Crittografia a chiave pubblica: uno sguardo alle vulnerabilità di RSA e Diffie-Hellman

Leonardo Alfreducci

Relatori
Dott. Gaspare Ferraro
Prof.ssa Anna Bernasconi

Università di Pisa
Dipartimento di Informatica

Pisa, 7 ottobre 2022

Indice

Parte 1

Introduzione

Introduzione

- Una grandissima quantità di informazioni viaggia attraverso la rete: è dunque di fondamentale importanza proteggere i dati che vengono scambiati.
- Si passeranno in rassegna i due protocolli più usati per lo scambio di chiave: RSA e Diffie-Hellman, quest'ultimo analizzato su campo primo e su curve ellittiche.
- Lo scopo della tesi è quello di andare al di là di una trattazione teorica di questi due protocolli, concentrandosi piuttosto sull'aspetto pratico.

Parte 2

RSA

RSA: la teoria dietro al protocollo

- È un cifrario asimmetrico. Sono dunque presenti due coppie di chiavi:
  - (e, n) utilizzata per cifrare (*chiave pubblica*);
  - (d, n) utilizzata per decifrare (*chiave privata*).
- Si scelgono due numeri primi p e q.
- Si calcola $n = p \cdot q$ e $\phi(n) = (p-1) \cdot (q-1)$.
- Si sceglie $e < \phi(n)$ tale che $gcd(e, n) = 1$.
- Si calcola $d = e^{-1} \mod \phi(n)$.
- Tutti i passi descritti possono essere svolti in tempo polinomiale.

RSA: cifratura e decifrazione

- Per cifrare un messaggio m è sufficiente calcolare il crittogramma c come:

$$c = m^e \mod n.$$

- Per ottenere il messaggio m dato c è sufficiente calcolarlo come:

$$m = c^d \mod n.$$

RSA: uno sguardo alla sicurezza

- La sicurezza di RSA è garantita grazie al problema della fattorizzazione di un numero n come prodotto di due fattori $p \cdot q$.
- Per questo è importante scegliere due fattori primi molto grandi, tale che il modulo sia almeno 2048 bit, meglio ancora se 3072 bit.
- Nel 1999 è stato fattorizzato RSA-512 in circa 7 mesi utilizzando centinaia di calcolatori e impiegando l'equivalente di 8400 anni di CPU.
  - Nel 2009 lo stesso attacco poteva essere effettuato in 83 giorni da un solo calcolatore.
- Nel 2020 il numero più grande fattorizzato ha 829 bit, impiegando l'equivalente di 2700 anni di CPU.

RSA: un'analisi sperimentale sulla sicurezza

- Sono stati implementati tre algoritmi per la fattorizzazione:
  - *Wheel factorization*: fondamentalmente un brute force sul numero, cercando i divisori;
  - *Pollard's rho factorization*: di natura probabilistica, è quello più efficiente;
  - *Fermat factorization*: è più veloce se i due numeri primi sono vicini tra loro.
- Sono stati fattorizzati moduli da 120 bit utilizzando l'algoritmo *Pollard's rho* in poco meno di un'ora su un moderno calcolatore.

RSA: l'esponente pubblico e

- L'esponente pubblico non dovrebbe essere troppo grande per velocizzare la cifratura.
- Con l'*algoritmo delle quadrature successive*, l'operazione può essere svolta in tempo $O(\log_2 e + hm(e))$, dove $hm(e)$ rappresenta il *peso di Hamming*.
  - Il peso di Hamming rappresenta il numero di simboli diversi dal simbolo 0 dell'alfabeto utilizzato.
- L'esponente pubblico, dato che non contiene alcuna informazione, viene generalmente riutilizzato per molteplici operazioni.

RSA: valori più utilizzati di e con i rispettivi *pesi di Hamming*

X.509			*PGP*			*Combinati*		
e	$hm(e)$	%	e	$hm(e)$	%	e	$hm(e)$	%
65537	2	98.4921	65537	2	48.8501	65537	2	95.4933
17	2	0.7633	17	2	39.5027	17	2	3.1035
3	2	0.3772	41	3	7.5727	41	3	0.4574
35	3	0.1410	19	3	2.4774	3	2	0.3578
5	2	0.1176	257	2	0.3872	19	3	0.1506
7	3	0.0631	23	4	0.2212	35	3	0.1339
11	3	0.0220	11	3	0.1755	5	2	0.1111
47	5	0.0101	3	2	0.0565	7	3	0.0596
13	3	0.0042	21	3	0.0512	11	3	0.0313
65535	16	0.0011	$2^{127}+3$	3	0.0248	257	2	0.0241
altri	-	0.0083	altri	-	0.6807	altri	-	0.0774

RSA: gli schemi di padding

- Prima di essere cifrato mediante RSA, ogni messaggio viene modificato con gli schemi di padding.
- Gli schemi di padding sono importanti in crittografia:
  - aggiungono una componente di casualità;
  - non rendono possibile un recupero anche parziale del messaggio, fissandone univocamente la lunghezza.
- Due degli schemi più utilizzati sono *PKCS1 v1.5* ed *OAEP*.

