Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea in Matematica

METODI DI AUTENTICAZIONE A CHIAVE PUBBLICA E PRIVATA

Relatore:
Prof.
DAVIDE ALIFFI

Presentata da:
FRANCESCA BALDUCCI

III Sessione
2013/2014

Indice

Capitolo 1

Introduzione

L'autenticazione è un processo svolto in tempo reale tramite il quale l'identità di un'entità può essere confermata. Comunemente sappiamo che esistono tre metodi per identificare un utente: si possono usare caratteristiche fisiche, quali per esempio sesso, altezza, colore della pelle, colore degli occhi, impronte digitali e scansione della retina, credenziali come carta d'identità, patente ecc., oppure nel caso in cui l'entità da certificare sia una macchina viene spesso utilizzata la conoscenza di un'informazione segreta, questo metodo è molto utile anche quando l'entità che vuole identificarsi non è materialmente nello stesso luogo fisico dell'entità che certifica, si usano in questo caso ad esempio i codici PIN e le parole d'ordine.
Nella realtà quotidiano troviamo molte situazioni in cui ci si avvale dei protocolli di autenticazione: negli accessi remoti ai network si deve essere a conoscenza di username e password validi, per i pagamenti con carte di credito, che se effettuati in un negozio necessitano del controllo della firma del proprietario della carta e se effettuati a distanza della conoscenza di un numero di carta valido insieme alla data di scadenza, nei prelievi con il bancomat nei quali si utilizza sia il possesso di una carta che un codice PIN.
L'autenticazione è una componente critica per la sicurezza, infatti il metodo più usato da coloro che tentano di introdursi in un sistema o disporre dei conti correnti altrui è quello di assumere una falsa identità, in modo da avere accesso ad informazioni riservate o al denaro. Nelle situazioni sopra descritte si possono rilevare molti punti deboli: se l'ID e la password che vengono trasmessi nel network non sono criptati possono essere letti da qualunque altro utente, le carte di credito possono essere clonate, oppure nel caso in cui un pagamento venga effettuato tramite chiamata telefonica chiunque sia in grado di intercettarla può usare i dati personali di chi si sta identificando a proprio piacimento.
Uno schema di identificazione sicuro ed efficiente dovrebbe far sì che chiunque

intercetti uno scambio di informazioni fra due utenti non possa automaticamente essere in grado di impersonare chi tenta di autenticarsi, neppure la stessa entità che in quel frangente identifica l'altra, e dovrebbe allo stesso tempo richiedere calcoli abbastanza semplici, visto che nella maggior parte delle applicazioni pratiche questi vengono eseguiti da delle smart card, le quali svolgono solo calcoli aritmetici.
Nei protocolli interattivi che analizzeremo in seguito verranno coinvolte due parti, che da qui in avanti converremo chiamare Alice e Bob, che comunicano fra loro seguendo un algoritmo che alternativamente invia e riceve informazioni. Ogni esecuzione del protocollo verrà chiamata sessione, mentre ogni passo interno del protocollo verrà chiamato flow. Alla fine di una sessione Bob, o chi ha iniziato la sessione accetta oppure rifiuta e l'interlocutore potrebbe non conoscere la risposta.
Osserviamo già da subito che una caratteristica fondamentale di qualsiasi schema di autenticazione efficiente è avere un certo grado di imprevedibilità: se l'informazione trasmessa tra i due interlocutori Alice e Bob fosse sempre la stessa sarebbe semplice per un terzo riutilizzarla e fingersi uno dei due interlocutori.
Adesso diamo qualche nozione di carattere matematico fondamentale per la comprensione dei prossimi capitoli.

1.1 Conoscenze preliminari

Definizione 1.1. Si chiama *funzione unidirezionale* un'applicazione h:$\mathbb{N} \longrightarrow \mathbb{N}$ (oppure $h : \{0,1\}^N \longrightarrow \{0,1\}^m$) tale che dato $y \in h(\mathbb{N})$ (oppure $y \in (\{0,1\}^n)$) non è possibile trovare $x \in \mathbb{N}$ (oppure $x \in \{0,1\}^n$) tale che $y = h(x)$.

Definizione 1.2. Sia p un numero primo, si ha $\mathbb{Z}_p^*$ è un gruppo ciclico di ordine $p-1$. Inoltre $\forall g \in \mathbb{Z}_p^*$, $\forall \beta \in \mathbb{Z}_p^*$ $\exists! x \in \{\mathbb{Z}_p^* : g^x = \beta$. Si definisce *logaritmo discreto* $x = Dlog_g(\beta)$.

Osservazione 1. Non esistono algoritmi efficienti per il calcolo del logaritmo discreto, perciò la funzione

$$f : \mathbb{Z}_p^* \longrightarrow \mathbb{Z}_p^* \tag{1.1}$$

$$x \longmapsto g^x (mod p) \tag{1.2}$$

potrebbe essere unidirezionale.

Definizione 1.3. Una funzione *hash* crittografica è un'applicazione $h : \{0,1\}^* \longrightarrow \{0,1\}^n$ dove $\{0,1\}^*$ è un insieme di stringhe di lunghezza indefinita, mentre $\mid \{0,1\}^n \mid = 2^n$ con n fissato a priori, tale che

1. $h(x)$ deve essere calcolato in modo efficiente per ogni $x \in \{0,1\}^*$.
2. Dato $y \in \{0,1\}^n$ deve essere impossibile trovare $x \in \{0,1\}^*$ tale che $h(x) = y$ (unidirezionalità).
3. Deve essere impossibile trovare $x_1, x_2 \in \{0,1\}^*$ con $x_1 \neq x_2$ tali che $h(x_1) \neq h(x_2)$.

NOTA: impossibile significa computazionalmente non risolvibile in tempi accettabili.
La terza condizione è detta resistenza alle collisioni.

