

# INTRODUZIONE ALLA CRITTOGRAFIA

## Corso di Sicurezza e Gestione delle reti

LEONARDO MACCARI: LEONARDO.MACCARI@UNIFI.IT  
LART - LABORATORIO DI RETI E TELECOMUNICAZIONI  
DIPARTIMENTO DI ELETTRONICA E TELECOMUNICAZIONI



1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

## 1 Recall..

## 2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

## 3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

## 4 Esempio

## Disponibilità

- Il servizio deve essere sempre raggiungibile
- La disponibilità viene violata se siete vittima di un attacco **DoS**: Denial of Service
- La disponibilità del servizio è la cosa più difficile da garantire:
  - Esistono sempre limiti fisici delle risorse
  - Realizzare un attacco DoS deve costare il più possibile
- La disponibilità si ottiene con una accurata progettazione della rete

# Segretezza dei dati scambiati

## Segretezza

- I dati scambiati devono rimanere riservati tra le parti che partecipano allo scambio
- Le reti ethernet permettono, generalmente, di fare *sniffing* dei pacchetti
- Per ottenere segretezza si devono utilizzare algoritmi di crittografia (simmetrici, asimmetrici, distribuiti. . . )

## Integrità

- I dati devono raggiungere la destinazione senza essere stati modificati
- Si possono modificare dati cifrati senza decifrarli (attacchi di *bit flipping*)
- Per ottenere l'integrità dei dati si devono utilizzare funzioni di *hashing*

## Autenticazione

- Chi riceve un'informazione deve essere sicuro che il mittente è effettivamente quello dichiarato
- I protocolli di internet spesso permettono di effettuare lo *spoofing* degli indirizzi mittente, ad esempio per le email

## Non ripudiabilità

- Chi invia un messaggio non può in seguito negare di averlo mandato
- Importante soprattutto a livello di applicazione, nello scambio di documenti

La possibilità di immettere informazioni in una rete senza che queste siano direttamente collegabili all'identità del mittente.

- Esistono reti anonimizzanti:
  - Tor
  - freenet
  - Remailer anonimi
- Perchè si vuole anonimato:
  - Forse perchè si vogliono commettere atti illeciti senza essere rintracciati...
  - ... o forse perchè non si è in condizione di esercitare i propri diritti civili.
- Le reti anonimizzanti producono entrambe le conseguenze, ma non bisogna considerarle con pregiudizio. In ogni caso, perche' AoL mantiene quei database?

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

# La crittografia

- Il termine crittografia viene dalle parole greche *kryptós* che significa nascosto, e *gráphein* che significa scrivere. E' quindi la scienza che si occupa di rendere segrete le informazioni.
- La crittografia ha una storia secolare, dal cifrario di Cesare in poi sono stati fatti molti passi avanti.
- Oggi nella crittografia confluiscono studi mirati ad ottenere segretezza ma anche tutti gli altri servizi di sicurezza che abbiamo visto.
- fa eccezione la disponibilità, che ha poco a che vedere con la crittografia, anzi, generalmente un uso troppo diffuso di tecniche di cifratura aumentano la possibilità di essere vittime di DoS.

# La crittografia garantisce:

- Protezione dei documenti
  - integrità
  - segretezza
  - autenticazione
  - non ripudiabilità
- Verifica dell'identità dei corrispondenti
  - controllo degli accessi

# Principi di base

- per spiegare i principi di base della crittografia useremo degli esempi svincolati da qualsiasi tecnologia, dove vengono coinvolti:
  - delle entità che comunicano (potrebbero essere persone, indirizzi IP, porte TCP ...) che chiameremo A e B (Alice e Bob)
  - uno scambio di un messaggio M. In questo momento non ci interessa specificare alcun protocollo, le considerazioni che faremo si applicano alle connessioni TCP così come alla posta tradizionale. Vedremo in seguito come queste tecniche si applicano alle comunicazioni.
  - Una terza entità E (Eve) è l'attaccante e proverà ad interferire con i servizi di sicurezza di questo scambio. Si immagina che E possa intercettare i messaggi mentre viaggiano, leggerli e sostituirli così come avviene per un router nella rete Internet.

# Differenti tipi di funzioni crittografiche:

- Funzioni hash
- funzioni di cifratura a chiave simmetrica
- funzioni di cifratura a chiave pubblica/privata
- da queste si derivano funzioni di firma digitale e certificazione degli utenti.

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash

- HMAC

- Cifratura a chiave simmetrica
  - Cifratura a chiave pubblica/privata
  - Certificati
  - Riassumendo

3 Accenni di teoria

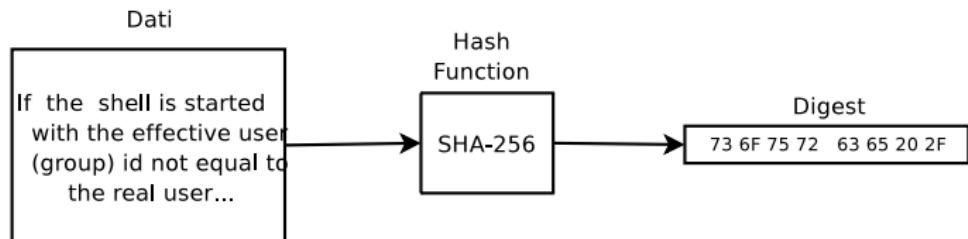
- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

# Funzioni hash - introduzione

- le funzioni hash risolvono il problema della garanzia di integrità di un documento trasmesso.
- una funzione hash è una funzione unidirezionale che si applica ad un'informazione (qualsiasi informazione in binario, un email, un pacchetto, un file ecc...) e genera un' impronta di dimensione fissa (un *digest* che può essere di 128, 160, 256 ... bit) che è funzione dei dati dati
- generalmente le funzioni hash sono sequenze di operazioni elementari quali shift e XOR sui dati, sono quindi molto veloci da computare.

# SHA



## Funzioni hash: esempio di utilizzo 1/2

- A invia a B il messaggio M, calcola anche  $H(M) = D$  ed invia anche D.
- B riceve M e D, ricalcola  $H(M) = D'$ . Se  $D = D'$  allora il messaggio non è stato modificato durante il percorso.
- E cosa può fare? se intercetta solo M può provare a cambiarlo, ma quando B riceverà anche D, l'hash non sarà più corrispondente, quindi si accorgerà della modifica avvenuta.
- Per riuscire a modificare il messaggio da M a M' deve intercettare anche D, e cambiarlo in  $H(M')$ .

## Funzioni hash: esempio di utilizzo 2/2

- Un'applicazione tipica è quella della distribuzione di immagini di file eseguibili:
- quando scaricate un eseguibile, il file che scaricate deve essere identico a quello che il produttore ha generato, anche solo un bit di differenza può provocarne il mancato funzionamento.
- Con le ISO dei sistemi operativi spesso vi viene dato anche il codice md5 del file, ovvero il digest creato con la funzione hash md5.

# Requisiti<sup>1</sup>

A hash function (in the unrestricted sense) is a function  $H$  which has, as a minimum, the following two properties:

- compression:  $H$  maps an input  $x$  of arbitrary finite bitlength, to an output  $H(x)$  of fixed bitlength  $n$ .
- ease of computation: given  $H$  and an input  $x$ ,  $H(x)$  is easy to compute.

