Alma Mater Studiorum · Università di Bologna

facoltà di scienze matematiche, fisiche e naturali

Corso di Laurea in Matematica

TIMING ATTACK DI PAUL C.KOCHER: attacco al sistema crittografico RSA mediante strumenti di statistica

Tesi di Laurea in Algoritmi della Teoria dei Numeri e Crittografia

Relatore:
Chiar.mo Prof.
DAVIDE ALIFFI

Presentata da:
CRISTINA CALZA

III Sessione
Anno Accademico 2012/2013

A te.

A noi.

Alle gocce di memoria.

A tutto quello che ancora sarà.

Indice

Introduzione

In questa tesi ho voluto descrivere il Timing Attack al sistema crittografico RSA, il suo funzionamento, la teoria su cui si basa, i suoi punti di forza e i punti deboli.
Questo particolare tipo di attacco informatico fu presentato per la prima volta da Paul C. Kocher nel 1996 all'"RSA Data Security and CRYPTO conferences".

Nel suo articolo "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems" l'autore svela una nuova possibile falla nel sistema RSA, che non dipende da debolezze del crittosistema puramente matematiche, ma da un aspetto su cui nessuno prima di allora si era mai soffermato: il tempo di esecuzione delle operazioni crittografiche.

Il concetto è tanto semplice quanto geniale: ogni operazione in un computer ha una certa durata.

Le variazioni dei tempi impiegati per svolgere le operazioni dal computer infatti, necessariamente dipendono dal tipo di algoritmo e quindi dalle chiavi private e dal particolare input che si è fornito.

In questo modo, misurando le variazioni di tempo e usando solamente strumenti statistici, Kocher mostra che è possibile ottenere informazioni sull'implementazione del crittosistema e quindi forzare RSA e altri sistemi di sicurezza, senza neppure andare a toccare l'aspetto matematico dell'algoritmo.

Di centrale importanza per questa teoria diventa quindi la statistica.

Questo perché entrano in gioco molte variabili che possono influire sul tempo di calcolo nella fase di decifrazione:

- La progettazione del sistema crittografico
- Quanto impiega la CPU ad eseguire il processo
- L'algoritmo utilizzato e il tipo di implementazione
- La precisione delle misurazioni
- Ecc.

Per avere più possibilità di successo nell'attaccare il sistema occorre quindi fare prove ripetute utilizzando la stessa chiave e input differenti per effettuare

analisi di correlazione statistica delle informazioni di temporizzazione, fino al punto di recuperare completamente la chiave privata.

Ecco cosa asserisce Kocher: "*Against a vulnerable system, the attack is computationally inexpensive and often requires only known ciphertext*.", cioè, contro sistemi vulnerabili, l'attacco è computazionalmente poco costoso e spesso richiede solo di conoscere testi cifrati e di ottenere i tempi necessari per la loro decifrazione.

Tutto ciò insegna come a volte nuove teorie e nuovi problemi possano scaturire da aspetti e ambiti che neppure si erano presi in considerazione, e cambiare completamente le carte in tavola!

Capitolo 1

Sistema Crittografico RSA e Protocollo di Diffie-Hellmann

Prima di approfondire il Timing Attack è necessario dare alcune nozioni riguardo il sistema crittografico RSA e il protocollo di Diffie-Hellmann.

Questi due capisaldi della crittografia risalgono agli anni'70, più precisamente RSA risale al 1977 e fu un'enorme rivoluzione per la crittografia.

Vediamo in breve la sostanziale differenza rispetto ai sistemi crittografici precedenti: fino a 1976 si usavano sistemi simmetrici a chiave privata, che però doveva essere condivisa in precedenza tra gli utenti (due persone o due computer o due sistemi); quindi, nonostante si fosse arrivati a una crittografia tradizionale molto sofisticata, rimaneva comunque il problema di come scambiarsi la chiave segreta prima della comunicazione attraverso un canale insicuro.

Nel 1976 viene introdotto da Diffie-Hellmann il sistema asimmetrico o a chiave pubblica: la novità è che entrambi gli utenti hanno due chiavi, le quali hanno funzioni differenti:

- CHIAVE PUBBLICA (come un indirizzo) per cifrare i messaggi
- CHIAVE PRIVATA (conosciuta solo dal singolo utente proprietario) per decifrare

Quindi non c'è più il bisogno di avere una chiave segreta condivisa; il nuovo problema che sorge allora è come sincerarsi che la chiave pubblica sia autentica, ma ciò si risolve con la firma digitale.

Nel 1977 infine Rivest-Shamir-Adleman (da cui RSA) implementano l'idea di Diffie-Helmann, che per primi avevano introdotto il concetto di sistema a chiave pubblica, non riuscendo però a trovare un'implementazione.

1.1 Protocollo di Diffie-Helmann

Come già detto questo protocollo serve nella crittografia a chiave pubblica per scambiare la chiave segreta su un canale insicuro.

In realtà la chiave non viene mai passata da un utente all'altro, perché Diffie-Hellmann sfrutta un problema matematico noto come Problema del Logaritmo Discreto, di cui si ha la seguente definizione:

> **Definizione 1.1** (Problema del Logaritmo discreto)**.** Dati un numero primo p, una radice primitiva α $(mod\ p)$ $\in \mathbb{Z}_p^*$, cioè $< \alpha > = \mathbb{Z}_p^*$, un numero $b \in \mathbb{Z}_p^*$, il problema del logaritmo discreto consiste nel trovare un numero intero x , con $0 \leq x \leq p-2$, tale che
>
> $$\alpha^x \equiv_p b$$
>
> e si ha che $\forall b \in \mathbb{Z}_p^*\ \exists!\, x \in \{0,1, \dots, p-2\}$ tale che $\alpha^x \equiv_p b$
> Tale x è detto logaritmo discreto di b rispetto alla base α.

