Tecniche di Progettazione Automatica **Anno Acc. 1998-99**
PROF. ALFREDO DE SANTIS

SCHEMI DI IDENTIFICAZIONE

a cura di Lucia Cirigliano, Ferdinando Ruggiero, Francesco Suppa *12 Febbraio 1999*

Indice

1 Introduzione

Il problema della costruzione di schemi di identificazione nasce dalla necessità di provare (elettronicamente) l'identità di qualcuno in modo sicuro; cioè dati due utenti A e B, deve essere impossibile, per un terzo utente C, potersi identificare a B come A.
Le proprietà fondamentali che un qualsiasi protocollo di identificazione deve verificare sono:

1. **Completezza:** *se l'utente che vuole identificarsi è onesto ed esegue correttamente il protocollo, allora l'identificazione ha sempre successo.*

2. **Soundness:** *una persona che non è Alice non riesce ad identificarsi a Bob come Alice.*

3. **Complessità bassa:** *si presuppone che Alice abbia a disposizione poca memoria e possa effettuare solo calcoli di bassa complessità, infatti, solitamente l'utente che vuole identificarsi ha a disposizione una smart card che effettivamente è fornita di poca memoria e ha la possibilità di effettuare calcoli di complessità limitata.*

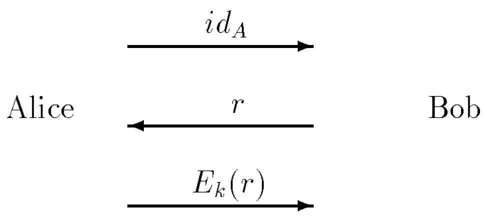
Inoltre è importante la presenza di un'Autorità che certifichi l'identità di Alice; nel nostro contesto parleremo di una Trusted Authority. Più dettagliatamente, ciò di cui si occupa una Trusted Authority è la produzione di certificati di identità per utenti della rete. Informalmente, costruisce certificati utilizzando informazioni come per esempio documenti di riconoscimento (certificati di nascita, passaporto, carta d'identità), da questi genera una stringa, $ID(utente)$, che successivamente firma utilizzando uno schema di firma sicuro.

2 Schema di Identificazione a chiave segreta

Problematica: Gli utenti A e B hanno in comune una chiave segreta k e l'utente A vuole identificarsi a B in modo sicuro. I due utenti utilizzano un algoritmo di cifratura E_k e il relativo algoritmo di decifratura D_k. **Protocollo:**

A: Manda a B id_A (il suo identificativo) **B:** Manda a A un numero r scelto a caso come sfida **A:** Calcola il valore $y{=}E_k(r)$ e lo manda a B **B:** Controlla che $r{=}D_k(y)$

Graficamente la comunicazione può essere riassunta nel seguente scambio di parametri:



Verifichiamo che le proprietà sopra elencate siano soddisfatte dal protocollo:

Completezza: Se entrambi A e B eseguono correttamente il protocollo, e nessuno dei due

vuole imbrogliare l'altro, sicuramente A riuscirà ad identificarsi a B senza nessun problema in quanto $D_k(E_k(r))=r$, quindi la proprietà di Completezza è verificata.
Soundness: Questa proprietà dipende dalla sicurezza dell'algoritmo di cifratura, in particolare dipende dalla sicurezza dell'algoritmo contro attacchi di tipo chosen plaintext.
Complessità bassa: Questa proprietà non è rispettata in quanto A deve conservare tutte le chiavi degli utenti ai quali vuole identificarsi.

Svantaggi:
L'utente A può identificarsi solo ad utenti con cui condivide una chiave.

3 Schema di Identificazione a chiave pubblica

Problematica: Ogni utente che vuole identificarsi ad altri possiede due chiavi, una pubblica e una privata, inoltre possiede un certificato di identità firmato da una Trusted Authority (TA) fidata che garantisce la sua identità. La TA usando le informazione contenute in documenti di identità tradizionali (es: certificato di nascita, passaporto) stabilisce l'identità di un utente costruendo una stringa ID(Alice). La coppia (ID(Alice), Firma(TA)) rappresenterà il certificato dell'utente.

Il protocollo più diffuso a chiave pubblica è quello di Schnorr.

3.1 Schema di identificazione di Schnorr

Lo schema richiede una TA (Trusted Authority) che si occuperà, tra l'altro, di scegliere i parametri per lo schema.
Valori comuni a tutti gli utenti:

- p è un numero primo abbastanza grande (per esempio $p \geq 2^{512}$) tale che il problema del logaritmo discreto in Z_p^* è intrattabile.
- q è un primo grande, che divide $p-1$ (per esempio $q \geq 2^{140}$).
- $\alpha \in Z_p^*$ ha ordine q (quindi α può essere calcolato come la $((p-1)/q)$-esima potenza di un elemento primitivo); $\alpha^q \equiv 1 \bmod p$.
- La TA fissa uno schema di firma sicuro con un algoritmo di firma sig_{TA} ed un algoritmo di verifica ver_{TA}.

Chiave privata:

- Ogni utente sceglie un parametro segreto a in [0,q-1] e pone $priv = a$.

Chiave pubblica:

- Ogni utente calcola $pub = \alpha^{-a} \bmod p$.

Costruzione del certificato da parte della TA:

1. La TA stabilisce l'identità di Alice usando forme convenzionali di identificazione (es. certificato di nascita, passaporto). Successivamente costruisce una stringa ID(Alice) che contiene le informazioni per la sua identificazione.

2. Alice sceglie a caso un esponente segreto a , dove $0 \leq a \leq q-1$. Successivamente calcola $v = \alpha^{-a} \bmod p$ e spedisce v alla TA.

3. La TA genera la firma

$$s = sig_{TA}(ID(Alice), v)$$

e poi il certificato firmato

$$C(Alice) = (ID(Alice), v, s)$$

che viene dato ad Alice.

Protocollo (fase di identificazione tra due utenti):

1. Alice sceglie a caso un numero k, dove $0 \leq k \leq q-1$ e calcola

$$\gamma = \alpha^k \bmod p.$$

2. Alice manda il suo certificato C(Alice)=$(ID(Alice), v, s)$ ed il valore γ a Bob.

3. Bob verifica la firma di TA testando che:

$$ver_{TA}(ID(Alice), v, s) = true.$$

4. Bob sceglie un numero a caso r $(1 \leq r \leq 2^t)$ che manda ad Alice (la scelta di t verrà discussa in seguito).

5. Alice calcola $y = k + ar \bmod q$ e lo manda a Bob.

6. Bob verifica che $\gamma = \alpha^y v^r \bmod p$.

Vediamo graficamente come avviene la comunicazione tra Alice e Bob:

$$\begin{array}{ccc} & \xrightarrow{\quad C, \gamma \quad} & \\ \text{Alice} & \xleftarrow{\quad r \quad} & \text{Bob} \\ & \xrightarrow{\quad y \quad} & \end{array}$$

Nel protocollo di verifica inizialmente la firma s prova la validità del certificato di Alice. Bob, quindi, verifica la firma della Trusted Authority sul certificato di Alice e si convince che il certificato è autentico. Verifichiamo quali proprietà il prorocollo soddisfa:

- **Completezza:** dobbiamo verificare che $\gamma = \alpha^y v^r$; infatti

$$\alpha^y v^r = \alpha^{k+ar} v^r = \alpha^{k+ar}\alpha^{-ar} = \alpha^k = \gamma$$

  La prima uguaglianza è valida in quanto $\alpha^q \equiv 1 \bmod p$.

- **Soundness:** sebbene tale proprietà non sia stata mai dimostrata, noi vedremo come un utente disonesto Tom potrebbe identificarsi come Alice:

  - Attacco 1: potrebbe dare un certificato falso $C'(Alice) = (ID(Alice), v', s')$ dove $v' \neq v$. Ma s' è supposto essere una firma di $(ID(Alice), v')$ e questo viene verificato da Bob nel passo 3 del protocollo. Se lo schema di firma del TA è sicuro, Tom non sarà capace di falsificare una firma s'.
  - Attacco 2: supponiamo che Tom riesca a conoscere r prima di iniziare il protocollo e che il certificato sia quello giusto. Potrebbe calcolare γ in funzione di r, facendo in modo che sia soddisfatta l'equazione $\gamma = \alpha^y v^r$. Questo lo può fare perchè conoscendo v, r, α e scegliendo y a caso, può calcolare $\gamma = \alpha^y v^r$. La scelta a caso di r implica che Tom ha probabilità 2^{-t} di indovinare il valore corretto di r; per questo motivo t è un *parametro di sicurezza*. Nelle applicazioni pratiche, in genere, $t = 40$.
  - Attacco 3: supponiamo che per un particolare valore γ Tom conosca due valori di r, r_1 ed r_2, tali che $\gamma = \alpha^{y_1} v^{r_1} = \alpha^{y_2} v^{r_2}$. Allora, se riceve r_1 dà come risposta y_1 tale che $\gamma = \alpha^{y_1} v^{r_1}$; se, invece, riceve r_2 risponde con y_2 tale che $\gamma = \alpha^{y_2} v^{r_2}$. In questo caso la probabilità di successo risulta essere $\leq \frac{2}{2^t} = \frac{1}{2^{t-1}}$. Un tale valore γ non può esistere perchè altrimenti riusciremmo a risolvere un logaritmo discreto in tempo polinomiale. Vediamo perchè. Siano $y_1,\ y_2,\ r_1,\ r_2$ tali che:

$$y_1 \neq y_2 \bmod p$$

    e

$$\gamma \equiv \alpha^{y_1} v^{r_1} \equiv \alpha^{y_2} v^{r_2} \bmod p.$$

    Si ha che

$$\alpha^{y_1 - y_2} \equiv v^{r_2 - r_1} \bmod p.$$

    Poichè $v = \alpha^{-a}$ abbiamo che

$$y_1 - y_2 \equiv a(r_1 - r_2) \bmod q.$$

    Essendo $0 \leq |r_2 - r_1| \leq 2^{40}$ ed essendo $q \geq 2^{140}$ primo allora $\gcd(r_2 - r_1, q) = 1$, quindi il nemico può calcolare $a = (y_1 - y_2)(r_1 - r_2)^{-1} \bmod q$ risolvendo quindi il logaritmo discreto in tempo polinomiale il che è assurdo.

