Contenuto di questo fascicolo

- Generatore di numeri pseudo-casuali Blum-Blum-Shub
- Cifrario sistema di Blum-Blum-Shub.

Il presente fascicolo contiene la lezione di Crittografia di martedì 7 maggio 2019. Scaricatelo e portatevelo in aula FS.

7. IL GENERATORE DI BLUM BLUM SHUB

7.1. I Numeri di Blum

Definizioni.

- Un **primo di Blum** è un numero primo p tale che $p \equiv 3 \pmod 4$.
- Un **intero di Blum** è un numero $n = pq$ dove p e q sono primi di Blum.

7.1.1. Proprietà dei Numeri di Blum

In questa sezione proveremo alcune delle proprietà degli interi di Blum che sono necessarie per capire le basi matematiche su cui si fondano il generatore di Blum Blum Shub e il crittosistema di Blum Goldwasser.

I primi di Blum e gli interi di Blum godono di un'importante proprietà: ogni residuo quadratico a ha una sola radice quadrata y che è essa stessa un residuo quadratico.

Chiamiamo tale y con il termine **radice quadrata principale** e la denotiamo $\sqrt{a}$.

Se $n = pq$ è un intero di Blum indichiamo con Q_n l'insieme dei residui quadratici mod n.

Teorema.

Sia $n = pq$ un intero di Blum e definiamo un'applicazione

$$f : Q_n \to Q_n$$

$$x \mapsto x^2 \pmod n$$

Allora f è una permutazione su Q_n.

Dimostrazione.

Innanzitutto osserviamo che se $p \equiv 3 \pmod 4$ allora:

$$x^2 + y^2 \equiv 0 \pmod p \Rightarrow x^2 \equiv -y^2 \pmod p \Rightarrow x^{p-1} \equiv (-1)^{\frac{p-1}{2}} y^{p-1} \pmod p$$

cosicché $x^{p-1} \equiv -y^{p-1} \pmod p$ e quindi $x \equiv y \equiv 0 \pmod p$.

Supponiamo ora che $f(x) = x^2 \equiv y^2 = f(y) \pmod n$ con $x, y \in Q_n$. Voglio dimostrare che $x \equiv y \pmod n$. Dato che per ipotesi $x, y \in Q_n$, sappiamo che esistono $u, v \in Z_n^*$ tali che $x \equiv u^2 \pmod n$ e $y \equiv v^2 \pmod n$. Quindi $u^4 \equiv v^4 \pmod n$ e quindi

(infatti $x^{p-1} \equiv 1 \pmod p$, $y^{p-1} \equiv 1 \pmod p$ e $1 \equiv -1 \pmod p \Rightarrow 2 \equiv 0 \pmod p$ che è assurda).

$$(u^2 - v^2)(u^2 + v^2) \equiv 0 (\mathrm{mod}\, n) \qquad (*)$$

Dato che $MCD(u,n) = MCD(v,n) = 1$, la nostra osservazione iniziale mostra che sono impossibili tanto $(u^2 + v^2) \equiv 0 (\mathrm{mod}\, p)$ quanto $(u^2 + v^2) \equiv 0 (\mathrm{mod}\, q)$ e da questo deduciamo che $(u^2 + v^2, n) = 1$. La congruenza (*) allora implica che $u^2 \equiv v^2 (\mathrm{mod}\, n)$, da cui segue che $x \equiv y (\mathrm{mod}\, n)$. Questo prova che f è iniettiva e che quindi è una permutazione di Q_n. □

Da questo teorema si giunge alla definizione di radice quadrata principale di a modulo n:

Corollario.

Dato un $a \in Q_n$ esiste un unico $x \in Q_n$ tale che $x^2 \equiv a (\mathrm{mod}\, n)$. Questo unico $x \in Q_n$ è definito **radice quadrata principale** di $a (\mathrm{mod}\, n)$.

Teorema.

Nel precedente Teorema abbiamo provato che $f^{-1} : Q_n \to Q_n$ esiste. Infatti

$$f^{-1}(x) \equiv x^{\frac{(p-1)(q-1)+4}{8}} (\mathrm{mod}\, n).$$

Dimostrazione.