Vediamo il modo ingegnoso con cui viene usato nel protocollo, sfruttando il fatto che il calcolo del logaritmo discreto è un problema difficile:

1.1.1 Scambio della chiave in Diffie-Hellmann

- Si fissa un primo p grande[1] , che può essere trasmesso su un

[1] *sufficientemente grande attualmente consiste in almeno un ordine di* ~300 *cifre decimali*

canale anche non sicuro, e non c'è bisogno che sia segreto, può essere pubblico

- Si fissa una radice primitiva $\alpha \ (mod\ p) \in \mathbb{Z}_p^*$, anch'essa pubblica
- Uno dei due utenti (chiamiamolo **A**) sceglie un x segreto , con $0 \leq x \leq p-2$, che verrà usato come esponente
- L'altro utente (chiamiamolo **B**) sceglie un y segreto, con $0 \leq y \leq p-2$
- **A** calcola $\alpha^x \ (mod\ p)$ e lo invia a **B**
- **B** calcola $\alpha^y \ (mod\ p)$ e lo invia ad **A**

Un avversario che intercetta la conversazione vede solo le potenze, protette dalla difficoltà del logaritmo discreto, che si ritiene abbia una difficoltà subesponenziale.

- **A** calcola $(\alpha^y)^x \ (mod\ p) = k$
- **B** calcola $(\alpha^x)^y \ (mod\ p) = k$

In questo modo entrambi gli utenti hanno ottenuto la chiave segreta k senza in realtà averla dovuta inviare attraverso alcun canale, perché l'hanno generata entrambi, protetti dalla difficoltà del problema matematico del logaritmo discreto.

1.2 Sistema Crittografico RSA

Il sistema RSA è la vera rivoluzione crittografica che stravolge il panorama dei sistemi si sicurezza, introducendo la chiave pubblica.

RSA usa metodi di teoria dei numeri, già noti ad Euclide[2] e a Gauss, e implementa l'idea di sistema a chiave pubblica già introdotta da Diffie-Hellmann, i quali però non l'avevano sviluppata.

[2] *per* l'algoritmo di Euclide si veda l'Appendice A

1.2.1 Generazione delle chiavi in RSA

- Si scelgono p,q primi grandi distinti
- Si calcola $n=pq$, che è il modulo di RSA, dove n è pubblico, ma non la sua fattorizzazione.
  Proprio la difficoltà di fattorizzazione è ciò che sostiene la sicurezza di RSA, anche se, come si mostrerà in seguito, non è del tutto sufficiente.
- si sceglie e , detto esponente di cifratura tale che $MCD(e,\varphi(n)^{[3]}) = 1$
- si determina d , detto esponente di decifrazione tale che $de \equiv_{\varphi(n)} 1$
- per la cifratura:
  - una delle due parti codifica il messaggio m con un elemento di $\mathbb{Z}_n$, ossia $m \in \{0,1,\dots,n-1\}$
  - poi si procura la chiave pubblica del destinatario e calcola $c \equiv_n m^e$
- per la decifrazione: si calcola $m \equiv_n c^d$

Per la condizione su e si usa l'algoritmo di Euclide, per la condizione su d si usa l'algoritmo di Euclide esteso.

Per la cifratura e la decifrazione si utilizzano quindi algoritmi di difficoltà polinomiale; mentre un avversario, che non conosce la chiave privata, deve risolvere un problema (la fattorizzazione) per cui si conoscono solo algoritmi di difficoltà subesponenziali.

Più grandi sono le chiavi maggiore è il gap tra la complessità delle operazioni legittime e quella che deve essere affrontata da un avversario che non conosce d.

[3] *funzione di Eulero* $\varphi(n) = \#\{z \in \mathbb{Z} \mid 1 \le z < n \text{ e } MCD(z,n) = 1\}$ *, cioè* $\varphi(n)$ *è il numero degli interi non negativi <n coprimi con n. In questo caso* $n=pq \Rightarrow \varphi(n) = \varphi(p)\varphi(q) = (p-1)(q-1)$ *perché p,q primi*

Queste ultime considerazioni si riferiscono però, come anche per il protocollo di Diffie-Hellmann, alla sicurezza in termini della complessità del problema matematico, ma come già detto, quello che noi andremo ad analizzare saranno il fattore tempo e certi aspetti dell'implementazione.

1.3 La Firma Digitale

La firma digitale, o firma elettronica, risolve il problema dell'accertamento dell'autenticità della chiave pubblica nei sistemi asimmetrici; è utile per inviare documenti validi legalmente e deve avere le stesse caratteristiche di una firma cartacea; consiste in un insieme di dati in forma elettronica – risultato di una procedura detta *validazione* – allegati al documento, che devono dipendere dal documento stesso, e fornisce un metodo di identificazione informatica.

1.3.1 Funzionalità della Firma Digitale

La firma digitale ha tre importanti funzioni:

1) INTEGRITÀ: Permette di verificare l'integrità di un documento, cioè che non sia stato alterato sia intenzionalmente sia casualmente

2) AUTENTICITÀ: Permette di autenticare il mittente

3) NON RIPUDIABILITÀ: Questa è una caratteristica esclusiva della firma digitale; vincola il firmatario alla conoscenza del documento e all'autenticazione di esso

1.3.2 Generazione della Firma Digitale

- La firma digitale in un sistema a chiave pubblica può essere generata solo dall'utente detto firmatario o titolare.
- Il firmatario, attraverso la propria chiave privata, può sia decifrare i messaggi che gli vengono inviati, sia firmare messaggi da lui generati.
- Chi riceve un messaggio firmato può verificare l'autenticità della firma usando solamente la chiave pubblica.

Il Timing Attack ad RSA (e ad altri sistemi crittografici a chiave pubblica, che utilizzano standard per la firma digitale come DSA – Digital Signature Algorithm, o DSS – Digital Signature Standard) va a colpire in particolare questa fase della generazione della firma, misurando il tempo di risposta necessario per la firma di un documento.