  In generale si pensa (non esiste nessuna dimostrazione) che anche se un nemico assistesse a tutte le comunicazioni, non riuscerebbe lo stesso ad impersonificare un utente senza conoscere il valore di a.
  Non è stata provata la sicurezza dello schema di identificazione di Schnorr.

- **Complessità bassa:** la proprietà è rispettata in quanto i calcoli che Alice deve effettuare hanno complessità abbastanza limitata, infatti, Alice deve computare solo:
  1. $\alpha^k \bmod p$. che può essere effettuato anche off_line dal momento in cui α e k sono parametri che non dipendono dalla conversazione con Bob.
  2. $y = k + ar \bmod q$ è semplice.

  Per quanto riguarda la complessità in spazio, c'è da dire che l'unica cosa che Alice deve memorizzare, è la sua chiave segreta a.

3.2 Schema di firme digitali basato sul protocollo di Schnorr

Lo schema di Schnorr si può trasformare in uno schema di firme digitali eliminando la fase di interazione tra Alice e Bob. In questo schema il parametro r saraà calcolato direttamente da Alice utilizzando una funzione hash (h). I parametri utilizzati per il protocollo, come per lo schema di identificazione, sono:

- **Parametri comuni a tutti gli utenti:**

  un numero primo p abbastanza grande (per esempio $p \geq 2^{512}$) tale che il problema del logaritmo discreto in Z_p^* è intrattabile.

  q è un primo grande, che divide $p - 1$.

  $\alpha \in Z_p^*$ ha ordine q (α potrebbe essere la $((p-1)/q)$-esima potenza di un elemento primitivo).

  uno schema di firma sicuro con un algoritmo di firma sig_{TA} ed un algoritmo di verifica ver_{TA}.

- **Chiave privata:** ogni utente sceglie un valore segreto a in [0,q-1] e pone $priv$=a.

- **Chiave pubblica:** ogni utente calcola $pub = \alpha^{-a} \bmod p$

Il protocollo è il seguente:

1. Calcola $\gamma = \alpha^k \bmod p$.
2. Calcola $r = h(x, \gamma)$.
3. Calcola $y = k + ar \bmod p$.
4. Firma il messaggio x con $sig_{ID}(x) = (x, \gamma, y)$.
5. Manda questi valori ed il certificato ad un utente.

Le informazioni che i due utenti si scambiano, sono visualizzate nel seguente schema:

$$\text{Alice} \xrightarrow{(\gamma, r, y, C_{Alice})} \text{Bob}$$

Per verificare che la firma è realmente di Alice, l'utente verifica l'equazione $\gamma \equiv \alpha^y v^r \bmod p$. La sicurezza di questo schema non è stata dimostrata. Questa appena descritta è una tecnica generale per la costruzione di schemi di firme digitali a partire da uno schema di identificazione.

3.3 Schema di identificazione di Guillou-Quisquater

Il problema trattato è lo stesso della sezione precedente, e cioè permettere ad un utente (Alice) di potersi identificare ad un altro utente della rete (Bob), in modo tale da evitare che un impostore (Tom) possa identificarsi a Bob come Alice. Lo schema di identificazione di Guillou-Quisquater si basa sull'RSA. Viene richiesta, come nel protocollo di Schnorr, la partecipazione di una Trusted Authority che provvederà alla scelta dei parametri da utilizzare nel protocollo:

- **Parametri pubblici:**
  1. Due numeri primi p e q abbastanza grandi e tali che sia difficile fattorizzare n dove $n = pq$.
  2. Un intero primo grande b (parametro di sicurezza di 40 bit), utilizzato come esponente pubblico per RSA.
  3. Uno schema di firme.
- **Chiave privata:** ogni utente sceglie un parametro segreto u in $[0,n\text{-}1]$ e calcola $v = (u^{-1})^b \bmod n$. La chiave privata consiste del parametro u.
- **Chiave pubblica:** la chiave pubblica è composta dai tre parametri u, b e v.

Viene richiesto un certificato di identità per Alice; la sua costruzione viene eseguita dalla Trusted Authority seguendo il seguente protocollo:

Costruzione del certificato

1. La TA stabilisce l'identità di Alice e costruisce una stringa di identificazione $ID(Alice)$.
2. Alice sceglie segretamente un intero u, dove $0 \leq u \leq n-1$, valuta $v = (u^{-1})^b \bmod n$ e lo invia alla TA.
3. La TA genera una firma
$$s = sig_{TA}(ID(Alice), v)$$
e poi il certificato
$$C(Alice) = (ID(Alice), v, s)$$
è dato ad Alice.

Il protocollo di identificazione è il seguente:

Protocollo di Guillou_Quisquater

- **Alice:** sceglie a caso un numero k, dove $0 \leq k \leq n-1$ e valuta
$$\gamma = k^b \bmod n.$$
- **Alice:** invia a Bob il certificato $C(Alice) = (ID(Alice), v, s)$ e γ.
- **Bob:** verifica la firma della TA testando che:
$$ver_{TA}(ID(Alice), v, s) = true.$$
- **Bob:** sceglie un numero a caso r, $0 \leq r \leq b-1$ e lo invia ad Alice.
- **Alice:** valuta $y = ku^r \bmod n$ ed invia y a Bob.
- **Bob:** verifica che $\gamma \equiv v^r y^b \bmod n$.

Il seguente disegno mostra quali sono i parametri scambiati dai due utenti durante la conversazione:

$$\begin{array}{ccc} & \xrightarrow{\quad (C_{Alice}, \gamma) \quad} & \\ \text{Alice} & \xleftarrow{\quad r \quad} & \text{Bob} \\ & \xrightarrow{\quad y \quad} & \end{array}$$

Verifichiamo la validità delle seguenti proprietà:

- **Completezza:** se il protocollo viene eseguito correttamente allora si può provare che

$$\begin{aligned} v^r y^b &\equiv (u^{-b})^r (ku^r)^b \bmod n \\ &\equiv u^{-br} k^b u^{br} \bmod n \\ &\equiv k^b \bmod n \\ &\equiv \gamma \bmod n. \end{aligned}$$

- **Soundness:** vediamo in che modo un utente (Tom) potrebbe identificarsi come un altro utente (Alice).

  1. Supponiamo che riesca ad indovinare la sfida, r, prima di iniziare il protocollo di identificazione. In tal caso potrebbe costruire un valore γ tale che sia soddisfatta l'equazione $\gamma = v^r y^b$. Questo può farlo in quanto conosce v, r, b e può scegliere y a caso. La probabilità che il nemico indovini la sfida è $\frac{1}{b} = \frac{1}{2^{40}}$.
  2. Supponiamo che il nemico conosca un particolare valore γ per il quale sia in grado di indovinare due sfide r_1 e r_2 in modo che se riceve r_1 dà come risposta y_1 tale che $\gamma \equiv v^{r_1} {y_1}^b \bmod n$ e se riceve r_2 dà come risposta y_2 tale che $\gamma \equiv v^{r_2} {y_2}^b \bmod n$. In tal caso la probabilità che il nemico abbia successo è $\frac{2}{b}$. Proveremo ora che se il nemico conoscesse un tale valore γ potrebbe calcolare u in tempo polinomiale.
  Sia
  $$\gamma \equiv v^{r_1} {y_1}^b \equiv v^{r_2} {y_2}^b$$
  e supponendo che $r_1 > r_2$ otteniamo
  $$v^{r_1 - r_2} \equiv \left(\frac{y_2}{y_1}\right)^b \bmod n.$$
  Poichè $0 < r_1 - r_2 < b$ e b è primo allora il nemico può valutare $t = (r_1 - r_2)^{-1} \bmod b$ in tempo polinomiale, usando l'algoritmo esteso di Euclide. Quindi valutando
  $$v^{(r_1 - r_2)t} \equiv \left(\frac{y_2}{y_1}\right)^{bt} \bmod n$$
  ed osservando che per qualche intero positivo l si ha
  $$(r_1 - r_2)t = lb + 1$$
  otteniamo
  $$v^{lb+1} \equiv \left(\frac{y_2}{y_1}\right)^{bt} \bmod n$$
  o equivalentemente
  $$v \equiv \left(\frac{y_2}{y_1}\right)^{bt} v^{-lb} \bmod n.$$
  Eleviamo entrambi i lati della congruenza alla potenza $b^{-1} \bmod \phi(n)$ ottenendo:
  $$u^{-1} \equiv \left(\frac{y_2}{y_1}\right)^{t} v^{-l} \bmod n.$$

Infine valutando l'inverso modulo n di entrambi i lati della congruenza precedente otteniamo la seguente formula per u:

$$u = \left(\frac{y_2}{y_1}\right)^{-t} v^l \bmod n.$$

Quindi un nemico può usare questa formula per valutare u in tempo polinomiale il che è assurdo.