Il nostro scopo è mostrare che $f^{-1}(x)$, come definita sopra, è sicuramente la funzione inversa di f. Innanzitutto osserviamo che dato un qualunque $y \in Q_n$, il risultato precedentemente dimostrato ci dice che esiste un unico $x \in Q_n$ tale che $f(x) \equiv y (\mathrm{mod}\, n)$. Quindi:

$$f^{-1}(y) \equiv f^{-1}(f(x)) \equiv f^{-1}(x^2) \equiv x^{\frac{(p-1)(q-1)+4}{4}} \equiv x \cdot x^{\frac{(p-1)(q-1)}{4}} (\mathrm{mod}\, n) \qquad (*)$$

Dato che $x \in Q_n$, esiste un $u \in Z_n^*$ con $x \equiv u^2 (\mathrm{mod}\, n)$. Pertanto $x^{\frac{(p-1)(q-1)}{4}} \equiv u^{\frac{(p-1)(q-1)}{2}} (\mathrm{mod}\, n)$.

Dato che $\frac{(p-1)(q-1)}{2}$ è multiplo sia di $p-1$ che di $q-1$ deduciamo che $u^{\frac{(p-1)(q-1)}{2}} \equiv 1 (\mathrm{mod}\, p)$ e che $u^{\frac{(p-1)(q-1)}{2}} \equiv 1 (\mathrm{mod}\, q)$. Quindi $u^{\frac{(p-1)(q-1)}{2}} \equiv 1 (\mathrm{mod}\, n)$.

Allora la congruenza $(*)$ implica che $f^{-1}(y) \equiv x(\mathrm{mod}\, n)$ e ciò prova il teorema. □

Notiamo che se $x \in Q_n$, allora $x \equiv u^2(\mathrm{mod}\, n)$ con $u \in Z_n^*$. Pertanto $f^{-1}(x) \equiv x^{\frac{(p-1)(q-1)+4}{8}} \equiv \left(u^{\frac{(p-1)(q-1)+4}{8}} \right)^2 (\mathrm{mod}\, n)$ e quindi $f^{-1}(x) \in Q_n$ cosicché f^{-1} è chiaramente una mappa da Q_n a Q_n.

7.2. Il Generatore di Blum-Blum-Shub

L'algoritmo Blum-Blum-Shub è un PRNG ("Pseudo Random Bit Generator") in uso oggi, notevolmente semplice ed efficiente. Fu proposto nel 1986 da Lenore Blum, Manuel Blum e Michael Shub (anche se era già stata pubblicata una prima bozza nel 1983), che dimostrarono come fosse un CSPRNG sotto l'assunzione dell'intrattabilità del problema dei resti quadratici, vale a dire della difficoltà dell'estrazione delle radici quadrate nell'aritmetica modulare. Il problema dei residui Quadratici è un'ipotesi più debole della difficoltà di fattorizzare, nel senso che nel momento in cui si potrà risolvere il problema della fattorizzazione allora automaticamente si risolverà il problema dei residui quadrati, mentre non è detto che, risolvendo il problema dei residui quadrati, si possa chiarire il problema della fattorizzazione. Se il problema dei resti quadratici e della fattorizzazione degli interi siano equivalenti è una domanda ancora aperta.

Definizione

Si definisce **algoritmo a tempo polinomiale** un algoritmo il cui tempo d'esecuzione nel caso peggiore è $O((\ln n)^k)$, dove n è la dimensione dei dati di ingresso e k una costante. Un algoritmo il cui tempo di lavoro non è polinomiale è chiamato a tempo esponenziale, ed è considerato inefficiente.

Definizione.

Un generatore pseudo - casuale è **crittograficamente sicuro** se dato un qualunque blocco di output $\{b_1, b_2, ..., b_k\}$ dal generatore, la probabilità di

indovinare b_1 da $\{b_2,...,b_k\}$ in tempo polinomiale è all'incirca la stessa che si ha cercando di indovinarlo come un numero casuale. Lo stesso vale se si cerca di indovinare b_k a partire da $\{b_1,b_2,...,b_{k-1}\}$.

