Algoritmi di Crittografia (2023/24)
Notebook 3

In [ ]:
```
from IPython.display import HTML
HTML('<style>{}</style>'.format(open('/home/mauro/.jupyter/custom/custom.css').read()))
```

Latex definitions

Il problema dello scambio di chiavi

- I protocolli simmetrici fanno uso di chiavi condivise fra le parti che devono comunicare.
- Le moderne applicazioni crittografiche amplificano enormemente il problema non solo della sicurezza ma anche quello legato alla gestione delle chiavi.

Breve richiamo "storico"

- In un articolo pubblicato nel 1976, dal titolo New Directions in Cryptography, due ricercatori della Stanford University, Whitfield Diffie e Martin Hellman, hanno esaminato in primis proprio il problema di efficienza e hanno proposto una (parziale) soluzione.
- L'intuizione di Diffie e Hellman è il seguente: se nella cifratura e nella corrispondente decifrazione si potessero usare chiavi differenti, si aprirebbe uno scenario del tutto nuovo: la chiave di cifratura potrebbe essere resa pubblica, e dunque utilizzabile da chiunque, mentre quella di decifrazione dovrebbe essere tenuta riservata.
- Dal punto di vista dell'efficienza, questo schema appare molto più promettente: ogni soggetto coinvolto in una comunicazione, se vuole poter ricevere messaggi confidenziali, deve gestire solo una coppia di chiavi, una delle quali da rendere pubblica.
- Chi invece ha solo necessità di inviare messaggi riservati non deve gestire alcuna chiave; deve solo procurarsi le chiavi pubbliche opportune al momento giusto.
- Lo schema introduce però un problema di gestione delle chiavi pubbliche, che non ha soluzione solamente algoritmica.
- Diffie e Hellman non riuscirono a trovare la "matematica" necessaria per implementare uno schema a chiave pubblica.
- Il protocollo introdotto in concreto da Diffie e Hellman (DH) è invece essenzialmente un metodo per la comunicazione di informazione segreta su un canale di comunicazione insicuro.
- Il protocollo DH è tutt'ora utilizzato all'interno di molti altri protocolli di sicurezza su Internet fra i quali, solo per fare due esempi, TLS e SSH.
- Per il loro contributi nel settore della crittografia, Diffie e Hellman hanno ricevuto il Premio Turing nel 2015.

Il protocollo DH su gruppi ciclici $\mathbf{Z}_p^*$

- Lo scenario prevede che, per la comunicazione confidenziale, Alice e Bob utilizzino un protocollo simmetrico.
- Per questo scopo, devono potersi accordare su una chiave condivisa da tenere segreta; questo però comunicando su un canale insicuro.
- Il protocollo di Diffie e Hellman consente alle due parti di concordare un segreto che altro non è che un elemento di un opportuno gruppo $\mathbf{Z}_p^*$.
- Utilizzando tale segreto, le due parti deriveranno la chiave da usare nel protocollo simmetrico.

L'algoritmo in dettaglio

1. Alice e Bob si accordano sul modulo p da utilizzare (un numero primo) e sull'uso di una radice primitiva g di $\mathbf{Z}_p^*$.
   - Tutto questo viene fatto pubblicamente e, dunque, l'informazione può venire in possesso anche di un malintenzionato (Eva).
   - In concreto, p e g sono tipicamente scelti da una delle due parti (ad esempio chi inizia il protocollo) e confermati dall'altra parte.
2. Alice sceglie, a caso, con distribuzione di probabilità uniforme, un numero $a \in \mathbf{Z}_p^*$; con questo calcola $x_a = g^a \bmod p$ e invia x_a a Bob.
3. Similmente, Bob sceglie, a caso, con distribuzione di probabilità uniforme, un numero $b \in \mathbf{Z}_p^*$; con questo calcola $x_b = g^b \bmod p$ e invia x_b ad Alice.
4. Con l'informazione x_a ricevuta da Alice, Bob calcola il valore $z_{Bob} = x_a^b \bmod p$.
5. Analogamente, con l'informazione x_b ricevuta da Bob, Alice calcola $z_{Alice} = x_b^a \bmod p$
6. La quantità $z = z_{Alice} = z_{Bob}$ è il segreto condiviso.
7. Nota: il valore a è anche detto chiave privata di Alice mentre x_a è la sua chiave pubblica. Lo stesso vale naturalmente per Bob.