Oltre a queste proprietà elementari si aggiungono:

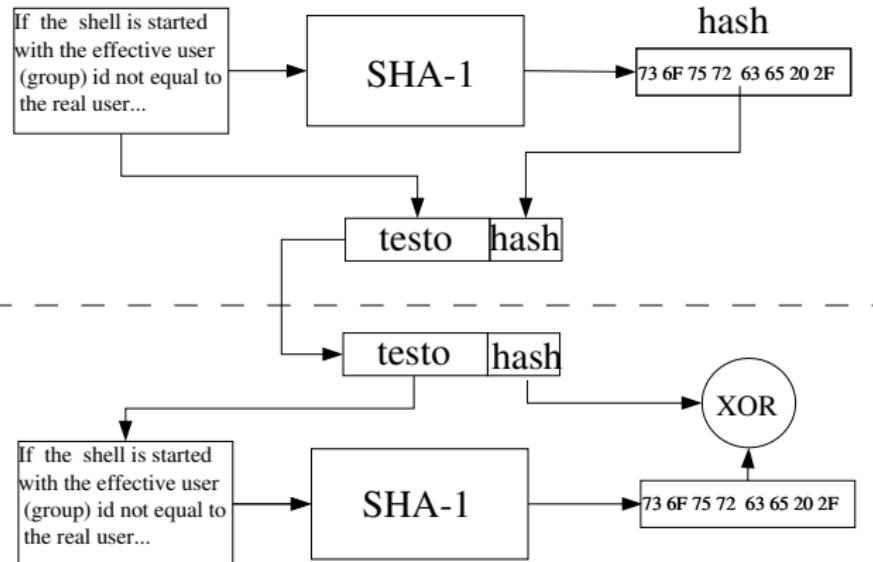
- preimage resistance: for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage  $x$  such that  $H(x) = y$  when given any  $y$  for which a corresponding input is not known.
- 2nd-preimage resistance: it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given  $x$ , to find a 2nd-preimage  $x' \neq x$  such that  $H(x) = H(x')$ .
- collision resistance: it is computationally infeasible to find any two distinct inputs  $x, x'$  which hash to the same output, i.e., such that  $H(x') = H(x)$ .  
*(Note that here there is free choice of both inputs.)*

---

<sup>1</sup>Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone *Handbook of Applied Cryptography*

# Hashed message

Entità A



Entità B

- Se mandate una email con il digest in attach?
- E' molto probabile che E se riesce ad intercettare l'email riesca ad intercettare anche il digest e modificarlo. Non è quindi un buon metodo quello di usare solo una funzione hash per l'integrità delle email.
- In generale, le funzioni hash si accompagnano spesso a metodi di autenticazione.

- un HMAC (keyed-hash message authentication code) accoppia l'utilizzo di una chiave simmetrica ad una funzione hash, per garantire oltre che l'integrità anche l'autenticazione dei dati.
- una chiave simmetrica non è altro che una stringa di lunghezza opportuna scelta in modo meno predicibile possibile.
- spesso, per generare una chiave simmetrica si usano delle funzioni hash a partire da una password alfanumerica:
  - es:  $K = \text{md5}(\text{password}) = 0x12ab5893092ba4183f3a345872b34f233$
- se si dispone di una buona funzione hash, si può generare un HMAC componendo la funzione hash con la chiave:

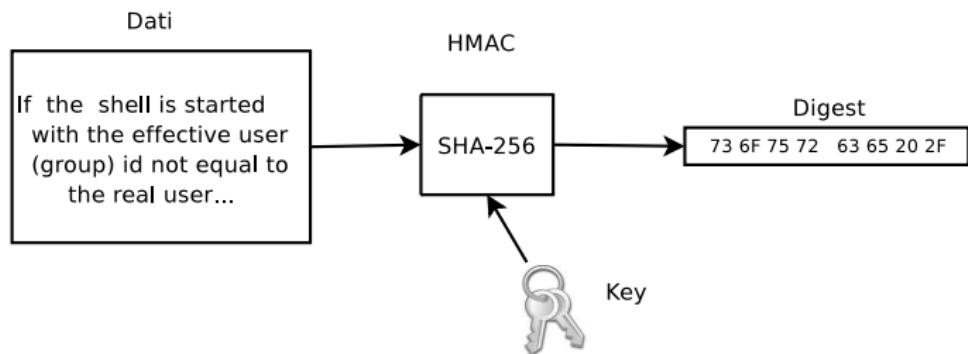
$$\text{HMAC}_K(M) = H((K \oplus opad) \vee H((K \oplus ipad) \vee M)) \quad (1)$$

- vedete che il digest a questo punto può essere calcolato solo se si conosce anche la chiave K.

# Come si usa un HMAC

- A e B si mettono d'accordo su una password. Per mettersi d'accordo devono usare un canale sicuro, si vedono di persona o si telefonano.
- A ha un messaggio M da inviare a B, compie le seguenti operazioni:
  - calcola  $K = \text{md5}(\text{password})$
  - calcola  $\text{HMAC}_K(M)$
  - invia a B la coppia  $M, \text{HMAC}_K(M)$
- Le informazioni  $M, \text{HMAC}_K(M)$  possono essere inviate accoppiate nello stesso pacchetto!

# Funzioni HMAC



# Vantaggi rispetto ad una funzione hash

- Se E intercetta entrambe le informazioni  $M$ ,  $HMAC_K(M)$  per cambiare il messaggio  $M$  dovrebbe:
  - modificare  $M$  in  $M'$ ,
  - ricalcolare  $HMAC_K(M)$
- ma E non è in possesso di  $M$ , quindi non può calcolare l'HMAC.
- Schemi di questo tipo vengono utilizzati per garantire l'integrità e implicitamente anche l'autenticazione di pacchetti di livello MAC in molte reti (es. WIFI).

# Problemi:

- il primo problema è di gestione: se A e B si devono scambiare una chiave in modo sicuro, un metodo del genere non è utile per comunicazioni via Internet.
- attacchi di forza bruta, E potrebbe intercettare un pacchetto e cercare di indovinare la chiave K:
  - dato M, e  $K = 0x00000000000000000000000000000000$
  - $D = HMAC_K(M)$  ? se è vero allora K è la chiave giusta, altrimenti
  - $K = 0x00000000000000000000000000000001$ , riprova...
- Un attacco di questo tipo è computazionalmente impegnativo: per calcolare tutte le chiavi possibili  $2^{128}$  sono necessari migliaia di anni con i computer di oggi.
- ma se la chiave K è generata da una password, allora l'attacco diventa possibile, si prende un dizionario e si comincia:
  - dato M e  $K = \text{md5(abaco)}$ ,  $D = HMAC_K(M)$
  - se è falso,  $K = \text{md5(abate)}$ ...
- Le parole di un dizionario possono essere decine di migliaia, per generarle tutte ci vogliono pochi minuti. Per questo le password non devono essere scelte come parole esistenti!

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

# Un po' di teoria. . .

- Elementi di un sistema crittografico:
  - cifrario (algoritmo)
  - chiave (informazione)
- Principio di Kerchoffs
  - il metodo si suppone noto a tutti
  - il segreto risiede nella chiave
- La conoscenza della chiave
  - consente di cifrare/decifrare documenti
  - in certi casi può costituire prova certa di identità
- Gli algoritmi di cifratura implementano la segretezza ed a volte l'autenticazione dei dati.