7.2.1. Come generare una sequenza di l bit casuali

Sia n un intero di Blum, definito come $n = p \cdot q$ dove $p \equiv q \equiv 3(\text{mod}\,4)$. Si sceglie un seme casuale s_0 da Q_n, l'insieme dei residui quadratici $(\text{mod}\,n)$, e per $i \geq 0$ si pone

$$s_{i+1} \equiv s_i^2 (\text{mod}\, n) \text{ con } 0 \leq i \leq l-1.$$

Quindi si definisce z_i in $\{0,1\}$ come

$$z_i \equiv s_i (\text{mod}\, 2) \text{ con } 1 \leq i \leq l.$$

Pertanto z_i altro non è che il bit che indica la parità di s_i, e che quindi è uguale a 1 se s_i è dispari ed è uguale a 0 se s_i è pari.

La sequenza $\{z_1, z_2, ..., z_l\}$ è la sequenza richiesta.

Il modo più semplice per scegliere s_0 consiste nel prendere un a arbitrario, $1 \leq a < n$, controllare che $MCD(a,n) = 1$ e quindi scegliere $s_0 \equiv a^2 (\text{mod}\, n)$.

7.2.2. Esempio

Sia $n = p \cdot q = 7 \cdot 19 = 133$ e sia $a = 5$. Allora abbiamo $s_0 \equiv 25(\text{mod}133) = 25$. La sequenza continua con $s_1 \equiv 25^2 (\text{mod}133) = 93$, $s_2 \equiv 93^2 (\text{mod}133) = 4$, $s_3 \equiv 4^2 (\text{mod}133) = 16$, $s_4 \equiv 16^2 (\text{mod}133) = 123$ e questo produce l'output 1,0,0,1.

7.2.2.1. Sulla scelta di p e q

La robustezza di BBS dipende, come già detto, dalla difficoltà di fattorizzare grandi numeri interi, per cui p e q devono essere scelti con cura. In particolare:

- p e q dovrebbero essere di dimensioni tali da rendere la fattorizzazione di $n = p \cdot q$ computazionalmente insostenibile. Scegliere n lungo 1024 bit è considerato il minimo per stare tranquilli.

- La differenza tra p e q non dovrebbe essere troppo piccola. Infatti se $p-q$ è un valore piccolo, allora $p \approx q$ e quindi $p \approx \sqrt{n}$, quindi n potrebbe essere fattorizzato efficientemente semplicemente provando come divisori tutti i numeri dispari vicini a $\sqrt{n}$. Se p e q sono scelti a caso la differenza $p-q$ sarà con ogni probabilità appropriatamente grande.
- Alcuni consigliano che p e q siano anche numeri "primi forti" o "strong primes". Un numero primo p si dice strong prime se:
  - p-1 ha un fattore primo "grande", detto r.
  - p+1 ha un fattore primo "grande".
  - r-1 ha un fattore primo "grande".

  In realtà se p e q sono scelti a caso, la probabilità che queste tre condizioni siano soddisfatte è elevata e inoltre è stato mostrato come gli strong primes offrano poca protezione in più rispetto a semplici numeri primi scelti a caso. D'altro canto esistono algoritmi efficienti per la loro generazione (come l'algoritmo di Gordon), per cui la loro utilizzazione non comporta un costo computazionale aggiuntivo insostenibile.

Si noti come anche RSA basi la propria sicurezza sulla difficoltà della fattorizzazione degli interi, per cui queste raccomandazioni sono applicabili anche alla scelta di p e q per la crittazione con RSA.

Osservazione

Notiamo come anche il generatore di Blum-Blum-Shub ricada nello schema ricorsivo generale

$$\begin{cases} x_0 = x_0 \\ x_{n+1} = f(x_n), \ \forall n \geq 0 \end{cases}$$

dove $f: A \to A$, A è un insieme qualunque ed f è una qualunque funzione. Nel caso di Blum si ha $A = Q_n$, $x_0 = s_0$ e $f(x) \equiv x^2 \pmod n$ (vedasi la definizione ricorsiva del precedente paragrafo 7.2.1).

Dato che Q_n è un insieme finito, infatti $|Q_n| = \frac{(p-1)(q-1)}{4}$, anche le sequenze pseudo-casuali di Blum sono definitivamente periodiche. Quanto è lungo il periodo di ogni singolo seme so? (Lo vedremo...) (6)

8. IL CRITTOSISTEMA A CHIAVE PUBBLICA DI BLUM GOLDWASSER

8.1.1. Breve introduzione alla crittografia a chiave pubblica probabilistica

La richiesta minima che si possa fare a un sistema di cifratura è che deve essere difficile, per un avversario, risalire al testo originario partendo dal testo cifrato. Tuttavia, in certe situazioni, è desiderabile imporre delle richieste di sicurezza più stringenti.