Correttezza

Al termine dell'esecuzione del protocollo, vale l'uguaglianza: $z_{Alice} = z_{Bob}$, come si può facilmente verificare.

$$\begin{aligned} z_{Alice} &= x_b^a \bmod p \\ &= (g^b \bmod p)^a \bmod p \\ &= (g^b)^a \bmod p \\ &= (g^a)^b \bmod p \\ &= (g^a \bmod p)^b \bmod p \\ &= x_a^b \bmod p \\ &= z_{Bob} \end{aligned}$$

Efficienza

- Tutte le quantità necessarie ad Alice per completare il protocollo sono calcolabili efficientemente. Lo stesso naturalmente vale per Bob.
- Le computazioni più onerose sono infatti due esponenziali modulari, che sappiamo avere complessità polinomiale nella lunghezza dei numeri.
- Il problema della scelta di p con l'individuazione della radice primitiva g viene risolto ricorrendo a safe-prime con generatore "fisso"

Sicurezza

- La sicurezza riguarda ovviamente la difficoltà (dal punto di vista computazionale) per Eva di calcolare z sulla base delle informazioni che ella può aver carpito sul canale insicuro, e cioè i parametri di gruppo p e g e le due chiavi pubbliche x_a e x_b.
- Eva può naturalmente effettuare tutte le operazioni che vuole in $\mathbf{Z}_p^*$, "combinando" le quantità note.
- z tuttavia non "sembra" essere il risultato di operazioni semplici che coinvolgono tali parametri.
- Conoscendo x_a e x_b, l'unico modo noto per risalire a z è di determinare a (o b), ma questo richiede il calcolo di un logaritmo discreto.
- Infatti, $x_a = g^a \bmod p$ implica $a = \log_g x_a \bmod p$.
- Ovviamente, la situazione è la stessa se, per risalire a z, Eva decidesse di usare b, in quanto $b = \log_g x_b \bmod p$.

Computational Diffie-Hellman assumption

- Il problema del calcolo di $z = g^{ab} \bmod p$, dati $x_a = g^a \bmod p$ e $x_b = g^b \bmod p$, è noto come Problema di Diffie e Hellman.
- Questo, come visto, sarebbe risolvibile "facilmente" se fosse facile il calcolo dei logaritmi discreti.
- Il viceversa non è provato ma si ritiene valere.
- Computational Diffie-Hellman (CDH) Assumption: Se g, a e b sono scelti a caso in $\mathbf{Z}_p^*$, allora, a partire da g, $g^a \bmod p$ e $g^b \bmod p$, la determinazione di $g^{ab} \bmod p$ è un problema computazionalmente intrattabile.

</P>

Il protocollo DH con OPENSSL

- Come utile esempio pratico possiamo simulare passo-passo l'esecuzione del protocollo usando la suite OPENSSL.
- Il primo passo è la generazione di un certificato DH (secondo lo standard di codifica X.509), con i parametri comuni.
- La generazione viene effettuata da uno dei due partner della comunicazione.
- I parametri sono memorizzati nel file di testo nel formato PEM (Privacy Enhanced Mail).

1. Creazione dei parametri condivisi (e loro visualizzazione)
   - openssl genpkey -genparam -algorithm DH -out dhparams.pem
   - openssl pkeyparam -in dhparams.pem -text
2. Creazione delle chiavi pubblica e privata di Alice (e loro visualizzazione)
   - openssl genpkey -paramfile dhparams.pem -out dhAlicekey.pem
   - openssl pkey -in dhAlicekey.pem -text -noout
3. Creazione delle chiavi pubblica e privata di Bob (e loro visualizzazione)
   - openssl genpkey -paramfile dhparams.pem -out dhBobkey.pem
   - openssl pkey -in dhBobkey.pem -text -noout
4. Estrazione della chiave pubblica da parte di Alice, per il successivo scambio
   - openssl pkey -in dhAlicekey.pem -pubout -out dhAlicepub.pem
   - openssl pkey -pubin -in dhAlicepub.pem -text
5. Estrazione della chiave pubblica da parte di Bob, per il successivo scambio
   - openssl pkey -in dhBobkey.pem -pubout -out dhBobpub.pem
   - openssl pkey -pubin -in dhBobpub.pem -text
6. Scambio dei file dhAlicepub.pem e dhBobpub.pem
7. Derivazione del segreto condiviso da parte di Alice
   - openssl pkeyutl -derive -inkey dhAlicekey.pem -peerkey dhBobpub.pem -out sAlice.bin
8. Derivazione del segreto condiviso da parte di Bob
   - openssl pkeyutl -derive -inkey dhBobkey.pem -peerkey dhAlicepub.pem -out sBob.bin