Per questo motivo, in seguito, quando si parlerà di tempo di esecuzione o di risposta della macchina ci si riferirà al tempo impiegato per apporre la firma a un documento.

Capitolo 2

Timing Attack

Come si è visto nel capitolo precedente l'operazione su cui si basa l'algoritmo di cifratura e decifrazione di RSA è l'esponenziazione rapida o modulare, noto anche come "*Left-to-Right binary method for exponentiation"* , che resiste bene agli attacchi "matematici" ma non altrettanto bene al Timing Attack

Vediamo più in particolare come lavora questa forma di *"Side Channel Analysis"*, ossia un attacco che sfrutta le misurazioni dei tempi, i costi di energia, ecc, , per ottenere informazioni sulla chiave segreta del sistema crittografico.

L'idea di Kocher – che egli espone nel suo articolo presentato nel 1996 – è questa: chi porta l'attacco deve utilizzare $Y_1, \dots, Y_k$ testi cifrati, e deve indurre la macchina a decifrarli; equivalentemente, gli Y_i possono essere anche messaggi qualsiasi che l'attaccante deve far firmare alla macchina.

Misurando il tempo di esecuzione di questi processi, e conoscendo bene l'implementazione usata, si può fare una stima del tempo impiegato dalla macchina per calcolare i vari passi, senza neppure avere accesso alla macchina stessa.

2.1 Algoritmo di Esponenziazione Modulare

Vediamo come funziona l'algoritmo di esponenziazione modulare:

INPUT:

- Y (messaggio = intero da esponenziare)
- n (il modulo RSA)
- $d = (b_0 b_1 \dots b_{w-1})_2 = \sum_{i=0}^{w-1} b_i \, 2^i$, (la chiave privata, b_i cifre binarie)

PROCESSO:

(1) k=0 (variabile contatore, con *k=0,…,w-1* che percorre i bit di d)

(2) $S_0 = 1$ (variabile che darà il risultato finale)

(3) Se $b_k = 1 \Rightarrow r_k \equiv_n S_k Y$, altrimenti se $b_k = 0 \Rightarrow r_k \equiv_n S_k$

(4) Si pone $S_{k+1} \equiv_n r_k^2$

(5) Finché $k < w - 1$ si incrementa k e si ritorna al punto (3)

(6) Quando $k = w - 1 \Rightarrow STOP$ (i bit sono stati percorsi tutti)

OUTPUT: $S = r_{w-1} \equiv_n Y^d$ (S è la firma del messaggio)

```
def mod_exp(Y, d, n):
    Let S0 = 1
    For k = 0 upto w-1:
     If  (bit k of d)is 1 then
        Let rk = (Sk.Y) mod n .
     Else
        Let rk = Sk .
     Let  Sk+1 = rk^2 mod n .
    EndFor .
    Return (rw-1) .
```

Fig.1 – *Binary method for exponentiation*

***Esempio 2.1*:** Vogliamo trovare il valore di Y^{34}

Eseguendo i passi indicati nell'algoritmo sopra si ha:
d = 34 , che espresso in cifre binarie ha il seguente valore: d = 1 0 0 0 1 0

Svolgiamo il ciclo per $k \in \{0, \dots, 5\}$ e si ha la seguente tabella:

K	b_k	S_k	r_k
0	1	1	$r_0=S_0Y=Y$
1	0	Y^2	$r_1=S_1=Y^2$
2	0	Y^4	$r_2=S_2=Y^4$
3	0	Y^8	$r_3=S_3=Y^8$
4	1	Y^{16}	$r=S_4Y=Y^{17}$
5	0	Y^{17}	$r_5=S_5Y=Y^{34}$

Come ci si aspettava da quanto visto nell'algoritmo, r_5 è proprio il valore cercato.

2.1.1 Stima Temporale

In crittografia, o comunque in ambito informatico, il tempo di esecuzione si misura in operazioni sui bit, cioè contando quante operazioni elementari[1] (binarie) necessitano per un dato procedimento.

L'algoritmo di esponenziazione modulare richiede:

- Quadrature: $(k-1)$
- Moltiplicazioni: da 0 a $(k-1)$

Abbiamo due possibili valori per il tempo di esecuzione :

Massimo: $(k-1)+(k-1)=2(k-1)$

Minimo: $(k-1)+0=k-1$

La media è: $\frac{(k-1)+2(k-1)}{2}=1{,}5(k-1)$

Il tempo necessario per il calcolo delle moltiplicazioni può essere stimato conoscendo le caratteristiche (hardware e software) della macchina utilizzata.
Inoltre, cosa molto importante, abbiamo individuato un tempo massimo e un tempo minimo.

Vediamo quale importanza ha ciò per il Timing Attack: vogliamo capire questa differenza da cosa dipende e che informazioni fornisce a chi muove l'attacco.

[1] *con operazione elementare si intende l'operazione di sommare o moltiplicare due cifre binarie, più un eventuale riporto*

2.2 L'Attacco

L'idea di Kocher di misurare i tempi di esecuzione gli fece osservare che, in base al valore dell'esponente *d*, l'algoritmo di esponenziazione modulare eseguiva o meno, ad ogni passo, un'operazione in più.

Infatti, in ogni iterazione:

- se il bit dell'esponente sul quale viene compiuto il ciclo è 1, vengono eseguiti sia l'elevazione al quadrato, sia la moltiplicazione (righe 3 e 4 dell'algoritmo)
- se il bit dell'esponente invece è 0 viene eseguita solo l'elevazione al quadrato (riga 4 dell'algoritmo)

Da qui segue che è possibile dedurre il valore dei corrispondenti bit di *d* solo osservando e confrontando il tempo di esecuzione dei vari cicli iterativi dell'algoritmo di esponenziazione modulare.

