tantum (one-time address) che rappresenta il destinatario sulla blockchain. Solo il destinatario conosce questo indirizzo una tantum e può collegarlo al proprio indirizzo pubblico. Questo meccanismo impedisce che osservatori esterni possano collegare facilmente le transazioni ai destinatari, proteggendo così la privacy di chi riceve Monero.

[13] L'analisi forense delle transazioni blockchain è un campo specializzato della forensica digitale che si occupa di esaminare e interpretare le attività e le transazioni registrate su una blockchain

Protocollo Dandelion++

Il protocollo Dandelion++ è progettato per nascondere l'origine delle transazioni durante la loro propagazione nella rete. Funziona suddividendo il percorso della transazione in due fasi distinte: Stem (gambo) e Fluff(soffione). Durante la fase di Stem, la transazione passa attraverso una catena di nodi selezionati casualmente. Nella fase di Fluff, la transazione viene diffusa a tutta la rete. Questo metodo riduce significativamente la possibilità di correlare l'indirizzo IP dell'utente con la transazione, offrendo un ulteriore strato di protezione per la privacy degli utenti.

Kovri: Un Progetto Futuro

Kovri è un progetto ancora in fase di sviluppo che mira a utilizzare la tecnologia I2P (Invisible Internet Project) per nascondere ulteriormente le transazioni e le comunicazioni all'interno della rete Monero. Implementando I2P, Kovri intende offuscare non solo le transazioni registrate sulla blockchain, ma anche l'origine delle transazioni stesse all'interno della rete. Questo miglioramento garantirebbe un ulteriore livello di anonimato, proteggendo gli utenti dall'analisi del traffico di rete.

Per comprendere meglio come le tecnologie di offuscamento di Monero lavorino insieme per garantire la privacy, consideriamo un esempio pratico di una transazione:

Scenario: Alice desidera inviare 5 XMR (Monero) a Bob.
Quando Alice invia i 5 XMR a Bob, la sua chiave pubblica viene mescolata con chiavi pubbliche di altri utenti in un anello. Supponiamo che l'anello contenga chiavi pubbliche di 10 utenti. Un osservatore esterno vede una firma ad anello composta da queste 10 chiavi, ma non può determinare quale di queste appartiene ad Alice, offuscando così l'identità del mittente.
L'importo di 5 XMR è nascosto utilizzando le transazioni confidenziali ad anello. Anche se la transazione è visibile sulla blockchain, l'importo effettivo non è divulgato. Un osservatore esterno può verificare che la transazione è valida senza conoscere l'importo trasferito, grazie alle prove a conoscenza zero.
Per garantire la privacy di Bob, Alice genera un indirizzo una tantum basato sull'indirizzo pubblico di Bob. Questo indirizzo una tantum è registrato sulla blockchain come destinatario della transazione. Solo Bob può collegare questo indirizzo una tantum al proprio indirizzo pubblico e riscattare i fondi, mantenendo così la sua identità privata.
Infine, quando Alice trasmette la transazione alla rete Monero, essa passa prima attraverso una serie di nodi selezionati casualmente (fase Stem) e poi viene propagata a tutta la rete (fase Fluff). Questo processo offusca l'origine della transazione, rendendo difficile per un osservatore esterno correlare l'indirizzo IP di Alice con la transazione.

Monero, attraverso l'implementazione di firme ad anello, RingCT, indirizzi stealth, e progetti futuri come Kovri e il protocollo Dandelion++, offre un livello di privacy e anonimato nelle transazioni che è tra i più elevati nel mondo delle criptovalute. Queste tecnologie combinate rendono Monero una scelta ideale per gli utenti che desiderano mantenere la loro attività finanziaria privata e sicura. La continua evoluzione delle tecnologie di offuscamento in Monero garantisce che la criptovaluta rimanga all'avanguardia nell'ambito della privacy delle transazioni [67].

Capitolo 5

5 Conclusioni

Il presente lavoro di tesi ha affrontato in maniera sistematica e approfondita il tema delle dimostrazioni a conoscenza zero (Zero-Knowledge Proofs, ZKP), una delle più significative innovazioni introdotte nel campo della crittografia moderna. Attraverso un percorso che parte dalle fondamenta teoriche per giungere alle applicazioni pratiche, si è cercato di evidenziare come tali protocolli costituiscano oggi un elemento imprescindibile per la sicurezza dei sistemi distribuiti e per la tutela della riservatezza nelle comunicazioni digitali.