Osserviamo che non è stata ancora dimostrata la sicurezza dello schema di Guillou-Quisquater. Questo schema viene utilizzato per l'autenticazione delle smart cards [8]. Ogni smart card è caratterizzata da un proprio insieme di *credenziali* (numero di acconto bancario, numero seriale di chip, periodo di validità, codice di servizio etc.) che rappresentano l'identità della scheda (in questo caso $C(Alice) = (ID(Alice), v, s)$). La chiave pubblica della scheda è rappresentata da un intero composto n e da un esponente pubblico b. L'insieme delle credenziali viene trasformato in un intero più lungo detto rappresentativo v grande quanto il modulo di n. Tale trasformazione viene specificata da regole di ridondanza pubblicamente note che saranno standardizzate in ISO/IEC 9796 (DIS 9796). L'autenticazione della scheda avviene grazie alla conoscenza da parte di quest'ultima di un valore segreto u detto *accreditamento* che è definito come la soluzione segreta di un'equazione pubblica che DIS 9796 specifica nel modo seguente:

$$vu^b \equiv 1 \bmod n.$$

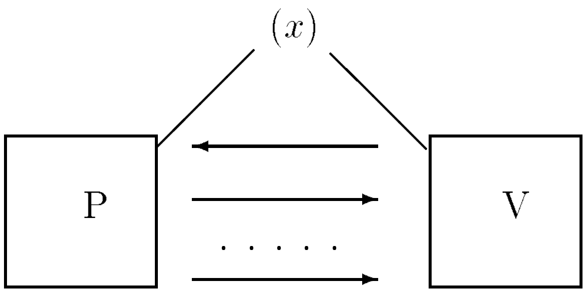
4 Sistemi di prova interattivi

Un sistema di prova interattivo si prefigge come scopo quello di far si che un utente possa convincerne un altro della veridicità di una certa affermazione senza rivelare alcuna informazione circa la dimostrazione. Solitamente, i partecipanti sono due: un provatore P e un verificatore V. Un sistema di prova interattivo deve verificare le seguenti due proprietà:

1. **Completezza:** *se x è una proposizione vera di Π ($x \in L$) e se P e V eseguono correttamente il protocollo, allora V accetta la prova di P con probabilità uguale ad 1.*

2. **Soundness:** *se x è una proposizione falsa di Π ($x \notin L$) allora la probabilità che V accetti la prova è molto bassa qualunque sia il provatore P'.*

Lo schema di un sistema di prova interattivo è:



Dal punto di vista teorico possiamo immaginare che V abbia risorse di tempo limitate (è una macchina polinomiale), mentre P è una macchina illimitata (complessità di tempo infinito); per questo motivo P è in grado di risolvere il problema a differenza di V. L'interazione di un sistema di prova consiste in un numero determinato di round durante i quali P e V alternativamente fanno quanto segue:

1. ricevere un messaggio dall'altro;

2. eseguire una computazione privata;

3. mandare un messaggio all'altro.

Un tipico round del protocollo consiste di una sfida lanciata da V e di una risposta di P. Alla fine della prova V, può accettare o rifiutare a seconda delle risposte di P.

4.1 Sistema di prova interattivo per grafi non-isomorfi

Isomorfismo tra grafi: Due grafi $G_0 = (V, E(G_0))$ e $G_1 = (V, E(G_1))$ (dove $V = \{1, 2, ..., n\}$) si dicono isomorfi ($G_0 \approx G_1$) se esiste una permutazione π dei vertici tale che $\forall\ i, j \in V$

$$(i, j) \in E(G_0) \Leftrightarrow (\pi(i), \pi(j)) \in E(G_1).$$

Per il problema dell'isomorfismo dei grafi non è stato ancora trovato nessun algoritmo che lo risolva in tempo polinomiale e non si sa se il problema appartiene alla classe NP.
Problematica: P vuole dimostrare a V che i due grafi G_0 e G_1 non sono isomorfi.

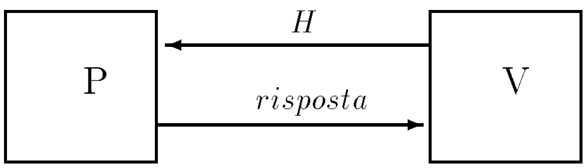
Protocollo

Input: due grafi G_0 e G_1 non isomorfi e aventi insieme dei vertici $\{1, ..., n\}$.

1. **V:** sceglie un bit b$\in \{0, 1\}$ a caso: se sceglie b=0 costruisce un grafo H isomorfo a G_0 (lo può costruire scegliendo a caso una permutazione π dell'insieme dei vertici $\{1, ..., n\}$). Se sceglie b=1 costruisce un grafo H isomorfo a G_1. V manda H a P (sfida P chiedendo a quale grafo è isomorfo H).

2. **P:** determina a quale grafo è isomorfo H (lo può fare perchè è una macchina illimitata) e manda la risposta a V.

3. **V:** verifica che la risposta è conforme a quello che aveva calcolato.



Verifichiamo che il protocollo soddisfi le seguenti proprietà:

- **Completezza:** supponiamo che P e V eseguano tutti i passi correttamente e che G_0 e G_1 siano non isomorfi, allora P può rispondere a qualunque sfida di V, in quanto effettivamente è in grado di calcolare a quale grafo è isomorfo H; quindi V accetterà con probabilitá uguale ad 1 la prova di P.

- **Soundness:** supponiamo che l'affermazione "G_0 non è isomorfo a G_1" sia falsa e che quindi un provatore P' voglia imbrogliare V. Allora H costruito isomorfo rispetto ad uno dei due grafi, risulterà isomorfo ad entrambi; a questo punto P' in nessun modo può determinare rispetto a quale grafo H è stato costruito isomorfo. Pertanto l'unica possibilità che ha P' di rispondere correttamente è quella di indovinare, scegliendo a caso, tra i due grafi, la risposta. Per questi motivi V accetterà la prova di P' con probabilità $\leq \frac{1}{2}$. Per ridurre questa probabilità possiamo iterare t volte il protocollo, facendo sí che la probabilità con cui V accetta sia $\leq \frac{1}{2^t}$. Si noti che le computazioni di V sono eseguibili tutte in tempo polinomiale. Questo sistema di prova in pratica non è utilizzabile perchè P avrebbe bisogno di una potenza di calcolo infinita che realmente non ha.

4.2 Prove a conoscenza-zero

Consideriamo l'esistenza di due utenti P e V che interagiscono, se P riesce a convincere V della veridicità di una certa affermazione senza dare nessuna informazione circa la dimostrazione, allora si dice che P ha utilizzato delle prove a **conoscenza-zero**. I sistemi di prova basati sulle prove a conoscenza-zero utilizzano, solitamente, problemi che appartengono alle classi NP e $NP-C$. In generale, un utente, nel nostro caso P, per identificarsi ad un altro utente, V, crea un'istanza di un problema che appartiene a NP o a $NP-C$ e utilizza delle prove a conoscenza-zero per mostrare a V di avere la soluzione del problema in questione. Utilizzando sistemi di prova a conoscenza-zero si fa in modo che V non riesca ad avere più informazioni rispetto a quelle che normalmente riesce ad ottenere da un'analisi del transcript della conversazione con P. Esistono tre tipi di conoscenza-zero: perfetta, computazionale e statistica. Qui ci occuperemo di due sistemi di prova che rispettano il primo e il secondo tipo di conoscenza_zero:

- Sistemi di prova interattivi a conoscenza-zero perfetta
- Sistemi di prova interattivi a conoscenza-zero computazionale

Di seguito diamo la definizione dei tre tipi di conoscenzazero.
Conoscenza-zero perfetta: *per ogni verificatore V^* esiste un algoritmo probabilistico polinomiale A_{V^*} tale che, per ogni $x \in L$*

$$A_{V^*}(x) = View_{V^*}(x)$$

dove $A_{V^*}(x)$ è la variabile aleatoria che rappresenta l'output dell'algoritmo quando ha in input x e $View_{V^*}(x)$ è la variabile aleatoria che rappresenta la vista del verificatore V^* quando interagisce con P sull'input x.

Informalmente, V^* può rappresentare sia un verificatore onesto (V), che un impostore. In pratica l'algoritmo A_{V^*}, chiamato *simulatore*, riesce a generare il transcript della conversazione, qualunque sia l'algoritmo utilizzato da V*. Osserviamo che la conoscenza zero-perfetta non esiste per problemi $NP-C$, per essi si può avere una forma di conoscenza più debole detta zero-computazionale.

Conoscenza-zero computazionale: *per ogni verificatore V^* esiste un algoritmo probabilistico polinomiale A_{V^*} tale che, per ogni $x \in L$ ed utilizzando l'algoritmo di V^*, riesce a generare* *una variabile aleatoria che è computazionalmente non distinguibile dalla vista del verificatore, cioè dalla variabile aleatoria che rappresenta la comunicazione tra P e V^*:*

$$A_{V^*}(x) \simeq View_{V^*}(x).$$

La conoscenza zero-computazionale è definita esattamente come la conoscenza zero-perfetta: l'unica differenza è che la distribuzione di probabilità è polinomialmente indistinguibile piuttosto che identica.

Conoscenza-zero di tipo statistico: *per ogni verificatore V^* esiste un algoritmo probabilistico polinomiale A_{V^*} tale che, per ogni $x \in L$,*

$$\sum_{y \in \{0,1\}^*} |Pr_{A_{V^*}}(y) - Pr_{View_{V^*}}(y)| \leq \frac{1}{|x|^c}$$

per ogni $c \geq 0$ e x sufficientemente grande, dove $Pr_{A_{V^*}}(y)$ è la probabilità che l'algoritmo A_{V^*} dia in output y su input x e $Pr_{View_{V^*}}(y)$ è la probabilità di y in $View_{V^*}(x)$.

4.2.1 Sistema di prova interattivo per grafi isomorfi

Problematica: In questo sistema il provatore P ed il verificatore V hanno come input comune due grafi G_0 e G_1 isomorfi. P vuole dimostrare a V che i due grafi sono isomorfi (da osservare che verificare che due grafi siano isomorfi tra loro è un problema difficile).

Protocollo:

Input: due grafi G_0 e G_1.

1. **P:** costruisce un grafo H scegliendo una permutazione a caso π dei vertici $\{1, 2, ..., n\}$ di G_0 e lo manda a V.

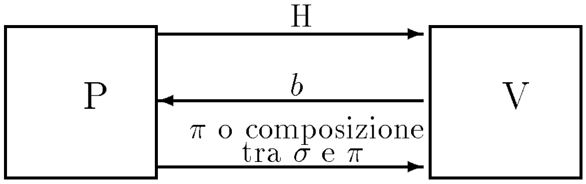
2. **V:** sceglie un bit b a caso: chiede a P di mostrare che H è isomorfo a G_b.

