Azzolini Riccardo 2021-05-03

Gestione delle chiavi simmetriche

1 Needham-Schroeder con chiave segreta

Il protocollo **Needham-Schroeder con chiave segreta** è un protocollo di key transport nel quale un'entità fidata S, il **Key Distribution Center**, genera una chiave simmetrica di sessione K_{AB} per due utenti A e B e la invia ai due utenti usando esclusivamente la cifratura simmetrica. Per poter trasportare la chiave in modo sicuro S condivide una diversa chiave simmetrica persistente (cioè non di sessione, usata per più esecuzioni del protocollo) con ciascuno dei due utenti:

- A e S condividono la chiave segreta persistente K_{AS};
- B e S condividono la chiave segreta persistente K_{BS}.

Questo protocollo è importante perché è alla base di un protocollo di autenticazione molto utilizzato, Kerberos, che verrà presentato più avanti.

1.1 Funzionamento

Quando A vuole iniziare la comunicazione con B, si rivolge a S per richiedere la generazione di una chiave di sessione. Il messaggio di richiesta contiene l'identificatore dell'utente richiedente A, l'identificatore dell'utente B con cui vuole comunicare, e un nonce N_a:

(1) $A \to S$: $\mathrm{ID}_A, \mathrm{ID}_B, N_a$

S risponde con il messaggio

(2) $S \to A$: $\{N_a, K_{AB}, \mathrm{ID}_B, \{K_{AB}, \mathrm{ID}_A\}_{K_{BS}}\}_{K_{AS}}$

che è cifrato con K_{AS} in modo che solo A possa leggerlo (oltre a S, dato che la cifratura è simmetrica). Inoltre, se la decifratura con K_{AS} dà il risultato corretto A è sicuro che tale messaggio sia stato generato da S, l'unica altra entità che possiede K_{AS}. Il messaggio contiene:

- il nonce N_a, che dà ad A la prova che questo messaggio sia attuale, "fresh", generato in risposta al precedente messaggio di richiesta e non invece riutilizzato da una sessione precedente per effettuare un attacco replay;
- la chiave di sessione K_{AB}, che A memorizza in modo da poterla successivamente utilizzare;

- l'identificatore dell'utente B con cui A ha chiesto di comunicare (cioè quello con cui verrà condivisa la chiave di sessione), che serve a prevenire attacchi in cui nel messaggio (1) si sostituisce ID_B con l'identificatore dell'attaccante;
- un componente ulteriormente cifrato con la chiave K_{BS}, che B possiede ma A no, quindi A non può decifrare questa parte del messaggio ma può inviarla così com'è a B.

Il passo successivo è appunto l'invio a B del messaggio cifrato con K_{BS}:

(3) $A \to B\colon\ \{K_{AB}, \mathrm{ID}_A\}_{K_{BS}}$

Decifrando questo messaggio, B ottiene le seguenti informazioni:

- il messaggio è stato veramente generato da S, l'unica entità oltre a B che possiede la chiave K_{BS};
- la chiave di sessione è K_{AB};
- questa chiave di sessione è condivisa con, e deve essere usata per comunicare con, l'utente il cui identificatore è riportato nel messaggio, cioè A.

Adesso B deve ottenere la prova che l'utente con cui sta attualmente comunicando sia effettivamente l'A con cui la chiave K_{AB} è condivisa. A tale scopo invia un nonce N_b cifrato con la chiave di sessione:

(4) $B \to A\colon\ \{N_b\}_{K_{AB}}$

Nel protocollo Needham-Schroeder con chiave pubblica al fine di realizzare quest'autenticazione è sufficiente che A decifri il nonce con la propria chiave privata e lo ricifri con la chiave pubblica di B. Invece, nel protocollo con chiave segreta ciò non funzionerebbe: siccome si usa la cifratura simmetrica con una chiave condivisa da entrambi gli utenti, il messaggio contenente il nonce decifrato e ricifrato sarebbe identico al messaggio (4) non decifrato, quindi un attaccante potrebbe semplicemente rispedire il messaggio senza bisogno di possedere K_{AB} per decifrarlo. La soluzione è richiedere che A modifichi il valore decifrato N_b del nonce in un qualche modo conosciuto da A e B prima di ricifrarlo e rispedirlo, in modo da ottenere un messaggio di risposta diverso ma comunque prevedibile. Ad esempio, si potrebbe sottrarre uno al nonce:

(5) $A \to B\colon\ \{N_b - 1\}_{K_{AB}}$

Se B decifrando questo messaggio ottiene il valore corretto $N_b - 1$, allora ha la prova che l'utente con cui sta comunicando sia in possesso di K_{AB}, e sapendo dal messaggio (3) che l'entità fidata S ha generato K_{AB} per l'utente A ha la prova che l'altro utente è effettivamente A. Dopo questo passo, A e B possono iniziare a scambiarsi dati usando la chiave di sessione appena condivisa.

Riassumendo, i messaggi scambiati in una sessione del protocollo sono:

(1) $A \to S\colon\ \mathrm{ID}_A, \mathrm{ID}_B, N_a$

(2) $S \to A$: $\{N_a, K_{AB}, \mathrm{ID}_B, \{K_{AB}, \mathrm{ID}_A\}_{K_{BS}}\}_{K_{AS}}$
(3) $A \to B$: $\{K_{AB}, \mathrm{ID}_A\}_{K_{BS}}$
(4) $B \to A$: $\{N_b\}_{K_{AB}}$
(5) $A \to B$: $\{N_b - 1\}_{K_{AB}}$

1.2 Vulnerabilità

Il protocollo definito in questo modo è vulnerabile a un **attacco replay**, un tipo di attacco nel quale l'attaccante riutilizza un messaggio ottenuto da una precedente sessione del protocollo per acquisire il controllo della comunicazione o invalidare il protocollo.

Si supponga che un attaccante E conosca una chiave di sessione K'_{AB} usata in precedenza tra A e B[1] e abbia tenuto traccia della sessione del protocollo Needham-Schroeder in cui tale chiave era stata scambiata, in particolare del messaggio

(3) $A \to B$: $\{K'_{AB}, \mathrm{ID}_A\}_{K_{BS}}$

Quando poi A e B iniziano una nuova sessione del protocollo, nella quale si dovrebbero scambiare una diversa chiave K_{AB}, l'attaccante può sostituire il messaggio

(3) $A \to B$: $\{K_{AB}, \mathrm{ID}_A\}_{K_{BS}}$

con quello precedentemente memorizzato. Così, B riceve la vecchia chiave K'_{AB} invece della nuova K_{AB}, e non ha modo di accorgersi che essa è compromessa, non è nuova. Se poi E intercetta i successivi messaggi inviati da B ad A allora B inizia a usare K'_{AB} per comunicare con E pensando che sia A.

La soluzione per prevenire l'attacco è aggiungere al messaggio (3) delle informazioni sulla validità temporale del messaggio stesso (come minimo un timestamp che indichi quando è stato emesso), in modo che B si possa accorgere se tale messaggio non è stato generato nella sessione corrente. Questa soluzione non verrà mostrata qui, ma la si vedrà direttamente nell'ambito del protocollo Kerberos.

2 Scambio di chiavi Diffie-Hellman

Lo **scambio di chiavi Diffie-Hellman** è un protocollo di key agreement che viene utilizzato come metodo efficiente per lo scambio di chiavi simmetriche in molti prodotti commerciali (ad esempio il protocollo SSL/TLS usato per la sicurezza delle comunicazioni in ambito Web). Esso è un algoritmo a chiave pubblica, cioè di crittografia asimmetrica, la cui sicurezza si basa sulla difficoltà del calcolo dei logaritmi discreti.

[1] Come l'attaccante conosca la chiave non è rilevante per il funzionamento dell'attacco, ma potrebbe ad esempio averla ottenuta tramite un attacco a forza bruta sui messaggi cifrati con K'_{AB} che A e B si sono scambiati in seguito alla condivisione della chiave. Un attacco a forza bruta è plausibile perché non è necessario che K'_{AB} sia recente, dunque un attaccante potrebbe anche molto tempo per eseguire l'attacco.

2.1 Funzionamento

In un'esecuzione dello scambio di chiavi Diffie-Hellman due utenti A e B devono computare localmente un numero, una stringa di bit, che verrà usata come chiave.

