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. È dunque presente una coppia 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 $d = e^{-1} \mod \phi(n)$.
- Tutte le operazioni descritte possono essere svolte in tempo polinomiale.

RSA: cifratura e decifratura

- 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: l'esponente pubblico e

- L'esponente pubblico, dato che non contiene alcuna informazione, viene generalmente riutilizzato per molteplici operazioni.

X.509		*PGP*		*Combinati*	
e	%	e	%	e	%
65537	98.4921	65537	48.8501	65537	95.4933
17	0.7633	17	39.5027	17	3.1035
3	0.3772	41	7.5727	41	0.4574
35	0.1410	19	2.4774	3	0.3578
altri	0.2264	altri	1.6271	altri	0.588

- Un bug di un'implementazione di SaltStack imponeva $e = 1$.
- Si deve prestare attenzione che intervenga la riduzione in modulo.

RSA: malleabilità

- RSA è *malleabile*.
  - Ad esempio, se un attaccante conosce $c = m^e \mod n$, 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: 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.

RSA: generazione errata della chiave

- L'esponente e deve essere scelto coprimo con $\phi(n)$.
- In una pre-release di Windows 10, nel 2019, 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 decifratura 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 venga scelto in modo errato è data da

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

- 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.
- 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.
- Per scartare i messaggi errati, si possono utilizzare gli schemi di padding.

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.
- Nel 2013, molti router e dispositivi della stessa linea di un produttore condividevano lo stesso modulo: si potevano decifrare i testi a vicenda.

Parte 3

Diffie-Hellman su campo primo

DH su campo primo: la teoria dietro al protocollo

- È un protocollo per lo scambio di chiave.
- A e B, che vogliono comunicare, si devono accordare su due parametri:
  - un numero primo p;
  - un generatore g di $\mathbb{Z}_p^*$.
- Per lo scambio di chiave:
  - A e B scelgono due interi casuali a, b tali che $1 < a, b < p - 1$;
  - A calcola $n_A = g^a \mod p$, B calcola $n_B = g^b \mod p$;
  - A e B si scambiano rispettivamente n_A ed n_B;
  - A calcola $k = n_B^a \mod p$, B calcola $k = n_A^b \mod p$;
  - k è condivisa poiché $k = g^{a \cdot b} \mod p$.

DH su campo primo: uno sguardo alla sicurezza

- Un attaccante per decifrare la comunicazione dovrebbe calcolare il logaritmo discreto:

$$a = \log_g n_A \quad \textit{oppure} \quad b = \log_g n_B.$$

  - Ad oggi non sono noti algoritmi eseguire questa operazione in tempo polinomiale.
  - L'attacco migliore è l'*Index calculus*, che ha complessità $O(2^{\sqrt{b \cdot \ln b}})$ per chiavi di b bit.
- p dal 2013 dovrebbe essere almeno di 2048 bit.
  - Il calcolo del logaritmo discreto con p a 530 bit è stato effettuato nel 2007.

DH su campo primo: moduli non primi

- In alcune implementazioni p non è un primo.
- È una vulnerabilità solo se p ha fattori primi molto piccoli.
  - Si potrebbe usare il *Teorema cinese del resto*.
  - Il calcolo del logaritmo discreto si presume sia difficile tanto quanto il problema della fattorizzazione.

DH su campo primo: valori da evitare

- Sottogruppi di ordine 1 e 2 sono da evitare come scambi di chiave.
- Se g genera l'intero gruppo, allora sono presenti i sottogruppi di ordine 1 e 2.
  - Se i primi i parametri una determinata struttura, ovvero $p = 2q + 1$, e g genera un sottogruppo di ordine q, questo non accade.
  - Se viene scambiato il valore 1, il segreto condiviso non può che essere 1.
  - Più grave il caso in cui venga scambiato il valore -1.

DH su campo primo: un bit insicuro

- Con il sottogruppo di ordine 2 la situazione è più grave.
  - B è un host che ricicla la stessa chiave Diffie-Hellman per molteplici connessioni.
  - A (l'attaccante) manda a B il valore di scambio $n_A = -1$.
  - La chiave condivisa non può che essere 1 o -1.
  - Avendo modo di verificare quale delle due sia la chiave risultante, A può capire se il segreto di B è un numero pari o dispari, facendo perdere così un bit di sicurezza a B.
- Nel 2016 il 3% dei server HTTPS e il 34% dei server SSH accettava -1 come valore di scambio.

Parte 4

Diffie-Hellman su curve ellittiche

Crittografia su curve ellittiche: una panoramica

- La crittografia su curve ellittiche, a parità di sicurezza, richiede chiavi di molti meno bit.
  - Di conseguenza, le operazioni sono più veloci.
- Si comparano nella seguente tabella i bit necessari a parità di sicurezza nei tre protocolli descritti.