3. **P:** se $b = 0$ gli invia la permutazione π, mentre se $b = 1$ allora gli manda la permutazione composta di σ e π, dove σ è una permutazione fissata tale che $G_0 \approx G_1$ tramite σ.

4. **V:** controlla che H è isomorfo al grafo selezionato G_b, tramite la permutazione avuta da P.

5. **V:** accetta la prova di P se la verifica ha successo.

Graficamente si può sintetizzare nel seguente modo:



Verifichiamo che il protocollo soddisfi le seguenti proprietà:

- **Completezza:** supponiamo che P e V siano onesti ed eseguano correttamente il protocollo, allora P sarà in grado di rispondere sempre alle sfide di V in modo esatto; questo perchè P può calcolare la permutazione come descritto nel passo 3 e pertanto V accetta la prova di P con probabilità uguale ad 1.

- **Soundness:** supponiamo che il provatore non sia onesto e che quindi l'affermazione $G_0 \approx G_1$ sia falsa.
  I casi da considerare sono i seguenti:

  - P' costruice H isomorfo a G_0
  - P' costruisce H isomorfo a G_1
  - P' costruisce H non isomorfo ad entrambi

  Nei primi due casi P' può rispondere ad una sola sfida di V, infatti, se siamo nel primo caso, H è costruito isomorfo a G_0, quindi P riuscirà a rispondere correttamente solo se la scelta del bit da parte di V sarà zero (V accetterà con probabilità uguale ad $\frac{1}{2}$); se siamo nel secondo caso, P riuscirà a rispondere correttamente solo se V sceglierà il bit uguale a uno; nel terzo caso invece, P non potrà rispondere a nessuna sfida di V (V accetterà con probabilità uguale a zero). Complessivamente V accetta la prova di P' con probabilità $\leq \frac{1}{2}$. Per ridurre questa probabilità possiamo iterare t volte il protocollo facendola diventare $\leq \frac{1}{2^t}$.

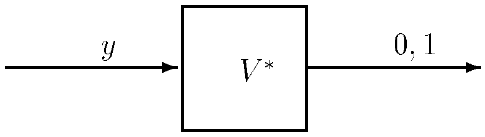
- **Conoscenza-zero:** concretamente quello che V vede durante il protocollo di comunicazione sono delle triple (H, b, ϕ) dove H è un grafo scelto a caso tra i grafi isomorfi a G_0 e G_1, b è un bit, ϕ è un isomorfismo tra H e G_b; quello che vogliamo dimostrare è che V può generare da solo, senza interagire con P, le stesse triple con la stessa distribuzione di probabilità. Formalmente per il verificatore V dobbiamo dimostrare l'esistenza di un algoritmo A_V polinomiale tale che

$$A_V(G_0 \approx G_1) = View_V(G_0 \approx G_1)$$

dove $View_V(G_0 \approx G_1)$ è una variabile casuale che rappresenta l'insieme di tutte le triple, ognuna con una certa probabilità. In effetti l'algoritmo A_V è in grado di costruire $View_V(G_0 \approx G_1)$ senza interagire con P. Vediamo come questo sia possibile nel nostro caso, ci sono due possibilità:

1. V segue fedelmente il protocollo per ottenere le triple
2. V non segue fedelmente il protocollo per ottenere le triple

Supponiamo che il verificatore segua fedelmente il protocollo, l'algoritmo A_V calcola un isomorfismo tra H e G_0. Successivamente sceglie un bit b a caso: se $b = 0$ è riuscito a costruire una tripla (H, b, ϕ) altrimenti fallisce. Ovviamente questo metodo non è efficace; poichè a noi interessa costruire triple che siano identicamente distribuite rispetto a quelle reali, si può anche non seguire il protocollo per ottenerle. Infatti, se il verificatore scegliesse prima b potrebbe calcolare un isomorfismo tra G_b ed H. Le triple ottenute sono statisticamente le stesse di quelle prodotte dal protocollo; infatti, b è scelto a caso tra due possibili valori e di conseguenza le altre due informazioni hanno le stesse distribuzioni di probabilità. Fino ad ora abbiamo fatto la supposizione che il verificatore era "buono", facciamo adesso la supposizione di avere a che fare con un verificatore "cattivo" l'algoritmo utilizzato da V' può essere visto come una *black box* che, preso in input y, dà in output un bit:



V^* per ottenere delle informazioni in più dalla conversazione con P, potrebbe non seguire correttamente il protocollo, per esempio potrebbe scegliere sempre $b = 1$ per avere maggiori conoscenze riguardanti l'isomorfismo tra G_0 *e* G_1. Facciamo vedere, però, che questo non è possibile, perchè esiste un algoritmo A_{V^*} che è in grado di generare le stesse informazioni di un verificatore cattivo V^*. Dimostriamo cioè che $\forall\ V^*\ \exists\ A_{V^*}$ tale che

$$A_{V^*}(G_0 \approx G_1) = View_{V^*}(G_0 \approx G_1)$$

Il modo in cui il simulatore effettivamente agisce è il seguente:

1. Sceglie un valore a caso $a \in \{0, 1\}$.

2. Si calcola un grafo H isomorfo a G_a attraverso una permutazione a caso σ.
3. Dà in input H a V^* il quale a sua volta gli restituisce un bit b.
4. Se $b = a$ allora genera una tripla (H, b, σ) altrimenti annulla la tripla e ripete il procedimento.

La probabilità con cui il simulatore genera una tripla valida è $\frac{1}{2}$ perchè tale è la probabilità che $b = a$. Quindi, il simulatore in media deve fare due tentativi per generare una tripla valida. Se il protocollo viene iterato t volte, questo procedimento viene ripetuto anch'esso t volte.

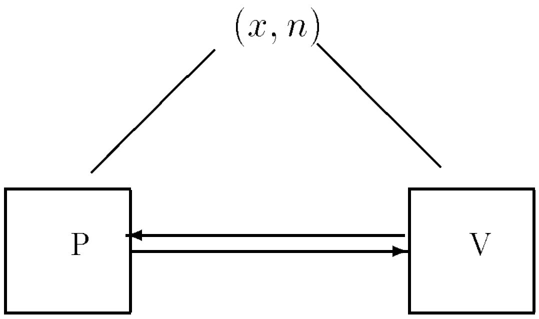
Nei sistemi di prova interattivi si prova anche la validità di un' altra proprietà, più precisamente si fanno delle "prove di conoscenza".

4.2.2 Prova di conoscenza

Se il provatore riesce a rispondere a tutte le domande in modo esatto allora o è molto fortunato oppure realmente conosce il segreto. Intuitivamente per verificare che il provatore sia onesto, cioè conosce realmente il segreto, si potrebbe fare in modo di chiedergli di rispondere a più domande in un unico round. Per fare ciò viene utilizzato un algoritmo chiamato "estrattore". Un sistema a prova di conoscenza è un sistema di prova interattivo che ammette un "estrattore". Un'estrattore è un algoritmo utilizzato dal verificatore che, utilizzando il provatore come un oracolo, riesce ad avere delle informazioni aggiuntive sulla dimostrazione dell'affermazione di cui il provatore vuole provare la veridicità. All'estrattore è permesso fare domande diverse, anche tutte insieme. Per esempio i protocolli per grafi isomorfi e non isomorfi visti precedentemente, non godono di questa proprietà, infatti non esiste un algoritmo estrattore che riesce ad ottenere informazioni aggiuntive sul segreto del provatore (il segreto è il modo in cui P riesce a costruire un isomorfismo o a verificarlo). Successivamente diamo alcuni esempi di sistemi di prova interattivi che godono di questa proprietà.

4.2.3 Prova a conoscenza-zero perfetta per quadrati residui

Problematica: Dato un intero n che è il prodotto di due primi p e q ed un intero $x \in \mathbf{Z}_n^*$, l'input in comune sarà la coppia (x, n)



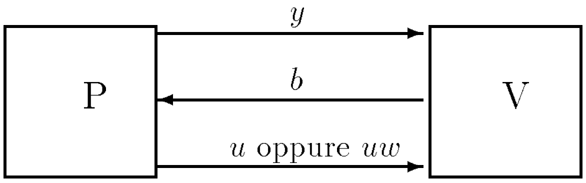
Il provatore vuole dimostrare al verificatore che x è un quadrato modulo n. Questo il verificatore non lo può fare perchè decidere la residuità quadratica è difficile ed il migliore algoritmo conosciuto consiste nel fattorizzare n e verificare che x è un quadrato sia modulo p sia modulo q.

Un modo con cui P può convincere V, è quello di inviargli una radice quadrata di x modulo

n; così però non verrebbe rispettata la proprietà di conoscenza-zero.
Uno schema che verifica le tre proprietà (completezza, soundness e conoscenza-zero) è il seguente:
Protocollo:

1. **P:** sceglie un numero a caso $u \in \mathbf{Z}_n^*$; calcola $y = u^2 \bmod n$ e lo invia a V (essendo y e x quadrati, lo sarà anche xy).
2. **V:** sceglie un bit a caso b e lo manda a P.
3. **P:** Se $b = 0$, invia a V u. Se $b = 1$, calcola w tale che $w^2 = x$, e risponde con uw che è una radice di xy.
4. **V:** verifica (per $b = 1$) che $(uw)^2 = xy$ oppure (per $b = 0$) che $u^2 = y$.

Graficamente, il provatore e il verificatore, si comportano nel seguente modo:



Verifichiamo la validità delle proprietà:

- **Completezza:** se l'affermazione è vera, cioè x è effettivamente un quadrato modulo n e se P esegue correttamente i passi del protocollo, allora riuscirà a convincere V con probabilità 1, perchè riesce a rispondere ad ogni sfida di V.
- **Soundness:** supponiamo che x non è un quadrato, i casi possibili sono i seguenti:
  1. Se y è un quadrato $\Rightarrow$ xy non è quadrato.
  2. Se xy è quadrato $\Rightarrow$ y non è quadrato.