Alcuni metodi di cifratura come l'RSA o Rabin sono di natura deterministica nel senso che, sotto una fissata chiave pubblica, il testo di partenza M avrà sempre lo stesso cifrato C. Uno schema di cifratura deterministico avrà sempre almeno uno dei seguenti inconvenienti:

1) Lo schema non è sicuro per ogni messaggio. Per esempio se si usa l'RSA i singoli numeri 0 e 1 saranno sempre cifrati in se stessi e per questo motivo saranno sempre facili da scoprire.
2) E' talvolta possibile ricavare informazioni parziali sul testo di partenza a partire dal testo cifrato. Per esempio nel caso dell'RSA, se $C \equiv M^e \pmod{n}$ è il testo cifrato corrispondente al messaggio originale M, allora $\left(\frac{C}{n}\right) = \left(\frac{M^e}{n}\right) = \left(\frac{M}{n}\right)^e = \left(\frac{M}{n}\right)$ in quanto e è dispari, e quindi un avversario può facilmente ottenere un'informazione su M, vale a dire il simbolo di Jacobi $\left(\frac{M}{n}\right)$.
3) E' facile scoprire quando lo stesso messaggio è stato inviato due volte.

La **cifratura probabilistica** fa uso della casualità per ottenere un livello di sicurezza dimostrabile e molto alto. C'è una nozione di sicurezza molto forte che si cerca sempre di raggiungere: l'essere semanticamente sicuro.

Definizione

Uno schema di cifratura a chiave pubblica è detto **semanticamente sicuro** se nessun avversario può, in tempo polinomiale, distinguere i cifrati di due assegnati

testi in chiaro o, dato un messaggio, distinguere il cifrato da una stringa casuale con una probabilità significativamente maggiore di ½.

Il crittosistema a Chiave Pubblica di Blum-Goldwasser è un crittosistema probabilistico la cui sicurezza dipende dalla difficoltà di fattorizzare. In questo senso e anche per la sua velocità di operazione ricorda l'RSA o il crittosistema di Rabin, che sono entrambi di natura deterministica. Tuttavia differisce dagli altri due sopracitati sistemi considerevolmente in quanto è un crittosistema probabilistico. Può essere dimostrato che è completamente sicuro nel senso che non esiste nessun intermediario che possa calcolare in "polynomial time" anche solo informazioni parziali riguardo al testo originario partendo dalla conoscenza del testo cifrato. Pertanto è un crittosistema semanticamente sicuro, se si suppone che non esista alcun algoritmo "polynomial time" per la fattorizzazione.

8.2. Setup Iniziale

Sia $n = pq$ dove p e q sono numeri primi di dimensioni elevate con $p \equiv q \equiv 3 \pmod 4$.
L'intero n è pubblico mentre la fattorizzazione $n = pq$ è privata.
Il messaggio che si intende cifrare può ovviamente essere espresso in binario:

$$x = (x_1, x_2, ..., x_l).$$

In tal modo $x_i = 0$ o $x_i = 1$ e ci sono esattamente l bit.

8.3. Cifratura

Per cifrare il messaggio x, si sceglie un seme iniziale s_0 da Q_n e si usa il generatore di Blum Blum Shub sopra descritto per ottenere una sequenza di bit che sia pseudo-casuale

$$(z_1, z_2, ...z_l).$$

Ricordiamo che la sequenza $(s_1, s_2, ...s_l)$ è calcolata con elevamenti al quadrato consecutivi $(\mathrm{mod}\, n)$ in modo tale che per ciascun i, con $1 \le i \le l$ abbiamo che $s_i \equiv s_0^{2^i} (\mathrm{mod}\, n)$. Pertanto z_i è l'ultimo bit significativo del corrispondente s_i.

Calcoliamo anche $s_{l+1} \equiv s_0^{2^{l+1}} (\mathrm{mod}\, n)$.