In [ ]:
```
!openssl genpkey -genparam -algorithm DH -out dhparams.pem
```

In [ ]:
```
!openssl pkeyparam -in dhparams.pem -text
```

In [ ]:
```
!openssl genpkey -paramfile dhparams.pem -out dhAlicekey.pem
```

In [ ]:
```
!openssl pkey -in dhAlicekey.pem -text -noout
```

In [ ]:
```
!openssl genpkey -paramfile dhparams.pem -out dhBobkey.pem
```

In [ ]:
```
!openssl pkey -in dhBobkey.pem -text -noout
```

In [ ]:
```
!openssl pkey -in dhAlicekey.pem -pubout -out dhAlicepub.pem
```

In [ ]:
```
!openssl pkey -pubin -in dhAlicepub.pem -text -noout
```

In [ ]:
```
!openssl pkey -in dhBobkey.pem -pubout -out dhBobpub.pem
```

In [ ]:
```
!openssl pkey -pubin -in dhBobpub.pem -text -noout
```

- Scambio di file ...

In [ ]:
```
!openssl pkeyutl -derive -inkey dhAlicekey.pem -peerkey dhBobpub.pem -out sAlice.bin
```

In [ ]:
```
!openssl pkeyutl -derive -inkey dhBobkey.pem -peerkey dhAlicepub.pem -out sBob.bin
```

In [ ]:
```
!cmp -b sAlice.bin sBob.bin  # confronta byte per byte, con l'opzione -b stampa i byte diversi
```

In [ ]:
```
!xxd sAlice.bin   # Hex dump
```

Uso dell'informazione segreta

- Il dato condiviso da Alice e Bob non è in generale utilizzabile come chiave segreta in un protocollo simmetrico.
- Le chiavi segrete ideali che vogliamo utilizzare sono infatti stringhe i cui i singoli bit hanno, in maniera indipendente, la stessa probabilità di essere 0 oppure 1.
- Il segreto condiviso invece è un numero uniformemente distribuito in $\mathbf{Z}_p^*$ e questa non è la stessa nozione.
- Un semplice esempio può aiutare a capire la questione.

- Consideriamo il gruppo $\mathbf{Z}_{11}^*$, i corrispondenti valori in decimale e come sequenze di bit.

1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
pr(b=1)	**0.2**	**0.4**	**0.5**	**0.5**

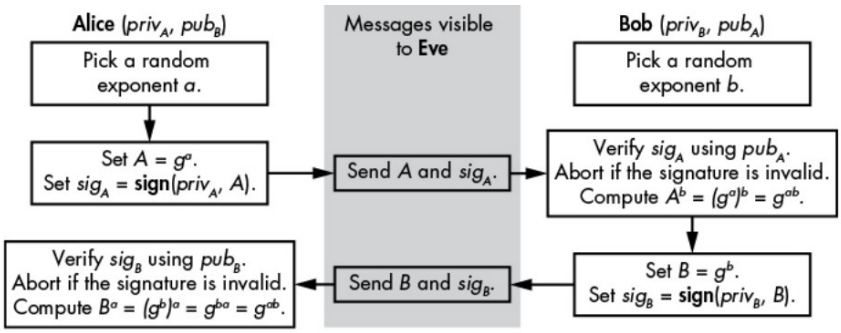
- Come si vede dalla tabella, alcuni bit non hanno probabilità $\frac{1}{2}$ di essere 1. Questo fatto costituisce una significativa deviazione dalla situazione ideale perché fornisce informazione ad un eventuale attaccante, riducendo la sicurezza dello schema crittografico.
- Esistono molti lavori scientifici che forniscono tecniche per estrarre entropia dal segreto e utilizzarla per la generazione delle chiavi.
- In pratica, al segreto "numerico" viene applicata una funzione hash crittografica (naturalmente identica per Alice e Bob) il cui valore è utilizzato come chiave.
- Non ci addentriamo ulteriormente in questa questione che è, tecnicamente, alquanto complessa.

Attacchi a DH

- Sotto l'ipotesi della CDH assumption, l'attacco diretto volto alla determinazione di $g^{ab} \bmod p$ è computazionalmente infattibile.
- Un attacco vincente per il protocollo base qui descritto è invece il cosiddetto man-in-the-middle.
- Se Eva riesce a farsi passare per Bob (e Bob per Eva), cosa perfettamente fattibile in assenza di un protocollo sicuro di autenticazione, allora può partecipare a due protocolli di scambio di chiavi (uno con Alice e uno con Bob) e quindi decodificare i messaggi prima di re-inviarli, nuovamente cifrati, al legittimo destinatario.
- A seconda dei propri obiettivi, Eva può decidere se alterare i messaggi originali (per un attacco a breve termine) o lasciarli inalterato (attacco a medio/lungo termine).