Il primo capitolo ha introdotto i principi fondamentali alla base delle ZKP, delineandone la genesi storica, le proprietà essenziali (completezza, correttezza e zero-conoscenza) e le distinzioni tra le versioni interattive e non interattive. È stato mostrato come la struttura formale delle prove di conoscenza, supportata dai Σ-protocol, consenta di verificare un'affermazione senza rivelare informazioni sensibili sul segreto posseduto dal dimostratore (Prover). L'analisi del protocollo di identificazione di Schnorr e della trasformata di Fiat-Shamir ha permesso di comprendere il processo di transizione dai sistemi interattivi ai sistemi non interattivi, ponendo le basi per l'adozione pratica delle dimostrazioni a conoscenza zero in contesti computazionali distribuiti.

Il secondo capitolo ha esteso la trattazione al contesto della blockchain, tecnologia che ha profondamente ridefinito il concetto di fiducia nel mondo digitale. In esso è stato illustrato come la blockchain rappresenti un registro distribuito, immutabile e trasparente, la cui sicurezza si fonda su meccanismi crittografici e di consenso, quali il Proof of Work (PoW) e il Proof of Stake (PoS). Tuttavia, pur offrendo vantaggi in termini di decentralizzazione e integrità, tale tecnologia presenta anche criticità legate alla scalabilità, al consumo energetico e alla riservatezza delle informazioni. In tale scenario, le dimostrazioni a conoscenza zero si rivelano strumenti fondamentali per garantire la privacy e la sicurezza delle transazioni, permettendo di validare i dati senza esporli, in perfetto accordo con i principi di trasparenza e verificabilità della blockchain.

Nel terzo capitolo, l'attenzione si è concentrata sugli ZK-SNARK (Zero-Knowledge Succinct Non-Interactive Argument of Knowledge), una delle implementazioni più note e avanzate delle ZKP. Attraverso lo studio dei Quadratic Arithmetic Programs (QAP) e dei Rank-1 Constraint Systems (R1CS), è stato mostrato come questi strumenti consentano di costruire prove estremamente compatte e verificabili in tempi ridotti. Gli ZK-SNARK rappresentano una soluzione tecnologica di grande rilevanza, già ampiamente utilizzata in ambito blockchain (come nei protocolli Zcash e Ethereum), grazie alla loro capacità di garantire privacy, efficienza e riduzione dei costi computazionali.

Il quarto capitolo ha introdotto i Bulletproofs, una più recente tipologia di prove a conoscenza zero che, a differenza dei sistemi SNARK, non richiedono una fase di configurazione fidata (trusted setup). Tale caratteristica rende i Bulletproofs particolarmente interessanti per applicazioni in cui la trasparenza e la fiducia distribuita sono requisiti fondamentali. La loro adozione nella blockchain di Monero ne testimonia la validità, consentendo un efficace offuscamento delle transazioni e un significativo miglioramento della privacy, senza sacrificare l'efficienza computazionale.

Dall'analisi complessiva emerge come le dimostrazioni a conoscenza zero costituiscano uno strumento di straordinaria versatilità, capace di coniugare sicurezza crittografica, riservatezza dei dati e efficienza operativa. Esse si configurano come una delle principali risposte alle esigenze di protezione e fiducia richieste dalle moderne infrastrutture digitali, rappresentando il punto d'incontro tra teoria della complessità computazionale e applicazioni pratiche nei sistemi distribuiti. Guardando al futuro, le prospettive di sviluppo di tali protocolli appaiono ampie e promettenti. L'integrazione delle ZKP con tecniche di computazione sicura multi-parte (MPC), con l'intelligenza artificiale e con i sistemi di verifica decentralizzata potrà aprire la strada a nuove soluzioni in ambiti come la gestione sicura dell'identità digitale, la protezione dei dati sensibili in ambito sanitario e finanziario, e la certificazione di processi basati su machine learning in conformità con i principi di privacy-by-design.

