Protocolli *Zero-Knowledge* (a Conoscenza Zero) e identificazione.

Fabrizio Luccio
Breve dispensa per il corso di Crittografia
Laurea triennale in Informatica, Novembre 2013.

1. Il concetto di zero-knowledge.

Nel marzo di quest'anno sono stati annunciati i vincitori per il 2012 del *Turing Award*, il più importante premio scientifico annuale per l'informatica, considerato equivalente al Premio Nobel per altre discipline: sono i professori **Shafi Goldwasser** e **Silvio Micali** del Massachusetts Institute of Technology. Per la prima volta riceve il premio un italiano, Micali appunto, ora anche cittadino americano. I loro studi che partono dal Dottorato di Ricerca a Berkeley (Micali si era prima laureato in matematica a Roma) e poi pubblicati nel 1984 sul Journal of Computer and Systems Sciences, hanno per primi illustrato l'importanza degli eventi casuali nello studio della crittografia dando poi luogo ai metodi di "dimostrazione a conoscenza zero". È dunque quanto mai opportuno parlarne oggi: chi volesse leggere una semplice e simpatica introduzione alle motivazioni del riconoscimento tributato a Goldwasser e Micali potrà riferirsi all'articolo [1] edito sul principale giornale dell'associazione ACM che ha istituito il premio.

Scopo iniziale di questa piccola dispensa è far comprendere il significato dello schema matematico detto *zero-knowledge* nell'ambito dei protocolli di scambio d'informazione tra un *prover* **P** (dimostratore) e un *verifier* **V** (verificatore). Il termine stesso nacque in una presentazione di Goldwasser, Micali e Rackoff nel convegno *STOC* 1985, il più importante evento annuale di informatica teorica, e la teoria ebbe poi completo sviluppo nell'articolo [2]. Questo articolo è molto complesso e adatto a chi volesse approfondire completamente l'argomento affrontandolo con una solida conoscenza di computabilità e complessità di calcolo. Altre fonti sono di più semplice lettura ma in genere piuttosto deludenti: si salva, senza alcuna pretesa di profondità, la voce "Zero Knowledge" di Wikipedia. Compreso il senso del discorso ridiscuteremo poi come impostare un protocollo di identificazione e come l'introduzione di un meccanismo di zero-knowledge ne consenta la massima sicurezza.

La caratteristica principale di uno scambio di messaggi zero-knowledge è che il prover P riuscirà a dimostrare al verifier V di essere in possesso di una facoltà particolare o di una conoscenza specifica su un argomento **senza comunicargli alcun'altra informazione** salvo l'evidenza del suo possesso di tale facoltà o conoscenza. Introduciamo l'argomento come un semplice gioco; un esempio diverso ma di simile stile può trovarsi in [3]. P afferma che, condotto su una qualsiasi spiaggia, è in grado con un'occhiata di contarne i granelli di sabbia (più seriamente potremmo dire che afferma di saper risolvere in tempo polinomiale un problema NP-hard). V verificherà che questa affermazione è vera con una probabilità arbitrariamente prossima a 1 (certezza) da lui stesso stabilita, senza però poter ricavare alcun'altra informazione sul metodo seguito da P per contare i granelli.

P e V adottano un protocollo iterativo di scambio concordato in precedenza, consistente in un numero $k+1$ di domande e risposte ove k è scelto da V. Per semplificare ulteriormente le cose ammettiamo che V si accontenti che le risposte di P siano costituite da un bit che indica se il numero di granelli è pari o dispari. All'iterazione $i = 0, 1, 2, \ldots, k$ la risposta deve essere $b_i = 0$ se il numero di granelli è dispari, $b_i = 1$ se è pari. Secondo il protocollo concordato V conduce P su una spiaggia a sua scelta e gli chiede di calcolare b_0

(come vedremo nulla importa se P conosce già la spiaggia e il suo numero di granelli). Quindi itera le seguenti operazioni: k volte se stabilirà che l'affermazione di P è vera, o arrestandosi prima se scoprirà che P è un impostore:

iterazione $i = 1, 2, \ldots, k$
V chiede a P di voltarsi;
sceglie un bit random e lanciando una moneta;
if $e = 0$ toglie un granello di sabbia e lo intasca **else** non fa nulla;
chiede a P di tornare a guardare la spiaggia e di calcolare il nuovo valore b_i;
if ($e = 0$ AND $b_i = \neg b_{i-1}$) OR ($e = 1$ AND $b_i = b_{i-1}$) (cioè se le risposte di P ai passi i e i-1 sono consistenti tra loro)
V passa all'iterazione i +1
else arresta il procedimento dichiarando che P è un impostore.