Authenticated Diffie-Hellman

- Un modo per "parare" l'attacco man-in-the-middle consiste nell'autenticazione delle parti
- Potremo capire meglio il protocollo quando parleremo di firma digitale
- In sintesi, ciascuna delle due parti deve possedere anticipatamente una chiave pubblica dell'altra parte con cui verificarne la firma
- Lo specifico protocollo può variare a seconda dei casi, ma l'idea di fondo è ben spiegata dal seguente schema (in cui, per semplicità, sono omessi i moduli), tratto dal testo Jean-Philippe Aumasson, Serious Cryptography, No Starch Press, illustra la situazione

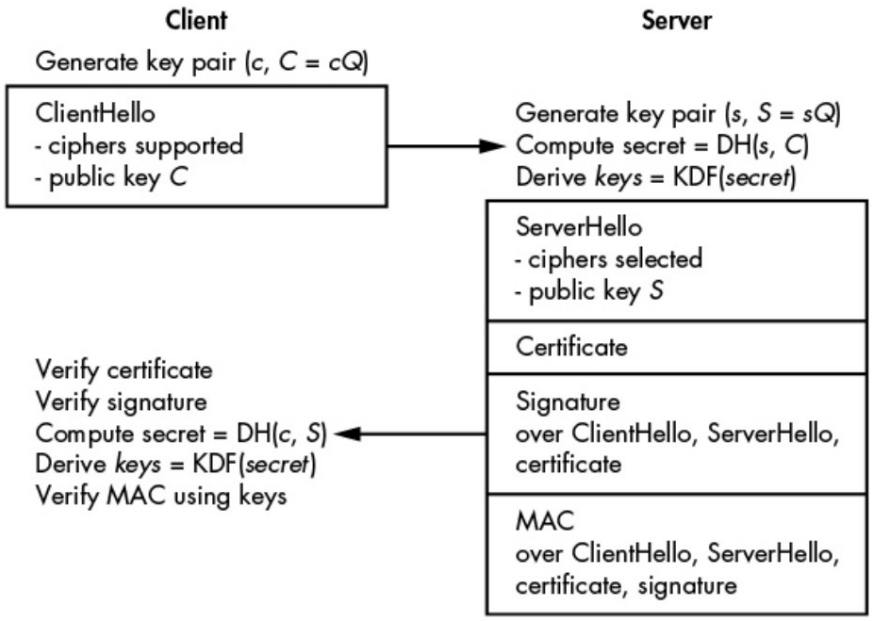


The authenticated Diffie–Hellman protocol

- In generale, dunque, in un protocollo per lo scambio sicuro di chiavi, c'è la necessità che una almeno una delle due parti disponga di un long-term secret, come ad esempio una chiave RSA.
- Se questa è la situazione, ci si potrebbe però lecitamente chiedere: se (poniamo) Alice possiede una chiave RSA, perché mai bisognerebbe usare DH per lo scambio di un segreto?
- Il segreto non potrebbe essere direttamente cifrato da Bob con la chiave pubblica di Alice e inviato a quest'ultima, che potrebbe (solo lei) decifrarlo con la propria chiave privata?
- In effetti esiste anche questa possibilità, soprattutto in suite di protocolli più vecchie (es. in TLS 1.2), che però tende ad essere abbandonata perché non garantisce forward-secrecy.

Chiavi DH effimere

- Per fissare le idee, consideriamo proprio il caso del TLS (Transport Level Security).
- In questo contesto, infatti, il "segreto a lungo termine" è proprio la chiave privata con cui il server si autentica presso il client.
- Forward-secrecy implica che, se anche tale segreto venisse compromesso, la sicurezza di precedenti scambi fra client e server non sarebbe a rischio.
- Ora, in uno scenario in cui la chiave simmetrica è cifrata con la chiave pubblica del server, la forward-secrecy non è in generale garantita (perché l'attaccante potrebbe aver intercettato e memorizzato le precedenti interazioni).
- In TLS 1.3 questo viene impedito e le chiavi simmetriche sono invece generate a partire da chiavi DH effimere (Ephemeral Diffie-Hellman keys).
- Questo vuol dire che le chiavi DH pubbliche e private, dopo essere state usate una sola volta, vengono scartate da client e server.
- Uno schema, tratto sempre dal testo Serious Cryptography sopra citato, illustra la situazione (lo schema fa implicito riferimento a DH su curve ellittiche).



The TLS 1.3 handshake process when connecting to HTTPS web-sites

- Per il momento, in relazione allo scambio di chiavi, ci fermiamo, in attesa di riconsiderare il protocollo di DH rispetto a gruppi formati da *punti appartenenti a una curva ellittica*