Le Zero-Knowledge Proofs incarnano una delle espressioni più avanzate della crittografia contemporanea. Esse permettono di superare la dicotomia tra trasparenza e riservatezza, garantendo che la sicurezza non derivi più dalla segretezza dell'informazione, ma dalla solidità dei principi matematici che la governano. La loro applicazione nei sistemi blockchain e nelle tecnologie emergenti rappresenta non soltanto una conquista teorica, ma un passo decisivo verso una nuova era della sicurezza digitale, fondata sulla fiducia computazionale e sulla conoscenza controllata.

Bibliografia

[1] S. Goldwasser e M. Sipser, «Private coins versus public coins in interactive proof systems,» *Proceedings of the eighteenth annual ACM symposium on Theory of computing,* pp. 59-68, 1986.

[2] «Zero-knowledge proof,» [Online]. Available: https://en.wikipedia.org/wiki/Zero-knowledge_proof.

[3] J. L. Villar, «Zero-Knowledge Proof Notes,» *Data Protection Master in Cyber Security,* pp. 2,9, 2021.

[4] S. Goldwasser, S. Micali e C. Rackoff, «The knowledge complexity of interactive proof systems,» *SIAM Journal on Computing,* vol. 18, n. 1, pp. 186-208, 1989.

[5] L. Babai, «Trading group theory for randomness,» *Proceedings of the seventeenth annual ACM symposium on Theory of computing,* pp. 421-429, 1985.

[6] L. Babai e S. Moran, «Arthur-Merlin Games: a Randomized Proof System, and a Hierarchy of Complexity Classes,» *Journal of Computer and System Sciences,* vol. 36, n. 2, pp. 254-276, 1988.

[7] L. Margara e A. Farneti, «I protocolli Zero-Knowledge,» 2021, pp. 12, 20, 28.

[8] B. Schoenmakers, «Lecture Notes Cryptographic Protocols,» p. 48, 2022.

[9] V. Iovino, «An introduction to Zero Knowledge Proofs,» 2019.

[10] R. J. F. Cramer, «Modular Design of Secure yet Practical Cryptographic Protocols,» p. 19, 1997.

[11] D. Kogan, «Lecture 6: Sigma Protocols, Secret Sharing,» p. 2, 2019.

[12] D. Wong, «Real-World Cryptography,» pp. 28-31, 134-138, 2021.

[13] D. Giuseppe, «The Fiat-Shamir Transform,» p. 13, 2022.

[14] M. Bellare e P. Rogaway, «ACM Conference on Computer and Communications Security,» pp. 62-73, 1993.

[15] A. Montinaro e P. Rizzo, «Funzione Hash,» *Introduzione alla Crittografia,* p. 68, 2019.

[16] M. Blum, S. Micali e P. Feldman, «Non-interactive zero-knowledge and its applications,» pp. 329-349, 2019.

[17] M. Blum, S. Micali, A. D. Santis e G. Persiano, «NonInteracative Zero-Knowledge,» vol. 20, n. 6, pp. 1084-1118, 1991.

[18] «Fiat–Shamir heuristic,» [Online]. Available: https://en.wikipedia.org/wiki/Fiat-Shamir_heuristic.

[19] S. Krenn e M. Orrù, «Proposal: Σ-protocols,» p. 13, 2021.

[20] I. Damgård, «On Σ-Protocols,» vol. 2, pp. 8-9, 2010.

[21] S. Rajput, A. Singh, S. Khurana, T. Bansal e S. Shreshtha, «Blockchain Technology and Cryptocurrenices,» in *2019 Amity International Conference on Artificial Intelligence (AICAI)*, 2019, pp. 909-912.

[22] Z. Zheng, S. Xie, H. Dai, X. Chen e W. Huaimin, «An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends,» in *2017 IEEE International Congress on Big Data (BigData Congress)*, 2017, pp. 557-564.

[23] M. J. Rennock, A. Cohn e J. R. Butcher, «Blockchain technology,» *The Journal,* vol. 1, n. 7, pp. 1-11, 2018.

[24] E. Tijan, S. Aksentijević, K. Ivanić e M. Jardas, «Blockchain Technology Implementation in Logistics,» *Sustainability,* vol. 11, n. 4, 2019.