Abbiamo detto che chi muove l'attacco deve indurre la macchina a firmare $Y_1, \dots, Y_k$ messaggi, e la macchina li firmerà con RSA attraverso l'algoritmo di esponenziazione modulare. L'attaccante misura e memorizza i tempi $T_1, \dots, T_k$ necessari per ottenere $Y_i^d \pmod n$, $i = 1, \dots, k$.
L'esponente d ha $b_0 = 1$ necessariamente, perché la prima cifra deve essere significativa; quindi le cifre che prendiamo in considerazione saranno da quella con indice 1 a quella con indice w-1, con i relativi cicli, tralasciando il ciclo 0, il cui tempo è noto.
Supponiamo inoltre che l'avversario conosca anche le cifre $b_1, \dots, b_{j-1}$ di d; allora può determinare bj (e tutte le successive cifre, quindi tutto d stesso).
Questo perché si ipotizza che l'attaccante sia in grado di simulare i tempi per le firme su un simulatore identico alla macchina in questione, e poiché conosce $b_0, \dots, bj_{-1}$, allora può stimare il tempo necessario per arrivare a r_{j-1}, e di conseguenza riesce a trovare bj.

2.2.1 Il Timing Attack nel dettaglio

Vediamo ora il Timing Attack nel dettaglio.

$\forall i \in \{1, \dots, k\}$, quindi $\forall Y_i$ messaggio:

2.2 L'Attacco

- Si misura il tempo totale T_i impiegato dalla macchina per firmare Y_i
- T_i dipende da:
  - il tempo $q_{i,j}$ per eseguire il modulo del quadrato[2] ad ogni ciclo iterato j, $j \in \{1, \dots, w-1\}$, w numero di cifre di d
  - il tempo $p_{i,j}$ per eseguire il modulo della moltiplicazione[3] ad ogni ciclo iterato j, che può essere anche nullo (se $b_j = 0$)
  - il tempo e_i dovuto ad eventuali errori, ad esempio errori di misurazione, rallentamenti della rete, ecc.
- $T_i = e_i + \sum_{j=1}^{w-1} (q_{i,j} + p_{i,j})$ dove $p_{i,j}$ dipende dal valore del bit b_j dell'esponente d

Come già detto, supponiamo che l'attaccante conosca le prime *j-1* cifre di d, oltre a b_0; quindi si può procedere facendo delle ipotesi a partire dal ciclo iterativo j:

- Abbiamo supposto che l'attaccante sia in grado di simulare i tempi necessari per le firme. Quindi per ogni messaggio Y_i si simula il procedimento due volte, cambiando l'ipotesi su b_j (ad esempio, prima con 0 e poi 1)
- In questo modo si avranno ovviamente due tempi diversi, dovuti alla diversa durata del ciclo j-esimo; si calcolano due candidati: $\tilde{T}_{i,j,0}$ e $\tilde{T}_{i,j,1}$[4]

(questi tempi misurano la durata dei primi j cicli; questo significa che si sta

[2] *riga (4) dell'algoritmo di esponenziazione modulare*

[3] *riga (3) dell'algoritmo di esponenziazione modulare*

[4] *i indica il messaggio, j il ciclo, 0 e 1 le ipotesi fatte sul bit b_j*

lavorando sui primi *j* bit, con le prime *j-1* cifre di *d* significative, mentre le ultime (*w-1)-j* cifre si suppongono non significative, e si fa un'ipotesi su b_j)

- Si costruisce la seguente tabella delle differenze tra i tempi T_i di risposta della macchina e i tempi $\tilde{T}_{i,j,0}$ e $\tilde{T}_{i,j,1}$ simulati , $\forall i \in \{1 , \dots , k\}$:

0	1
$T_1 - \tilde{T}_{1,j,0}$	$T_1 - \tilde{T}_{1,j,1}$
$\vdots$	$\vdots$
$T_k - \tilde{T}_{k,j,0}$	$T_k - \tilde{T}_{k,j,1}$

Tabella.1 – *Differenze tra tempi macchina e tempi simulati*

- Si pone $T_{i,0} = T_i - \tilde{T}_{i,j,0}$ e $T_{i,1} = T_i - \tilde{T}_{i,j,1}$
- Si calcolano le varianze (campionarie corrette)[5]:

$$\sigma_0^2 = \frac{1}{k-1}\sum_{i=1}^{k}(T_{i,0} - \bar{T}_0)^2$$

e

$$\sigma_1^2 = \frac{1}{k-1}\sum_{i=1}^{k}(T_{i,1} - \bar{T}_1)^2$$

dove $\bar{T}_0$ e $\bar{T}_1$ sono la media della colonna 0 e della colonna 1

- La varianza che risulterà minore indicherà la corrispondente colonna con l'ipotesi corretta sul bit b_j

Per il bit successivo si procede allo stesso modo, e così via fino all'ultima cifra dell'esponente *d* .

[5] *per la varianza campionaria corretta e altre nozioni di statistica si veda l'Appendice B*

2.2 L'Attacco

Ovviamente la differenza fra le due varianze deve essere significativa, altrimenti non si ottiene nessuna informazione e questo attacco non porta l'avversario ad alcuna conclusione su *d* e le cifre mancanti.

