Alma Mater Studiorum · Università di Bologna

Scuola di Ingegneria e Architettura
Dipartimento di Informatica · Scienza e Ingegneria · DISI
Corso di Laurea in Ingegneria Informatica

SISTEMI ANTI-DENIAL OF SERVICE IN AMBIENTI ANONIMI BASATI SU ZK-SNARK

Relatore:
Prof. Paolo Bellavista

Presentata da:
Straccali Leonardo

Anno Accademico 2022/2023

Indice

Elenco delle figure

Introduzione

L'implementazione di strumenti atti a combattere gli attacchi di tipo **denial of service**, sono da molto tempo materia di studio e di ricerca. Questi attacchi possono rappresentare una minaccia grave, soprattutto quando si verificano su servizi cruciali e sensibili. Per tale motivo, negli anni sono state sviluppate molte strategie diverse per impedire a singoli o ai gruppi di computer (noti anche come "zombie armies") di attaccare servizi o reti. Attualmente una delle metodologie di protezione più comuni agli attacchi **DoS** è il **Rate-limiting**, che permette di imporre una frequenza massima accettabile per le richieste, al fine di evitare sovraccarichi e congestionamenti delle risorse. In particolare, nei sistemi su larga scala, la limitazione della frequenza rappresenta uno strumento essenziale per garantire la disponibilità e l'integrità delle risorse e dei servizi.

Un altro campo di interesse in cui si sono fatti molti progressi negli ultimi anni, grazie soprattutto all'avvento delle tecnologie blockchain, è stato quello dell'anonimato in rete. Con il termine anonimato in rete ci si riferisce alla condizione in cui, sulla base di una conoscenza parziale o totale delle interazioni di un utente in una rete, non è possibile risalire all'identità dell'utente stesso; permettendo un'interazione con i servizi offerti dalla rete in assoluta riservatezza. L'anonimato in rete presenta molteplici vantaggi, soprattutto in contesti in cui la privacy costituisce un requisito fondamentale, come ad esempio nelle votazioni o nelle transazioni finanziarie. Inoltre, la possibilità di mantenere l'anonimato può rivelarsi altrettanto utile in ambiti più diffusi, come le conversazioni online o i social network, garantendo la libertà di espressione e la tutela della propria sfera privata.

Tuttavia, l'anonimato in rete presenta anche alcune criticità, tra cui la principale è rappresentata dalla difficoltà di controllo. La ragione di questa difficoltà è da individuare nel fatto che le metodologie di sicurezza a livello applicazione (pila ISO/OSI) generalmente si basano sull' analisi del comportamento degli utenti o dei dispositivi, nel corso del tempo, al fine di rilevare i pattern di attività che potrebbero suggerire la presenza di un attacco. In un contesto anonimo, in cui le identità degli utenti non sono tracciabili, ciò diviene notevolmente più arduo. Esistono strumenti di rate-limiting efficaci a livelli inferiori della pila ISO/OSI, come a livello di trasporto o di sessione. Tuttavia, la loro attuazione comporta alcuni svantaggi, come il rischio di limitare o bloccare numerosi indirizzi IP tradotti sotto una NAT, minare la privacy degli utenti disattivando protezioni utili come TLS o ancora essere elusi da tecniche di mascheramento, come l'IP spoofing. Per tali motivazioni, per un servizio diffuso in ambiente anonimo, non è possibile fare affidamento esclusivamente su questi strumenti.

La presente tesi affronta la discussione di un protocollo chiamato **RLN (Rate-Limiting Nullifier)**[1] basato sulla tecnologia **zk-SNARK**, che permette di attuare un rate-limiting in ambiente anonimo. Il protocollo è composto da tre parti generali, le quali si differenziano, nei dettagli, a seconda del dominio applicativo. Fasi del protocollo:

- **Registrazione**: Durante questa fase, gli utenti che desiderano accedere al servizio devono registrarsi fornendo una prova del possesso di determinati requisiti. Questa prova di possesso è denominata "identity commitment" e viene conservata e utilizzata nelle fasi successive del protocollo. I dati personali che soddisfano i requisiti sono definiti "stake" e possono assumere forme diverse a seconda del contesto applicativo. Ad esempio, possono essere costituiti da un profilo su un social network, dall'indirizzo di un portafoglio di criptovalute o da un'identità digitale come lo SPID o la CIE.

  Gli stake non vengono conservati dal protocollo. Alla fine della fase di registrazione, il sistma è a conoscenza unicamente nel fatto che un nuovo utente anonimo che soddisfa le specifiche della registrazione è stato aggiunto al servizio. La presenza di uno stake non è strettamente necessaria per la registrazione. In effetti, è possibile registrarsi semplicemente creando un codice univoco e fornendolo come identity commitment. Tuttavia, l'utilizzo di uno stake risulta essere molto utile al fine di prevenire attacchi Sybil, ovvero la generazione di numerosi utenti malevoli da parte di un attaccante.

- **Interazione**: Dopo essersi registrati, gli utenti hanno la possibilità di interagire con il servizio attraverso l'invio di richieste. Ad ogni richiesta, gli utenti sono tenuti a fornire una prova di appartenenza al sistema. Tale prova è generata tramite la tecnologia zk-SNARK, che consente di dimostrare al servizio che l'utente è effettivamente un membro legittimo senza rivelare alcuna informazione sulla sua identità. Inoltre, il protocollo consente di implementare una regola di rate-limiting, che, se non rispettata, porta l'utente a rivelare la propria stake. Ciò consente di scoprire e gestire chi attua comportamenti di spam o Dos, e di procedere alla fase successiva.

- **Punizione**: La tipologia e il grado di punizione dipendono molto dal contesto applicativo. Ad esempio, alcune punizioni o misure di "slashing" possono comportare la rimozione del membro dal gruppo, con conseguente impossibilità di interagire con il sistema e perdita dell'anonimato. In presenza di una stake, è possibile prevedere sanzioni più articolate, come l'inserimento dell'identità virtuale in un registro di esclusione per altri servizi, o la rivelazione dell'indirizzo del portafoglio criptovalute dell'utente e il sequestro dei fondi.

Di seguito si cercherà di guardare a tutto tondo le tecnologie e le idee alla base di zk-SNARK e del protocollo RLN. Inoltre si mostrerà l'implementazione di un piccolo prototipo che utilizza questo protocollo per evitare il sovraccarico di risorse in un servizio basato su API.

[1] `https://rate-limiting-nullifier.github.io/rln-docs/`

Capitolo 1

Stato dell'arte

1.1 Denial-of-Service (DoS)

Gli attacchi di tipo Denial-of-Service (DoS) sono una tipologia di attacchi informatici estremamente diffusi, la cui frequenza è cresciuta negli anni sia in termini quantitativi che qualitativi. Questo incremento è stato determinato dalla relativa facilità con cui è possibile condurre un attacco di questo tipo, nonché dalle sue conseguenze, che in base alle contromisure adottate e alla struttura del servizio attaccato possono essere più o meno gravi. Nella tesi ci concentreremo solamente, sull'analisi della metodologia di protezione denominate rate-limiting. Queste strategie possono risultare estremamente efficaci in alcune situazioni, come dimostrato dall'evento accaduto ad agosto 2022 a Google [1], che grazie all'attivazione di un rate-limiter è stato in grado di contrastare con successo il più grande attacco di tipo DDoS a livello applicativo mai registrato. Tale attacco, proveniente da oltre 5.256 sorgenti IP dislocate in 132 paesi differenti, ha raggiunto il picco di 46 milioni di richieste al secondo.

In generale, un una regola per il rate limiting consiste in un semplice conteggio delle occorrenze delle richieste in un lasso di tempo. Tuttavia, esistono diverse tecniche per misurare e limitare la frequenza di tali richieste, ognuna con i propri usi e implicazioni.