Le funzioni hash servono a garantire la sicurezza dell'integrità dei dati. Una particolare famiglia di funzioni hash è quella dei Message Authentication Code, o più brevemente MAC, che utilizzano una chiave k. Un modo per crearli è incorporare la chiave k all'interno del messaggio. Supponiamo che Alice e Bob abbiano a disposizione un canale insicuro per scambiarsi informazioni, e una chiave segreta k condivisa, la quale determina una funzione hash h_k. Sia Alice che Bob riescono a calcolare $y = h_k(x)$, quale che sia il messaggio x, e Alice può inviare la coppia (x, y) a Bob sul canale insicuro. Quando Bob riceve (x, y) controlla che $y = h_k(x)$ e in tal caso è sicuro che il messaggio non è stato alterato e che è stato inviato da Alice, visto che è l'unica a conoscere k. La differenza fondamentale tra una funzione hash che non prevede l'uso di una chiave segreta e una che la utilizza è che nel primo caso il valore y deve essere tenuto segreto, mentre con la chiave può essere trasmesso su un canale insicuro, dato che k è nota solo ad Alice e Bob.
Definiamo più precisamente come è fatta una famiglia di funzioni hash con chiave k:

Definizione 1.4. Una famiglia di funzioni hash con chiave k è una quattrupla (X, Y, H, K) che soddisfa le seguenti condizioni:

- X è un insieme di possibili messaggi.
- Y è un insieme finito di possibili digest.
- K è l'insieme delle possibili chiavi ed è finito.
- Per ogni k$\in K$ esiste un'unica funzione hash $h_k \in H$

Se X è un insieme finito si usa chiamare la funzione hash compressione e vale sempre che $| X | > | Y |$.

Capitolo 2

Schemi a chiave segreta

2.1 Introduzione

Ci proponiamo in questo capitolo di trattare gli schemi di autenticazione a chiave segreta, sia unilaterale che reciproca. Negli scambi a chiave segreta gli interlocutori condividono la stessa chiave segreta K e si servono di schemi a sfida e risposta basati sull'uso dei MAC.

2.2 Schemi unilaterali

Un primo esempio di protocollo a sfida e risposta molto semplice prevede che Bob invii un numero random r di sfida ad Alice rappresentata da una stringa di lunghezza prefissata (nella pratica 100 bit sono sufficienti), la quale dopo averlo ricevuto calcolerà $y = MAC_k(r)$ e lo invierà a sua volta a Bob, che dopo aver verificato la correttezza di y sarà in grado di accettare o rifiutare l'identità di Alice.

PROTOCOLLO SCHEMA SFIDA-RISPOSTA (INSICURO)

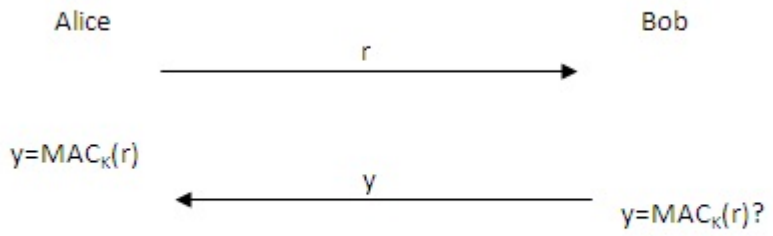
1. Bob sceglie una sfida random r e la invia ad Alice.
2. Alice calcola

$$y = MAC_k(r)$$

   e lo invia a Bob.
3. Bob calcola

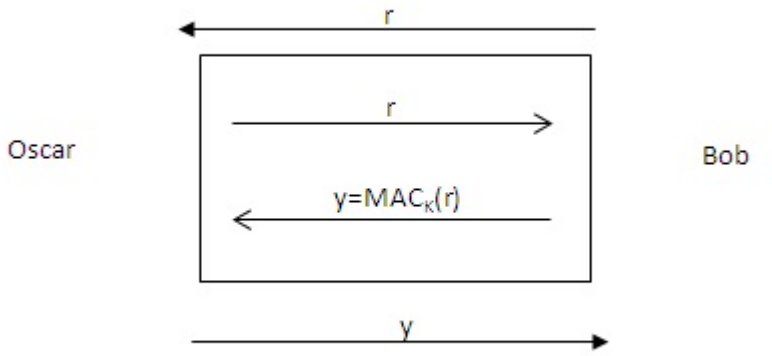
$$y' = MAC_k(r)$$

   se $y = y'$, allora Bob accetta, altrimenti rifiuta.



Si vede subito però che questo protocollo è insicuro, anche disponendo di un MAC affidabile risulta infatti suscettibile ad un attacco di sessione in parallelo in cui un terzo individuo, qui identificato con Oscar, finge di essere Alice aprendo una seconda sessione all'interno della prima in cui chiede a

Bob di identificarsi inviandogli la sfida r che Bob stesso aveva inviato nella prima sessione, e una volta ottenuta risposta riapre la prima sessione nella quale manda a Bob la sua stessa risposta.



Questo attacco è realistico e semplice da eseguire, per questo è prudente creare uno schema di identificazione che possa impedirlo. Un modo semplice è inserire nel MAC l'identità dell'entità che lo ha calcolato.

SCHEMA SFIDA-RISPOSTA (SICURO)

1. Bob sceglie una sfida random r e la invia ad Alice
2. Alice calcola

$$y = MAC_k(ID(Alice) \parallel r)$$

   e lo manda a Bob.

3. Bob calcola

$$y' = MAC_k(ID(ALICE) \parallel r)$$

   se $y = y'$ allora Bob accetta, altrimenti rifiuta.