[25] Shiksha, «Structure of a Block in Blockchain,» 9 Agosto 2023. [Online]. Available: https://www.shiksha.com/online-courses/articles/structure-of-a-block-in-blockchain/.

[26] S. Omohundro, «Cryptocurrencies, smart contracts, and artificial intelligence,» *AI Matters,* vol. 1, n. 2, pp. 19-21, 2014.

[27] Z. Zibin, X. Shaoan, D. Hong-Ning, C. Xiangping e W. Huaimin, «Blockchain challenges and opportunities: a survey,» *International Journal of Web and Grid Services,* vol. 14, n. 4, pp. 352-375, 2018.

[28] B. Academy, «Binance,» 12 Dicembre 2018. [Online]. Available: https://academy.binance.com/it/articles/positives-and-negatives-of-blockchain.

[29] S. Nakamoto, «Bitcoin: A Peer-to-Peer Electronic Cash System,» 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[30] Bitpanda, «The Bitcoin Whitepaper simply explained,» [Online]. Available: https://www.bitpanda.com/academy/en/lessons/the-bitcoin-whitepaper-simply-explained/.

[31] M. Trust, «What is the RGB Protocol on Bitcoin?,» [Online]. Available: https://trustmachines.co/learn/what-is-the-rgb-protocol-on-bitcoin/.

[32] Ajian, Cyberorange, J. Pow, Shawn e DaPangDun, «Nervos Talk,» 13 Febbraio 2024. [Online]. Available: https://talk.nervos.org/t/rgb-protocol-light-paper/7733.

[33] Criptosociety, «Cos'è il protocollo RGB di Bitcoin,» 21 Dicembre 2023. [Online]. Available: https://criptosociety.net/protocollo-rgb-bitcoin-contratti-intelligenti/.

[34] A. Chiesa, Y. Hu, M. Maller, P. Mishra, N. Vesely e N. Ward, Marlin: Preprocessing zksnarks with universal and updatable srs., Cryptology ePrint Archive, 2019.

[35] M. Maller, S. Bowe, M. Kohlweiss e S. Meiklejohn, «Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings,» in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, London, United Kingdom, Association for Computing Machinery, 2019, p. 2111–2128.

[36] T. Xie, J. Zhang, Y. Zhang, C. Papamanthou e D. Song, «Libra: Succinct zero-knowledge proofs with optimal prover computation,» in *Advances in Cryptology*, Cham, Springer International Publishing, 2019, pp. 733-764.

[37] A. Gabizon, Z. J. Williamson e O. Ciobotaru, «Plonk: Permutations over lagrangebases for oecumenical noninteractive arguments of knowledge.,» vol. 2019, p. 953, 2019.

[38] E. Ben-Sasson, A. Chiesa, M. Riabzev, N. Spooner, M. Virza e N. P. Ward, «Aurora: Transparent succinct arguments for R1CS,» in *Advances in Cryptology*, Cham, Springer International Publishing, 2019, pp. 103-128.

[39] E. Ben-Sasson, I. Bentov, Y. Horesh e M. Riabzev, «Scalable, transparent, and post-quantum secure computational integrity.,» *IACR Cryptol. ePrint Arch.,* vol. 2018, p. 46, 2018.

[40] A. Chiesa, D. Ojha e N. Spoone, «Fractal: Post-quantum and transparent recursive proofs from holography.,» in *Advances in Cryptology*, Cham, Springer International Publishing, 2020, pp. 769-793.

[41] S. Setty, «Spartan: Efficient and general-purpose zksnarks without trusted setup.,» in *Advances in Cryptology*, Cham, Springer International Publishing, 2020, pp. 704-737.

[42] S. Bowe, J. Grigg e D. Hopwood, «Halo: Recursive proof composition without a trusted setup.,» *IACR Cryptol. ePrint Arch.,* vol. 2019, p. 1021, 2019.

[43] S. Patel, G. Persiano e K. Yeo, «Leakage cell probe model: Lower bounds for key-equality mitigation in encrypted multi-maps.,» *IACR Cryptol. ePrint Arch.,* vol. 2019, p. 1132, 2019.