  Nel primo caso, un provatore P', che cerca di indurre V ad accettare, può rispondere solo quando V sceglie zero, mentre, nel secondo caso, P' riesce a rispondere solo quando V sceglie uno. Quindi essenzialmente P' potrà rispondere solo ad una delle due sfide di V. Questo vuol dire che comunque P' scelga y, la probabilità che ha di convincere il verificatore V è $\leq \frac{1}{2}$. Se si itera il protocollo t volte, la probabilità diventa $\leq \frac{1}{2^t}$.
- **Conoscenza-zero:** formalmente dobbiamo dimostrare l'esistenza di un simulatore A_{V^*}, che, qualunque sia l'algoritmo del verificatore V^* (il quale potrebbe anche non seguire il protocollo, scegliendo ad esempio sempre b=0), riesce a generare il transcript della conversazione.
  Essenzialmente questo simulatore può far girare l'algoritmo di V^* dandogli in input i valori y. È chiaro che il simulatore deve essere una macchina polinomiale, in quanto deve

poter essere utilizzabile anche da V^* (macchina polinomiale e probabilistica). Quando il provatore onesto interagisce con V^* quello che si vede sono delle triple del tipo (y, b, z) dove b è la risposta che dà V^* quando ha in input y; z è tale che:

1. Se $b = 0 \Rightarrow z^2 = y$.
2. Se $b = 1 \Rightarrow z^2 = xy$.

Vogliamo che il simulatore generi una variabile casuale che sia perfettamente identica a $View_{V^*}$ ma questo non lo può fare se segue esattamente il protocollo. Quello che A_{V^*} può fare è costruirsi prima il valore di b e poi prepararsi a rispondere ad una domanda che dipende da questo valore. A_{V^*} sceglie dapprima un bit a, se $a = 0$ costruisce delle triple in cui $b = 0$, cioè $(y, 0, z)$, dove z deve essere esattamente la radice di y (si calcola prima z scegliendolo a caso in Z_n^* e poi $y = z^2$). Nel secondo caso $(a = 1)$ il simulatore genera delle triple del tipo $(y, 1, z)$ dove z è la radice quadrata di xy; anche in questo caso si calcola prima z e poi $y = z^2 x^{-1}$. Una volta calcolata una di queste due triple, A_{V^*} dà in input a V^* il valore y e controlla il bit di risposta: se questo corrisponde al bit calcolato nella sua tripla allora questa è valida, altrimenti annulla e ripete tutto il procedimento. Dobbiamo verificare che:

1. la tripla che ha costruito è identicamente distribuita alla tripla nella $View_{V^*}(x, n)$;
2. il tempo medio richiesto per il procedimento è polinomiale.

La prima è facile da verificare, perchè y è sempre un residuo quadratico modulo n scelto a caso tra tutti i residui quadratici; b è esattamente la risposta di V^* quando riceve in input y; z è o una radice quadrata di y oppure una radice quadrata di xy. Possiamo quindi concludere che le triple hanno la stessa distribuzione di quella originale.
Per quanto riguarda la seconda, bisogna tener conto che la probabilità che il simulatore in un passo riesca a costruirsi un tripla è $\frac{1}{2}$. Questa probabilità è indipendente da come V^* calcola il suo bit; anche se V^* calcolasse il suo bit per esempio sempre uguale ad 1, questa probabilità sarebbe sempre $\frac{1}{2}$ perchè a, scelto da A_{V^*}, ha probabilità $\frac{1}{2}$ di essere uguale a b (a è indipendente dalla scelta di b). Per questo motivo in media il simulatore impiegherà due tentativi per costruire una tripla. Se il protocollo viene iterato t volte questo processo del simulatore deve essere iterato t volte, sempre utilizzando un bit casuale ed indipendente dai precedenti; la variabile quindi, è identicamente distribuita a quella generata dalla interazione tra P e V.

- **Prova di conoscenza della radice quadrata:** il protocollo in analisi, verifica tale proprietà. Il provatore, non solo dimostra che x è un quadrato, ma che conosce anche la sua radice. Questo concetto di conoscenza può essere esposto nel seguente modo: supponiamo che il provatore sia limitato computazionalmente (macchina polinomiale); se ha successo nell'eseguire questo protocollo allora deve necessariamente conoscere la radice quadrata di x. Questo perchè mandando y a V, P è in grado di rispondere ad entrambe le domande del verificatore. In effetti, se ci riesce in tutte le iterazioni, allora o è estremamente fortunato perchè V gli fa sempre la domanda a cui sa rispondere, oppure effettivamente P riesce a calcolare sia la radice di y che la radice di xy, cioè riesce a calcolare r e s tali che:

$$r^2 = y$$

$$s^2 = xy$$

e quindi può calcolarsi la radice quadrata di x, in quanto $x = (sr^{-1})^2$. In questo protocollo l'estrattore può chiedere sia la radice di y che la radice di xy; se il provatore è in grado di rispondere a tutte le domande dell'estrattore allora conoscerà la radice quadrata di x.

Per utilizzare realmente il protocollo, bisogna verificare che sia sufficiente per P una potenza di calcolo polinomiale. Questo è vero in quanto le operazioni che P deve svolgere sono elevamenti a potenza e prodotti che possono essere computati facilmente se conosce w.

4.2.4 Prova a conoscenza-zero perfetta per logaritmi discreti

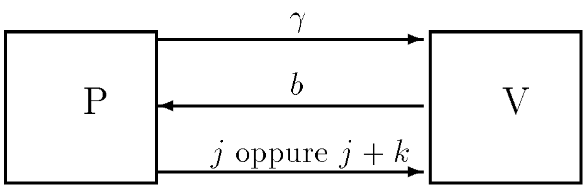
Problematica: Dato un intero positivo n e due elementi distinti α e $\beta \in Z_n^*$, P deve dimostrare che esiste un k tale che $\beta = \alpha^k \bmod n$.

Protocollo:

Input: Un intero positivo n e due elementi distinti α di ordine l, $\beta \in Z_n^*$

1. **P:** sceglie un valore a caso j tale che $1 \leq j \leq l-1$, calcola $\gamma \equiv \alpha^j \bmod n$ e lo manda a V.
2. **V:** sceglie un intero a caso $b \in \{0, 1\}$ che manda a P.
3. **P:** se $b = 0$, manda j a V, invece se $b = 1$ gli manda $j + k \bmod l$.
4. **V:** controlla che (per $b = 0$) $\alpha^j = \gamma$ oppure (per $b = 1$) $\alpha^{j+k} = \gamma\beta \bmod n$.

Graficamente si può riassumere nel seguente modo:



Vediamo quali sono le proprietà del protocollo:

- **Completezza:** supponiamo che P e V eseguano tutti i passi del protocollo correttamente, allora P sarà in grado di rispondere sempre alle sfide di V in modo esatto in quanto riesce a fornire l'esponente del logaritmo discreto. Pertanto V accetta la prova di P con probabilità uguale a uno.

- **Soundness:** supponiamo che il provatore non sia onesto e quindi non esista un valore k tale che $\alpha^k \equiv \beta \bmod n$. P allora invia a V un valore γ. Si possono verificare due diverse situazioni:

caso 1: supponiamo che non esista un valore j tale che $\alpha^j = \gamma$ allora se V sceglie il valore $b = 0$ P non può rispondere perchè non esiste j. Al contrario, se V sceglie il valore $b = 1$ P potrebbe rispondere inviando tale j.

caso 2: supponiamo che esista j tale che $\alpha^j = \gamma$, allora, se P riceve $b = 0$ da V, può rispondere inviando proprio j. Se invece P riceve $b = 1$, quello che può fare è indovinare un valore h che successivamente V andrà a verificare controllando che $\alpha^h = \alpha^j\beta$. Se però valesse questo allora si avrebbe che $\alpha^{h-j} = \beta$ contraddicendo l'ipotesi iniziale della non esistenza di un esponente k per α.

In conclusione: se γ è una potenza di α, P non sa rispondere per $b = 1$ e se γ non è una potenza di α, P non sa rispondere per $b = 0$. La probabilità di successo di P allora è $\leq \frac{1}{2}$ in ogni caso.
Per ottenere una probabilità ancora più bassa basta iterare più volte il processo (così come è stato proposto per i protocolli visti in precedenza).

- **Conoscenza-zero:** vogliamo dimostrare che esiste un simulatore in grado di generare una tripla (del tipo generato dal protocollo) senza interagire con P ottenendo, quindi una variabile casuale perfettamente identica a $View_{V^*}$. Fare questo è semplice, quello che V vede è una tripla del tipo (γ, b, q) dove se $b = 0$: $\alpha^q = \gamma$ e se $b = 1$: $\alpha^q = \gamma\beta$; il simulatore allora può generare a caso $b \in \{0,1\}$ e scegliere γ e q opportuni (essendo noti α e β), se quello che ha ottenuto è una tripla buona il processo termina altrimenti si itera. La probabilità con cui il simulatore genera una tripla valida è $\frac{1}{2}$. Il simulatore in media farà quindi due tentativi per generare una tale tripla.

- **Prova di conoscenza di un logaritmo discreto:** senza formalizzare si può dire che il protocollo è a prova di conoscenza poichè se $b = 0$, P invia j a V, se $b = 1$, P invia $j + k$ a V quindi solo se conosce k è in grado di rispondere a entrambe le richieste di V. In questo caso l'algoritmo "estrattore", durante lo stesso round, manda a P sia $b = 1$ che $b = 0$, quindi chiede a P di rispondere ad entrambe le domande e dalle sue risposte può calcolarsi k.

4.2.5 Protocollo per grafi 3-colorabili

3Colorabilità: Un grafo è 3-colorabile se è possibile dare ad ogni nodo un colore scelto tra tre (per esempio giallo-rosso-verde), in maniera tale che comunque scelti due nodi che sono connessi da un arco, questi hanno colori diversi. Il problema della 3-colorabilità è $NP - C$ e quindi in questo caso parleremo di conoscenza-zero di tipo computazionale. Vediamo un protocollo in cui P e V hanno come input comune un certo grafo G, che è 3-colorabile.
Problematica: P vuole identificarsi a V dimostrandogli che il grafo G è 3-colorabile e che conosce una sua colorazione.