RSA: malleabilità senza il padding

- Il padding aiuta ad evitare che RSA sia *malleabile*.
  - Ad esempio, se un attaccante conosce $c = m^e \mod n$, che non utilizza il padding, può sostituire $c' = c \cdot 2^e \mod n$.
  - Quando c' verrà decifrato, si otterrà $2m$ invece che l'originario m.
- Con il padding questa modifica molto semplice non è più possibile.

RSA: generazione errata della chiave

- L'esponente e deve essere scelto coprimo con $\phi(n)$.
- In una pre-release di Windows 10, non veniva effettuato il controllo che $gcd(e, \phi(n)) = 1$ nel momento in cui veniva scelto l'esponente pubblico.
- Il corretto funzionamento di RSA è compromesso e la decifrazione non è più possibile.

RSA: la probabilità di scegliere l'esponente pubblico errato

- Ma quanto spesso questo problema si verifica nella pratica?
- Per $e = 65537$, la probabilità P che $e|(p-1)$ *oppure* $e|(q-1)$ è data da

$$P < \frac{1}{32000}$$

  - Il caso $e^i \,|(p-1)\cdot(q-1)$ con $i > 2$ non verrà trattato.
- Windows 10 è utilizzato da oltre un miliardo di dispositivi.
  - Più di 30000 utenti coinvolti.

RSA: il recupero dei messaggi erroneamente cifrati

- Se la chiave viene generata in modo errato ad ogni crittogramma potrebbero corrispondere e messaggi che lo generino.
- E se i messaggi perduti sono importanti?
- Il recupero dei messaggi è esponenziale nella dimensione di e.
  - Per fortuna, e viene generalmente scelto basso.
  - Per $e = 65537$ il recupero dei messaggi con un moderno calcolatore avviene in circa 30 secondi.

RSA: scartare i messaggi durante il recupero

- Come si possono scartare i messaggi automaticamente?
- Analizzando il contesto.
- Analizzando i messaggi che rispettano le caratteristiche degli schemi di padding:
  - con *OAEP*, ci si aspetta solo un risultato (il messaggio originario);
  - con *PKCS1 v1.5* ci si aspetta almeno un falso positivo.

RSA: esponente pubblico corto

- Un bug imponeva $e = 1$.
- Si deve prestare attenzione che intervenga la riduzione in modulo.
- In generale moduli corti possono facilitare gli attacchi basati sui reticoli.

RSA: moduli ripetuti

- È comune che uno stesso modulo n sia condiviso tra più host.
  - Il 4% dei moduli usati in HTTPS risulta condiviso tra più host.
  - Il 60% delle chiavi SSH e il 65% di quelle usate per IPv4 risultano condivise.
  - Non è una vulnerabilità se gli host non sono correlati.
- Molti router e dispositivi della stessa linea di un produttore condividono lo stesso modulo: si possono decifrare i testi a vicenda.
- A causa di problemi con il PRNG e con i seed iniziali molte chiavi sono risultate uguali.

RSA: un fattore in comune

- Se un primo è condiviso tra due moduli n_1 e n_2 è possibile trovare facilmente l'altro primo come $gcd(n_1, n_2)$.
- Esistono dataset pubblici contenenti moduli RSA, per verificare se uno dei due primi è condiviso.

RSA: vulnerabilità DROWN

- Nella sua prima versione sfruttava il semplice schema di padding *PKCS1 v1.5*.
- Si può effettuare se il server manda un errore se il padding è errato.
- Per questo attacco sono necessarie milioni di query.
  - Sfruttando il messaggio di errore un attaccante può restringere i possibili valori assunti da m.
- Si stima che nel 2016 il 33% dei siti HTTPS fossero vulnerabili a questo attacco.

RSA: conclusioni

- Al giorno d'oggi esistono alternative migliori per lo scambio della chiave.
- RSA non è sicuro per utilizzi post-quantistici.
  - L'*algoritmo di Shor* sui computer quantistici permette di calcolare la fattorizzazione in tempo polinomiale probabilistico.
- Ridurre l'utilizzo di RSA è difficile per ragioni di retrocompatibilità.
  - Di TLS v1.3 è stata ritardata l'uscita perché al suo interno ha eliminato lo scambio di chiavi basato su RSA.

Parte 3

Diffie-Hellman

Diffie-Hellman: la teoria dietro al protocollo

-

IN SUPREMÆ DIGNITATIS
1343

IN SUPREMÆ DIGNITATIS
1343

IN SUPREMÆ DIGNITATIS
1343