Se le k iterazioni vanno a buon fine V può stabilire che P ha la capacità asserita con probabilità $1 - 1/2^k$. Infatti se P fosse un impostore avrebbe a ogni iterazione i, una probabilità ½ di indovinare il valore corretto di b_i pur senza saperlo calcolare, e tutti questi valori si moltiplicano tra loro poiché sono indipendenti in quanto funzione dalla scelta casuale di e fatta da V e non nota a P. In questo caso cioè la probabilità complessiva che P possa ingannare V è $1/2^k$. Se invece P possiede la capacità asserita risponderà sempre esattamente e tutte le operazioni andranno a buon fine per qualunque valore k scelto da V.

Iniziamo ora uno studio generale del problema. I due interlocutori possono essere **onesti**, cioè sia P che V seguono esattamente il protocollo concordato e P possiede effettivamente la conoscenza affermata; o **disonesti** se qualcuna di queste caratteristiche è violata. Un protocollo zero-knowledge deve possedere le tre seguenti proprietà:

Completezza: se l'affermazione di P è vera, V ne accetterà sempre la dimostrazione.

Correttezza: se l'affermazione di P è falsa (P è disonesto), V può essere convinto della veridicità dell'affermazione solo con "bassa probabilità", cioè una probabilità $\leq 1/2^k$ per un valore arbitrario k scelto da lui stesso. Alternativamente la correttezza può essere riformulata dicendo che se P è onesto sarà in grado di convincerne V con probabilità $\geq 1 - 1/2^k$.

Conoscenza Zero: Se l'affermazione di P è vera nessun verificatore V, anche se disonesto (nell'uso del protocollo), può acquisire alcuna informazione su questo fatto salvo la sua veridicità.

Per quanto riguarda l'esempio della spiaggia pensiamo che le tre proprietà appaiano intuitivamente verificate senza darne una dimostrazione formale. In genere, come vedremo, la proprietà più difficile da dimostrare è la terza. Un esempio serio, tratto da uno dei primi articoli sull'argomento e qui un po' trasformato, riguarda il problema dell'*isomorfismo tra grafi* per cui non è noto un algoritmo polinomiale di calcolo anche se non è stato a oggi dimostrato che questo problema sia NP-completo (un algoritmo polinomiale potrebbe cioè esistere senza alcun impatto sulle classi di complessità).

Consideriamo due grafi $G_1 = (V_1,E_1)$, $G_2 = (V_2,E_2)$, con $|V_1| = |V_2| = n$: i vertici sono numerati da 1 a n e ogni spigolo è individuato dalla coppia di interi relativi ai vertici connessi. G_1 e G_2 sono **isomorfi** se esiste una permutazione π degli interi $\{1, \ldots, n\}$ tale che, applicata alla numerazione di V_1 e rinominati in conseguenza gli spigoli di E_1, risulta $E_1 = E_2$. Indicheremo questa proprietà con $G_2 = \pi\, G_1$. Determinare se G_1 è isomorfo a G_2 o, in

caso affermativo determinare la permutazione π, sono problemi esponenzialmente difficili. Ora il prover P prende un grafo arbitrario G_1 di *n* vertici e una permutazione random π di {1, …, *n*}, costruisce il grafo $G_2 = \pi\, G_1$ isomorfo a G_1, comunica i due grafi a V asserendo di conoscere la permutazione π ma senza rivelare null'altro a riguardo (in particolare senza rivelare π). Il protocollo prosegue con *k* iterazioni dei seguenti passi:

1. V **chiede** a P di iniziare un'iterazione;
2. P **genera** una permutazione random θ degli interi {1, …, *n*};
   genera il grafo $G_3 = \theta\, G_2$;
   comunica G_3 a V;
   \\ commento: si genera un nuovo valore di θ, quindi un nuovo grafo G_3, a ogni iterazione: il valore di θ può esere "bruciato" al passo 4
3. V **genera** un bit random *e* e lo comunica a P;
4. P **genera** la permutazione ρ tale che **if** ($e = 0$) $\rho = \theta$ **else** $\rho = \pi\, \theta$;
   comunica ρ a V;
   \\ commento: $\pi\, \theta$ è la permutazione che si ottiene applicando in sequenza π e θ su {1, …, *n*}; si noti che poiché θ è random la comunicazione di $\rho = \pi\, \theta$ nulla svela su π
5. V **genera** il grafo G_4 tale che **if** ($e = 0$) $G_4 = \rho\, G_2$ **else** $G_4 = \rho\, G_1$;
   if ($G_4 = G_3$) ripete dal passo 1
   else blocca il protocollo dichiarando che P è disonesto.