Analizziamo la correttezza di questo procedimento:

- Sia m un determinato valore di j, cioè stiamo considerando l'm-esimo ciclo nell'algoritmo di esponenziazione modulare e l' m-esima cifra di *d.*
- L'ipotesi è sempre quella che b_1, ... , b_{m-1} siano conosciuti e corretti; l'attaccante $\forall i \in \{1, \dots, k\}$ calcola: $T_i - \tilde{T}_{i,m,h}$

  dove: - h è l'ipotesi fatta su b_m
  - $\tilde{T}_{i,m,h} = (q_{i,m} + \tilde{p}_{i,m,h}) + \sum_{j=1}^{m-1}(q_{i,j} + p_{i,j})$, con $\tilde{p}_{i,m,h}$ tempo candidato per $p_{i,m}$

  notiamo che se h=0 risulta $\tilde{p}_{i,m,h}$=0, mentre se h=1 risulta $\tilde{p}_{i,m,h}$>0
- Quindi si ha:

$$T_i - \tilde{T}_{i,m,h} = \left[e_i + \sum_{j=1}^{w-1}(q_{i,j} + p_{i,j})\right] - \left[(q_{i,m} + \tilde{p}_{i,m,h}) + \sum_{j=1}^{m-1}(q_{i,j} + p_{i,j})\right]$$

$$= e_i + \sum_{j=m+1}^{w-1}(q_{i,j} + p_{i,j}) + (q_{i,m} + p_{i,m}) - (q_{i,m} + \tilde{p}_{i,m,h})$$

$$= e_i + \sum_{j=m+1}^{w-1}(q_{i,j} + p_{i,j}) + (p_{i,m} - \tilde{p}_{i,m,h})$$

- Se l'ipotesi sul bit b_m è corretta, allora: $p_{i,m} = \tilde{p}_{i,m,h}$, e quindi:

$$T_i - \tilde{T}_{i,m,h} = e_i + \sum_{j=m+1}^{w-1}(q_{i,j} + p_{i,j})$$

- Se invece $p_{i,m} \neq \tilde{p}_{i,m,h}$, non si hanno cancellazioni.

Consideriamo ora le stesse variabili come variabili aleatorie.

Siano:

- T variabile aleatoria relativa al tempo necessario per firmare il messaggio
- q variabile aleatoria relativa al tempo per eseguire l'elevazione al quadrato modulare

- p variabile aleatoria relativa al tempo per eseguire la moltiplicazione modulare
- $\tilde{T}_{m,h}$ variabile aleatoria che indica il tempo necessario per firmare il messaggio Y considerando significative solo le prime *m-1* cifre di *d*, e con l'ipotesi $b_m = h$
- e variabile aleatoria che indica l'eventuale errore

Supponiamo inoltre che i tempi di esecuzione della moltiplicazione modulare siano indipendenti tra i vari cicli .

- Nel caso in cui $b_m = h$ sia corretto, si ottiene:

$$\sigma^2\left(T - \tilde{T}_{m,h}\right) = \sigma^2\left(e + \sum_{j=m+1}^{w-1} q_j + \sum_{\substack{j=m+1 \\ b_j \neq 0}}^{w-1} p_j\right) \quad [5]$$

$$= \sigma^2(e) + (w - m - 1)\,\sigma^2(q) + l\,\sigma^2(p)$$

dove *l* è il numero di bit non nulli di *d* tra quelli maggiori di $b_m = h$

- Nel caso in cui $b_m = h$ non sia corretto, come già detto, si ha:

$$T_i - \tilde{T}_{i,m,h} = e_i + \sum_{j=m+1}^{w-1} \left(q_{i,j} + p_{i,j}\right) + \left(p_{i,m} - \tilde{p}_{i,m,h}\right)$$

Questo perché, se l'ipotesi è sbagliata, la differenza $\left(p_{i,m} - \tilde{p}_{i,m,h}\right)$ risulta non nulla.

Quindi si ottiene:

$$\sigma^2\left(T - \tilde{T}_{m,h}\right) = \sigma^2(e) + (w - m - 1)\,\sigma^2(q) + (l + 1)\,\sigma^2(p)$$

Abbiamo così dimostrato che la colonna che ha la varianza minore (in modo significativo) è quella che contiene l'ipotesi corretta sul bit.

Si può così concludere che se i cicli sono simulati con l'ipotesi corretta si ottiene ad ogni ciclo una diminuzione della varianza attesa, mentre se l'ipotesi sul bit è errata ad ogni ciclo si avrà al contrario un aumento della varianza attesa;

quindi da un'operazione semplice, come lo è il calcolo delle varianze, otteniamo un buon metodo per individuare le cifre dell'esponente d .

Un altro aspetto di questo tipo di attacco che possiamo analizzare , che è molto rilevante per poter stabilire quanto il Timing Attack sia davvero efficace, è la probabilità di successo, anch'essa calcolabile utilizzando noti strumenti statistici; analizzeremo questo aspetto nel capitolo successivo.

Capitolo 3

Probabilità di Successo

Nel suo articolo sul Timing Attack, Kocher non si limita a descrivere il funzionamento dell'attacco, ma, utilizzando risultati base di probabilità e statistica, mostra che è possibile stabilire la probabilità di successo dell'attacco.

Riprendiamo le ipotesi della dimostrazione precedente, cioè consideriamo le variabili come variabili aleatorie indipendenti; supponiamo che:

- l'errore e sia trascurabile, cioè $\sigma^2(e) = 0$
- le variabili aleatorie q,p abbiano distribuzione normale $\mathcal{N}(\mu_q, \sigma_q^2)$, $\mathcal{N}(\mu_p, \sigma_p^2)$, cioè q ha media μ_q e varianza σ_q^2, lo stesso per p.
- Essendo che la combinazione lineare di variabili aleatorie normalmente distribuite è una variabile aleatoria con distribuzione normale, allora si ha che le variabili relative alla colonna corretta e a quella sbagliata,

$$\sum_{j>m} q_j + \sum_{\substack{j>m \\ b_j \neq 0}} p_j \qquad e \qquad \sum_{j>m} q_j + \sum_{\substack{j>m \\ b_j \neq 0}} p_j + p \ ,$$

  hanno distribuzione normale
- Sia $\mathcal{N}(\mu_0, \sigma_0^2)$ la distribuzione della prima variabile, quella dell'ipotesi corretta, con: $\mu_0 = (w - m - 1)\,\mu_q + l\,\mu_p$
  e $\sigma_0^2 = (w - m - 1)\,\sigma_q^2 + l\,\sigma_p^2$
- Sia $\mathcal{N}(\mu_0 + \mu_p, \sigma_0^2 + \sigma_p^2)$ la distribuzione della seconda variabile, quella relativa alla colonna sbagliata
- Consideriamo le k misurazioni del tempo impiegato per la restituzione del messaggio firmato, e indichiamo con X e J le variabili normali standardizzate[1] (X riferita all'ipotesi corretta, J viceversa a quella errata); possiamo compilare la seguente tabella:

[1] *una variabile aleatoria A si dice standardizzata quando la si riconduce a una variabile aleatoria* $A' = \frac{A-\mu}{\sigma}$ *con distribuzione standard, ossia che presenta media 0 e varianza 1.*

$\sigma_0 X_1 + \mu_0$ $\vdots$ $\sigma_0 X_k + \mu_0$	$(\sigma_0 X_1 + \mu_0) + (\sigma_p J_1 + \mu_p)$ $\vdots$ $(\sigma_0 X_k + \mu_0) + (\sigma_p J_k + \mu_p)$

Tabella.2 – Riscrittura della tabella 1 attraverso medie, varianze e le ipotesi X,J

- Poniamo: $V_i = \sigma_0 X_i + \mu_0$ e $W_i = (\sigma_0 X_i + \mu_0) + (\sigma_p J_i + \mu_p)$
  Dove V e W risulteranno anch'esse variabili con distribuzione normale, rispettivamente $\mathcal{N}(\mu_0, \sigma_0^2)$ e $\mathcal{N}(\mu_0 + \mu_p, \sigma_0^2 + \sigma_p^2)$, perché V e W sono combinazioni lineari di variabili con distribuzione normale.

Allora per la probabilità di successo vediamo che risulta:

$$P(\sigma_W^2 > \sigma_V^2) = P\left(\frac{1}{k-1}\sum_{i=1}^{k}(W_i - \bar{W})^2 > \frac{1}{k-1}\sum_{i=1}^{k}(V_i - \bar{V})^2\right)$$

$$= P\left(\sum_{i=1}^{k}(W_i - \bar{W})^2 > \sum_{i=1}^{k}(V_i - \bar{V})^2\right)$$

Essendo V e W normalmente distribuite, per valori di k sufficientemente grandi si ha che: $\bar{V} \approx \mu_0$ e $\bar{W} \approx \mu_0 + \mu_p$

Allora:

$$P(\sigma_W^2 > \sigma_V^2) \approx P\left(\sum_{i=1}^{k}\left(\sigma_0 X_i + \sigma_p J_i\right)^2 > \sum_{i=1}^{k}(\sigma_0 X_i)^2\right)$$

$$= P\left(\sum_{i=1}^{k}\left(\sigma_0^2 {X_i}^2 + \sigma_p^2 {J_i}^2 + 2\sigma_0\sigma_p X_i J_i\right) > \sum_{i=1}^{k}(\sigma_0^2 X_i^2)\right)$$

$$= P\left(\sum_{i=1}^{k}\left(2\sigma_0\sigma_p X_i J_i\right) + \sum_{i=1}^{k}\left(\sigma_p^2 J_i^2\right) > 0\right)$$

$$= P\left(2\sigma_0\sigma_p \sum_{i=1}^{k}(X_i J_i) + \sigma_p^2 \sum_{i=1}^{k}(J_i^2) > 0\right)$$

$$= P\left(2\sigma_0 \sum_{i=1}^{k}(X_i J_i) + \sigma_p \sum_{i=1}^{k}(J_i^2) > 0\right)$$

Ricordiamo che X e J sono standardizzate, e sfruttando la relazione $\sigma^2(X) = E(X^2) - E(X)^2$, si ottiene $E(X^2) = E(J^2) = 1$, quindi:

$$\sum_{i=1}^{k}(J_i^2) \approx \sum_{i=1}^{k} E(J_i^2) = k$$

Inoltre X e J sono indipendenti, perciò valgono le relazioni:

$$E(XJ) = E(X)E(J) = 0$$

e

$$\sigma^2(XJ) = E(X^2J^2) - E(XJ)^2 = E(X^2J^2) = E(X^2)E(J^2) = 1$$

Per quello che abbiamo appena visto si ha che la variabile $\sum_{i=1}^{k}(X_i J_i)$ ha una distribuzione del tipo $\mathcal{N}(0, k)$.

In conclusione, se si approssima la variabile $\sum_{i=1}^{k}(J_i{}^2)$ con k, e se si considera la variabile aleatoria $Z = \frac{XJ-0}{\sqrt{k}}$ con distribuzione normale standard, si ha:

$$P(\sigma_W^2 > \sigma_V^2) \approx P\left(2\sigma_0\left(\sqrt{k}Z\right) + \sigma_p k > 0\right) = P\left(Z > -\frac{\sigma_p}{\sigma_0}\frac{\sqrt{k}}{2}\right) = \Phi\left(\frac{\sigma_p}{\sigma_0}\frac{\sqrt{k}}{2}\right),$$

dove $\Phi(z)$ è l'area sottesa dalla curva normale standard.

Dal risultato appena ottenuto si può facilmente notare che la probabilità di successo dipende dal numero *k*, il numero di testi che l'avversario invia per indurre la macchina a firmarli. Più grande è questo numero, maggiore è la possibilità di successo; ciò è in accordo con la teoria statistica che supporta il Timing Attack. Infatti in statistica, e in particolare nella teoria degli errori, più ampio è il collettivo o maggiore è il numero di misurazioni su cui si attua l'analisi, con più precisione si individuerà il valore "vero".

Inoltre, da questo risultato si può osservare che la probabilità di successo cresce come $\sqrt{k}$, e ciò ci permette di affermare che l'attacco è efficace e attuabile, perché il numero di testi campione che l'avversario deve inviare non esorbitante, e quindi il tempo necessario per l'attacco è relativamente breve.