RSA e DH (bit del modulo)	ECC (bit dell'ordine)
1024	160
2048	224
3072	256
7680	384
15360	512

Crittografia su curve ellittiche: il protocollo Diffie-Hellman

- Diffie-Hellman si presta molto bene all'applicazione su curve ellittiche.
- È stato adottato su larga scala negli ultimi 10 anni.
- Ad oggi è utilizzato da più del 65% degli scambi di chiave.

DH su curve ellittiche: la teoria dietro al protocollo

- A e B, che vogliono comunicare, si devono accordare su due parametri:
  - una curva ellittica prima $E_p(a,b) = \{(x,y) \in \mathbb{Z}_p^2 | y^2 \mod p = (x^3 + a \cdot x + b) \mod p\} \cup \{0\}$;
  - un generatore G primo facente parte di un sottogruppo della curva di ordine n.
- Per lo scambio di chiave:
  - A e B scelgono due interi casuali a, b tali che $a, b < n$;
  - A calcola $P_A = a \cdot G$, B calcola $P_B = b \cdot G$;
  - A e B si scambiano rispettivamente P_A ed P_B;
  - A calcola $k = a \cdot P_B$, B calcola $k = b \cdot P_A$;
  - k è condivisa poiché $k = n_A \cdot n_B \cdot G$.

DH su curve ellittiche: uno sguardo alla sicurezza

- La sicurezza è basata sulla difficoltà di calcolare, dati due punti P e Q, il più piccolo intero k tale che $P = k \cdot Q$.
  - Chiamato *problema della risoluzione del logaritmo discreto su curve ellittiche*.
  - Il migliore attacco ad oggi è *Pollard rho* che ha complessità $O(2^{b/2})$.
- Si fa affidamento a curve sicure, scelte da una lista del NIST (National Institute of Standards and Technology).

DH su curve ellittiche: i parametri della curva *P-384*

- È una delle curve ellittiche più utilizzate e raccomandate.
- $p = 2^{284} - 2^{128} - 2^{96} + 2^{32} - 1$;
- $a = -3$;
- $b = 27580193559959705877849011840389048093056905856361568521\backslash$ $428707301988689241309860865136260764883745107765439761230575$, avente la proprietà che $2^{383} < b < 2^{384}$;
- $N \simeq 2^{384} - 2^{190}$, che rappresenta l'*ordine della curva*, ovvero il numero di punti che ammette al suo interno.

DH su curve ellittiche: attacco con curva invalida

- Attacco simile a quello descritto per DH su campo primo.
- Se il valore inviato non giace sulla curva ed ha un ordine basso q_i, un attaccante può calcolare il segreto della vittima modulo q_i.
- Se la vittima ricicla il suo segreto per molteplici connessioni, effettuando questo attacco più volte si può recuperare l'intero segreto.
- Si dovrebbe controllare che il punto inviato giaccia sulla curva corretta.
  - Nel 2015 tre di otto popolari librerie TLS non effettuavano questo controllo.
  - Si è stimato che nel 2015 lo 0,8% dei siti HTTPS non effettuavano questo controllo.

Parte 5

Conclusioni

Conclusioni

- RSA è molto più soggetto ad errori di implementazione e per questo si cerca di non utilizzarlo.
- La strada è sempre più verso le curve ellittiche.
- I protocolli presentati non sono sicuri per utilizzi post-quantistici.
  - L'*Algoritmo di Shor* sui computer quantistici permette di calcolare la fattorizzazione e il logaritmo discreto per le curve ellittiche in tempo polinomiale probabilistico.
- La crittografia è in continuo studio ed evoluzione: ciò che oggi è considerato sicuro domani potrebbe non esserlo.
  - La fattorizzazione e il calcolo del logaritmo discreto sono problemi che ad oggi non conoscono algoritmi polinomiali per il loro calcolo, ma potrebbero esistere.
  - È importante e fondamentale restare aggiornati con gli studi.
- A luglio 2022 il NIST ha pubblicato quattro algoritmi resistenti ai computer quantistici.
  - Un mese dopo uno di questi algoritmi è stato violato utilizzando un classico calcolatore.

Fine

Grazie per l'attenzione!

Parte 6

Extra

Extra: attacco con curva invalida

- Si supponga che A sia l'attaccante e B la vittima.
- Si supponga che A mandi il punto P_A come valore di scambio, avente un basso ordine 5.
  - Questo significa che $5 \cdot P_A = O$.
- La chiave k non potrà che assumere uno tra i seguenti valori:

$$1 \cdot P_A \qquad 2 \cdot P_A \qquad 3 \cdot P_A \qquad 4 \cdot P_A \qquad 5 \cdot P_A = O.$$

- Si supponga che $k = 3 \cdot P_A$.
- Allora n_B avrà necessariamente la forma $n_B = 3 + 5 \cdot l$, per qualche l.
  - La chiave k può essere scritta come

$$k = (3 + 5 \cdot l) \cdot P_A = 3 \cdot P_A + 5 \cdot P_A \cdot l = 3 \cdot P_A.$$

- A è riuscito a calcolare $n_B \mod 5 = 3$.

-