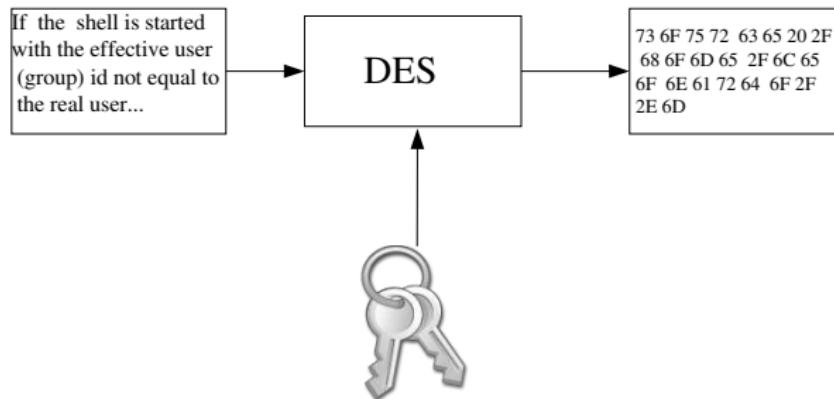
Il principio di Kerchoffs ci dice che:

- gli algoritmi crittografici devono essere noti a priori
- un prodotto che vi garantisce una cifratura con un algoritmo segreto, non è un buon prodotto.

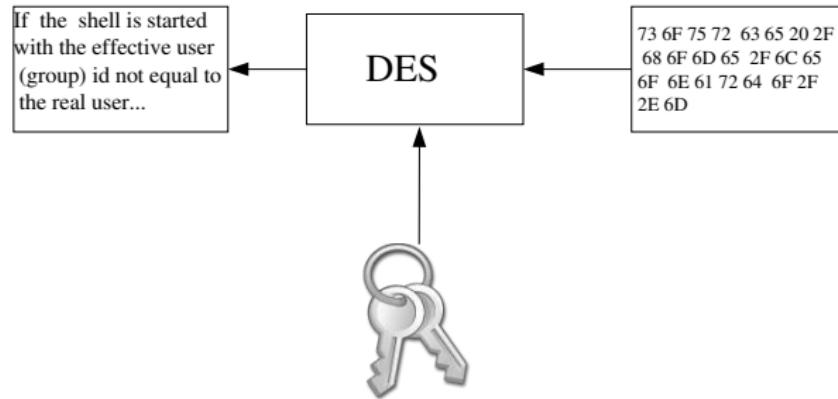
# Cifratua a chiave simmetrica

- come per un HMAC A e B si accordano su una chiave K, o una password da cui generare K
- solo con la chiave K si possono cifrare e decifrare i messaggi.
- generalmente l'algoritmo è lo stesso sia per cifrare che per decifrare

# Cifratura



# Decifrazione



- Sono gli stessi di prima, da una parte la necessità di scambiarsi le chiavi in anticipo rende questo strumento poco flessibile
- dall'altra ci espone ad attacchi di forza bruta, anche se più difficili del caso precedente. Infatti per l'attaccante non è facile sapere ad ogni tentativo se il testo decifrato è quello corretto.
- per questo motivo le due tecniche possono essere combinate:
  - prima si genera l' HMAC con una chiave K,
  - poi si cifra tutto il pacchetto, compreso l'HMAC con una seconda chiave K'. In questo modo si è ragionevolmente sicuri di avere segretezza e integrità dei dati, oltre che una forma di autenticazione che dipende da come si sono scambiate le chiavi.
- questo tipo di approccio si ritrova, come detto nelle reti LAN in cui è facile pre-impostare le chiavi segrete a mano sulle macchine.

## 1 Recall..

## 2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- **Cifratura a chiave pubblica/privata**
- Certificati
- Riassumendo

## 3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

## 4 Esempio

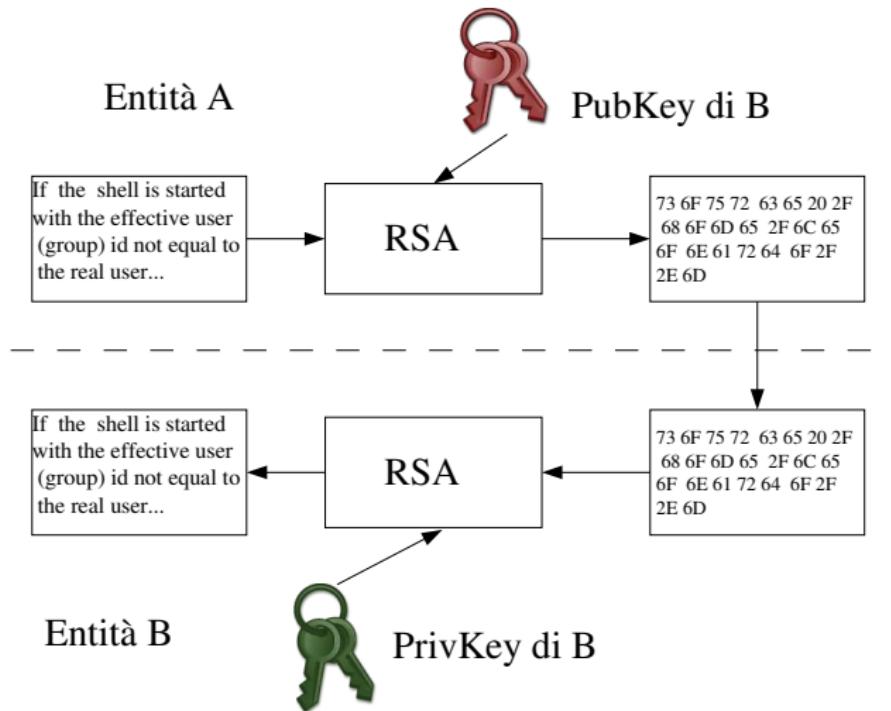
# Cifratura a chiave pubblica/privata

- A e B possiedono due chiavi ciascuno.
  - Una chiave pubblica  $Pub_A$ ,  $Pub_B$
  - Una chiave privata  $Priv_A$ ,  $Priv_B$
- La chiave privata deve essere mantenuta segreta. E' vitale che A sia l'unico possessore di  $Priv_A$  e lo stesso vale per B.
- La chiave pubblica invece è pubblica, A può pubblicare la sua chiave  $Pub_A$  su Internet.

# Cifratura a chiave pubblica/privata

- Ciò che viene cifrato con una chiave pubblica può essere decifrato solo con la corrispondente chiave privata
- E' computazionalmente impossibile risalire ad una chiave privata tramite la chiave pubblica
- Se A utilizza la chiave pubblica di B per cifrare un messaggio, allora solo B può decifrarlo, perchè è l'unico che possiede la corrispondente chiave privata. Si ottiene quindi il servizio di sicurezza della **segretezza**.