Protocollo:

1. **P:** colora i vertici del grafo scegliendo una tricolorazione a caso e nasconde i colori in buste. Il grafo così costruito viene mandato a V.
2. **V:** sfida P scegliendo un arco a caso e chiedendogli di aprire le buste corrispondenti.
3. **P:** apre le buste.
4. **V:** verifica che i colori contenuti nelle buste sono diversi.

Mostriamo ora le operazioni che farebbe il provatore nel nostro esempio. Dopo la tricolorazione ottiene:

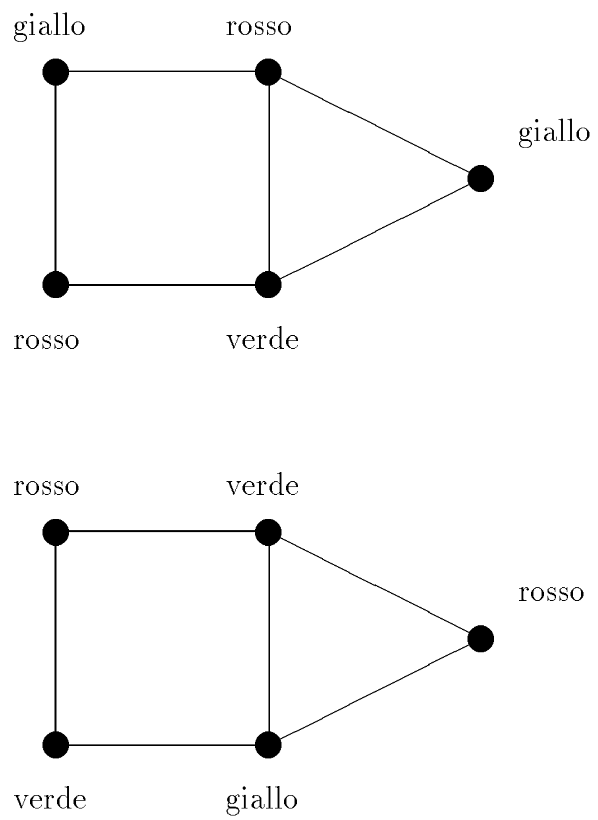
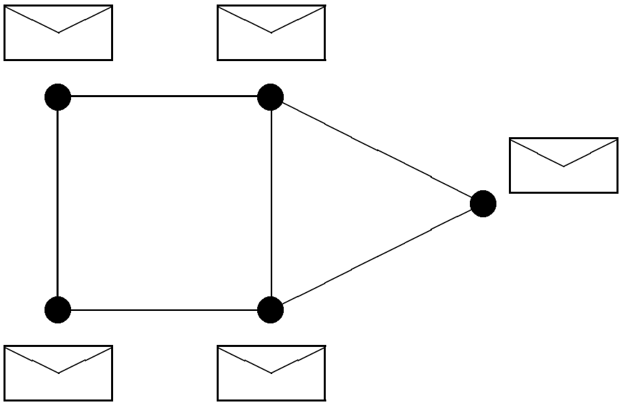


Figura 1: Esempio di grafo 3-colorabile



Vediamo ora quali sono le proprietà soddisfatte:

- **Completezza:** effettivamente se il provatore segue correttamente il protocollo, allora riesce sempre a convincere il verificatore. Questo perchè il grafo all'origine era 3-colorabile, P potrà quindi soddisfare ogni richiesta di V.

- **Soundness:** supponiamo che il grafo non è 3-colorabile e quindi non è vero che per coppie di vertici connesse da un arco, le buste nascondono colori diversi; in questo caso c' è almeno una coppia di buste in cui o ci sono colori uguali oppure non c' è nessun colore associato. La probabilità che ha il provatore "cattivo" di ingannare il verificatore è $\leq 1 - \frac{1}{m}$, se $m = |E|$ dove E è l'insieme degli archi di G (la probabilità è minore o uguale ad $1 - \frac{1}{m}$ perchè c' è almeno un arco che non verifica i requisiti della 3-colorabilità). Iterando il protocollo m^2 volte la probabilità che P' ha di ingannare V è $\leq (1 - \frac{1}{m})^{m^2} \approx e^{-m}$.

- **Conoscenza-zero:** dal punto di vista intuitivo il verificatore vede un grafo con le buste chiuse; successivamente sceglie un arco e vede i colori (diversi) contenuti in esse; in realtà quello che vede sono due colori scelti a caso, questo tipo di informazione poteva calcolarsela da solo, cioè poteva scegliersi un arco e associargli due colori scelti a caso e coprire con le buste tutti quanti gli altri. Formalmente dobbiamo far vedere come opera un simulatore avendo a disposizione queste buste: quello che si vede sono delle triple del tipo
$$(G_{busta}, (u, v), [C(u), C(v)])$$
dove G_{busta} è un grafo tricolorato con le buste, (u, v) è un arco, $C(u)$ e $C(v)$ sono i colori associati ad (u, v). Supponiamo di voler simulare il protocollo: la prima cosa che può fare il simulatore S (visto che non conosce i colori nelle buste) è scegliere a caso un arco (u, v), scegliere a caso i colori da associare ad u e v e metterli nelle buste; nelle altre, o metterà colori a caso, oppure metterà sempre lo stesso colore. Supponiamo che metta sempre lo stesso colore. A questo punto presenta il grafo con le buste al verificatore, che sceglie un arco a caso: se l'arco scelto è proprio (u, v) (quello che S aveva preparato) allora non ci sono problemi (in quanto S può aprire le buste e mostrare che i colori dei nodi sono differenti). Se, però, l'arco scelto da V fosse diverso dall'arco (u, v), S ripeterebbe il processo. Poichè la probabilità che V scelga esattamente l'arco (u, v) è $\frac{1}{m}$, il simulatore deve effettuare, in media, m tentativi per riuscire a calcolare una tripla. Questa variabile è "essenzialmente" la stessa della variabile originale. Diciamo "essenzialmente" perchè

quello che c'è nelle buste è diverso nei due casi: in un caso c'è una reale permutazione dei colori, nell'altro ci sarà sempre lo stesso colore; in entrambi i casi questi vengono coperti da buste, quindi, dal punto di vista del verificatore questi due casi non sono distinti tra loro. Questo è un protocollo a conoscenza-zero computazionale, il motivo per il quale abbiamo trattato questo protocollo, basato su un problema $NP-C$, risiede nel fatto che nella teoria della complessità la dimostrazione dell'esistenza di un protocollo a conoscenza-zero perfetta per un linguaggio $NP-C$ implicherebbe risultati improbabili (del tipo P=NP).

- **Prova di conoscenza di una 3-colorazione del grafo:** intuitivamente se P riesce a soddisfare le richieste di V per m^2 volte, allora o è molto fortunato oppure veramente conosce una 3-colorazione del grafo. Dal punto di vista formale l'algoritmo "estrattore" riesce a determinare una 3-colorazione del grafo chiedendo a P di aprire tutte le buste relative agli archi durante lo stesso round.

L'importanza dei protocolli basati su problemi $NP-C$ è puramente teorica: questo perchè l'esistenza di un "testimone" (segreto di un utente) fa sì che sia possibile per l'utente, che possiede il segreto, dimostrare che sta seguendo correttamente il protocollo senza esporre il proprio segreto.

4.2.6 Prova di conoscenza per grafi hamiltoniani

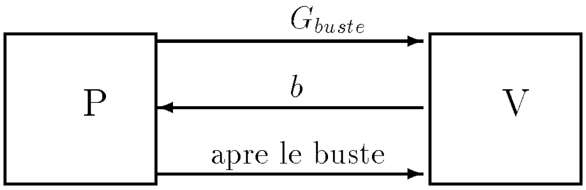
Grafi hamiltoniani: Un grafo non diretto G si dice hamiltoniano se esiste un ciclo che attraversa tutti i vertici una ed una sola volta. Determinare se un grafo è hamiltoniano o meno, è un problema $NP-C$.

Problematica: il provatore vuole mostrare al verificatore che un certo grafo G (che è dato in input ad entrambi) è hamiltoniano, conoscendone il ciclo.

Protocollo:

Input: un grafo G.

1. **P:** costruisce un grafo completo G' permutando a caso i vertici del grafo G e rappresentando gli archi attraverso dei singoli bit ('1' se l'arco esiste, '0' altrimenti). P nasconde sia i vertici che gli archi di G' in buste chiuse (in pratica, questi vengono cifrati); invia il grafo G' cosìcifrato a V.
2. **V:** vede un grafo con n buste che riguardano i vertici ed $\binom{n}{2}$ buste che riguardano gli archi, sceglie un bit b a caso e lo manda a P
3. **P:** se $b = 0$, P dimostra che G' è stato costruito correttamente seguendo il protocollo e risponde aprendo tutte le buste e mostrando la permutazione utilizzata. Se $b = 1$, P dimostra l'esistenza di un ciclo hamiltoniano nel grafo cifrato aprendo le buste relative agli archi che formano un ciclo hamiltoniano, non fornendo in tal modo, alcuna informazione addizionale a V dato che non apre le buste relative ai vertici del ciclo hamiltoniano.