Infine per ogni i, con $1 \le i \le l$ calcoliamo

$$y_i \equiv x_i + z_i (\mathrm{mod}\, 2)$$

Questo è equivalente a dire che

$$y_i = \begin{cases} 1 & \text{se } x_i \text{ e } z_i \text{ hanno parità opposta} \\ 0 & \text{se } x_i \text{ e } z_i \text{ hanno la stessa parità} \end{cases}$$

Il messaggio cifrato è $y = (y_1,....,y_l,s_{l+1})$.

Notiamo che la lunghezza del testo cifrato è superiore di un solo bit alla lunghezza del testo originario.

8.4. Decifratura

Per decifrare il messaggio, la procedura è la seguente:

- Innanzitutto si calcolano $a_1 \equiv \left(\frac{p+1}{4}\right)^{l+1} (\text{mod}\, p-1)$ e $a_2 \equiv \left(\frac{q+1}{4}\right)^{l+1} (\text{mod}\, q-1)$
- Quindi si calcola $b_1 \equiv s_{l+1}^{a_1} (\text{mod}\, p)$ e $b_2 \equiv s_{l+1}^{a_2} (\text{mod}\, q)$.
- Si usa il Teorema Cinese del Resto per trovare $s_0 (\text{mod}\, n)$ dato che s_0 soddisfa sia $s_0 \equiv b_1 (\text{mod}\, p)$ che $s_0 \equiv b_2 (\text{mod}\, q)$ (proveremo poi questa affermazione).
- Si usa il generatore di Blum Blum Shub con seme s_0 per calcolare $z_1, z_2,...z_l$.
- Per ogni i, con $1 \le i \le l$ si calcolano i numeri x_i che sono definiti da $x_i \equiv y_i + z_i (\text{mod}\, 2)$.

La sequenza $(x_1, x_2,..., x_l)$ è il testo originario.

L'idea che sta alla base del procedimento è che nel generatore di Blum Blum Shub, dato un qualunque s_{i+1} e i primi p e q, possiamo sempre calcolare s_i. Quindi partendo da s_{l+1}, che è parte del testo cifrato, possiamo calcolare il seme s_0. In tal modo possiamo determinare la sequenza di bit $z_1, z_2,...z_l$ e di conseguenza procedere come indicato sopra per recuperare $(x_1, x_2,..., x_l)$.

8.5. Correttezza

Dato che s_l è un residuo quadratico $\bmod n$, allora è anche un residuo quadratico $\bmod p$.

Quindi $s_l^{\frac{p-1}{2}} \equiv 1 (\bmod p)$.

Noto che $s_{l+1}^{\frac{p+1}{4}} \equiv \left((s_l)^2\right)^{\frac{p+1}{4}} = (s_l)^{\frac{p+1}{2}} = (s_l)^{\frac{p-1}{2}} \cdot s_l \equiv s_l (\bmod p)$.

Analogamente si mostra che $s_l^{\frac{p+1}{4}} \equiv s_{l-1} (\bmod p)$. Pertanto $s_{l+1}^{\left(\frac{p+1}{4}\right)^2} \equiv s_{l-1} (\bmod p)$ in quanto $s_{l+1}^{\left(\frac{p+1}{4}\right)^2} \equiv s_{l+1}^{\left(\frac{p+1}{4}\right)\left(\frac{p+1}{4}\right)} = \left(s_{l+1}^{\frac{p+1}{4}}\right)^{\frac{p+1}{4}} \equiv s_l^{\frac{p+1}{4}} \equiv s_{l-1} (\bmod p)$. Ripetendo questo ragionamento si prova che $b_1 \equiv s_{l+1}^{a_1} \equiv s_{l+1}^{\left(\frac{p+1}{4}\right)^{l+1}} \equiv s_0 (\bmod p)$.

In modo analogo si dimostra che $b_2 \equiv s_{l+1}^{a_2} \equiv s_{l+1}^{\left(\frac{q+1}{4}\right)^{l+1}} \equiv s_0 (\bmod q)$.

Infine basta risolvere il sistema con il Teorema Cinese del Resto perché $(p,q) = 1$.

Alice trova così lo stesso seme iniziale s_0 che Bob aveva usato nella cifratura, e di conseguenza trova il cifrato.