# Cifratura e Decifratura



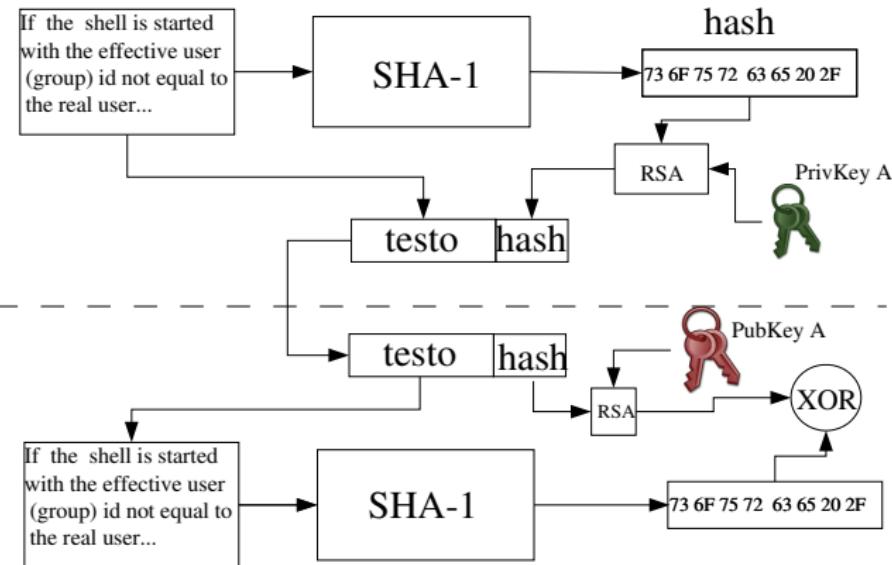
# Riassumendo

- Ogni utente ha due chiavi legate in modo inscindibile, queste chiavi possono essere generate assieme da programmi appositi.
  - una chiave viene resa pubblica, possibilmente in un elenco pubblico accessibile da chiunque.
  - l'altra è in possesso del solo utente
- Non è necessario concordare preventivamente una chiave di cifratura comune per scambiarsi un documento riservato
- La chiave privata di un utente è sempre segreta

- Le chiavi pubblica e privata sono invertibili, se A che è l'unico possessore della chiave  $Priv_A$ , usa questa chiave privata per cifrare un messaggio, allora chiunque possieda  $Pub_A$  può decifrarlo
- ma  $Pub_A$  è pubblica, quindi la possiedono tutti. Quindi il messaggio è decifrabile da chiunque. Allora a che serve?
- visto che solo A è in possesso di  $Priv_A$ , chi decifra il messaggio sicuro che il messaggio proviene direttamente da A.
- Con questo utilizzo la cifratura a chiave pubblica/privata non serve a garantire segretezza ma serve a garantire l'autenticazione del mittente.
- tutto questo si chiama **firma digitale** e fornisce l'autenticazione e la non ripudiabilità dei dati.

# Firma digitale

Entità A



Entità B

# Problemi

Il tipico attacco che si può compiere contro la cifratura a chiave pubblica/privata è il man in the middle (MITM) che è possibile quando A e B non possiedono le chiavi l'uno dell'altro. Un esempio è il seguente:

- A invia a B la sua chiave pubblica  $Pub_A$
- E intercetta il messaggio, scambia la chiave  $Pub_A$  con  $Pub_E$
- B riceve la chiave  $Pub_E$  convinto che sia la chiave di A
- B invia un messaggio M cifrato con la chiave  $Pub_E$
- E intercetta il messaggio, lo decifra, lo cifra nuovamente con la chiave  $Pub_A$  e lo invia ad A
- A riceve il messaggio e lo decifra, ma E ha potuto intercettare il contenuto ed eventualmente modificarlo.

- Un attacco MITM è sempre possibile quando A e B non conoscono in anticipo le rispettive chiavi.
- Le chiavi quindi non devono essere scambiate durante la comunicazione stessa, ma con qualche altro mezzo.
- Si ricrea il problema che abbiamo visto con la chiave simmetrica, ovvero deve esistere un mezzo sicuro attraverso cui A e B si scambiano la chiave.
- La grande differenza è che per *sicuro* non si intende segreto, ma semplicemente autenticato.

# Soluzioni: fingerprint

Per risolvere questo problema si usano fingerprint, Keyserver e web of trust.

- Una fingerprint è una piccola parte della chiave pubblica, i primi 24 byte.
- E' altamente improbabile che due chiavi pubbliche diverse abbiano i primi 24 byte in comune.
- E' più facile distribuire una fingerprint che una chiave pubblica. Ad esempio la si può usare come signature nella posta elettronica o la si può inserire in un biglietto da visita.
- Se ricevete posta elettronica da qualcuno da anni e lui usa la sua fingerprint nella signature, il giorno che avete bisogno di utilizzare la sua chiave pubblica lui ve la invierà e potrete verificare se la chiave che avete ricevuto corrisponde alla fingerprint che lui ha usato in passato.

# Soluzioni: keyserver

- In Internet potete trovare dei keyserver su cui caricare la vostra chiave pubblica.
- Il keyserver non garantisce niente, non si fa carico di stabilire l'associazione tra utente e chiave, semplicemente accetta l'upload e il download delle chiavi stesse.
- Nella chiave si possono inserire informazioni come il nome utente o il suo indirizzo di posta elettronica, quindi cercano in un motore di ricerca potete trovare la chiave di chi volete.
- NB: trovare la chiave di George W. Bush su un keyserver non significa avere la certezza che il presidente americano usa veramente quella chiave. I keyserver sono solo un modo comodo per tenere chiavi pubbliche, dovete poi essere voi ad accertarvi che la chiave corrisponda all'identità che viene dichiarata.
- link: <http://keyserver.linux.it/>

- Un Web Of Trust è una rete di contatti attraverso la quale i partecipanti certificano l'identità altrui.
- Il principio che sta alla base di un wot e che se A conosce personalmente B, A può *certificare*  $Pub_B$ , ovvero garantire agli altri che una certa chiave è effettivamente quella di B.
- Se C, pur non conoscendo B conosce A ha un livello maggiore nella chiave  $Pub_B$  se vede che la chiave è certificata da qualche altro utente.

- Ogni utente quindi ha interesse affinchè la propria chiave sia certificata dal maggior numero di persone possibile
- Le certificazioni avvengono utilizzando la propria chiave privata per firmare la chiave privata altrui.

Esempio:

- A genera una sua coppia di chiavi  $Pub_A$  e  $Priv_A$ . Nella chiave pubblica c'e' scritto che la chiave appartiene ad A.
- A va da B, gli mostra un documento e gli consegna la fingerprint della chiave.
- B scarica la chiave da un keyserver, controlla la fingerprint.
- B a quel punto può firmare con la propria  $Priv_B$  la chiave pubblica  $Pub_A$ .
- B carica la chiave firmata sul keyserver.

# In pratica; GPG

- Il primo programma che permetteva agli utenti di generare chiavi pubbliche/private e inviarsi messaggi cifrati è stato PGP ovvero Pretty Good Privacy
- il programma ha avuto molti problemi di distribuzione all'inizio della sua vita, perchè le leggi statunitensi trattavano la crittografia alla stregua di armi, e ne vietavano l'*esportazione*.
- PGP in principio era distribuito con il codice sorgente, in seguito il codice sorgente è stato chiuso e il programma è diventato commerciale.
- Nasce GPG GNU Privacy Guard, che implementa le stesse funzioni ma è rilasciato con una licenza libera, la GNU GPL
- GPG può essere usato anche con programmi di posta elettronica come Thunderbird.

# GPG: comandi utili

- come creare chiavi: gpg --gen-key
- come esportare chiavi: gpg --export --armor C93F299D
- come usare un keyserver:  
gpg --keyserver pgp.mit.edu --send-key C93F299D
- come cifrare messaggi: gpg --encrypt -r C93F299D file

## 1 Recall..

## 2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

## 3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

## 4 Esempio

- Il wot di GPG è comodo, ma si basa sulla fiducia reciproca e sul grande numero di persone che vi partecipano.
- In contesti più formali, perchè una chiave pubblica sia associata ad una persona ci vogliono garanzie più forti che possono essere date solo da terze parti riconosciute, gli enti certificatori.