[44] R. Gennaro, C. Gentry, B. Parno e M. Raykova, «Quadratic Span Programs and Succint NIZKs without PCPs,» in *Advances in Cryptology -- EUROCRYPT 2013*, vol. 7881, Heidelberg, Berlin, Springer Berlin Heidelberg, 2013, pp. 626-645.

[45] B. Parno, J. Howell, C. Gentry e M. Raykova, «Pinocchio: Nearly Practical Verifiable Computation,» in *Symposium on Security and Privacy*, Berkeley, CA, USA, IEEE, 2013, pp. 238-252.

[46] E. Ben Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer e M. Virza, «Zerocash: Decentralized Anonymous Payments from Bitcoin,» in *Symposium on Security and Privacy*, IEEE, 2014, pp. 459-474.

[47] A. Nitulescu, «zk-SNARKs: A Gentle Introduction,» pp. 14, 23-25, 32-33, 2020.

[48] V. Buterin, «Quadratic Arithmetic Programs: from Zero to Hero,» 10 Dicembre 2016. [Online]. Available: https://medium.com/@VitalikButerin/quadratic-arithmetic-programs-from-zero-to-hero-f6d558cea649.

[49] «Schwartz–Zippel lemma,» [Online]. Available: https://en.wikipedia.org/wiki/Schwartz-Zippel_lemma.

[50] J. Groth, «On the Size of Pairing-Based Non-interactive Arguments,» *Proceedings, Part II, of the 35th Annual International Conference on Advances in Cryptology,* vol. II, n. 9996, pp. 305-326, 2016.

[51] M. Binello, «The Zero Knowledge Blog,» 2019. [Online]. Available: https://www.zeroknowledgeblog.com/index.php/groth16.

[52] «Zcash: Privacy-protecting digital currency,» 2023.

[53] Electric Coin Company, «Technical explainer: Halo on Zcash,» 2021. [Online]. Available: https://electriccoin.co/blog/technical-explainer-halo-on-zcash/#:~:text=We%E2%80%99re%20proud%20to%20introduce%20ZIP,deployment%20of%20the%20%208.

[54] L. T. Thibault, T. Sarry e A. S. Hafid, «Blockchain Scaling Using Rollups: A Comprehensive Survey,» *IEEE Access,* vol. 10, pp. 93039-93054, 2022.

[55] B. Yee, D. Song, P. McCorry e C. Buckland, «Shades of Finality and Layer 2 Scaling,» *Cryptography and Security,* 2022.

[56] R. Lavin, X. Liu, H. Mohanty, L. Norman, G. Zaarour e B. Krishnamachari, «A Survey on the Applications of Zero-Knowledge Proofs,» *Cryptography and Security,* 2024.

[57] M. Labs, «zksync era docs,» 2023.

[58] StarkWare, «Native account abstraction: Opening blockchain to new possibilities,» 2023.

[59] Scroll, «Ethereum & scroll differences,» 2023.

[60] Starknet, «Cairo and sierra,» 2023.

[61] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille e G. Maxwell, «Bulletproofs: Short Proofs for Confidential Transactions and More,» in *2018 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, 2018, pp. 315-334.

[62] P. Team, «Panther Academy,» PantherProtocol.io, 16 Settembre 2022. [Online]. Available: https://blog.pantherprotocol.io/bulletproofs-in-crypto-an-introduction-to-a-non-interactive-zk-proof.

[63] D. Chaum, R. L. Rivest e A. T. Sherman, «Blind signatures for untraceable payment,» in *Advances in Cryptology*, Boston, MA, Chaum, David, 1983, pp. 199-203.

[64] G. G. Dagher, B. Buenz, J. Bonneau, J. Clark e D. Boneh, «Provisions: Privacy-preserving proofs of solvency for Bitcoin exchanges,» 2015.

[65] R. Canetti, B. Riva e G. N. Rothblum, «Practical Delegation of Computation Using Multiple Servers,» in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, Chicago, Illinois, USA, Association for Computing Machinery, 2011, pp. 445-454.

[66] «Monero,» [Online]. Available: https://en.wikipedia.org/wiki/Monero.

[67] «Moneropedia,» [Online]. Available: https://web.getmonero.org/resources/moneropedia.

Indice figure