- **Fixed window**: Tra tutte le strategie che vedremo è la più semplice da implementare, consiste nello stabilire un limite massimo al numero di richieste che possono essere inviate in un determinato intervallo di tempo, detto "finestra". Ad esempio, si potrebbe limitare il numero di richieste a 100 ogni minuto. Questo limite viene applicato in modo uniforme all'interno della finestra temporale. Ciò significa che, una volta raggiunto il limite massimo di richieste consentite, l'utente o l'applicazione deve attendere il termine della finestra temporale per poter inviare nuove richieste. Uno svantaggio significativo di questa strategia è la possibile concentrazione di richieste tutte in una porzione della finestra, rischiando un sovraccarico del servizio. Caso notevole è quello in cui si concentrino tutte le richieste di una finestra al margine della fine e tutte le richieste della finestra successiva al margine dell'inizio questo comporta che il sistema in una durata equivalente a quella della finestra ha

ricevuto il doppio delle richieste consentite.

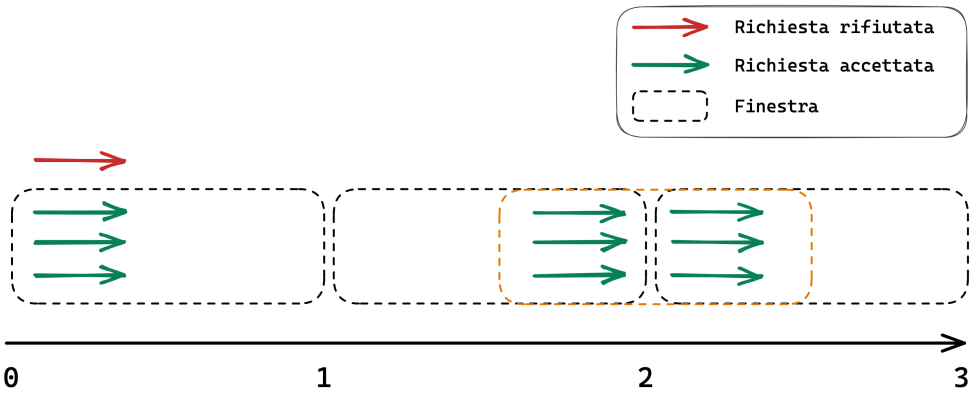

Figura 1.1: Diagramma funzionamento della strategia Fixed window

- **Token bucket**: Il funzionamento del token bucket prevede che non tutte le richieste di un servizio vengano mappate 1:1 con la richiesta di risorse, poiché alcune richieste potrebbero richiedere più risorse di altre. Per questo motivo, viene mantenuto un contatore di risorse, che viene scalato per ogni richiesta, del numero di token necessari per portare a termine il lavoro. Il contatore ha una frequenza di riempimento, ovvero a ogni unità di tempo viene reimpostato al suo valore massimo. Quando un utente o un'applicazione invia una richiesta al sistema, il sistema verifica se ci sono abbastanza token disponibili per soddisfare la richiesta. Se non ci sono abbastanza token, la richiesta viene respinta. In questo modo, le richieste vengono accettate solo se c'è abbastanza capacità per soddisfarle.

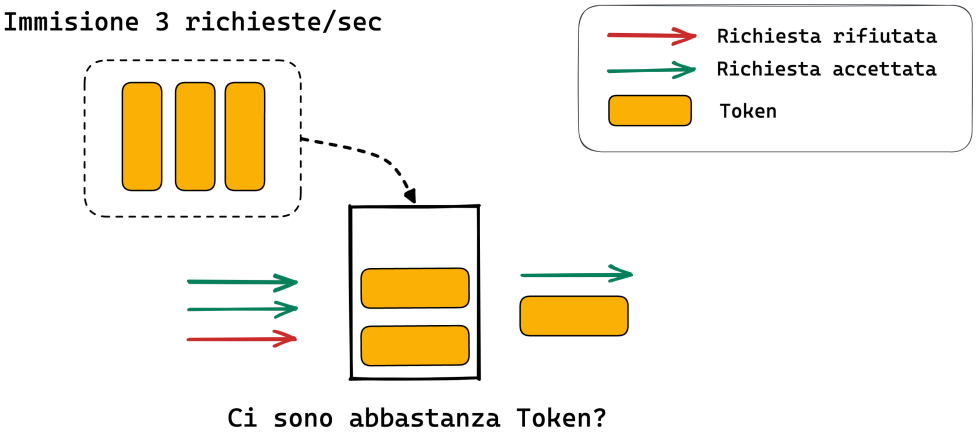

Figura 1.2: Diagramma funzionamento della strategia Token bucket

- **Leaky bucket**: L'idea di base è simile a quella del Token bucket ma invece di rispondere al numero di richieste liberando i token necessari fino a esaurimento, il tasso di elaborazione delle richieste viene regolato in modo uniforme. Questo significa che quando un pacchetto di dati arriva al sistema, se il secchio è già pieno, il pacchetto viene scartato. Nel mentre ad ogni unità di tempo viene elaborata una quantità di richieste corrispondente alla velocità di uscita del

sistema. In questo modo, il flusso di dati in uscita non supera mai una certa soglia.

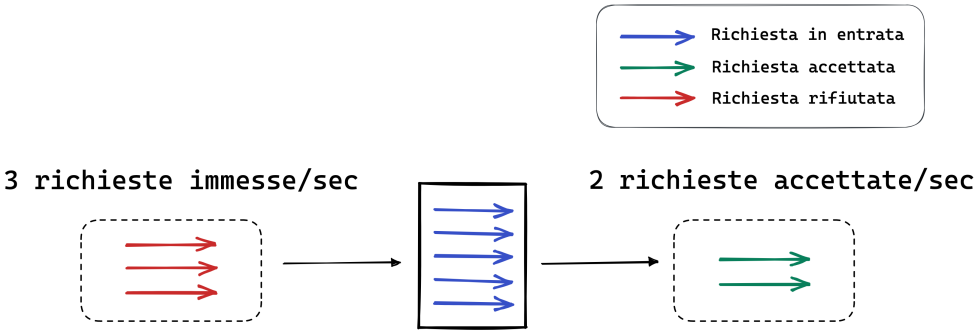

Figura 1.3: Diagramma funzionamento della strategia Leaky bucket

- **Sliding window**: La strategia di sliding window adotta un approccio simile alla Fixed Window, ma con una differenza fondamentale: esamina il tasso di richieste effettuate in un periodo di tempo continuo piuttosto che in intervalli fissi. Ad esempio, se il limite di richieste è fissato a 100 al minuto, la strategia prevede di controllare il numero di richieste effettuate nell'ultimo minuto e, nel caso superi il limite, rifiutare la richiesta. Questo approccio evita il potenziale sovraccarico del sistema alla fine di una finestra con un tempo prefissato, ma richiede la gestione di una finestra scorrevole, aumentando la complessità di implementazione.