Capitolo 4

Difese Attuabili

Per prevenire il Timing Attack si possono prendere varie misure; gli algoritmi possono essere implementati (o mascherati da un proxy – programma che si interpone tra le due parti facendo da tramite, ad esempio ricevendo e inoltrando le richieste) in modo da ridurre o eliminare la dipendenza dal tempo delle informazioni sulle chiavi.

4.1 Algoritmo con Tempi di Calcolo Costanti

Si consideri ad esempio un algoritmo che genera la firma con tempi per ogni fase di calcolo costanti, qualsiasi messaggio cifrato venga inviato. Al suo interno ogni ciclo ha una determinata durata, precisamente la massima durata possibile, perché si deve considerare l'ipotesi peggiore in termini di tempo di calcolo.

In un'implementazione di questo tipo, la misurazione dei cicli non fa trapelare alcuna informazione riguardante la chiave segreta, perché si ha un appiattimento dei dati e l'analisi statistica risulta inefficace.

Questa contromisura però non è sempre pratica, perché innanzitutto il tempo uguale per tutti i cicli deve appunto essere quello dell'ipotesi con la durata di calcolo maggiore, e quindi non sono possibili delle ottimizzazioni; inoltre, anche se i tempi sono i medesimi, controllando il lavoro della CPU durante lo svolgimento dell'algoritmo, si riescono ad ottenere delle informazioni sul tipo di operazione svolta.

4.2 Algoritmo con Rumore Casuale

Un'altra possibilità è quella di introdurre un rumore casuale (dei ritardi e/o errori) per rendere inefficace l'analisi statistica, così che, per attuare l'attacco ed ottenere dei risultati significativi, il numero di messaggi cifrati da far firmare alla

macchina per ottenere l'esponente d cresce esponenzialmente e ciò rende vano il Timing Attack.

Ma anche in questo caso questa difesa non è del tutto efficiente, perché l'avversario può contrattaccare effettuando più misurazioni inviando lo stesso testo cifrato, eliminando quindi l'errore casuale sempre grazie all'analisi statistica. Inoltre, anche per questo algoritmo non sono possibili delle ottimizzazioni.

4.3 RSA Blinding

Il metodo per ora più valido ed usato è l'RSA Blinding.

Vediamone velocemente il funzionamento:

- prima di apporre la firma $S \equiv_n Y^d$, si sceglie un numero random r, e si calcola $S' \equiv_n (r^e Y)^d$, dove e è la chiave pubblica
- si ricava poi r^{-1}, l'inverso di r, e infine si calcola: $r^{-1}S' \equiv_n Y^d$

  Quest'ultima è la firma del messaggio.

Così facendo il tempo impiegato per firmare il messaggio non dipende più dal messaggio in input, di conseguenza il Timing Attack non può ottenere informazioni riguardo l'esponente d.

Quest'ultimo è ancora uno dei metodi utilizzati per difendere RSA da attacchi che sfruttano la misurazione dei tempi di esecuzione, come il Timing Attack.

Conclusioni

In conclusione, algoritmi che si basano sull'algoritmo di esponenziazione rapida, come RSA, sono vulnerabili al Timing Attack; se si misura accuratamente il tempo di risposta si può recuperare la chiave *d*, inviando messaggi selezionati da firmare, il cui numero risulta proporzionale alla lunghezza della chiave.

Kocher sottolinea il fatto che gli attacchi del tipo Side Channel Analysis ci dimostrano che, anche se un sistema è fondato su problemi matematici forti, la loro sicurezza dipende anche dall'implementazione del sistema.

Quindi, in definitiva, il Timing Attack dimostra come, per la progettazione di un sistema crittografico sicuro, sia sempre necessario includere e analizzare più aspetti, crittografici e non.

Appendice A: Algoritmo di Euclide

L'algoritmo di Euclide è uno dei più antichi algoritmi conosciuti e prende il nome da colui che per la prima volta lo teorizzo nella sua opera gli *Elementi*, che risale al 300 a.C. , anche se era sicuramente già noto in tempi più antichi.

Questo algoritmo è un metodo diretto per il calcolo del massimo comun divisore MCD di due numeri interi a,b – lo indicheremo con due parentesi tonde: (a,b) – senza richiedere la fattorizzazione in numeri primi; per quanto riguarda la crittografia, è l'algoritmo su cui si basa RSA.

Vediamo come funziona: siano $a, b \in \mathbb{Z},\ a > b$ e $a, b \geq 2$ per semplicità. Sappiamo che $(a, b) = (b, r_1)$, con r_1 resto della divisione di a per b.
Applichiamo la divisione con resto più volte, finché il resto ottenuto non sarà nullo:

(1) $a = bq_1 + r_1\ ,\ 0 \leq r_1 \leq b$ e se $r_1 = 0 \Rightarrow (a, b) = b$

(2) se $r_1 > 0 : b = r_1 q_2 + r_2\ ,\ 0 \leq r_2 \leq r_1$ e se $r_2 = 0 \Rightarrow (a, b) = (b, r_1) = r_1$

(3) se $r_2 > 0 : r_1 = r_2 q_3 + r_3,\ 0 \leq r_3 \leq r_2$ e se $r_3 = 0 \Rightarrow (a, b) = \cdots = (r_1, r_2) = r_2$

⋮

(i+1) se $r_i > 0 : r_{i-1} = r_i q_{i+1} + r_{i+1},\ 0 \leq r_{i+1} \leq r_1$ e se $r_{i+1} = 0 \Rightarrow (a, b) = \cdots = (r_1, r_2) = r_2$

⋮

Così facendo si costruisce una successione di resti con $r_i < r_{i-1}$ convergente a zero, cioè al passo n-esimo si trova: $r_{n-2} = r_{n-1} q_n + r_n\ ,$ con $r_n = 0$
Si conclude quindi che: $(a, b) = r_{n-1}$

Appendice A

Appendice B: Nozioni di Statistica

B.1 Varianza

La varianza è il più noto indice di variabilità basato sullo scostamento dalla media aritmetica.