# Enti certificatori e certificati

- Una CA (Certification authority) è un ente che garantisce l'associazione tra chiave pubblica e persona fisica.
- L'ente possiede una sua coppia di chiavi pubblica/privata.
- Gli utenti conoscono l'ente (e la sua chiave pubblica) e si fidano delle certificazioni che rilascia.
- La certificazione avviene esattamente come per le chiavi GPG ma si utilizza un formato di file diverso, lo standard X.509.
- Alcuni enti di certificazione: Poste Italiane, Verisign ecc. . . .

# Certificazione

- Come avviene la certificazione:

- L'utente A genera una coppia di chiavi pubblica/privata.
- L'utente A invia all'ente la propria chiave pubblica e un documento.
- L'ente restituisce la chiave pubblica dell'utente A firmata con la propria chiave privata. Il contenitore in cui si sposta la chiave è un certificato.
- Questa procedura si fa una sola volta, alla creazione della chiave.
- In questo modo l'ente certificatrice non conosce la chiave privata, che è una maggiore garanzia dell'utente. In un caso più semplice la CA invia entrambe le chiavi e il certificato all'utente.

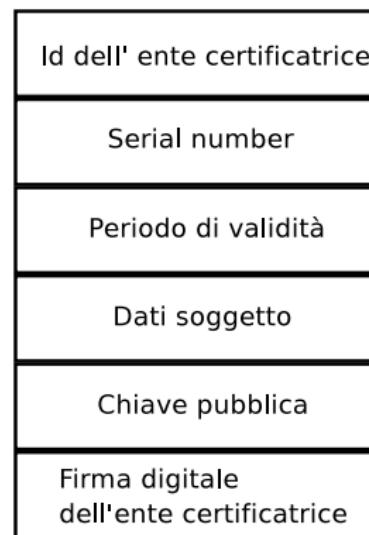
# Certificazione

- Quando un secondo utente B deve parlare con A, chiede ad A il suo certificato
- dal certificato estrae la chiave pubblica di A, verifica che la firma digitale dell'ente certificatrice sia corretta, utilizzando la chiave pubblica dell'ente certificatrice,
- a quel punto è sicuro che l'utente A è davvero chi dichiara di essere, perchè l'ente certificatrice è testimone per lui.
- Se io voglio utilizzare la mia firma digitale per vendere un'automobile, se sono certificato da un'ente lo posso fare. La firma digitale certificata ha lo stesso identico valore legale della firma su carta.
- Attraverso un sistema di certificati si può ottenere un accurato controllo degli accessi.

# Certificato digitale

Contiene:

- Dati identificativi dell'entità che fa da garante
- un serial number che identifica univocamente il certificato
- periodo di validità (da/a)
- dati identificativi del soggetto a cui è rilasciato (utente o dispositivo)
- la chiave pubblica del soggetto
- la firma digitale dell'ente che ha emesso il certificato



# Certificati per tutto

- Lo stesso modello può essere riprodotto per qualsiasi contesto, anche senza bisogno di contattare una CA ufficiale. Un esempio utile:
- Nell'azienda di cui amministrate la rete avete i seguenti servizi:
  - Posta elettronica per gli utenti
  - Sito web
  - Rete interna a cui collegare computer fissi e portatili
- Potete creare una vostra CA, con la sua coppia di chiavi e il suo certificato. Il certificato è *autofirmato*, ovvero la CA certifica se stessa
- Con la vostra CA rilasciate certificati validi a tutti gli utenti, in modo che invino solo posta elettronica firmata digitalmente. Ogni volta che un utente riceve una email questa è autenticata e cifrata.
- I browser sui computer aziendali possiedono un certificato proprio, in questo modo il server può accettare connessioni solo dalle macchine autorizzate
- Quando un portatile si connette alla rete, prima di essere abilitato a trasmettere e ricevere traffico dovrà autenticarsi con un server, utilizzando un certificato valido.

# Certificati per tutto, segue

- Anche se la vostra CA non è ufficiale, potete usarla all'interno della vostra organizzazione per aumentare il livello di sicurezza delle comunicazioni.
- È inutile sottolineare che se per qualche motivo il server su cui risiedono le chiavi pubbliche e private della vostra CA viene compromesso, è compromessa la sicurezza di tutta la rete.

# Certificate Revocation List: CRL

- Che succede nei seguenti casi:
  - uno dei vostri utenti perde il portatile con dentro un certificato valido?
  - uno dei vostri server con un certificato viene compromesso?
  - scoprite che uno dei vostri utenti si *comporta male*?
- In questi casi dovete essere in grado di revocare le credenziali di alcuni utenti, ovvero riuscire ad invalidare i certificati già rilasciati.
- Per farlo esistono le CRL.

# Certificate Revocation List: CRL

- Una CRL è semplicemente una lista di certificati che sono stati revocati dall'ente certificatore.
- Revocare un certificato significa che il certificato non deve essere più utilizzato, nella pratica avviene che la CA mantiene una lista di certificati non più validi e la distribuisce firmata con la sua chiave privata.
- Nella gestione di una rete sicura quindi si deve tenere conto anche del fatto che deve esistere un servizio attraverso il quale un utente può scaricare la CRL più aggiornata.
- Le CRL introducono un elemento di complicazione in più ma sono necessarie. Rendono infatti i certificati non più autonomi (auto-verificabili) ma emerge la necessità di richiedere un parere ad un ente terzo.

# Un approccio misto, il wot di Thawte

- Thawte è un'azienda che possiede una CA autorizzata, ma rilascia anche certificati secondo la logica del wot.
- Thawte ha dei *notai*. I notai possono certificare altre persone, pur non essendo direttamente dipendenti di Thawte. Funziona così:
  - Fate un account Thawte
  - Trovate un notaio, gli portate il numero del vostro account e una fotocopia di due documenti
  - Il notaio ha il potere di andare sul sito di Thawte e accreditare dei *punti* sul vostro account
  - Quando raggiungete un numero sufficiente di punti, vi viene consegnato un certificato per la vostra identità *rilasciato dalla CA autorizzata di Thawte*.
  - Se continuate a incontrare notai, potete accumulare punti e diventare voi stessi notai.
- Chiaramente i domini dei certificati Thawte per il wot e per le attività commerciali sono separati.

# In pratica: tinyca

- TinyCA è un programma per gestire una certification authority
- vedremo come generare certificati e firmare chiavi pubbliche

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

## C. Simmetrica

- Ha bisogno di un canale sicuro
- È computazionalmente semplice

## C. Asimmetrica

- Non ha bisogno di un canale sicuro
- È computazionalmente molto pesante

Le due tecniche vengono normalmente utilizzate insieme, per garantire sicurezza e performance.

# Sistema misto

- Vediamo un esempio di un possibile sistema misto per la sicurezza dello scambio di informazioni tra A e B:
  - A → B: certificato di A,  $C_a$
  - A ← B: certificato di B,  $C_b$
  - A → B: A genera un numero casuale R, trasmette  $\{R\}_{C_b}$ <sup>2</sup>
  - A ← B: B genera un numero casuale P, trasmette  $\{P\}_{C_a}$
  - la chiave segreta generata è  $K = \text{hash}(P \oplus R)$
- Dopo lo scambio effettuato le due parti possono smettere di utilizzare le chiavi pubbliche e continuare a cifrare e autenticare il traffico solo con la chiave K, in modo computazionalmente vantaggioso.
- **NB:** l'algoritmo proposto è una semplificazione e presenta molti difetti, serve a rendere l'idea di algoritmi più complessi come RSA.