Le stringhe che rappresentano l'identità di Alice hanno lunghezza prefissata come per le sfide random, contengono le informazioni necessarie per specificare univocamente di quale entità nel network stiamo parlando e hanno un formato standard.
Osserviamo che con questa modifica non si può portare un attacco in parallelo, infatti Oscar riceverebbe da Bob il valore $MAC_k(ID(Bob) \parallel r)$ nella seconda sessione, che non è di nessun aiuto per calcolare $MAC_k(ID(Alice) \parallel r)$, ovvero l'informazione richiesta nella prima sessione per rispondere con successo alla sfida di Bob.
Queste considerazioni ci convincono dell'impossibilità di un attacco in parallelo, ma diamo subito una prova di sicurezza contro gli altri possibili attacchi, con l'ipotesi che la chiave k sia nota solo ad Alice e Bob, che entrambi dispongano di un generatore di numeri casuali perfetti per le sfide random, e che il MAC sia sicuro, più precisamente che non esista un $(\epsilon, Q) - secure$ MAC per appropriati ϵ e Q. Oscar potrebbe aver osservato molte sessioni fra Alice e Bob, ma vediamo che con queste premesse difficilmente riuscirà ad avere successo, cioè a farsi accettare da Alice o Bob. Supponiamo per esempio che Bob accetti, significherebbe che $y = MAC_k(ID(Alice) \parallel r)$ è stato costruito da Alice in una sessione precedente perché solo Alice e Bob conoscono la chiave k e quindi nessun altro al di fuori di loro potrebbe calcolare il MAC utilizzandola. Segue che o Oscar ha calcolato y senza conoscere la chiave k oppure y è stato ricavato da una sessione precedente, copiato e riutilizzato. Entrambe queste opzioni sono da scartare: non può essere stato calcolato né da Bob né da Alice perché Bob calcola solo MAC del tipo $MAC_k(ID(Bob)$ $parallelr)$ mentre se lo avesse calcolato Alice significherebbe che la sfida r è stata utilizzata due volte, evento quasi impossibile con un generatore di numeri casuali perfetto. Non può averlo calcolato Oscar perché, avendo scelto un MAC sicuro, ha possibilità trascurabili di riuscirci.
Forniamo una dimostrazione ancora migliore: con una precisa definizione di sicurezza per il MAC sottostante potremo dare una precisa garanzia di sicurezza per lo schema di identificazione.
Cominciamo con una definizione:

Definizione 2.1. un MAC è detto essere $(\epsilon, Q) - secure$ se l'avversario non può costruire un MAC valido per nessun nuovo messaggio con probabilità più grande di ε, avendo visto MAC validi per al più Q messaggi validi.

Anche in questo caso assumiamo che la chiave k fissata non sia conosciuta dall'avversario, e sia stata usata per costruire tutti i Q messaggi MAC. Questi messaggi esistono per ogni valore Q, ma più esso aumenta più la chiave richiesta è grande, per questo si usano spesso i CBC-MAC. In questo caso

si può assumere anche il tempo necessario all'avversario per computare un messaggio come parametro di sicurezza:

Definizione 2.2. un MAC si dice $(\epsilon, Q, T) - secure$ nel caso in cui l'avversario non possa ingannare Alice o Bob con probabilità più grande di ϵ avendo osservato al massimo Q precedenti sessioni e avendo un tempo computazionale massimo T.

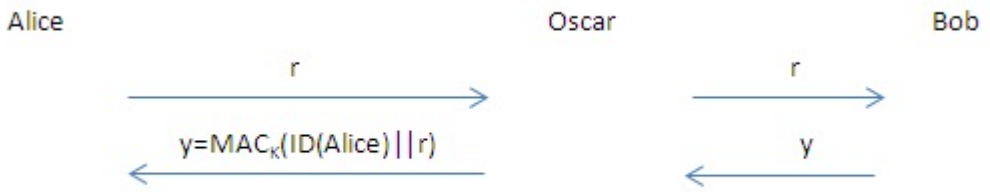
Se supponiamo di basare lo schema di autenticazione su un MAC incondizionatamente sicuro allora lo schema di autenticazione risultante sarà esso stesso incondizionatamente sicuro purché l'avversario abbia accesso ad al più Q MAC validi che utilizzano la stessa chiave k, e che essa sia appropriata. Adesso possiamo stabilire facilmente un limite superiore alla possibilità dell'avversario di ingannare gli interlocutori. Consideriamo i tre casi visti precedentemente:

- come visto prima il valore $y = MAC_k(ID(Alice) \parallel r)$ non poteva essere stato costruito da Bob stesso in qualche altra sessione precedente.(Quindi questo caso è in realtà da non considerarsi)
- supponiamo che il valore y sia stato costruito da Alice in qualche sessione precedente. Si suppone che la sfida r sia un nuovo numero casuale creato da Bob. La probabilità che sia già stato usato in una sessione è $1/2^k$, e ci sono al più Q precedenti sessioni da tenere in considerazione, di conseguenza la probabilità che sia stato usato in una di queste è $Q/2^k$. Se questo avviene allora l'avversario può riutilizzare una sessione precedente.
- supponiamo che y sia un nuovo MAC costruito da Oscar. Allora Oscar avrà probabilità di successo al più ϵ e ciò segue dalla definizione del MAC.

In conclusione la probabilità di Oscar di ingannare Bob è al massimo $Q/2^k + \epsilon$. Ecco che abbiamo dato una misura di sicurezza dello schema in funzione dei parametri utilizzati.

2.3 Modello di attacco intruder in the middle

Illustriamo ora lo scenario intruder in the middle.