Le dimostrazioni di completezza e correttezza sono semplici; la dimostrazione di zero-knowledgess è molto più complicata ed è discussa nel seguito solo in via schematica.

1. Completezza. Se P è onesto la condizione $G_4 = G_3$ del passo 5 è sempre verificata e V accetta l'asserzione di P per qualunque valore di *k*.

2. Corrrettezza. Se P è disonesto e non conosce π può tentare di ingannare V inviandogli, al passo 2, il grafo $G_3 = \theta\, G_2$ se prevede di ricevere il valore $e = 0$ al passo 3, o il grafo $G_3 = \sigma\, G_1$ se prevede di ricevere $e = 1$, ove σ è una permutazione abitraria. Invierà poi a V rispettivamente $\rho = \theta$ o $\rho = \sigma$ al passo 4. Se la previsione di P su *e* è corretta, al passo 5, in entrambi i casi, V troverà $G_4 = G_3$ e accetterà l'iterazione; ma se la previsione è sbagliata G_4 risulterà diverso da G_3 e sarà scoperto l'inganno. La proprietà di correttezza segue dall'osservazione che, poiché *e* è generato a caso, le previsioni di P su *e* sono corrette con probabilità ½.

3. Zero-knowledgeness. La dimostrazione che V non scopre nulla su P salvo che conosce la permutazione π è molto complessa. Come per tutte le dimostrazioni di zero-knowlegness essa si basa sull'esistenza di un *simulatore* (Macchina di Turing) *probabilistico* disponibile a qualunque verifier onesto o meno, che, *senza accesso al prover*, può costruire una falsa sequenza di iterazioni tra P e V che con alta probabilità coincide con l'esecuzione di un protocollo corretto di accettazione. Il non accesso a P garantisce che V non può acquisire alcuna informazione su di esso.

Dalla proprietà di zero-knowledgeness deriva anche un'altra caratteristica molto interessante di questi protocolli. Dopo che un onesto V ha terminato i suoi scambi con un onesto P non può ripetere a un altro attore T (per esempio a un giudice) la dimostrazione di

veridicità dell'asserzione di P. Infatti V potrebbe eseguire una registrazione di tutti i passi del protocollo e mostrarla a T; ma questa registrazione potrebbe essere stata costruita ad arte da un disonesto V senza accedere a P, simulando il funzionamente di un protocollo truccato ove P conosce a priori le scelte (ora non) random di V e si comporta in conseguenza.
Nell'esempio sui grafi sopra descritto V potrebbe costruire una registrazione in cui P conosce a priori i bit *e* generati da V come descritto nella prova di correttezza.

2. L'identificazione.

Vediamo ora quale contributo sostanziale abbia dato il concetto di zero-knowledge ai protocolli crittografici di identificazione. Un prover P – nella pratica, spesso un singolo utente - deve dimostrare la propria identità a un verifier V - di solito un'organizzazione; e, idealmente, non deve svelare altro che la sua identità, né a V né a possibili intrusi. Anche qui ci limiteremo a una semplice introduzione discutendo il primo protocollo zero-knowledge proposto e rimandando per esempio al testo [4] per approfondimenti.

Per stabilire un confronto ricordiamo anzitutto le due tecniche base di identificazione non zero-knowledge descritte per esempio in [5], che sono largamente impiegate quando non occorra una maggiore sicurezza.

1. *Identificazione basata su password*. P e V concordano su una password w di P e l'identificazione avviene in un solo round con la spedizione di w da P a V. Per evitare che V memorizzi esplicitamente w che potrebbe così essere letta da un infedele appartenente alla sua organizzazione, la *tabella delle password* posseduta da V contiene due sequenze r, $h(rw)$ associate al nome di P, ove r è una sequenza random assegnata da V a P e $h(rw)$ è l'hash della concatenazione delle sequenze r e w calcolato con una funzione hash *one-way*, cioè "facile" da calcolare e "difficile" da invertire. Per identificare P, alla ricezione della password w il verifier V estrae r dalla tabella, calcola $h(rw)$ possibilmente in hardware, e confronta questo valore con quello assegnato a P nella tabella. Dunque la password w non è esplicitamente memorizzata in V e non può essere estratta "facilmente' da $h(rw)$.

2. *Identificazione basata su chiave pubblica*. In un sistema a chiave pubblica siano k_s, k_p rispettivamente le chiavi segreta e pubblica di P; e siano C, D le funzioni di cifratura e decifratura. Il cifrario deve essere "commutativo", cioè per qualsiasi messaggio m deve valere $D(k_s,C(k_p,m)) = C(k_p,D(k_s,m)) = m$ (ciò accade per esempio nel cifrario RSA). Per l'identificazione:

V invia a P una sequenza casuale r ;
P usa la funzione D e la sua chiave segreta per calcolare $r' = D(k_s,r)$ e invia r' a V;
V usa la funzione C e la chiave pubblica di P per calcolare $r'' = C(k_p,r')$;
if $(r'' = r)$ V identifica P.