<sup>2</sup>con  $\{x\}_y$  si indica il messaggio x cifrato con la chiave y

- *Handbook of Applied cryptography*: Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone.
- Online <http://www.cacr.math.uwaterloo.ca/hac/>
- Capitolo 1 par: 1.1, 1.2, 1.5.1, 1.5.4, 1.7, 1.8, 1.9, 1.11.3
- Un ottimo libro William Stallings *Cryptography and Network Security* (<http://williamstallings.com/Crypto3e.html>)

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

# Un po' di teoria: principi di crittografia

## Campo finito $F$

Insieme di elementi con due operatori (somma e moltiplicazione) tali che valgono le seguenti proprietà:

- Chiusura per la somma
- Associatività per la somma
- Identità additiva
- Inverso additivo
- Commutatività della somma
- Chiusura per il prodotto
- Associatività per il prodotto
- Leggi distributive
- Commutatività del prodotto
- Identità moltiplicativa
- Annullamento del prodotto
- Inverso moltiplicativo

# Il campo $Z_n$

L'insieme dei numeri interi minori di  $n$  con  $n$  numero primo è un campo, considerando le operazioni di somma e moltiplicazione modulo  $n$

- Se  $a, n$  sono interi, si definisce  $a \bmod n$  come il resto della divisione di  $a$  per  $n$ .
- operatore modulo:  $4 = 11 \bmod 7$ ,  $3 = (-11) \bmod 7$  ( $-2*7 + 3 = -11$ )
- operazioni in modulo:
  - somma:  $[a \bmod n + b \bmod n] \bmod n = (a+b) \bmod n$
  - moltiplicazione:  $[a \bmod n \times b \bmod n] \bmod n = (ab) \bmod n$
- NB: Per qualsiasi elemento  $a$  di  $Z_n$  esiste un unico  $a^{-1}$  per cui  $aa^{-1} = 1 \bmod n$  ovvero un unico inverso moltiplicativo.

# Teorema di Eulero

## Funzione Totiente di Eulero $\phi(x)$

$\phi(x)$  è il numero di interi postivi minori di  $x$  e primi relativi di  $x$ . Si dimostra che se  $p, q$  sono due numeri primi e  $x = pq$  allora:

- $\phi(x) = (p - 1)(q - 1)$

Notate che calcolare  $\phi(x)$  per un  $x$  qualsiasi è computazionalmente oneroso, impossibile per  $x$  sufficientemente grandi. Se  $x = pq$  e conosco almeno uno tra  $p$  e  $q$  è invece possibile.

## Teorema di Eulero

Per  $a, x$  primi tra loro vale che  $a^{\phi(x)} \equiv 1 \pmod{x}$

# RSA for dummies

- dati  $p, q$  primi e  $n = pq$ , dato  $m$  un numero di dimensione  $< n$  (n può essere la codifica binaria di qualsiasi testo in chiaro) si dimostra che se  $e, d$  sono inversi moltiplicativi modulo  $\phi(n)$  (ovvero  $ed \equiv 1 \pmod{\phi(n)}$ ) allora
  - $m^{ed} \pmod{n} = m^{1+k\phi(n)} \pmod{n} = m(m^{k\phi(n)}) \pmod{n} = (m \pmod{n})(m^{k\phi(n)} \pmod{n}) = (m \pmod{n})((m^{\phi(n)})^k \pmod{n}) = m \pmod{n}$  (per il teorema di Eulero).
- dato  $m^e = c$  è computazionalmente oneroso ricavare  $m$ , oggi impossibile per numeri grandi. Quindi si può cifrare esponenziando per  $e$ , e si può decifrare esponenziando ancora per  $d$ .
- Quindi la coppia  $e, n$  è la chiave pubblica e la coppia  $d, n$  è la chiave privata.
- Ad oggi, per valori di  $e, n$  non piccoli ( $> 1\text{Kbit}$ ) non esistono algoritmi che permettano di:
  - dato  $e, n$  ricavare  $d$ .
  - dato  $m^e$  ricavare  $m$

# Attacchi su RSA: fattorizzare $n$

- La sicurezza di RSA si basa sull'impossibilità di derivare  $\phi(n)$  o  $d$  da  $e, n$ . Entrambi questi problemi hanno complessità equivalente a fattorizzare  $n$  nei suoi fattori primi.
- tra il 1991 e il 2003 sono stati fattorizzati numeri da 332 a 663 bit utilizzando algoritmi di fattorizzazione diversi (e decine di macchine in cluster a lavoro per mesi). Ad oggi si ritiene che utilizzare chiavi di dimensione superiore a 1024 bit sia sufficiente.
- E' importante notare che la sicurezza di una chiave dipende dalla lunghezza e dal migliore algoritmo noto di fattorizzazione allo stato dell'arte. Tra il 1994 e il 1996 è stato introdotto un nuovo algoritmo che ha potuto fattorizzare un numero di 431 bit con il 20% delle risorse rispetto all'algoritmo noto in precedenza. Non è detto che la cosa non si ripeta in futuro, magari con risultati più estremi.
- La sicurezza di RSA dipende quindi tutta dallo stato dell'arte in questo campo e nel campo della potenza computazionale.

## Dalle FAQ di rsa.com

Quantum computing is a new field in computer science that has been developed with our increased understanding of quantum mechanics. It holds the key to computers that are exponentially faster than conventional computers (for certain problems). A quantum computer is based on the idea of a quantum bit or qubit [...]

It has been proven that a quantum computer will be able to factor and **compute discrete logarithms in polynomial time**. Unfortunately, the development of a practical quantum computer still seems far away because of a phenomenon called quantum decoherence, which is due to the influence of the outside environment on the quantum computer

## Potenza computazionale

Quello che crediamo *infeasible* oggi potrebbe non esserlo domani.

# Attacchi su RSA: timing attacks

- Nel processo di decodifica è necessario produrre un esponenziazione con la chiave privata.
- Le operazioni in hardware hanno un costo computazionale diverso se il bit usato per l'esponenziazione è 0 o 1.
- Osservando i tempi di esecuzione della CPU, si può risalire alla chiave privata solo osservando operazioni di decodifica.
- E' un attacco laborioso ma che arriva da una direzione inattesa.
- Le implementazioni di RSA introducono dei ritardi casuali nell'esponenziazione per rendere impredicibile i tempi di esecuzione.

# Altri algoritmi a chiave pubblica

Gli stessi problemi computazionalmente intrattabili utilizzati in RSA (fattorizzazione di numeri primi, logaritmo di numeri interi) sono alla base di altri algoritmi frequentemente utilizzati:

- Diffie-Hellman: genera una chiave segreta condivisa a partire da due chiavi pubbliche note senza bisogno di scambiare esplicitamente alcun segreto.
- ElGamal: schema a chiave pubblica basato su logaritmi discreti
- DSA: schema di firma digitale basato su logaritmi discreti

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- **Diffie-Hellman**
- Algoritmi a chiave simmetrica
- Altri algoritmi

4 Esempio

# Scambio di chiavi Diffie-Hellman

Si basa sull'intrattabilità del logaritmo discreto. Dato un numero primo  $p$  si definisce radice primitiva di  $p$  un numero  $\alpha$  per cui vale che

- $\alpha^{mod p} \neq \alpha^2 mod p \neq \alpha^3 mod p \neq \dots \neq \alpha^i mod p$  con  $i < p$

Nello scambio DH entrambi gli utenti A e B posseggono due parametri noti  $p$ ,  $\alpha$ . Ciascuno genera un numero casuale  $X_a$  e  $X_b$ .