A prima vista potrebbe sembrare un attacco a sessione parallela, ma non possiamo considerarlo tale: unendo le due sessioni se ne ottiene una unica il cui risultato è che Alice si è identificata a Bob. Bob ha emesso una sfida r e Alice ha calcolato la risposta corretta alla sfida, Oscar ha semplicemente inoltrato i messaggi ai loro destinatari senza modificarli. In questo caso si dice che Oscar è un avversario benigno.
Diamo una formulazione più chiara che ci permetta di capire che non è un attacco. Innanzi tutto diremo che un avversario è **attivo** se sussiste una di queste condizioni:

- Oscar crea un nuovo messaggio e lo immette nel canale.
- Oscar cambia un messaggio nel canale.
- Oscar devia un messaggio, cioè lo fa arrivare ad un altro destinatario.

Lo scopo dell'avversario è far sì che chi ha iniziato lo schema lo accetti in una sessione in cui è attivo, e nello schema che abbiamo descritto prima Oscar non lo è.
Ovviamente questa discussione è valida se i partecipanti sono onesti, ovvero se seguono lo schema, eseguono calcoli corretti, e non rivelano informazioni all'avversario. Adesso osserviamo che Oscar effettua una fase di raccolta di informazioni prima di intervenire per ingannare Bob, e può essere un avversario passivo se si limita ad osservare, oppure attivo se per esempio ha modo di accedere per un periodo limitato ad un oracolo che calcola i $MAC_k(\cdot)$ per la chiave k (sconosciuta) usata da Alice e Bob. In entrambi i casi si può applicare l'analisi di sicurezza precedente: si può provare che lo schema è sicuro nel modello passivo di raccolta informazioni purché il MAC sia $(\epsilon, Q) - \textit{secure}$ contro gli attacchi ai messaggi conosciuti, e inoltre che sia sicuro nella raccolta attiva purché il MAC sia $(\epsilon, Q) - \textit{secure}$ contro gli attacchi a testo scelto.

2.4 Schemi di autenticazione reciproca

In questi schemi entrambe le parti sono chiamate a provare la loro identità e ad accettare se la sessione termina con successo. L'obiettivo dell'avversario è far sì che uno dei partecipanti onesti accetti dopo uno scambio di messaggi in cui l'avversario è stato attivo. I possibili esiti di uno schema sicuro sono:

- Alice e Bob accettano entrambi al termine di una sessione in cui l'avversario è stato passivo.

- Nessun partecipante onesto accetta dopo una sessione in cui l'avversario è stato attivo.

L'avversario potrebbe essere stato inattivo in una sessione finché uno dei due partecipanti onesti accetta e poi diventare attivo, in questo caso uno dei due accetta e l'altro rifiuta, così che l'avversario comunque fallisce, visto che in una parte della sessione è stato inattivo. Ciò può considerarsi una rottura dello schema, ma non un attacco terminato con successo. Facciamo una lista delle modalità che l'avversario ha per essere attivo in una sessione:

- L'avversario impersona Alice sperando che Bob accetti.

- L'avversario impersona Bob sperando che Alice accetti.

- L'avvesario è attivo in una sessione in cui sono coinvolti entrambi Alice e Bob e prova a fare in modo che accettino.

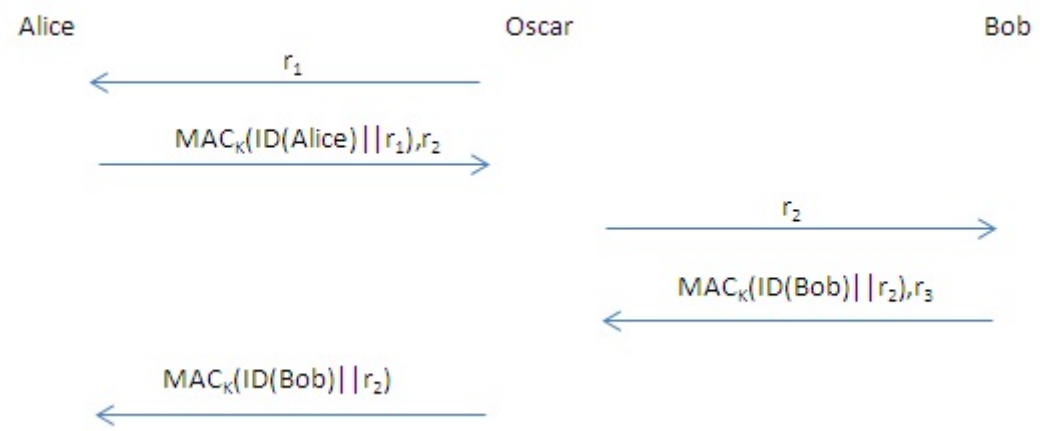
Per realizzare uno schema di autenticazione reciproca si potrebbe ripetere due volte lo schema unilaterale in cui Alice e Bob verificano la loro identità in sessioni separate, ma è più efficiente costruirne uno in cui entrambi si identificano in una sessione sola. Vediamo un primo esempio di protocollo:

PROTOCOLLO DI AUTENTICAZIONE RECIPROCA A SFIDA E RISPOSTA (INSICURO)

1. Bob sceglie una sfida random r_1 e la manda ad Alice.
2. Alice sceglie una sfida random r_2, calcola
$$y_1 = MAC_k(ID(Alice) \parallel r_1)$$
e invia r_2 y_1 a Bob.
3. Bob calcola
$$y' = MAC_k(ID(Alice) \parallel r_1)$$
se $y'_1 = y_1$ allora Bob accetta, altrimenti rifiuta. Bob calcola anche
$$y_2 = MAC_k(ID(Bob) \parallel r_2)$$
e lo invia ad Alice.
4. Alice calcola
$$y'_2 = MAC_k(ID(Bob) \parallel r_2)$$
se $y'_2 = y_2$, allora Alice accetta, altrimenti Alice rifiuta.