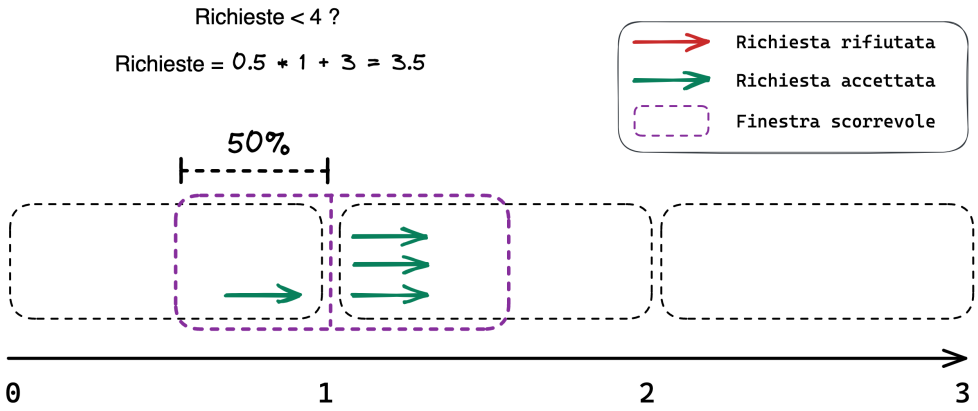

Figura 1.4: Diagramma funzionamento della strategia Sliding window

Vale la pena notare che pur essendo un metodo ampiamente utilizzato il rate-limiting da solo potrebbe risultare insufficiente per garantire un livello di sicurezza adeguato contro gli attacchi DoS, pertanto, è necessario adottare contromisure di diversa natura al fine di garantire una protezione completa.

1.2 zk-SNARK

L'acronimo zk-SNARK rappresenta l'espressione completa "Zero-Knowled Succinct Non-Interactive Argument of Knowledge". Questa tecnologia è stata introdotta per la prima volta in un articolo scientifico pubblicato nel 2013 intitolato "Pinocchio: Nearly Practical Verifiable Computation". Da allora, sono state apportate diverse migliorie alla tecnologia e sono emerse numerose applicazioni in svariati settori. Le prime applicazioni significative di zk-SNARK sono state implementate nel contesto della Blockchain, che rimane il settore dove la tecnologia è più conosciuta e applicata, per fare alcuni esempi Etherium una delle Blockchain più accreditate ha iniziato ad implementare la tecnologia nella sua rete dal 2016, e il suo fondatore Vitalik Buterin ha scritto in un articolo sull'argomento: "Perhaps the most powerful cryptographic technology to come out of the last decade is general-purpose succinct zero knowledge proofs, usually called zk-SNARKs" (Forse la tecnologia crittografica più potente emersa nell'ultimo decennio è quella chiamata zk-SNARK).

Zk-SNARK, come suggerisce il nome, rappresenta una tecnologia fondata sul protocollo crittografico zero-knowledge proof, il quale consente a una parte (il dimostratore) di dimostrare a un'altra (il verificatore) la veridicità di un'affermazione senza rivelare nessuna informazione ulteriore. Inoltre questo processo di dimostrazione, viene effettuato in modo da ottenere una prova, in cui sia la dimensione della prova, che il tempo necessario per verificarla crescono molto più lentamente rispetto al calcolo da verificare e senza la necessità di interazione bidirezionale tra le due parti coinvolte.

Di seguito esamineremo meglio le singole parti che compongono la tecnologia zk-SNARK, analizzando i suoi componenti chiave per ottenere una panoramica completa e concisa.

Prima di addentrarci nella descrizione delle componenti che utilizzeremo, è importante introdurre il campo di applicazione dei teoremi e delle proprietà che andremo a utilizzare. In particolare, nei prossimi paragrafi lavoreremo in un dominio di campi finiti, che è un dominio algebrico dove l'insieme dei numeri è finito e rispetta determinate proprietà algebriche come l'addizione, la sottrazione, la moltiplicazione e la divisione. Per le loro proprietà i campi finiti svolgono un importante ruolo in diversi algoritmi crittografici.

1.2.1 Zero-Knowledge

Come accennato precedentemente, Zero-Knowledge proof è un protocollo crittografico in cui una parte dimostra di conoscere una determinata informazione a un'altra parte, senza rivelare alcuna informazione aggiuntiva su di essa. Per esempio, se Alice vuole dimostrare a Bob di conoscere la password del suo account, senza rivelargli la password stessa, può utilizzare un protocollo di tipo Zero-Knowledge. In questo modo, Alice può dimostrare a Bob di sapere qual è la password corretta senza rivelarla, proteggendo così la sua privacy e la sicurezza del suo account. Come è possibile tutto questo? grazie a diverse tecniche crittografiche e tanta matematica.

Formalmente le prove di tipo Zero-Knowledge non sono dimostrazioni di carattere

matematico, ma probabilistiche, il che significa che c'è sempre una probabilità che un dimostratore scorretto riesca a dimostrare la veridicità di un'affermazione a un verificatore onesto. Esistono tuttavia tecniche per ridurre questa probabilità a valori piccoli a piacere.

Il primo articolo che definisce il costrutto è "The Knowledge Complexity of Interactive Proof-Systems"[2] pubblicato nel 1985, dove gli autori introducono il concetto di "Zero-Knowledge proof" come un tipo di "interactive proof systems[3]" (un modello computazionale che simula lo scabio di messagi tra due individui) in cui il verificatore non apprende nulla oltre alla verità dell'affermazione che viene dimostrata.

Per poter costruire un sistema Zero-Knowledge proof per una particolare affermazione abbiamo bisogno che la prova generata soddisfi le seguenti prorpietà :

- **Completezza**: Se l'affermazione da dimostrare è vera, allora un verificatore onesto (cioè che segue correttamente le regole del protocollo) verrà convinto della veridicità da un dimostratore onesto.

- **Correttezza**: Se l'affermazione è falsa, nessun dimostatore disonesto può convincere un verificatore onesto che essa è vera, se non con una piccola probabilità.

- **Zero-knowledge**: se l'affermazione è vera, nessun verificatore apprende altro se non il fatto che l'affermazione è vera.

Le tre proprietà appena viste descrivono Zero-Knowledge proof da un punto di vista formale ma per avere una visione più intutiva del protocollo è utile vedere il flusso di funzionamento attraverso un esempio. L'esempio in questione è tratto da un famoso articolo di Jean-Jacques Quisquater "How to Explain Zero-Knowledge Protocols to Your Children"[4] ne estrapolerò solo le parti esenziali alla tratazione, ma ne consiglio la lettura.

L'esempio riguarda una caverna a forma di anello, nella quale è posta a metà del percoso una porta che impedisce il completamento del tragitto, a meno di non conoscere una parola segreta. Supponiamo l'esistenza di due parti, Peggy - che conosce il segreto della porta e agirà da dimostratrice - e Victor - che invece non conosce il segreto e sarà il verificatore. Peggy vuole dimostrare a Victor di sapere come superare la barriera senza però rivelare il segreto. Per fare ciò, Peggy propone a Victor di seguire una strategia: dapprima stabiliscono un nome per identificare i due percorsi (ad esempio, A e B), poi Victor rimane fuori dalla caverna mentre Peggy sceglie uno dei due percorsi. Dopo qualche minuto, Victor entra nella caverna e chiama ad alta voce il nome di uno dei due percorsi, a quel punto Peggy dovrà uscire dal percorso chiamato da Victor. Victor accetta la proposta, ma con una condizione: il processo dovrà essere ripetuto più volte. Dopo diversi tentativi, Victor si convince che Peggy conosca effettivamente il segreto per superare la barriera. Possiamo calcolare questa probabilità utilizzando la distribuzione binomiale, dove la probabilità di successo p è del 50% e il numero totale di prove n è 10. La probabilità di indovinare correttamente tutte le 10 scelte di Victor è data dalla seguente formula: $P = p^n = 0,5^{10} = 0,0009765625$

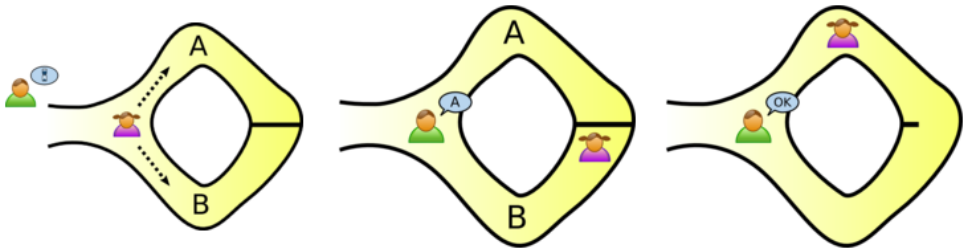

Figura 1.5: Disegno della caverna di Ali Baba. Jean-Jacques Quisquater

L'esempio proposto è molto efficace nel far comprendere come sia possibile dimostrare il possesso o la conoscenza di un informazione senza rivelarla. Nell'articolo citato inoltre si prevendono diversi scenari in cui una o entrambe le parti potrebbero essere disoneste. Per gestire tali situazioni, è possibile applicare una procedura denominata "trusted setup", che verrà approfondita nelle sezioni successive.