- A invia a B  $Y_a = \alpha^{X_a} mod p$
- B riceve  $Y_a$  ed invia ad A  $Y_b = \alpha^{X_b} mod p$
- A calcola  $K_a = Y_b^{X_a} mod p = (\alpha^{X_b})^{X_a} mod p = \alpha^{X_b * X_a} mod p$
- B calcola  $K_b = Y_a^{X_b} mod p = (\alpha^{X_a})^{X_b} mod p = \alpha^{X_a * X_b} mod p$
- Ma  $K_a = K_b$ . Quindi A e B si sono scambiati una chiave segreta **senza avere nessuna credenziale comune**. Un attaccante che vede passare solo  $Y_a$  e  $Y_b$  non è in grado di calcolare il logaritmo discreto quindi non può ricavare la chiave!

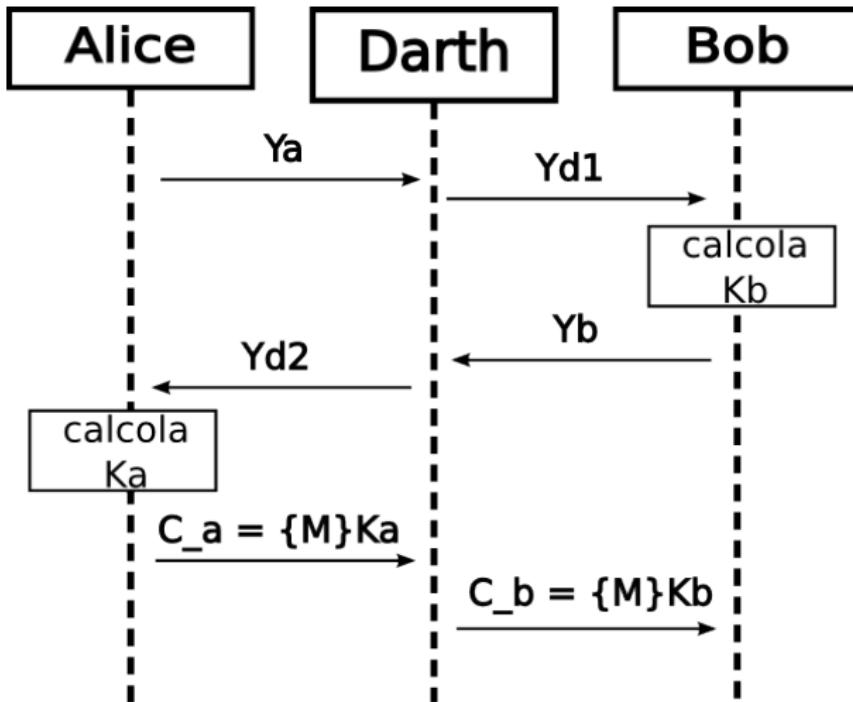
# Scambio di chiavi Diffie-Hellman

Si dimostra abbastanza facilmente che DH non è sicuro contro attacchi man-in-the-middle in cui l'attaccante D(arth) è in grado di modificare il traffico tra A e B.

- D genera  $X_{d1}$  e  $X_{d2}$  e le chiavi pubbliche corrispondenti  $Y_{d1}$  e  $Y_{d2}$ .
- A invia  $Y_a$  a B
- C intercetta il messaggio e trasmette  $Y_{d1}$  a B
- B riceve  $Y_{d1}$  e calcola  $K_b = Y_{d1}^{X_b} \text{ mod } p$
- B invia  $Y_b$  ad A
- D intercetta  $Y_b$  e invia  $Y_{d2}$  ad A
- A calcola  $K_a = Y_{d2}^{X_a} \text{ mod } p$

Alla fine dello scambio A e B non condividono nessuna chiave, ma entrambi condividono una chiave con D. D può intercettare il traffico, decifrarlo e ricifrarlo

# Scambio di chiavi Diffie-Hellman



## 1 Recall..

## 2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

## 3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

## 4 Esempio

## One time pad

L'algoritmo One-Time-Pad è quello che in linea teorica produce la sicurezza più elevata, ma nella pratica non è facilmente utilizzabile

- Dato un testo in chiaro  $m$  di  $n$  bit si sceglie una chiave  $k$  di  $n$  bit generata con generatore perfetto di numeri casuali.
- La cifratura è semplicemente  $c = m \oplus k$
- Ad ogni trasmissione si deve cambiare  $k$

Con questo algoritmo si scorrela completamente  $c$  da  $m$  e non può essere effettuata crittanalisi. Il problema ovviamente è che si deve trasportare con un canale sicuro una chiave lunga quanto il testo da spostare.

## Algoritmi di sostituzione

Un algoritmo di sostituzione ideale che mappa un messaggio in chiaro di 2 bit in un messaggio cifrato di 2 bit funziona utilizzando una mappa statica:

- $00 \rightarrow 01$
- $01 \rightarrow 11$
- $10 \rightarrow 00$
- $11 \rightarrow 10$

Anche qui se il messaggio è lungo 2 bit e la mappa ha  $2^2$  righe l'attaccante può solo provare attacchi di forza bruta, non esiste correlazione statistica tra testo in chiaro e testo cifrato.

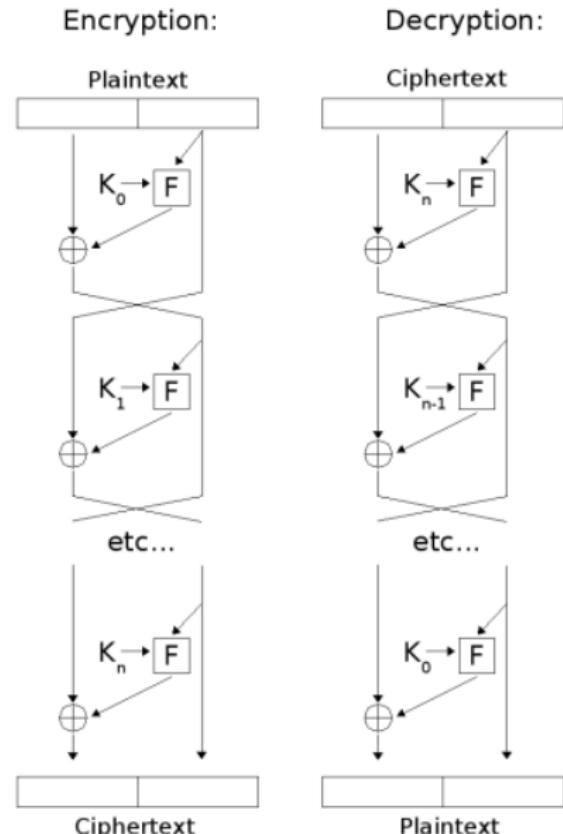
Se il messaggio è più lungo si possono cifrare blocchi di due bit per volta.