Oscar può facilmente portare un attacco a sessione parallela: fingendo di essere Bob inizia una sessione con Alice, e non appena riceve da lei la sfida r_2 nel secondo flow accetta per poi iniziare una seconda sessione (impersonando Alice) con Bob. In questa seconda sessione Oscar invia r_2 a Bob come sfida nel primo flow e quando riceve la risposta di Bob la invia ad Alice (come terzo flow nella prima sessione). A questo punto Alice accetta facendo concludere con successo a Oscar la prima sessione, invece la seconda parte non verrà mai completata.

Notare che in questo scambio Bob non partecipa neppure, ma Oscar riesce a farsi autenticare da Alice come Bob. Vedremo subito che il problema risiede nel fatto che r_2 non è protetto. Apportiamo ora qualche modifica al fine di renderlo sicuro:



SCHEMA DI AUTENTICAZIONE RECIPROCA A SFIDA E RISPOSTA (SICURO)

1. Bob sceglie una sfida random r_1 e la invia ad Alice.
2. Alice sceglie una sfida random r_2, calcola

$$y_1 = MAC_k(ID(Alice) \parallel r_1 \parallel r_2)$$

   e manda r_2 e y_1 a Bob.
3. Bob calcola

$$y_1' = MAC_k(ID(Alice) \parallel r_1 \parallel r_2)$$

   se $y_1' = y_1$, allora Bob accetta altrimenti rifiuta. Bob calcola anche

$$y_2 = MAC_k(ID(Bob) \parallel r_2)$$

   e invia y_2 ad Alice.
4. Alice calcola

$$y_2' = MAC_k(ID(Bob) \parallel r_2)$$

   se $y'2 = y_2$, allora Alice accetta, altrimenti rifiuta.

L'unico cambiamento è nella definizione di y_1 nel secondo passo, infatti adesso il MAC dipende da due sfide random, r_1 e r_2, e cioè il secondo flow dal terzo flow. Il protocollo potrebbe essere analizzato come in precedenza, anche se con un po' più di difficoltà visto che l'avversario può fingersi sia Alice che Bob. Si può calcolare la possibilità che un valore y_1 o y_2 sia riusato da una sessione precedente, così come quella che un avversario calcoli un nuovo MAC da zero. Prima di tutto è impossibile che un valore y_1 sia preso da una sessione e riutilizzato come valore y_2 in un'altra sessione, visto che i valori usati come sfide sono impredicibili. Oscar potrebbe provare ad impersonare Bob per ingannare Alice o viceversa determinando y_2 o y_1 rispettivamente. La probabilità che questi valori possano essere riusati da una sessione precedente è al più $Q/2^k$, assunto che Oscar abbia intercettato al massimo Q MAC. La probabilità che Oscar possa calcolare un nuovo y_2 è al più ϵ . Per questo la probabilità di ingannare Alice o Bob è al più $Q/2^k + 2\epsilon$. Otteniamo il teorema seguente come conclusione:

Teorema 2.4.1. *Supponiamo che il MAC sia un* $(\epsilon, Q)-$*secure MAC, e supponiamo che le sfide random abbiano k bits di lunghezza. Allora il protocollo precedente è un* $(Q/2^k + 2\epsilon, Q/2) -$ *secure schema di mutua identificazione.*

Capitolo 3

schemi a chiave pubblica

3.1 introduzione

Vediamo ora gli schemi a chiave pubblica, nei quali anche se Alice e Bob non dispongono di una chiave segreta scambiata precedentemente, assumiamo che siano entrambi membri di un network nel quale ogni partecipante ha una chiave pubblica e una privata da usare in nei crittosistemi e/o negli schemi di firma. Data questa situazione è necessario fornire una procedura che permetta di autenticare la chiave pubblica degli altri individui nel network. Per semplicità assumiamo che ci sia una trusted authority, denotata da TA, la quale firma la chiave pubblica di tutti gli individui del network pubblicamente. La firma, denotata con VER_TA è data per nota come per magia da tutti nel network. Questa semplificazione, pur non essendo realistica ci permette di concentrarci sulla struttura degli schemi.

3.2 schemi di identificazione e certificati

I protocolli di autenticazione a chiave pubblica utilizzano al posto dei messaggi MAC le firme digitali, che vengono generate da ogni individuo del network con la propria chiave segreta e sono strettamente personali, e si servono per la verifica di certificati di identità distribuiti dal TA. I certificati consistono di informazioni identificative, chiave pubblica e la firma del TA su queste informazioni e permettono a ogni utente del network che conosce il ver_TA di verificare l'autenticità delle chiavi di ogni altro utente ma non danno nessuna prova di identità, visto che contenendo solo informazioni pubbliche non attestano l'identità di chi li fornisce. Vediamo come vengono emessi:

PROCEDURA DI EMISSIONE DEI CERTIFICATI

1. Il TA stabilisce l'identità di Alice con le convenzionali forme di identificazione (certificato di nascita, passaporto...) e genera una stringa che che denotiamo con ID(Alice), che contiene le informazioni sull'identità di Alice.