1.2.2 Succinct

Una dimostrazione di tipo succinct che potremmo tradurre con la parola concisa, è una prova in cui sia la dimensione della prova che il tempo necessario per verificarla crescono molto più lentamente rispetto alla computazione da verificare. Ad esempio se Alice volesse provare a Bob di possedere un array composto da un milione di elementi, dove ogni elemento è uguale all'indice dell'array più uno.

$$a = [1, 2, 3, ..., 1000000] \tag{1.1}$$

Una prova di tipo "succinct", così come definita, non può essere realizzata attraverso un processo di controllo "uno per uno" degli elementi dell'array. Questo perché, se così fosse, si otterrebbe un processo che richiederebbe tanto tempo e spazio di memoria quanto il calcolo effettuato per generare l'array. Una possibile miglioria per il calcolo puntuale potrebbe essere quella di campionare l'array in un punto arbitrario e controllare se l'elemento selezionato rispetta la regola. Dopo un singolo campionamento, il grado di fiducia di Bob rispetto ad Alice sarebbe di soli 0,0001%, ovvero un milionesimo. Se in uno qualsiasi dei campionamenti effettuati Bob dovesse prelevare un numero non congruo, la dimostrazione verrebbe invalidata completamente senza bisogno di ulteriori controlli. È possibile aumentare il grado di fiducia del verificatore eseguendo più controlli, ma anche in questo caso, le prove generate da dimostratori disonesti in cui uno o pochi elementi sono errati all'interno dell'intera prova richiederebbero un numero eccessivo di passaggi per raggiungere un grado di fiducia accettabile, rendendo cosi questo secondo approccio fragile e non attuabile.

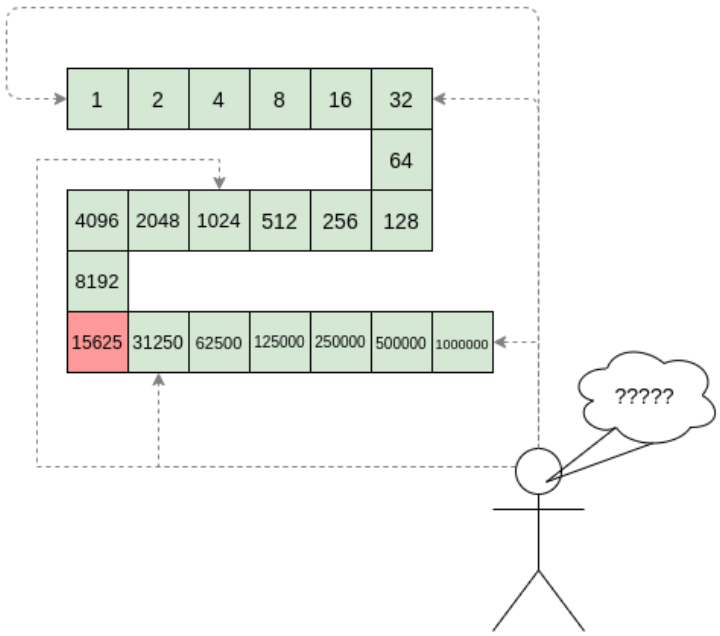

Figura 1.6: Disegno esempio di valutazione a tentativi fallita

Polinomi

Per trovare una soluzione al problema, è necessario fare riferimento ai polinomi, in particolare alla tecnica crittografica nota come "polynomial commitments". Questa tecnica ci consente di generare delle funzioni hash per i polinomi, chiamate "polynomial commitment", sulle quali è ancora possibile effettuare operazioni algebriche. Ciò significa che possiamo eseguire operazioni sui polinomi senza conoscerli, attraverso i commitment. Inoltre, l'uso di questa tecnica ci fornirà notevoli vantaggi durante la fase di verifica. Infatti la possibilità di verificare le informazioni del dimostratore senza dover operare un controllo puntuale è possibile grazie alla proprietà descritta dal lemma di Schwartz-Zippel, un risultato importante in teoria della complessità computazionale e della teoria degli algoritmi che stabilisce una condizione sufficiente per determinare se un polinomio multivariato non nullo ha radici in un campo finito.

Per comprendere come questo strumento possa esserci utile, partiamo da un equivalenza intuitiva: se abbiamo due polinomi $f(x_1, ..., x_n)$ e $g(x_1, ..., x_n)$, chiedersi se $f \equiv g$ è equivalente a chiedersi se

$$p(x_1, ...x_n) = f(x_1, ...x_n) - g(x_1, ...x_n) \equiv 0 \tag{1.2}$$

L'intuizione su cui possiamo basarci per capire l'utilità del lemma, è che Bob (il verificatore) abbia il polynomial commitment $g(x_1, ..., x_n)$ del polinomio corretto $f(x_1, ..., x_n)$ e voglia verificare che Alice (il dimostratore) lo conosca. Possiamo per esempio immaginare un semplice protocollo dove:

1. Bob sceglie un punto qualsiasi s e valuta il suo polynomial commitment in s, $g(s_1, ..., s_n)$ 2. Bob invia s a Alice che provvederà a valutare il suo polinomio in s, $f(s_1, ..., s_n)$ 3. Alice invia il risultato della sua valutazione a Bob che la confronta

con il suo valore, se i valori risultato uguali Bob si convince che nel punto s Alice consce il corretto polinomio

In questa fase ci potremmo chiedere quale beneficio abbiamo ottenuto passando dalla formulazione dell'array in cui venivano fatti campionamenti casuali, alla formulazione dei polinomi in cui vengono fatte valutazioni del polinomio in variabili casuali. Il beneficio è dovuto al lemma di Schwartz-Zippel, che permette di dimostrare che:

prendendo un polinomio $p(x_1, ..., x_n)$ di grado $d > 1$ con coefficienti in un campo finto $\mathbb{K}$ e prendendo $S \subset \mathbb{K}$ e $r_1, ..., r_n \in S$ scelti in modo arbitrario, allora se $p(x_1, ..., x_n)$ non è il polinomio nullo abbiamo che $p(r_1, ..., r_n) = 0$ con probabilità $\leq d/|S|$