Vediamo le proprietà soddisfatte da questo protocollo:

- **Completezza:** se il grafo è realmente hamiltoniano e P esegue correttamente il protocollo, convincerà sempre V perchè, nel caso in cui V sceglie $b = 0$ potrà far vedere che il grafo con le buste nasconde una copia isomorfa del grafo originario; nel caso in cui V sceglie $b = 1$, P sarà in grado di far vedere che il grafo cifrato contiene un ciclo hamiltoniano perchè conosce il ciclo hamiltoniano del grafo originario.
- **Soundness:** supponiamo che $G \notin HAM$, vogliamo determinare la probabilità con cui P' (che può alterare il protocollo in maniera arbitraria) può ingannare V; se quest'ultimo segue fedelmente il protocollo, allora ha uguale probabilità di chiedere una delle due domande. P' potrà rispondere alla prima domanda ($b = 0$) costruendo G' isomorfo a G, oppure alla seconda ($b = 1$) modificando il protocollo in uno dei seguenti modi:

  1. mette tutti valori '1' nelle buste relative agli archi;
  2. mette tutti valori '1' nelle buste di un ciclo scelto a caso e valori casuali nelle altre;

  In ogni caso potrà rispondere solo alla seconda domanda, perchè, se esiste un ciclo hamiltoniano in G', questo non può essere isomorfo a G. Da ciò si deduce che la probabilità che ha P' di ingannare V è $\leq \frac{1}{2}$ che è indipendente dal comportamento e dalle capacità computazionali di P'. La disuguaglianza è dovuta al fatto che P' potrebbe mettere tutti valori '0' nelle buste, avendo probabilità nulla di ingannare V. Per diminuire questa probabilità possiamo iterare il processo t volte, portando la probabilità ad essere $\leq \frac{1}{2^t}$. Questo tipo di prova ha bisogno di un minor numero di iterazioni rispetto a quelle necessarie nel protocollo basato sulla 3-COL dei grafi (in quel caso la probabilità di ingannare era $1 - \frac{1}{m} > \frac{1}{2}$).

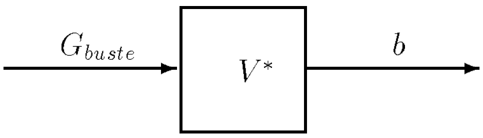
- **Conoscenza-zero:** intuitivamente questa proprietà è soddisfatta perchè il verificatore può riprodurre quello che vede, cioè:

  1. una copia isomorfa del grafo G i cui vertici ed archi sono nascosti nelle buste;
  2. un ciclo hamiltoniano ed un insieme di buste che coprono i rimanenti archi.

Queste informazioni possono essere calcolate da V stesso; infatti, per il primo caso può costruire una copia isomorfa di G, nel secondo caso può calcolarsi un ciclo hamiltoniano in quanto G' è un grafo completo. Dal punto di vista formale dobbiamo dimostrare l'esistenza di un algoritmo, simulatore (S), che riesce a generare la stessa conversazione che avviene tra P e V. Questa è costituita da triple del tipo

$$(G_{buste},\ b,\ \text{“apri buste“})$$

dove G_{buste} è il grafo cifrato, b è il bit scelto da V; "apri buste" corrisponde all'apertura di tutte le buste se $b = 0$, oppure all'apertura di n buste se $b = 1$; questo perchè nel caso in cui $b = 0$, P deve mostrare a V che G' é isomorfo a G, mentre nel caso in cui $b = 1$, P deve mostrare un ciclo hamiltoniano del grafo cifrato G_{buste}. Il simulatore deve generare triple di questo tipo che abbiano la stessa distribuzione di probabilità di quelle realmente generate. Supponiamo di avere un verificatore V* (di cui non conosciamo il comportamento; potrebbe ad esempio scegliere sempre zero), il quale, avendo in input un grafo con le buste, restituisce in output un bit:



Vediamo quali sono le operazioni eseguite da S per generare triple distribuite come quelle reali. Dapprima sceglie a caso un bit a: se $a = 0$ prepara una tripla in cui $b = 0$; G_{buste} lo costruisce permutando i vertici di G. "Apri buste" è la permutazione utilizzata nella costruzione di G_{buste}. Se $a = 1$ prepara una tripla in cui $b = 1$; G_{buste} è un grafo completo costruito sui vertici di G; "apri buste" mostra l'esistenza di un ciclo hamiltoniano che può essere ottenuto in uno dei seguenti modi:

- Dopo aver scelto un ciclo a caso nel grafo completo mette nelle buste relative agli archi di questo ciclo dei valori uguali ad 1, e poi lascia chiuse le altre buste.
- Mette nelle buste relative agli archi tutti valori uguali ad 1 ed apre solo le n buste relative ad un ciclo (essendo il grafo completo, questo lo può fare).

Quindi, dopo aver generato una tripla dà in input a V* il grafo G_{buste} e se il valore di b (dato in output da V*) è uguale al valore di a scelto da S la tripla generata è valida, altrimenti la annulla e ripete il procedimento. La probabilità con cui S trova una tripla valida è $\frac{1}{2}$, questo perchè il verificatore V* ha probabilità $\frac{1}{2}$ di generare il suo bit b uguale ad a. In media S con due tentativi riesce a trovare una buona tripla. Questo processo viene iterato t volte avendo scelto un numero di round uguale a t.

- **Prova di conoscenza di un ciclo hamiltoniano:** intuitivamente questa proprietà è verificata perchè, a meno che P non sia così fortunato da poter rispondere ad ogni domanda nelle diverse iterazioni, concludendo con successo il protocollo, deve necessariamente conoscere il ciclo hamiltoniano del grafo G. Formalmente dovremmo dimostrare l'esistenza di una macchina polinomiale, estrattore, che interagendo con P ed avendo la possibilità di fare più domande nello stesso round, riesca a calcolarsi il ciclo hamiltoniano del grafo G.

Questo protocollo potrebbe essere utilizzato per realizzare uno schema di identificazione (anche se, in realtà, non è pratico) in cui P genera in qualche modo un grafo hamiltoniano (potrebbe prendere n nodi, calcolarsi un ciclo su questi nodi, ed aggiungere degli archi a caso), lo mette in un file pubblico e per l'identificazione dimostra di conoscere il ciclo hamiltoniano del grafo. In genere nell'ambito dei protocolli di identificazione si genera un'istanza di un problema difficile partendo dalla soluzione e successivamente, nella fase di identificazione, si dimostra di essere in grado di risolvere il problema. Nel nostro caso, qualunque altra persona che voglia identificarsi come P, a meno che non sia così fortunato, deve necessariamente conoscere il ciclo hamiltoniano. La 'non-praticità' del protocollo è dovuta al fatto che per diminuire la probabilità di ingannare V, bisogna iterare il protocollo un numero di volte abbastanza elevato.

4.3 Sistema di prova a conoscenza-zero di Feige-Fiat-Shamir

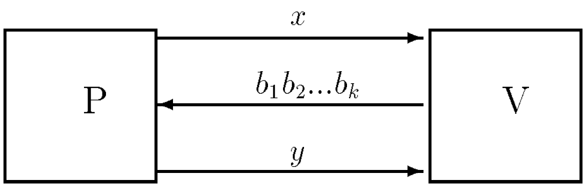
Questo protocollo, realmente utilizzato, si basa sui quadrati residui modulo n. La costruzione della chiave da parte di P avviene nel modo seguente:

1. P sceglie un valore n che è il prodotto di due numeri primi p e q ($n = pq$).
2. Sceglie k valori $v_1, v_2, ..., v_k$ appartenenti a Z_n^* che sono tutti quadrati modulo n.
3. Calcola i valori $s_1, s_2, ..., s_k$ tali che $s_i^2 = \frac{1}{v_i}$ per $i = 1, 2, ..., k$.

I valori n,$v_1, v_2, ..., v_k$ diventano la chiave pubblica, mentre i valori $s_1, s_2, ..., s_k$ costituiscono la chiave privata.

1. P sceglie un valore a caso $r \in Z_n^*$ e poi calcola il valore $x = r^2 \bmod n$ che viene mandato a V.
2. V sceglie a caso una stringa $b_1 b_2 ... b_k$ con $b_i \in \{0,1\}$ e la manda a P.
3. P manda a V il valore $y = r s_1^{b_1} s_2^{b_2} ... s_k^{b_k} \bmod n$.
4. V controlla che $x = y^2 v_1^{b_1} v_2^{b_2} ... v_k^{b_k} \bmod n$.

Il comportamento del provatore P e del verificatore V può essere schematizzato nella seguente figura:



L'identificazione di P a V, attraverso questo protocollo, avviene dimostrando che la persona che ha come chiave pubblica $(n, v_1, v_2, ..., v_k)$ è proprio P (lo dimostra facendo vedere a V che conosce la radici quadrate s_i). Quando V sceglie, durante il protocollo, i valori $b_1, b_2, ..., b_k$, quello che succede è che alcuni di questi b_i sono uguali a zero; di conseguenza, nel calcolo di y, i valori corrispondenti s_i non vengono considerati; per questo motivo P calcola il prodotto di un sottoinsieme delle k radici e lo moltiplica per r. V, a sua volta, controllerà che P ha calcolato y nel modo corretto, verificando che:

$$\begin{aligned}
x &= y^2 v_1^{b_1} v_2^{b_2} ... v_k^{b_k} \\
&= r^2 (s_1^2)^{b_1} (s_2^2)^{b_2} ... (s_k^2)^{b_k} v_1^{b_1} v_2^{b_2} ... v_k^{b_k} \\
&= r^2 \left(\frac{1}{v_1}\right)^{b_1} \left(\frac{1}{v_2}\right)^{b_2} ... \left(\frac{1}{v_k}\right)^{b_k} v_1^{b_1} v_2^{b_2} ... v_k^{b_k} \\
&= r^2 \bmod n.
\end{aligned}$$

Se il provatore esegue correttamente il protocollo allora riesce sempre a calcolarsi questi valori. Se un utente volesse impersonificare P e conoscesse in anticipo i valori $b_1, b_2, ..., b_k$ scelti da V, potrebbe calcolare una coppia di valori (x, y) tali che:

$$x = y^2 v_1^{b_1} v_2^{b_2} ... v_k^{b_k}.$$

Per fare questo basta scegliere un valore a caso di y e valutare x. Per conoscere i valori b_i in anticipo, il provatore potrebbe indovinarli a caso, ma la probabilità di indovinarli è $\frac{1}{2^k}$ che è la probabilità di ingannare il verificatore. Se iteriamo t volte il protocollo questa probabilità diventa $\frac{1}{2^{kt}}$.