2. si generano una chiave privata per la firma di Alice sig_{Alice}, e una corrispondente chiave pubblica di verifica ver_{Alice}

3. Il TA genera la sua firma sulla stringa identificativa di Alice e la chiave di verifica.

$$s = sig_{TA}(ID(Alice) \| ver_{Alice})$$

Il certificato

$$Cert(Alice) = (ID(Alice) \| ver_{Alice} \| s)$$

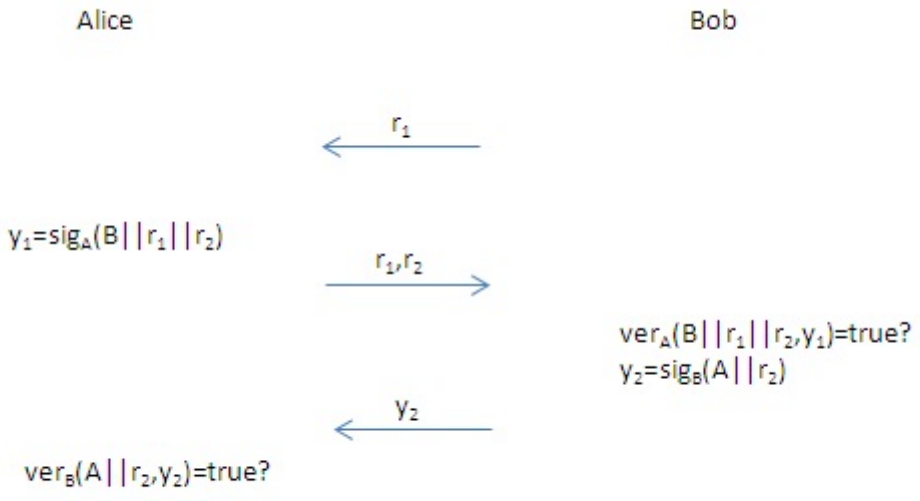
viene dato ad Alice assieme alla sua chiave segreta.

Ogni utente utilizza la chiave di verifica del destinatario per verificare le firme ricevute nella sessione successiva, e inserisce il nome di quest'ultimo in ogni firma che crea all'interno dello schema. Vediamo un classico protocollo di autenticazione reciproca:

PROTOCOLLO DI AUTENTICAZIONE A CHIAVE PUBBLICA

1. Bob sceglie una sfida random r_1 e invia $Cert(Bob)$ e r_1 ad Alice.
2. Alice sceglie una sfida random r_2 e calcola $y_1 = sig_{Alice}(ID(Bob)\|r_1\|r_2)$ e invia $cert(Alice), r_2$ e y_1 a Bob.
3. Bob verifica la chiave pubblica di Alice, ver_{Alice} sul certificato $Cert(Alice)$ e successivamente controlla che $ver_{Alice}(ID(Bob)\|r_1\|r_2, y_1) = true$, nel qual caso accetta, altrimenti rifiuta. Bob calcola anche $y_2 = sig_{Bob}(ID(Alice)\|r_2)$ e invia y_2 ad Alice.
4. Alice verifica la chiave pubblica di Bob $ver(Bob)$ sul certificato $Cert(Bob)$ e controlla che $ver_{Bob}(ID(Alice)\|r_2, y_2) = true$ nel qual caso accetta, altrimenti rifiuta.

In questa figura illustriamo lo schema omettendo la trasmissione dei certificati di Alice e Bob, supponendo che questi siano già noti:



Il seguente teorema stabilisce come la sicurezza del protocollo dipenda direttamente dalla sicurezza dello schema di firma:

Teorema 3.2.1. *Supponiamo che sig sia uno schema di firma* $(\epsilon, Q)-$*secure, e supponiamo che le sfide random abbiano lunghezza di k bits. Allora il protocollo è un* $(Q/2^k - 1 + 2\epsilon, Q) -$ *secure mutual identification scheme.*

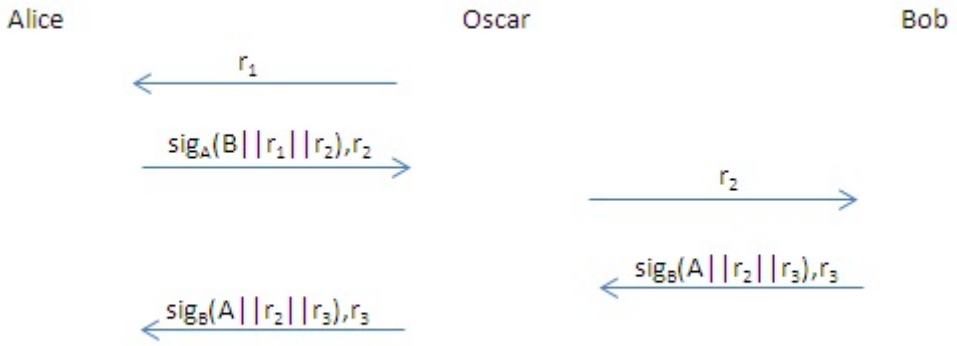
In questo teorema il numero di sessioni è Q mentre in quello del capitolo precedente era Q/2. Questo perché adesso Alice e Bob usano chiavi differenti per le firme, quindi l'avversario può vedere Q firme create da Alice e Bob.

Vediamo una variante di questo schema nel quale Bob inserisce e firma un terzo numero casuale r_3, vulnerabile ad attacchi a sessione parallela.

PROTOCOLLO DI AUTENTICAZIONE A CHIAVE PUBBLICA INSICURO

1. Bob sceglie una sfida random r_1 e la invia insieme a $Cert(Bob)$ ad Alice
2. Alice sceglie una sfida random r_2 calcola $y_1 = sig_{Alice}(ID(Bob)\|r_1\|r_2)$ e invia Cert(Alice), r_2 e y_1 a Bob.
3. Bob verifica la chiave pubblica di Alice, ver_{Alice}, sul certificato $Cert(Alice)$. Poi controlla che $ver_{Alice}(ID(Bob) \parallel r_1 \parallel r_2, y_1) = true$. se è così allora Bob accetta , altrimenti rifiuta. Bob sceglie anche una sfida random r_3, calcola $y_2 = sig_{Bob}(ID(Alice) \parallel r_2 \parallel r_3)$ e invia r_3 e y_2 ad Alice.
4. Alice verifica la chiave pubblica di Bob, ver_{Bob}, sul certificato $Cert(Bob)$, controlla che $ver_{Bob}(ID(Alice) \parallel r_2 \parallel r_3, y_2) = true$. Se è così accetta altrimenti rifiuta

Come vediamo nella figura questo schema cede ad attacchi a sessione parallela perché la firma nel terzo flow e costruita in modo simile a quella nel secondo così che Oscar può aprire una sessione con Alice fingendosi Bob e successivamente aprirne una seconda con Bob fingendosi Alice, per inviare a ad Alice nel terzo flow della prima sessione la risposta di Bob nel secondo flow della seconda sessione.