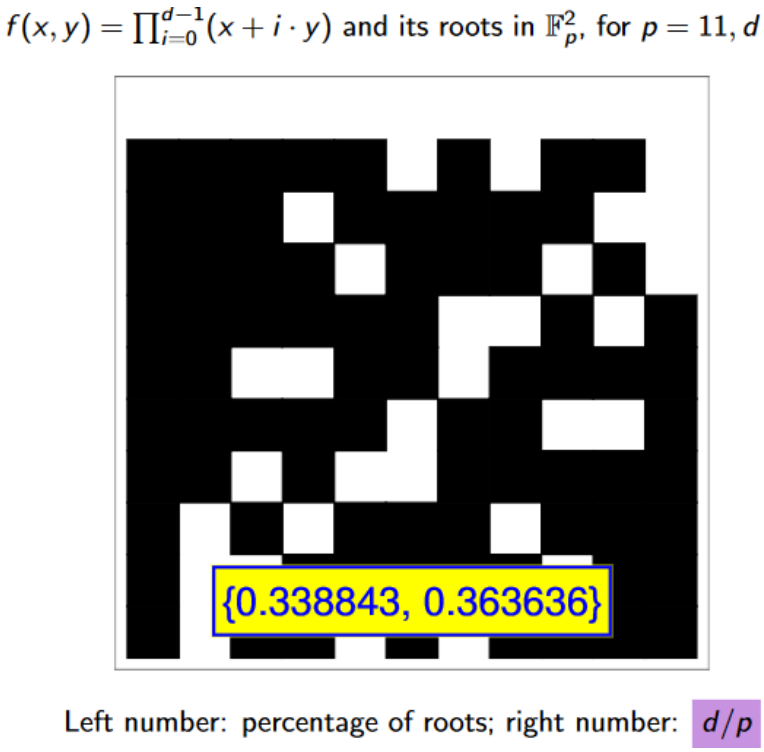

Figura 1.7: Esempio applicazione del lemma di Schwartz-Zippel[5]

la conseguenza del lemma è che la probabilità di trovare una radice del polinomio in un gruppo di valori appartenenti al campo è inferiore o uguale al rapporto tra il grado del polinomio e la cardinalità del gruppo di elementi selezionati, operazione molto più veloce da calcolare di un controllo puntuale. Inoltre se decidessimo di ripetere il processo di selezione degli $r_1, ..., r_n \in S$ un numero k di volte otterremo che la probabilità che $p(r_1^k, ..., r_n^k) = 0$ sarebbe obbligatoriamente $\leq (d/|S|)^k$ e per valori di S abbastanza grandi il il rapporto tende molto velocemente a 0. Intuitivamente il lemma ci dimostra che se un'equazione che coinvolge alcuni polinomi è vera in una coordinata selezionata arbitrariamente, allora è quasi certamente vera per i polinomio nel suo insieme.

Come calcolare i polinomi

Una volta compreso il vantaggio nell'affrontare la nostra computazione mediante l'utilizzo di polinomi, possiamo procedere alla descrizione del processo che ci permette di ottenere questi polinomi.

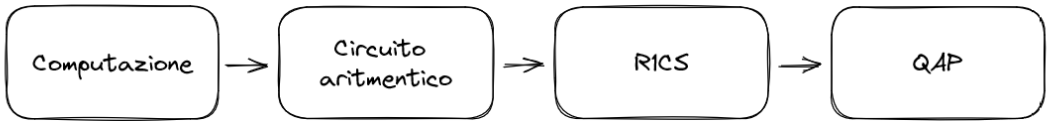

Figura 1.8: Passi da compiere per passare dalla Computazione alla rappresentazione polinomiale

Per illustrare in dettaglio i passaggi che conducono dalla Computazione alla rappresentazione polinomiale desiderata, ovvero la QAP (Quadratic Arithmetic Programs), utilizzerò un compilatore scritto in Rust chiamato "circom"[1]

1. La prima fase, ovvero il passaggio dalla computazione al circuito algebrico, può essere un processo non immediato, soprattutto quando si tratta di computazioni articolate.A titolo di esempio, consideriamo la computazione $2 * x^2 = 18$ con l'ovvia soluzione da dimostrare $x = 3$. Per ottenere un circuito algebrico a partire da questa computazione, si può procedere come segue:

```
pragma circom 2.1.4;

template Multiplier2 () {

   // Declaration of signals.
   signal input x;
   signal output y;

   // Constraints.
   y <== (x**2)*2;
}
```

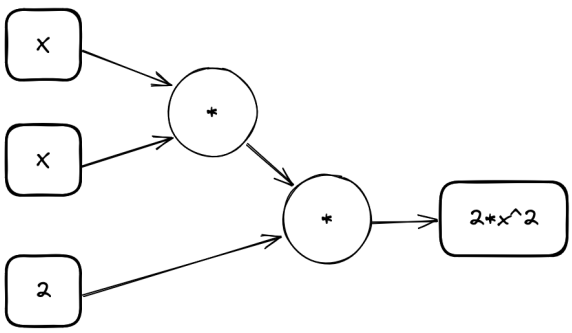

Figura 1.9: Codice esemplificativo per la creazione di un circuito algebrico

2. Ora dobbiamo trasformare il nostro circuito aritmetico in un formato chiamato R1CS, R1CS è un formato in cui ogni vincolo (gate del circuito) viene trasformato in una terna di vettori (a,b,c) e sul quale viene calcolato un vettore chiamo s che rappresenta la soluzione del sistema di vincoli R1CS, il vettore s è costruito in tal modo Successivamente, è necessario trasformare il circuito aritmetico in un formato chiamato R1CS. R1CS è un formato in cui ogni vincolo (gate del circuito) viene

[1] https://docs.circom.io/

rappresentato da una terna di vettori (a, b, c). Viene anche calcolato un vettore chiamato s, che rappresenta la soluzione del sistema di vincoli R1CS. Il vettore s è costruito nel seguente modo:

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

dove con il simbolo $\cdot$ di intende il prodotto scalare. quindi dal circuito precedente possiamo calcolare le terne di vettori (a, b, c) per i due gate

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Possiamo osservare che per ognuno dei due gate del circuito, sono stati creati una terna di vettori (a, b, c). Per quanto riguarda il vettore s, esso viene calcolato a partire dal vettore delle variabili, inserendo al posto di ogni variabile il valore assunto durante la valutazione del circuito. In questo modo, il vettore s rappresenta la soluzione del sistema di vincoli R1CS.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

semplici, mentre "one" è una variabile di sistema utilizzata per effettuare operazioni algebriche. Per verificare che il vettore s sia effettivamente una soluzione del sistema di vincoli R1CS, è possibile calcolare la formula precedente per ogni gate del circuito.

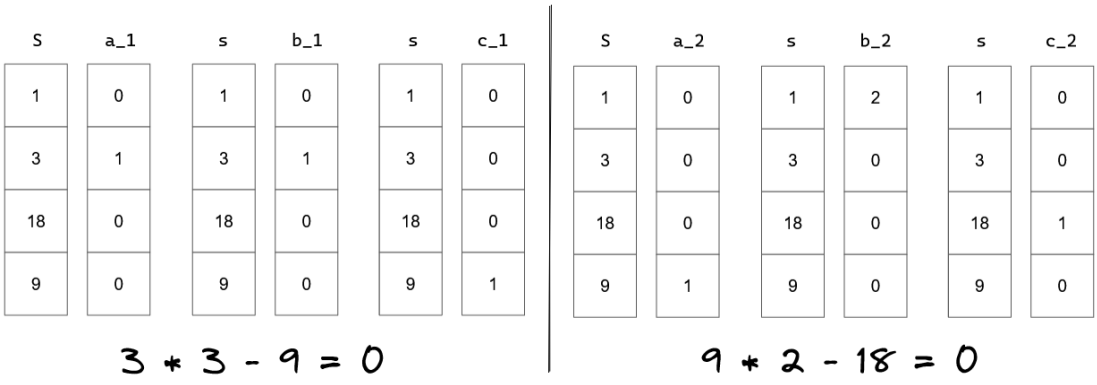

Figura 1.10: Contrllo dei vincoli R1CS

Ovviamente questa operazione di verifica per circuiti con molti gate non può essere percorribile.