Questi due metodi di identificazione non sono completamente sicuri anche se, da questo punto di vista, il secondo è assai migliore del primo ed è adottato nei protocolli https. Nel metodo basato su una password trasmessa in chiaro chiunque spii sulla linea, incluso un infedele appartenente all'organizzazione di V che si procuri un accesso alla porta d'ingresso, può rubare la password di P e può in seguito identificarsi al suo posto. La debolezza del metodo basato sulla chiave pubblica è più sottile e dipende dal fatto che V chiede a P di applicare la sua chiave segreta a una sequenza r che V stesso ha generato. Un disonesto verifier potrebbe generare una r non-random, ma scelta di proposito per ricavare qualche informazione su k_s attraverso l'esame di $D(k_s,r)$, soprattutto nel caso che P debba

identificarsi molte volte con V e questi possa quindi impiegare una serie mirata di scelte di r. Si può dimostrare che questo tipo di attacco può ridurre notevolmente il campo di variabilità noto a V in cui può trovarsi k_s. In sostanza questi due metodi soddisfano le condizioni di completezza e correttezza ma non quella di zero-knowledgess.

Il primo protocollo di identificazione zero-knowledge fu proposto da Fiat e Shamir e costituisce uno schema generale per gli studi successivi. Esso si basa sul problema dei residui quadratici nell'aritmetica modulare. Scelto n come modulo, un intero t è detto *residuo quadratico modulo n* se esiste un intero s tale che $(t = s^2) \bmod n$. Ovviamente dati n, s calcolare t è un'operazione polinomiale per qualsiasi n; ma è possibile dimostrare che per $n = pq$ con p, q primi l'operazione inversa, cioè calcolare la "radice quadrata" s dati n, t ha la stessa complessità della fattorizzazione di n, quindi non è noto un algoritmo polinomiale di calcolo. L'algoritmo di identificazione di Fiat-Shamir nella sua forma di base è il seguente.

P sceglie $n = pq$ con p, q primi, e un intero positivo $s < n$. Calcola $(t = s^2) \bmod n$. Comunica la coppia $<n,t>$ a chi dovrà identificarlo come una sorta di chiave pubblica: il suo segreto è la terna $<p,q,s>$ e nessuno di questi valori può essere desunto in tempo polinomiale da $<n,t>$. Quindi può partire il protocollo i cui passi, iterati per k volte, consentono a V di stabilire l'identità di P con probabilità $1-1/2^k$ senza nulla poter inferire sul segreto di P.

1. V **chiede** a P di iniziare un'iterazione;
2. P **genera** un intero random $r < n$ e **calcola** $(u = r^2) \bmod n$;
   comunica u a V;
   \\ commento: si genera un nuovo valore di r e quindi di u in ogni iterazione: il valore di r può essere "bruciato" al passo 4
3. V **genera** un intero random e in $\{0,1\}$;
   comunica e a P;
4. P **calcola** $(z = r s^e) \bmod n$;
   comunica z a V;
   \\ commento: se $e = 0$ si ha $z = r$; se $e = 1$ si ha $(z = rs) \bmod n$
5. V **calcola** $(x = z^2) \bmod n$;
   if $(x = u t^e) \bmod n$ ripete dal passo 1
   else blocca il protocollo senza identificare P;

Le dimostrazioni di completezza, correttezza e zero-knowledgeness sono lasciate al lettore come estensione di quelle già viste per il problema dell'isomorfismo di grafi. Per una spiegazione esauriente si veda [4].

[1] N. Savage. Proofs Probable. *Communications of the ACM*, Vol.56, 6 (2013) pp. 23-24.

[2] O. Goldreich, S. Micali e A. Widgerson. Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. *Journal of the ACM*, Vol.38, 1, pp. 691-729.

[3] J-J. Quisquater et al. How to Explain Zero-Knowledge Protocols to Your Children. In: *Advances in Cryptology* – Proceedings CRYPTO '89, Santa Barbara, CA (1989) pp. 628-631.

[4] H. Delfs e H. Knebl. *Introduction to Cryptography*. Cap. 4. Springer, Berlin 2007.

[5] P. Ferragina e F. Luccio. *Crittografia*. Cap. 9. Bollati-Boringhieri, Torino 2001, rist. 2007.