Innanzitutto, A e B devono essersi messi d'accordo su due *parametri pubblici*, q e g. Essi sono pubblici in quanto il sistema rimane sicuro se l'attaccante li conosce. Inoltre, siccome non è necessario mantenerli segreti, g e q possono essere condivisi anche da altri utenti oltre ad A e B.

Successivamente, si eseguono i seguenti passi:

1. Gli utenti A e B generano rispettivamente i *parametri privati* X_a e X_b, aventi valori compreso tra 1 e $q-1$:

   $$A\colon X_a \in \{1,\ \ldots,\ q-1\} \qquad B\colon X_b \in \{1,\ \ldots,\ q-1\}$$

   Si può allora intuire che il parametro q deve essere abbastanza grande da avere molte possibili scelte di valori dei parametri privati in $\{1,\ \ldots,\ q-1\}$.

2. A e B calcolano rispettivamente

   $$A\colon Y_a = g^{X_a} \bmod q \qquad B\colon Y_b = g^{X_b} \bmod q$$

3. A e B si scambiano i valori Y_a e Y_b, che come q e g sono parametri non sensibili: se anche un attaccante li leggesse, la sicurezza del protocollo non sarebbe compromessa.

4. A e B calcolano rispettivamente

   $$A\colon K = Y_b^{X_a} \bmod q \qquad B\colon K = Y_a^{X_b} \bmod q$$

All'ultimo passo i valori calcolati da A e B sono entrambi chiamati K perché i due utenti ottengono *lo stesso risultato*, K è uguale per entrambi. Allora, A e B si sono messi d'accordo su una stringa di bit K che possono usare come chiave segreta.

Il fatto che K sia uguale per entrambi gli utenti può essere dimostrato usando le proprietà delle potenze insieme alla proprietà dell'aritmetica modulare già sfruttata per l'ottimizzazione della cifratura/decifratura in RSA:

$$ab \bmod n = (a \bmod n)(b \bmod n) \bmod n \tag{M}$$

Grazie a tale proprietà, il valore K calcolato da A può essere riscritto come

$$\begin{aligned}
K &= Y_b^{X_a} \bmod q && \text{[definizione di } K \text{ per } A\text{]} \\
&= (g^{X_b} \bmod q)^{X_a} \bmod q && \text{[definizione di } Y_b\text{]} \\
&= \underbrace{(g^{X_b} \bmod q) \cdots (g^{X_b} \bmod q)}_{X_a \text{ volte}} \bmod q && \text{[definizione di potenza]} \\
&= \underbrace{g^{X_b} \cdots g^{X_b}}_{X_a \text{ volte}} \bmod q && \text{[proprietà (M)]} \\
&= (g^{X_b})^{X_a} \bmod q && \text{[definizione di potenza]}
\end{aligned}$$

poi con un ragionamento analogo il K calcolato da B si può riscrivere come

$$K = Y_a^{X_b} \bmod q = (g^{X_a})^{X_b} \bmod q$$

e per le proprietà delle potenze e la commutatività del prodotto questi due valori sono uguali,

$$(g^{X_b})^{X_a} \bmod q = g^{X_b X_a} \bmod q = g^{X_a X_b} \bmod q = (g^{X_a})^{X_b} \bmod q$$

dunque il valore calcolato da entrambi gli utenti è

$$K = g^{X_a X_b} \bmod q$$

In conclusione, si è verificato che tramite lo scambio di dei parametri pubblici e dei calcoli locali con dei parametri privati i due utenti riescono a ricavare una chiave segreta condivisa. L'uso dei parametri pubblici e privati è il motivo per cui Diffie-Hellman rientra nella famiglia degli schemi di crittografia asimmetrica, anche se il suo scopo è la condivisione di una chiave e non la cifratura dei messaggi. Rimane ora da capire come generare i vari parametri per fare in modo che il protocollo sia sicuro.