3. Il processo di trasformazione dal formato R1CS al formato QAP è il più complesso e richiede l'uso dell'interpolazione di Lagrange. L'idea principale è quella di costruire dei polinomi tali che se valutati nelle coordinate relative ai vincoli, restituiscano i

valori dei vettori corrispondenti. Per il nostro esempio le matrici ottenute dal calcolo sono

$$\text{Ap} = \begin{bmatrix} 0.0 & 0.0 \\ 2.0 & -1.0 \\ 0.0 & 0.0 \\ -1.0 & 1.0 \end{bmatrix} \qquad \text{Bp} = \begin{bmatrix} -2.0 & 2.0 \\ 2.0 & -1.0 \\ 0.0 & 0.0 \\ 0.0 & 0.0 \end{bmatrix} \qquad \text{Cp} = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ -1.0 & 1.0 \\ 2.0 & -1.0 \end{bmatrix}$$

dove ogni riga delle matrici rappresenta i coefficienti di un polinomio, questi polinomi sono costruiti in modo tale che valutando i polinomi in x = 1 (coordinata del primo vincolo) otteniamo i valori corrispondenti ai vettori del primo vincolo

$$\text{Ap} = \begin{bmatrix} 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} \qquad \text{Bp} = \begin{bmatrix} 0.0 \\ 1.0 \\ 0.0 \\ 0.0 \end{bmatrix} \qquad \text{Cp} = \begin{bmatrix} 0.0 \\ 0.0 \\ 0.0 \\ 1.0 \end{bmatrix}$$

grazie a questa formato ora se volessimo verificando i vincoli tramite la formula precedente avremmo

$$A(x) * B(x) - C(x) = P(x)$$

dove $P(x)$ non è obbligatoriamente il polinomio nullo, ma se la dimostrazione è corretta $P(x)$ deve avere delle radici in tutti le coordinate corrispondenti ai vincoli. Per verificarlo non abbiamo bisogno di controllare tutti i vincoli perché lavorando con i polinomi possiamo sfruttare il lemma di Schwartz-Zippel per valutare la condizione molto più velocemente

Valutazioni sul protocollo

Fino ad ora abbiamo descritto un processo che ci consente di trasformare i nostri vincoli numerici in polinomi, permettendoci di calcolare con grande velocità e un elevato grado di certezza la validità dei vincoli imposti. Tuttavia, il protocollo spiegato in precedenza presenta due falle:

1. Arbitrarietà : Nel secondo passaggio del protocollo, Bob condivide con Alice un punto s, che è stato scelto arbitrariamente per valutare il polinomio. Tuttavia, passando s in chiaro ad Alice ci esponiamo al rischio che Alice costruisca un polinomio ad hoc, che soddisfi i vincoli solo nel punto specifico scelto, permettendole così di convincere Bob di una affermazione non vera. Ciò violerebbe il principio di correttezza dei protocolli Zero-Knowledge. 2. Verificabilità : la capacità di Alice di valutare correttamente il polinomio in s non garantisce che essa abbia utilizzato effettivamente il polinomio p per restituire il risultato a Bob, In altre parole, non è garantito che Alice abbia utilizzato il polinomio descritto dal polynomial commitment per raggiungere il risultato atteso.

Per affrontare questi due problemi, vengono utilizzate altre tecniche matematiche, denominate Homomorphic Encryption per risolvere il primo problema e Knowledge of Coefficient (KC) Test per risolvere il secondo. In questa sede, si accennerà brevemente a come è possibile ottenere un Homomorphic Encryption in grado di soddisfare

la proprietà di correttezza, mentre la trattazione del KC Test non verrà approfondita. È possibile trovare una introduzione ad entrambe le tecniche in questa serie di articoli: https://electriccoin.co/blog/snark-explain/.

Homomorphic Encryption

L'idea alla base dell'Homomorphic Encryption è quella di costruire un'operazione simile all'operazione di hashing con la proprietà di poter applicare delle operazioni aritmetiche ai valori criptati senza decifrarli, ovvero ottenere una funzione $E(x)$ che soddisfi tali proprietà :

1. Difficoltà di inversione: Dato un $E(x)$ sia complesso risalire a x 2. Resistenza alle collisioni: se $x \neq y \Rightarrow E(x) \neq E(y)$ 3. Applicazione di operazioni aritmetiche: Se si conoscono E(x) ed E(y), si può generare delle espressioni aritmetiche in x e y. Per esempio, si può calcolare E(x+y) da E(x) ed E(y)

Utilizzando funzioni di questo tipo, è possibile che il Dimostratore e il Verificatore si scambino informazioni criptate che non rivelino il valore in chiaro del contenuto, ma che permettano ancora di effettuare operazioni aritmetiche su di esse.

1.2.3 Non-Interactive

Tutti i protocolli Zero-Knowledge proof esaminati analizzati funo a questo punto erano di tipo interattivo. Questo significa che per generare e verificare la prova, le due parti coinvolte, ovvero il verificatore e il dimostratore, devono interagire in modo bidirezionale. Ad esempio, nel caso della "grotta di Alibaba", Victor, il verificatore, indicava il passaggio da intraprendere e Peggy, il dimostratore, che "rispondeva" uscendo dalla giusta direzione; mentre nel protocollo dei polinomi, il verificatore forniva le coordinate del punto in cui valutare il polinomio e il dimostratore rispondeva con la valutazione corretta. In alcune situazioni tuttavia, può risultare vantaggioso utilizzare le cosiddette NIZKP, ovvero le Non-Interactive Zero-Knowledge proof, in cui il dimostratore trasmette al verificatore una prova contenente tutte le informazioni necessarie per la verifica, senza richiedere ulteriori interazioni tra le parti.

Questa metodologia presenta numerosi vantaggi sia in termini di prestazioni, poiché consente di costruire le prove più velocemente senza dover attendere la risposta e il tempo di comunicazione, sia in termini di sicurezza, in quanto l'assenza di uno scambio di messaggi elimina il rischio di intercettazioni da parte di individui malintenzionati. Inoltre, l'utilizzo di prove NIZKP potrebbe risultare obbligato in quei contesti applicativi in cui la comunicazione in tempo reale non è possibile. L'adozione di questo tipo di prove comporta però un aumento della complessità, poiché, essendo privi di interazione, tutto il lavoro di generazione e preparazione alla verifica viene svolto dal dimostratore e affinché il dimostratore non sia arbitrariamente libero di agire, generando prove scorrette, è necessario vincolarlo al rispetto delle regole.

A tal fine, si utilizza una tecnica nota come "trusted set-up" che consente di generare una Common Reference String (CRS), ovvero un dato pubblico conosciuto sia dal dimostratore che dal verificatore usato per aggiungere casualità all'interno della generazione della prova. Sempre riferendoci ai due esempi precedenti, la CRS ottenuta tramite un trusted set-up consentirebbe di rendere le scelte di Victor nella caverna il più casuali possibile, impedendo ad Alice di individuare dei pattern o a entrambi di collaborare per ingannare un terzo osservatore e nel caso dei polinomi, invece, consentirebbe di rendere aleatoria la scelta del punto in cui valutare il polinomio.

La procedura di costruzione della CRS rappresenta un passaggio critico per la robustezza del protocollo e dipende fortemente dal tipo di cerimonia di trusted set-up scelta.

Modello di affidabilità

Una cerimonia di trusted set-up è una procedura che viene eseguita una sola volta per generare un CRS che poi potrà essere utilizzata ogni volta che viene eseguito il protocollo. Il termine "trusted" deriva dal fatto che una o più di persone devono partecipare alla cerimonia in modo "fidato" contribuendo alla creazione della CRS con dei segreti anche chiamati "toxic waste" che non appena inseriti nella procedura dovranno essere eliminati.