Alice
Oscar
Bob
r1
sigA(B||r1||r2),r2
r2
sigB(A||r2||r3),r3
sigB(A||r2||r3),r3

Capitolo 4

Schema di identificazione Schnorr

Un altro approccio agli schemi di identificazione è creare schemi ex novo, cioè che non usino nessun altro strumento crittografico, pensati per essere più veloci ed efficienti sia nei costi computazionali che nella quantità di informazioni da scambiare, e che per queste caratteristiche sono preferibili nelle applicazioni pratiche rispetto a quelli visti nei capitoli precedenti, dato che solitamente l'utente che vuole identificarsi dispone di una smart card a basso potere computazionale per svolgere i calcoli necessari nei processi di autenticazione.
Questi schemi prevedono entità che si identificano provando di conoscere il valore di una qualche segreta quantità senza rivelarla. Vedremo che questo sarà possibile tramite la risoluzione di una serie di congruenze, e che la sicurezza dello schema è garantita dalla proprietà di unidirezionalità della funzione logaritmo discreto.
Un utente che vuole identificarsi, in questo caso Alice, utilizzerà una chiave segreta a, la quale funge più o meno come un codice PIN in quanto serve a convincere Bob che l'entità che cerca di identificarsi è realmente Alice, anche se a differenza di un codice PIN il valore della chiave a non viene mai rivelato, semplicemente ne viene dimostrato il possesso. Da qui in avanti assumiamo che α sia un elemento di ordine q nel gruppo$\mathbb{Z}_p$(dove p è un primo e $p-1 \equiv 0(modq)$), che $log_\alpha \beta$ sia definito per qualsiasi elemento $\beta \in \langle\alpha\rangle$, e $0 \leq log_\alpha \beta \leq q-1$. Come nel capitolo precedente richiediamo l'esistenza di una trusted authority TA che distribuisce certificati ad ogni utente del network e sceglie i seguenti parametri comuni (domain parameters):

1. p è un primo grande ($p \approx 2^1024$ per rendere lo schema sicuro).

2. q è un primo grande che divide p-1 ($q \approx 2^160$ per rendere lo schema sicuro).

3. $\alpha \in \mathbb{Z}*_p$ ha ordine q.

4. t è un parametro di sicurezza tale che $q > 2^t$ (la probabilità che l'avversario ha di ingannare Alice o Bob sarà di 2^-t, quindi t=40 garantirà una sicurezza adeguata per la maggior parte delle applicazioni pratiche.

I domain parameters p,q e t sono tutti resi pubblici, e verranno usati da chiunque nel network. Ogni utente nel network sceglie la sua chiave privata personale a tale che $0 \leq a \leq q-1$, e costruisce chiave pubblica $v = \alpha^{-a}$ mod p, che può anche essere calcolato come $(\alpha^a)^{-1}$ mod p, oppure come a^{q-a} mod p. Vediamo come funziona lo schema:

SCHEMA DI IDENTIFICAZIONE DI SCHNORR

1. Alice sceglie un numero random k tale che $0 \leq k \leq q-1$, e calcola $\gamma = \alpha^k modp$. Invia $Cert(Alice)$ e γ a Bob.

2. Bob verifica la chiave pubblica v di Alice sul certificato Cert(Alice) e sceglie una sfida random r tale che $1 \leq r \leq 2^t$, che invia ad Alice.

3. Alice calcola $y = k + armodq$ e lo invia come risposta a Bob.

4. Bob verifica che $\gamma \equiv \alpha^y v^r (modp)$. in tal caso accetta, altrimenti rifiuta.

Ora, convenendo che entrambe le parti siano oneste e svolgano calcoli corretti vediamo che Alice riesce ad identificarsi a Bob con una serie di congruenze:

$$\alpha^y v^r \equiv \alpha^k + arv^r (modp) \tag{4.1}$$
$$\equiv \alpha^k + ar\alpha^- ar (modp) \tag{4.2}$$
$$=\equiv \alpha^k (modp) \tag{4.3}$$
$$\equiv \gamma (modp). \tag{4.4}$$

Il fatto che Bob accetti la prova di identità di Alice, è chiamato proprietà di completezza dello schema.

Analizziamo adesso i costi computazionali e verifichiamo che il carico di Alice sia esiguo, condizione necessaria come abbiamo detto nelle applicazioni

pratiche: il primo step richiede il calcolo di una potenza (mod p), abbastanza costoso dal punto di vista computazionale ma che può essere svolto offline prima dell'esecuzione dello schema, e nel terzo step si effettuano un addizione e una moltiplicazione (mod q) che hanno un costo molto modesto. Attraverso il seguente schema analizziamo ora il flusso di informazione e cerchiamo di calcolare i bits scambiati:
Alice dà a Bob 1024 bits di informazione (trascuriamo in questa analisi i bits usati per trasmettere i certificati) nel primo flow, nel secondo flow Bob gliene rinvia 40 e infine Alice trasmette 160 bits nel terzo flow, possiamo quindi dire che i requisiti per la comunicazione sono esigui.

4.1 Sicurezza dello schema di identificazione di Schnorr

Elenchiamo i requisiti fondamentali per la sicurezza dello schema:

1. Il parametro t deve essere abbastanza grande da impedire a una terza persona, che qui identificheremo con Olga, di indovinare la sfida random r, altrimenti potrebbe scegliere un qualsiasi valore y e calcolare $\gamma = \alpha^y v^r mod p$, inviarlo a Bob nel primo flow e rinviare y subito dopo aver ricevuto la sfida r da Bob, il quale accetterà dopo aver verificato la congruenza con γ. La probabilità che Olga indovini il valore di r correttamente è 2^{-t} per i valori r scelti ad arbitrio da Bob.