2.2 Logaritmo discreto

Per analizzare la sicurezza dello scambio di chiavi Diffie-Hellman bisogna studiare come l'attaccante potrebbe computare K conoscendo i valori dei parametri pubblici q, g, Y_a e Y_b ma non quelli dei parametri privati X_a e X_b. L'unico modo di farlo è ricavare almeno il valore di X_a o, equivalentemente, quello di X_b. X_a può essere determinato a partire da Y_a trovando il valore X tale che $Y_a = g^X \bmod q$, mentre partendo da Y_b si determina in modo analogo il valore di X_b. In generale, l'operazione di trovare un X tale che $Y = g^X \bmod q$ è l'inverso dell'operazione di esponente, cioè il logaritmo, che nell'ambito dell'aritmetica modulare prende il nome di *logaritmo discreto*.

Più formalmente, dato un numero primo q e dato l'insieme $\mathbb{Z}_q^* = \{1, \ldots, q-1\}$ di tutti i possibili valori di resto modulo q eccetto 0 (l'esclusione dello 0 è indicata dall'asterisco

nel nome dell'insieme), esiste un numero g che è **radice primitiva** di q: per ogni $y \in \mathbb{Z}_q^* = \{1, \ldots, q-1\}$ esiste un i tale che

$$y = g^i \bmod q$$

Allora, il valore i è il **logaritmo discreto** di y per la base g modulo q. Ad esempio, $g = 2$ è radice primitiva di $q = 11$ in quanto genera tutti i possibili valori di resto $\mathbb{Z}_{11}^* = \{1, \ldots, 10\}$,

$$\begin{aligned} 2^1 \bmod 11 &= 2 & 2^6 \bmod 11 &= 9 \\ 2^2 \bmod 11 &= 4 & 2^7 \bmod 11 &= 7 \\ 2^3 \bmod 11 &= 8 & 2^8 \bmod 11 &= 3 \\ 2^4 \bmod 11 &= 5 & 2^9 \bmod 11 &= 6 \\ 2^5 \bmod 11 &= 10 & 2^{10} \bmod 11 &= 1 \end{aligned}$$

e si dice ad esempio che:

- il logaritmo discreto di $y = 1$ per la base $g = 2$ modulo $q = 11$ è $i = 10$, in quanto $1 = 2^{10} \bmod 11$;
- il logaritmo discreto di 9 per la base 2 modulo 11 è 6, in quanto $9 = 2^6 \bmod 11$.

Il **problema del logaritmo discreto**, cioè di trovare un i tale che $y = g^i \bmod q$ dati un numero primo q, una sua radice primitiva g e un numero $y \in \mathbb{Z}_q^*$, è molto difficile da risolvere (il suo calcolo ha una complessità paragonabile a quella della fattorizzazione), e risolverlo è proprio ciò che un attaccante dovrebbe fare per computare K dai parametri pubblici dello scambio di chiavi Diffie-Hellman. Invece, durante una normale esecuzione del protocollo gli utenti A e B devono calcolare y avendo i, g e q, il che è relativamente facile (si calcola la potenza g^i e poi il resto della sua divisione per q). In sostanza, il problema del logaritmo discreto fornisce un calcolo che è facile se si conoscono alcuni dati e difficile altrimenti (per la precisione, una funzione che è facile da calcolare ma difficile da invertire); in questo senso, tale problema è simile alla funzione toziente di Eulero $\phi(n)$ usata in RSA.

Il modo più semplice per cercare di ricavare X_a (oppure X_b) è un attacco a forza bruta: si provano tutti i possibili valori $X \in \mathbb{Z}_q^* = \{1, \ldots, q-1\}$ finché non si trova quello tale che $Y_a = g^X \bmod q$. Siccome l'insieme dei possibili valori di X_a e di X_b è appunto $Z_q^* = \{1, \ldots, q-1\}$, la dimensione di q determina il numero di valori da provare, ovvero il costo di un attacco a forza bruta. È allora importante che q sia molto grande: tipicamente, è un numero codificato in 1024–2048 bit. Riassumendo, i parametri iniziali dello scambio di chiavi Diffie-Hellman devono essere selezionati in questo modo:

- q deve essere un numero primo molto grande;
- g deve essere una radice primitiva di q;
- X_a e X_b devono essere scelti tra i possibili valori di resto $Z_q^* = \{1, \ldots, q-1\}$.

Il problema del logaritmo discreto è ampiamente sfruttato nella crittografia: lo scambio di chiavi Diffie-Hellman fu il primo algoritmo a farne uso, ma anche ad esempio la crittografia a curve ellittiche si basa su di esso.