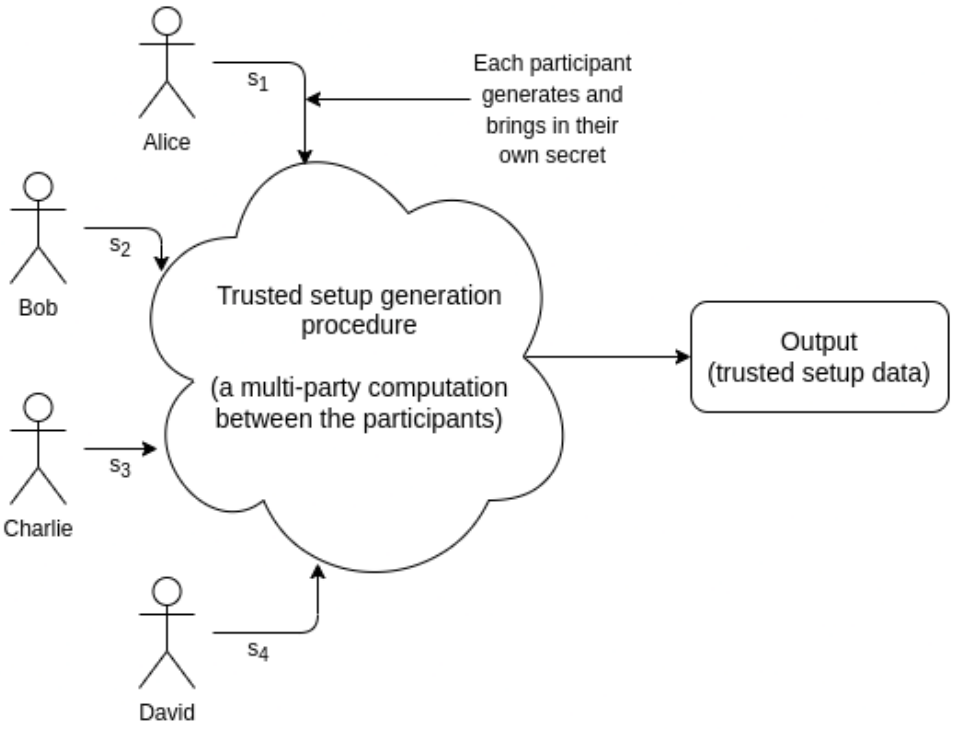

Figura 1.11: immaggine generica di una cerimonia di trusted set-up. presa da [6]

Una volta completata l'elaborazione e dopo aver eliminato i segreti, il risultato dell'elaborazione (la CRS) diventa irreversibile e non può essere riprodotto volontariamente, il che garantisce la sicurezza del protocollo. Tuttavia, se le parti coinvolte non si comportano in modo leale e conservano o pubblicano i loro segreti, la validi-

tà della cerimonia dipenderà dal modello di affidabilità utilizzato dall'algoritmo di trusted set-up.

Poiché i protocolli operano in diversi domini applicativi, non tutti utilizzano lo stesso modello di affidabilità. La trattazione dei vari modelli di affidabilità sarebbe lunga e non verrà spiegata qui. Tuttavia, riducendo la classificazione a due proprietà degli algoritmi, è possibile descrivere brevemente alcune tipologie utili per la trattazione successiva. Le due proprietà su cui si basa la classificazione sono il numero di parti fidate che partecipano alla cerimonia e il numero di parti che devono comportarsi "lealmente" per garantire la segretezza della CRS.

In base a queste proprietà, si possono individuare tre gruppi di interesse (che non sono gli unici):

- **1 - 1**: questo modello è simile a quello utilizzato nei servizi centralizzati, in cui ci si affida a un singolo individuo fidato per la gestione dei dati.
- **N/2 - 1**: la maggior parte delle blockchain utilizza questo modello, in cui il processo viene considerato valido se la maggioranza dei partecipanti rimane onesta.
- **1 - N**: in questo caso, è sufficiente che almeno una delle parti rimanga onesta durante la cerimonia per garantire la segretezza della CRS. Questo modello verrà analizzato in modo più approfondito successivamente.

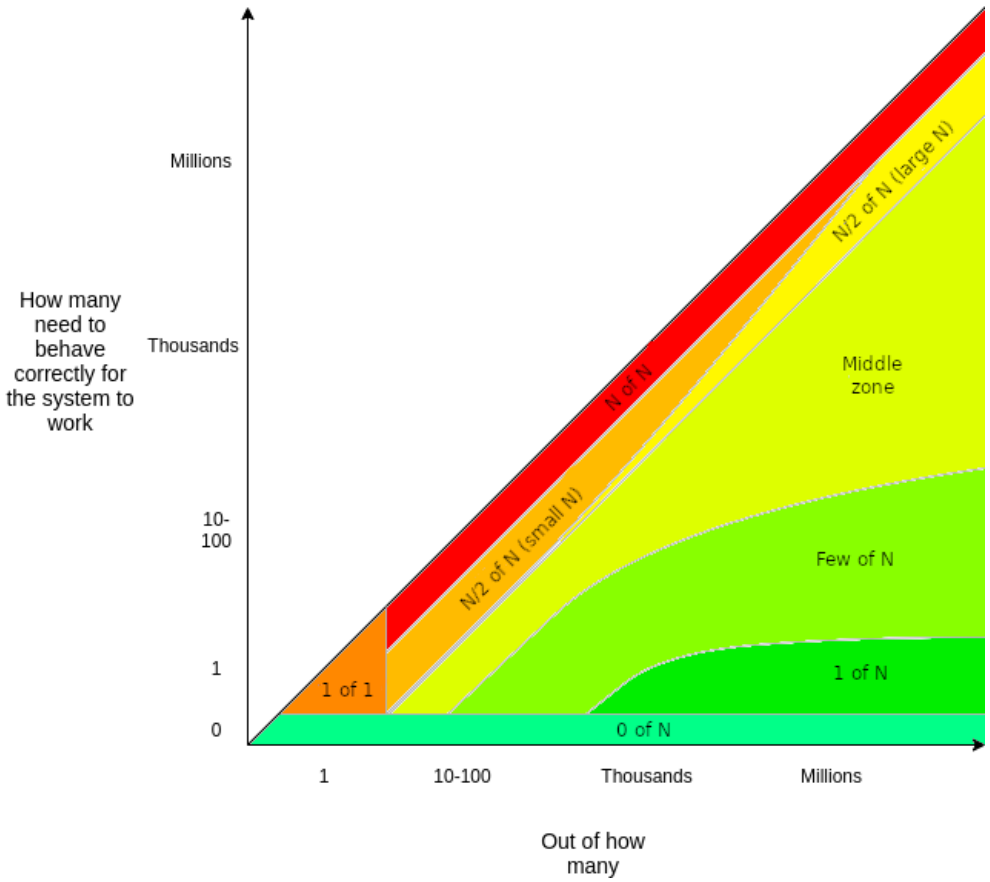

Figura 1.12: immaggine generica di una cerimonia di trusted set-up. presa da [6]

MPC (multi-party computation)

La tecnologia MPC (multi-party computation), anche se non è stata ancora esplicitamente menzionata, è alla base dei trusted set-up. Infatti per generare una CRS sicura e non riproducibile, è necessario coinvolgere molte parti e garantire che nessuno possa ricavare informazioni sui segreti dei partecipanti o sulla CRS stessa. Per questo motivo la procedura non si limita a una semplice combinazione dei segreti, ma può essere vista come una "black box" che applica elaborazioni ai segreti in input e restituisce la CRS in output.

Ogni algoritmo di MPC deve soddisfare due proprietà fondamentali:

1. Nel caso in cui uno o più partecipanti disonesti decidano di rivelare il loro segreto, il protocollo MPC deve impedire loro di obbligare i partecipanti onesti a rivelare le loro informazioni riservate o influenzare il risultato finale. 2. Nessuno dei partecipanti deve essere in grado di dedurre i segreti degli altri partecipanti dagli elaborati del protocollo. In altre parole, il calcolo effettuato dall'algoritmo non deve fornire alcun indizio sui segreti che hanno portato al risultato.