2. Bob deve sempre scegliere una sfida random diversa ogni volta che Alice vuole identificarsi, o Olga potrà impersonare Alice nel modo appena visto.

3. Il problema del logaritmo discreto deve essere irrisolvibile o Olga potrebbe calcolare $a = -log_\alpha v$ in $\mathbb{Z}_p$.

Abbiamo detto che Olga può indovinare la sfida r di Bob con probabilità 2^{-t}, vediamo che chiunque sia in grado di impersonare Alice con probabilità maggiore deve conoscere la chiave a, in altre parole che questi sono fatti equivalenti. Se Olga ha probabilità maggiori di 2^{-t} di impersonare Alice allora è plausibile che conosca un valore γ e due possibili sfide r_1 e r_2 che gli permettano di calcolare le risposte y_1 e y_2 rispettivamente che facciano accettare Bob. Quindi assumiamo che Olga conosca (o sappia calcolare) i valori r_1, r_2, y_1 e y_2 tali che

$$\gamma \equiv \alpha^{y_1} v^{r_1} \equiv \alpha^{y_2} v^{r_2} (mod p)$$

segue che

$$\alpha^{y_1-y_2} \equiv v^{r_2-r_1}(modp)$$

dato che $v \equiv \alpha^{-a}(modp)$, dove a è la chiave privata di Alice. Quindi,

$$\alpha^{y_1-y_2} \equiv \alpha^{-a(r_2-r_1)}(modp).$$

L'elemento α ha ordine q, quindi deve essere

$$y_1 - y_2 \equiv a(r_1 - r - 2)(modq).$$

Ora, $0 <| r_2 - r_1 |< 2^t$ è un primo. Quindi $MCD(r_2 - r_1, q) = 1$, e per questo $(r_1 - r_2)^{-1} modq$ esiste, e Olga può calcolare la chiave a così:

$$a = (y_1 - y_2)(r_1 - r_2)^{-1} modq.$$

Gli schemi di identificazione in cui l'essere in grado di impersonare un utente equivale a conoscere la chiave sono detti avere la proprietà di soundness. Gli schemi che sono sia completi che sound sono chiamati prove di conoscenza.
Per concludere ci resta adesso da vedere quale sia la possibilità che un terzo utente riesca a ottenere informazioni riservate. Consideriamo una situazione in cui Alice prova la sua identità a Olga un numero considerevole di volte, dopo le quali Olga cercherà di determinare il valore di a per poter impersonare Alice. Gli schemi in cui Olga non riesce ad ottenere nessuna informazione su a nemmeno dopo un numero ragionevole di sessioni in cui Alice si identifica sono chiamate zero-knowledge identification scheme, vogliamo provare che lo schema di Schnorr rientra fra questi se specifichiamo che Olga scelga sempre sfide diverse in maniera random.
Definiamo come transcript di una sessione le triplette $T = (\gamma, r, y)$ tali che $\gamma \equiv \alpha^y v^r$. Un osservatore può ottenere un transcript T(S) per ogni sessione S. Un set di possibili transcript è

$$\tau = (\gamma, r, y) : 1 \leq r \leq 2^t, 0 \leq y \leq q - 1, \gamma \equiv \alpha^y v^r (modp).$$

Si vede subito che $| \tau |= q2^t$, e che la probabilità che un particolare transcript capiti in una data sessione è $1/(q2^t)$, assumendo che le sfide r siano scelte in maniera random. Il punto chiave degli schemi a conoscenza zero è una proprietà chiamata simulability: Olga può generare transcript simulati, senza partecipare allo schema, che abbiano la stessa distribuzione di probabilità dei veri transcript, e ciò si vede da questi semplici tre steps:

1. sceglie $r \in_R \{1, ..., 2^t\}$
2. sceglie $y \in_r \{0, ..., q - 1\}$

3. calcola $\gamma = \alpha^y v^r mod p$.

La probabilità che un $T \in \tau$ sia generato con questa procedura è $1/(q2^t)$ cioè la stessa dei transcript generati realmente. In altre parole stiamo dicendo che se anche Olga cercasse di calcolare la chiave a inviando sfide scelte in maniera random in alcune sessioni dello schema non le risulterebbe più semplice rispetto a calcolarlo solo con sessioni simulate. Cosa succederebbe se Olga riuscisse ad ottenere informazioni utili scegliendo le sfide r non in maniera uniforme? Supponiamo che Olga scelga le sfide r in base a una qualche funzione dipendente dalle scelte di γ fatte da Alice, in questo caso non ci sono prove di sicurezza conosciute da poter applicare.
Per riassumere, uno schema interattivo è una prova di conoscenza se è impossibile (se non con bassissima probabilità) impersonare Alice senza conoscere la chiave a di Alice, e l'unico modo per rompere lo schema è saper calcolare a. Uno schema è chiamato zero-knoledge se non rivela alcuna informazione riguardo la chiave di Alice, cioè il calcolo di a non è reso più semplice prendendo parte allo schema impersonando il ruolo di Bob anche per un numero elevato di sessioni. Se lo schema è a conoscenza zero allora è sicuro.

Bibliografia

Stin [1] Douglas R. Stinson *Cryptography theory and practice 3ed*(2006). Springer-verlag, Milano

Stall [2] Stallings W. *Crittografia e sicurezza delle reti 2ed*2007. McGraw-hill

Wade [3] Wade Trappe, Washington Lawrence C. *Crittografia con elementi di teoria dei codici 2ed*2009.