Definizione B.1 (Varianza).
La varianza di un insieme di valori $x_1, \dots, x_k$ di una variabile X, con media aritmetica $\bar{M}$ è data da:

$$\sigma^2(X) = \frac{1}{k}\sum_{i=1}^{k}(x_i - \bar{M})^2$$

Quindi σ^2 è definita come la media dei quadrati degli scarti dalla media aritmetica.

Nella varianza lo scostamento tra la modalità x_i e la media $\bar{M}$ viene perciò misurata elevando al quadrato la differenza tra i due valori. L'elevamento al quadrato trasforma tutte le differenze in quantità positive e inoltre le differenze più grandi sono messe in maggior risalto, perché aumentano maggiormente rispetto a quelle piccole.

B.1.1 Varianza Campionaria Corretta

Nel caso in cui si studi un campione di cardinalità k, si utilizzano solitamente due stimatori al posto della varianza, uno dei quali è la varianza campionaria corretta.

Nei capitoli precedenti, supponendo che l'avversario si serva di un numero considerevole di testi campione Y_i , abbiamo utilizzato la varianza campionaria corretta:

$$\sigma_{k-1}^2 = \frac{1}{k-1}\sum_{i=1}^{k}(x_i - \bar{M})^2$$

La varianza campionaria corretta è detta così a causa della sua proprietà di correttezza, cioè il suo valore atteso è proprio la varianza: $\mathbb{E}[\sigma_{k-1}^2] = \sigma^2(X)$

Dimostrazione.

$$\begin{aligned}
\mathbb{E}[\sigma_{k-1}^2] &= \mathbb{E}\left[\frac{1}{k-1}\sum_{i=1}^{k} x_i^2 - \frac{k}{k-1}\bar{M}^2\right] \\
&= \frac{1}{k-1}\sum_{i=1}^{k}\mathbb{E}[x_i^2] - k\,\mathbb{E}[\bar{M}^2] \\
&= \frac{1}{k-1}\left(k\,\mathbb{E}[X^2] - k\,\mathbb{E}[\bar{M}^2]\right) \\
&= \frac{k}{k-1}\left(\sigma^2(X) + \mathbb{E}[X^2] - \sigma^2(\bar{M}) - \mathbb{E}[\bar{M}^2]\right) \\
&= \frac{k}{k-1}\left(\sigma^2(X) + \bar{M} - \frac{1}{k}\sigma^2(X) - \bar{M}\right) \\
&= \frac{k}{k-1}\left(\frac{k-1}{k}\sigma^2(X)\right) \\
&= \sigma^2(X)
\end{aligned}$$

□

La spiegazione del termine $k-1$ è data dalla necessità di stimare anche la media.

B.1.2 Varianza di una Somma di Variabili Indipendenti

La varianza della somma di due variabili indipendenti è pari alla somma delle loro varianze:

$$\sigma^2(X+Y) = \sigma^2(X) + \sigma^2(Y)$$

Dimostrazione. Supponiamo per semplicità il caso:
$\mathbb{E}[X] = \mathbb{E}[Y] = 0 \quad \Rightarrow \quad \mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y] = 0$

Si ha:

$\sigma^2(X+Y) = \mathbb{E}[(X+Y)^2] = \mathbb{E}[X^2] + 2\mathbb{E}[XY] + \mathbb{E}[Y^2] = \sigma^2(X) + \sigma^2(Y)$

- Il caso generale si dimostra allo stesso modo, traslando le variabili (*cio*è: $X' = X - \mathbb{E}[X]$) in modo che abbiano valore atteso nullo.

□

Bibliografia

[1] Paul C. Kocher: "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems". Advances in Cryptology - CRYPTO 1996, *16th Annual International Cryptology Conference,* Santa Barbara, California*, USA*. (Pdf file)

[2] Baldoni, Ciliberto, Piacentini Cattaneo, "Aritmetica, Crittografia e Codici" (2006). Springer-Verlag, Milano.

[3] W. Trappe, L.C. Washington, "Crittografia con elementi di teoria dei codici" (2009). Pearson-Pentice Hall.

[4] Kobliz, "A Course in Number Theory and Cryptography" (1987). Springer, New York.

[5] F.Biagini, M.Campanino, "Elementi di Probabilità e Statistica" (2006). Springer.

[6] A. Di Ciaccio, S. Borra, "Statistica" (2008). Mc Graw-Hill.

[7] S.Mac Lane, G.Birkhoff, "Algebra" (1999). Mursia.

[8] Eli Bendersky, "Efficient modular exponentiation algorithms" (2009).
http://eli.thegreenplace.net/2009/03/28/efficient-modular-exponentiation-algorithms/

[9] Francisco Rodríguez Henríquez, "Modular Exponentiation".
http://delta.cs.cinvestav.mx/~francisco/arith/expo.pdf

Ringraziamenti

Ringrazio il Professor Aliffi per il grande aiuto concessomi, per la Sua disponibilità, e soprattutto per la passione verso il Suo lavoro che mi ha colpita e galvanizzata ogni volta che ci siamo confrontati.

Ringrazio affettuosamente tutta la mia famiglia allargata, e soprattutto la mia mamma, Alberto, i miei cari nonni Paola e Gianfranco, che hanno avuto tanta pazienza e mi dimostrano ogni giorno tutto il loro bene; ora è giunto il mio turno, non vedo l'ora di ricambiare tutte le attenzioni che mi avete dedicato nonostante tutto.

Voglio rivolgere un pensiero speciale a Giuseppe, non smetti mai di stupirmi, voglio vivere questo giorno insieme a te!

Infine, voglio stringere in un abbraccio le persone a me care: la mia amica Alma compagna di avventure matematiche e non, la mia Mucci sempre al mio fianco anche quando lontana, i cari Manuela, Amalia, Gianni, Michele, Lorenzo, e tutti gli amici che non si dimenticano mai!

Non potrei desiderare persone migliori di voi al mio fianco in questo giorno importante!