Groth16

Groth16 è un protocollo Zero-knowledge proof proposto da Jens Groth nell'articolo "On the Size of Pairing-based Non-interactive Arguments"[7] pubblicato nel 2016, questo protocollo è diventato motlo diffuso perchè è molto efficiente dal punto di vista computazionale e permette quindi di creare zk-SNARK molto performati sia in termini di tempo che di memoria, questa tecnologia è alla base per esempio del progetto Z-cash una criptovaluta cofondata da alessanro chiesa uno dei ricercatori di [articolo].

Il trusted setups di Groth16 richiede due fasi distinte:

- **1.**: Nella prima si applica una procedura di trusted set-up chiamata "power of tau" che consiste in una algoritmo MPC basato sul modello di affidabilità 1-N che utilizzando le curve ellitice permette di generare una CRS. Il processo viene chiamato powers of tau o Pot perchè duarnte le sue fasi vengono usate le potenze di un numero tao generalmente il 2 assime ai contributi dei partecipanti per generare punti casuali e ottener una CRS. il numero N di partecipati per una cerimonia di tipo power of tau puo essere dell ordine del miglaio di partecipanti rendendo cosi la cerimonia molto sicura in quanto absta che almento uno di essi mantenga il segreto per far si che la CRS rimanga valdia. Questo procedimento deve essere svolto una volta sola e deve essere combinato con il processo ella fase successiva.

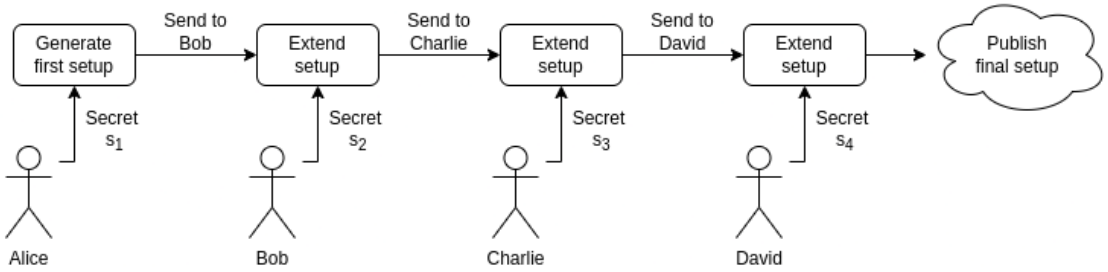

Figura 1.13: immaggine generica di una cerimonia di trusted set-up. presa da [6]

- **2.**: Nella seconda fase invece viene elaborato un trusted set-up dipendente dalla computazione che si vuole provare (o meglio dipendete dal circuito). Quindi questa fase deve essere riprodotta dal dimostratore ogni volta che la dimostrazione cambia. Non entrerò nei dettagli implementativi, ma in questa fase viene utilizzata la Fiat-Shamir heuristic, che permette di scegliere il punto in cui valutare il polinomio che descrive la computazione senza che il dimostratore possa interferire. Il punto viene scelto sulla base dei calcoli fatti per ottenere il polinomio stesso. È intuitivo comprendere che se il dimostratore volesse falsificare la prova, non potrebbe farlo perché per scegliere arbitrariamente il polinomio e soddisfare un determinato vincolo, dovrebbe conoscere il valore di s. Tuttavia, essendo $s$derivato dal polinomio stesso questo non è possibile

- **1 - N**: in questo caso, è sufficiente che almeno una delle parti rimanga onesta durante la cerimonia per garantire la segretezza della CRS. Questo modello verrà analizzato in modo più approfondito successivamente.

Sicuramente il fatto che la robustezze della tecnologia zk-SNARK sia subordinata a una procedura cosi elaborata è uno svantaggio, infatti esistono, protocolli di Zero-knowledge proof che non necessitano di una fase di trust set-up iniziale. Bulletproof e le STARK (Scalable Transparent Arguments of Knowledge), ad esempio, non richiedono alcuna configurazione di questo tipo, tuttavia, le tecnologie zk-SNARK basati su Groth16 risultano essere meglio in termini di efficienza.

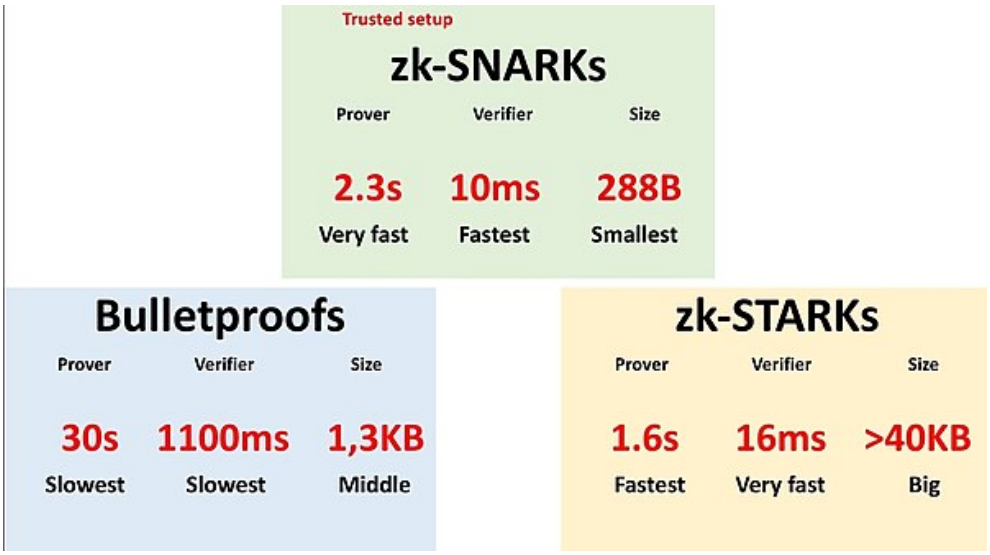

Figura 1.14: immaggine generica di una cerimonia di trusted set-up. presa da [6]

Bibliografia

[1] Cloud Armor Emil Kiner Senior Product Manager. «Google blocked largest Layer 7 DDoS attack». In: (). URL: https://cloud.google.com/blog/products/identity-security/how-google-cloud-blocked-largest-layer-7-ddos-attack-at-46-million-rps.

[2] S Goldwasser, S Micali e C Rackoff. «The Knowledge Complexity of Interactive Proof-Systems». In: *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing*. STOC '85. Providence, Rhode Island, USA: Association for Computing Machinery, 1985, 291–304. ISBN: 0897911512. DOI: 10.1145/22145.22178. URL: https://doi.org/10.1145/22145.22178.

[3] «Interactive proof system». In: (). URL: https://en.wikipedia.org/wiki/Interactive_proof_system.

[4] Jean-Jacques Quisquater et al. «How to Explain Zero-Knowledge Protocols to Your Children». In: *Advances in Cryptology — CRYPTO' 89 Proceedings*. A cura di Gilles Brassard. New York, NY: Springer New York, 1990, pp. 628–631. ISBN: 978-0-387-34805-6.

[5] dtubbenhauer. «Slide on Schwartz-Zippel Lemma». In: (). URL: https://www.dtubbenhauer.com/slides/my-favorite-theorems/23-schwartz-zippel.pdf.

[6] Vitalik Buterin. «How do trusted setups work?» In: (). URL: https://vitalik.ca/general/2022/03/14/trustedsetup.html.

[7] Jens Groth. «On the Size of Pairing-Based Non-interactive Arguments». In: *Advances in Cryptology – EUROCRYPT 2016*. A cura di Marc Fischlin e Jean-Sébastien Coron. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 305–326. ISBN: 978-3-662-49896-5.