8.6. Sicurezza

Blum-Glodwasser è semanticamente sicuro, supponendo che la fattorizzazione di numeri interi sia un problema intrattabile; in particolare si considera il caso in cui $n = p \cdot q$ dove p e q sono numeri primi molto grandi. Quest'algoritmo ha diversi vantaggi su altri algoritmi di crittografia probabilistici. Primo, la sua sicurezza è basata solamente sul problema della fattorizzazione, senza bisogno di altre ipotesi. È anche piuttosto efficiente in termini computazionali, perché non sfigura a confronto con altri algoritmi quali RSA.

Poiché per criptare si utilizza un algoritmo probabilistico, uno stesso testo in chiaro può produrre risultati molto diversi ogni volta che viene criptato. Questo porta dei vantaggi significativi, poiché impedisce a un avversario di riconoscere messaggi intercettati confrontandoli con altri intercettati precedentemente.

A seconda della dimensione del testo in chiaro, Blum-Goldwasser può essere più o meno computazionalmente costoso di RSA. Poiché la maggior parte delle implementazioni di

(10)

RSA usa un esponente fisso per la crittazione per ottimizzarne il tempo, RSA generalmente è molto più efficiente di BG tranne che per messaggi molto brevi. Tuttavia, poiché l'esponente usato da RSA per decriptare è distribuito casualmente, l'esponenziazione in modulo può richiedere un numero di moltiplicazioni comparabile a quello richiesto da BG per un messaggio criptato della stessa lunghezza.

BG ha infine il vantaggio di funzionare senza difficoltà per messaggi molto lunghi, nei quali invece RSA richiede più crittazioni separate. In questi casi, BG può essere significativamente più efficiente.

8.7. Esempio

Generazione delle chiavi. Alice sceglie i primi $p = 499$, $q = 547$, ognuno dei quali è congruo a 3 modulo 4, e calcola $n = pq = 272953$. Usando l'algoritmo euclideo, Alice calcola gli interi $a = -57$ e $b = 52$ che soddisfano a $ap + bq = 1$. La chiave pubblica di Alice è pertanto $n = 272953$, mentre la chiave privata è (p, q, a, b).

Cifratura. Bob vuole inviare un messaggio ad Alice. Supponiamo che il messaggio da cifrare, espresso in binario, sia: $M = (1, 0, 0, 1, 1) \equiv (m_1, m_2, m_3, m_4, m_5)$. Per cifrare M, Bob sceglie un seme casuale s_0 da Q_n, l'insieme dei residui quadratici modulo n. Per esempio $s_0 = 159201$ $(\equiv 399^2 (\mathrm{mod}\, n))$. Usando poi il Generatore di Blum Blum Shub, Bob ottiene la sequenza casuale $(s_1, s_2, s_3, s_4, s_5) = (180539, 193932, 245613, 130286, 40632)$.

Bob calcola anche $s_6 = 139680$.

Pertanto $(z_1, z_2, z_3, z_4, z_5) = (1, 0, 1, 0, 0)$.

Ricordando che, per ogni i con $1 \le i \le 5$, $y_i \equiv x_i + z_i \pmod 2$, Bob calcola $y = (0, 0, 1, 1, 1)$, e invia ad Alice il cifrato $C = (0, 0, 1, 1, 1, 139680)$.

Decifratura. Per decifrare C, Alice calcola:

- $a_1 \equiv \left(\frac{p+1}{4}\right)^{l+1} (\mathrm{mod}(p-1)) = 463$
- $a_2 \equiv \left(\frac{q+1}{4}\right)^{l+1} (\mathrm{mod}(q-1)) = 337$
- $b_1 \equiv s_{l+1}^{a_1} (\mathrm{mod}\, p) = 20$

- $b_2 \equiv s_{l+1}^{a_2} (\mathrm{mod}\, q) = 24$
- $s_0 \equiv bqb_1 + apb_2 \equiv 159201 \ (\mathrm{mod}\, n)$.

Infine, Alice usa il generatore di Blum Blum Shub con seme s_0 per calcolare $(z_1, z_2, ...z_5)$, e ottiene il testo originario in quanto sa che per ogni i, con $1 \leq i \leq 5$ i numeri m_i che sono definiti da $m_i \equiv y_i + z_i (\mathrm{mod}\, 2)$.