- **Conoscenza-zero:** Ciò che vede il verificatore durante le iterazioni del protocollo sono delle triple del tipo $(x, b_1b_2...b_k, y)$. Queste informazioni possono essere ottenute da un simulatore nel seguente modo: ad ogni round il simulatore sceglie una sequenza a caso $b_1b_2...b_k$, sceglie un $y \in Z_n^*$ e calcola il corrispondente valore di x. Il valore x viene dato in input a V^*. Se V^* sceglie la stessa sequenza $b_1b_2...b_k$ il simulatore termina il processo con successo e dà in output la tripla $(x, b_1b_2...b_k, y)$, altrimenti scarta la tripla e ripete il processo. La probabilità che ha V^* di scegliere la stessa sequenza $b_1b_2...b_k$ scelta dal simulatore è $\frac{1}{2^k}$, quindi il tempo medio necessario per calcolare una tripla esatta è 2^k. Visto che il simulatore deve essere polinomiale allora 2^k deve essere polinomiale in $\log n$. Pertanto $2^k \lessapprox (\log n)^c$ con c costante, da cui $k \lessapprox c \log\log n$ e cioè $k = O(\log\log n)$. Per quanto riguarda il parametro t anch'esso deve essere polinomiale nella lunghezza dell'input. Da notare che per t devono essere rispettati sia un limite superiore che un limite inferiore. Questo perchè se il numero di round è troppo basso la probabilità che avrebbe un provatore di ingannare il verificatore sarebbe alta. Nelle applicazioni reali Feige-Fiat-Shamir [7] suggeriscono di utilizzare $k = 5$ e $t = 4$.

5 Bit commitment

Problematica: Il problema che si vuole risolvere è quello di realizzare delle buste digitali (utilizzabili per esempio per nascondere gli archi e i vertici di un grafo). Come sempre abbiamo due utenti, P e V, che interagiscono tra loro: P manda a V la busta chiusa che contiene un bit, V riceve la busta ma non può aprirla fino a che P non gli invia la chiave. È importante il fatto che P non può in alcun modo modificare il contenuto della busta dopo averla inviata a V. Citeremo due protocolli che realizzano il bit commitment.
Protocollo 1
Parametri utilizzati:

- n calcolato come il prodotto di due primi ($n = pq$)
- y tale che non sia un quadrato modulo n.

P cifra i bit 0 ed 1 nel seguente modo:

- sceglie un valore a caso $w \in Z_n^*$
- per cifrare '1' calcola $x = w^2$ (che è un quadrato modulo n)
- per cifrare '0' calcola $x = w^2 y \bmod n$ (che è un non quadrato modulo n)
- nel momento in cui V vuole sapere il contenuto delle buste, P esibisce il valore w
- V, per risalire al bit cifrato, vede quale delle equazioni, $w^2 = x$ oppure $w^2 y = x$, è verificata. Se è verificata la prima allora stabilisce che il bit "nascosto" è 1, altrimenti il bit è 0.

Per evitare di avere informazioni preziose sul bit che vogliamo nascondere, sarà bene utilizzare valori diversi ogni volta che si cifra un arco. Questo sistema di cifratura, però, non può essere utilizzato nella pratica, perchè P dovrà memorizzare $\binom{|V|}{2}$ valori di w (uno per ogni arco) dove V è l'insieme dei vertici del grafo.
Dal punto di vista teorico questo sistema è valido e può essere utilizzato anche per altri tipi di problemi. Quando le buste vengono costruite in questo modo, si dice che si fa il **commitment** ad un valore, e la caratteristica principale è che questo valore non può essere cambiato in seguito; quindi nel nostro caso P non può, per esempio, "spacciare" un bit per un altro. Questo è dovuto al fatto che la verifica che un numero sia un quadrato o meno è un problema tanto difficile quanto quello della fattorizzazione (l'algoritmo migliore per la verifica si basa sulla fattorizzazione di n).

- **Soundness:** la soundness non si basa su nessun tipo di assunzione computazionale, in quanto il punto fondamentale di questa proprietà è il fatto che P non conosce quale sarà il bit che V gli manderà.

- **Conoscenza-zero:** il verificatore non conosce il contenuto delle buste. Se si utilizzano delle reali buste che soddisfano le nostre proprietà, il simulatore riuscirà a costruire delle triple perfettamente uguali a quelle originali, se invece si utilizza un sistema di commitment per realizzare le buste, allora il simulatore non riesce a costruire delle triple perfettamente identiche a quelle originali. Sono diverse in quanto, per determinati archi, il contenuto delle buste del simulatore può essere diverso dal contenuto delle buste reali (ad esempio il simulatore potrebbe mettere tutti uno nelle buste). Nonostante le due triple non siano uguali, affermiamo che il protocollo è a conoscenza-zero, in quanto dal punto di vista del verificatore continuano a non esserci delle differenze (vede due triple apparentemente uguali) e, quindi, poichè esiste un algoritmo simulatore che riesce a riprodurre (anche se non in maniera fedele) la comunicazione tra P e V in tempo polinomiale, non ottiene informazione addizionale.

- **Prova di conoscenza:** il verificatore, dopo aver interagito con il provatore, potrebbe cercare di sfruttare le informazioni acquisite per rompere il sistema di cifratura, ma, supponendo che sia una macchina polinomiale, lo riuscirebbe a fare in tempo altissimo (uno o due anni). Una volta rotto il sistema di cifratura, viene a conoscenza del segreto.

La considerazione precedente potrebbe causare problemi in alcuni protocolli, perciò ne sono stati inventati altri che invertono la difficoltà computazionale della soundness e della conoscenza-zero, cioè la conoscenza-zero sarà di tipo perfetto, mentre la soundness si baserà sul fatto che il provatore è una macchina polinomiale. Quindi anche se il verificatore volesse rompere il sistema, non ci riuscirebbe mai.

Protocollo 2: Supponiamo di avere due utenti A e B che interagiscono in questo modo:

1. B si calcola un numero n che è il prodotto di due numeri primi p e q ($n = pq$) e poi si calcola un quadrato $z = r^2 \bmod n$ dove r è scelto a caso in Z_n^*, ed invia la coppia (n, z) ad A.

2. A si calcola un valore a caso $w \in Z_n^*$. Successivamente si calcola un valore x, come commitment ad un valore 0 oppure ad un valore 1, nel seguente modo:

   (a) se A deve cifrare uno zero, $x = w^2 \bmod n$;

   (b) se A deve cifrare un uno, $x = zw^2 \bmod n$.

3. A farà vedere (quando richiesto da B) che x è una cifratura del bit 0 oppure del bit 1 mandando a B il valore w.

4. B stabilisce il bit cifrato con x dalla equazione che è verificata tra le due $x = w^2$ e $x = zw^2$.

- **Soundness:** dimostriamo che questo protocollo funziona bene se il provatore, in questo caso A, è computazionalmente limitato. Supponiamo che $x = w^2$ e che A voglia cambiare idea sul bit cifrato da x (vuole mandare a B un 1 al posto di uno 0); per fare questo A dovrebbe calcolare un valore $\bar{w}$ (che manda poi a B) tale che $x = z\bar{w}^2$. Se ci riuscisse allora A avrebbe calcolato due valori w, $\bar{w}$ tali che $w^2 = z\bar{w}^2$ e di conseguenza avrebbe calcolato la radice quadrata di z ($(\frac{w}{\bar{w}})^2 = z$). Poichè il calcolo della radice quadrata modulo n (numero composto) è equivalente alla fattorizzazione di n, si deduce che A non può calcolare il valore $\bar{w}$ e quindi non può riuscire a cambiare idea su nessun bit perchè limitato computazionalmente. Questo significa che, costruendo le buste in questo modo, la proprietà di soundness è verificata anche se non più incondizionatamente come prima (la soundness si basa sul fatto che il provatore non può estrarre la radice quadrata).

- **Conoscenza-zero:** le variabili reali e quelle generate dal simulatore sono identiche perchè le buste sono dei quadrati ($x = w^2$ oppure $x = zw^2$) scelti a caso tra tutti i possibili quadrati modulo n.

Le proprietà di conoscenza-zero perfetta e soundness incodizionata per problemi $NP - C$ non si possono ottenere entrambe. Il fatto che P sia una macchina polinomiale significa che non riesce a risolvere i problemi difficili durante l'esecuzione del protocollo con V (non ha importanza che P possa riuscirvi in un momento successivo).

Riferimenti

[1] C. P. Schnorr, *Efficient signature generation by smartcards*, Journal of Cryptology, vol. 4, 1991, pp. 161-174.

[2] S. Goldwasser, S. Micali, C. Rackoff, *The knowledge complexity of interactive proof systems*, Proceedings of the 17th ACM Symposium on the Theory of Computing, 1985.

[3] O. Goldreich, S. Micali, A. Wigderson, *Proofs that yield nothing but their validity and a methodology of cryptographic protocol design*, Proceedings of CRYPTO '86, Advanced in Cryptology, Springer-Verlang, Berlin, 1987.

[4] M. Blum, *How to prove a theorem so no one else can claim it*, Proceedings of the International Congress of Mathematicians, Berkeley CA, 1986.

[5] D. Chaum, J. H. Evertse, J. Van De Graff, *An improved protocol for demonstrating possession of discrete logarithms and some generalizations*, Proceedings, Springer-Verlang, Berlin, 1988.

[6] U. Feige, A. Fiat, A. Shamir, *Zero-knowledge proofs of Identity*, Journal of Cryptology, vol. 1, n. 2, 1988, pp. 77-94.

[7] A. Fiat., A. Shamir, *How to prove yuorself: Practical solutions to identification and signature problems*, Proceeding of CRYPTO '86, Lecture Notes in Computer Science, vol. 263, Springer-Verlang, Berlin, 1987, pp. 186-194.

[8] L. C. Guillou, J. J. Quisquater, *A practical zero-knowledge protocol fitted to security microprocessor minimizing both trasmission and memory*, Lecture Notes in Computer Science, vol. 330, Advances in Cryptology: Proceeding Eurocrypt '88, C. G. Günther, Ed., Davos, Switzerland, May 25-27, 1988, pp. 123-128, Springer-Verlang, Berlin, 1988.