## Problema

Per essere sicuro i blocchi devono essere di grandi dimensioni. In tal caso la chiave (la mappa) diventa molto grande (es:  $n=64 \rightarrow 64 \times 2^{64} = 2^{70}$  bit).

# Cifratura di Feistel

- Dalla chiave  $K$  si generano una serie di sottochiavi  $K_i$  con una funzione generatrice
- Si eseguono una serie di fasi di cifratura che hanno come parametro  $K_i$  e il risultato della fase precedente.
- Tutti gli algoritmi a blocchi moderni come AES usano questo schema di cifratura.



## 1 Recall..

## 2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

## 3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- **Altri algoritmi**

## 4 Esempio

# Curve ellittiche

- Un gruppo può essere definito nello spazio dei punti definito da una curva ellittica (del tipo  $y^2 = x^3 + ax^2 + b$ ) sotto certe condizioni.
- Ragionamenti analoghi a quelli fatti per RSA possono essere riportati al contesto delle curve ellittiche.
- Il vantaggio maggiore rispetto ad RSA è che la stessa sicurezza si ottiene con chiavi di dimensione minore. Quindi le operazioni di codifica e decodifica sono più veloci. L'NSA americana ha inserito alcuni algoritmi a curve ellittiche nel *suite B*, un insieme di algoritmi ufficialmente supportati.

Symmetric Scheme (key size in bits)	ECC-Based Scheme (size of $n$ in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

# ID-based cryptography: IBC

Il problema di distribuire i certificati è un limite significativo della cifratura a chiave pubblica. Impone un'infrastruttura e introduce costi di set-up di una sessione. La ID-based cryptography nasce per eliminare questo limite.

- Con IBC la chiave pubblica di un utente è derivata direttamente da un identificativo dell'utente ad es.  $e = \text{hash}(\text{leonardo.maccari@unifi.it})$
- Non c'e' bisogno di scambiarsi certificati, il canale di trasmissione (email, IP, ecc..) definisce esso stesso l'associazione chiave-utente.
- Tecnicamente esistono ancora molti limiti perchè la IBC prenda piede. Uno su tutti è che affinchè il sistema funzioni è necessario che le chiavi vengano generate tutte da un ente fidato. Al contrario di RSA in cui la chiave privata può non essere mai rivelata a nessuno se non al proprietario.
- Esistono aziende che vendono soluzioni basate su IBC (Trend Micro)

# Quantum Cryptography

- Si basa sull'utilizzo di informazioni immodificabili associate ad una trasmissione di dati.
- Ad esempio, in una fibra ottica, l'arrivo di un fotone implica la ricezione di un'informazione.
- Ogni fotone possiede uno stato di polarizzazione che secondo il principio di indeterminazione di Heisenberg non può essere misurato senza interferirci.
- Negli stati dei fotoni viene codificata una chiave simmetrica che verrà utilizzata per cifrare il resto della comunicazione.
- Ad oggi si applica solo a comunicazioni su fibra.

# TOC

1 Recall..

2 Principi di crittografia:

- Funzioni hash
  - HMAC
- Cifratura a chiave simmetrica
- Cifratura a chiave pubblica/privata
- Certificati
- Riassumendo

3 Accenni di teoria

- RSA
- Diffie-Hellman
- Algoritmi a chiave simmetrica
- Altri algoritmi

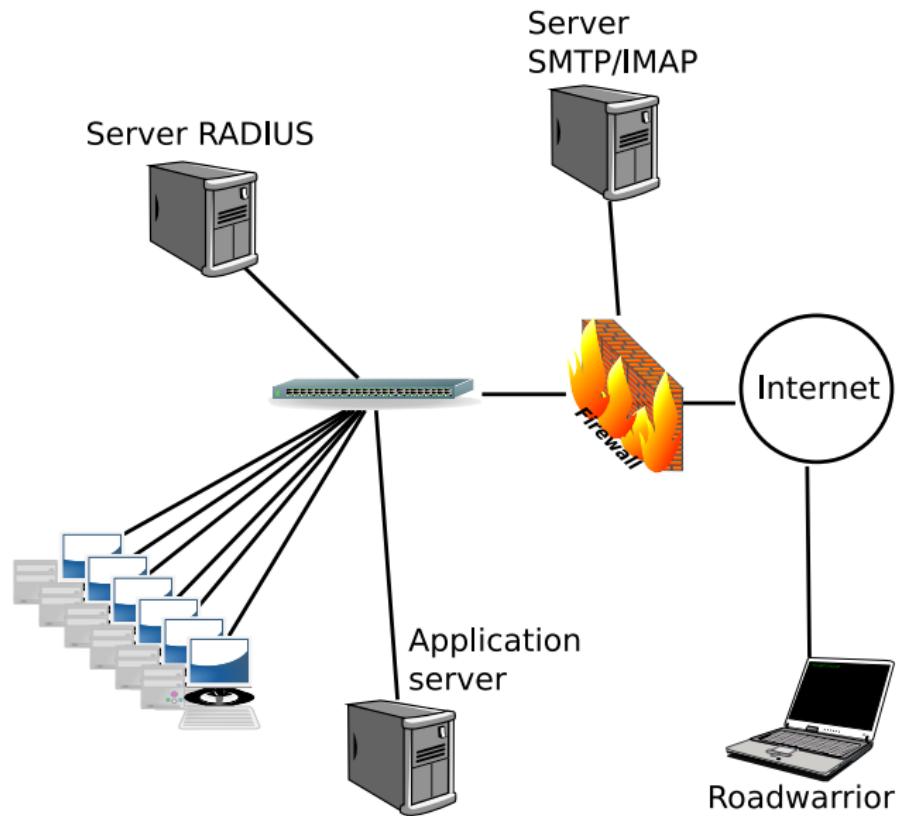
4 Esempio

# Organizzazione di una rete con certificati: esempio

La vostra rete è composta da

- Una rete di computer fissi per i vostri utenti
- Un server per applicativi web necessari ai vostri utenti
- Un server per la posta elettronica
- Dei possibili utenti remoti, **roadwarrior**.

# Esempio: la rete



# Esempio: i servizi

Nella vostra rete volete assicurare che:

- non sia possibile collegarsi alla rete interna se non utilizzando i client fissi della rete,
- che ogni ad client della rete possa effettuare l'accesso solo personale autorizzato
- che i servizi del vostro webserver siano accessibili solo dai terminali della rete
- che la posta elettronica sia certificata
- che gli utenti possano inviare e ricevere posta elettronica in modo sicuro
- che tutto quello che può fare un utente dal proprio terminale interno lo possa fare anche un utente *roadwarrior*

# Esempio: la CA

Avete bisogno di una CA

- un macchina il più possibile isolata dalla rete, non accessibile dall'esterno, non accessibile direttamente dagli utenti.
- Su questa macchina generate un certificato master, protetto da password che utilizzerete per generare tutti i certificati dei servizi. Possibilmente, in un luogo fisicamente inaccessibile (cassaforte?) dovete tenere un backup del certificato.
- Se la vostra rete è contenuta potete usare un programma come tinyca per creare le chiavi degli utenti e dei servizi, che poi distribuire nei singoli terminali, altrimenti avrete bisogno di un'interfaccia web accessibile dagli utenti.
- Le CRL della CA devono essere pubblicate in un luogo liberamente accessibile a cadenza regolare.

# Esempio: i certificati

Avete bisogno dei seguenti certificati per i server:

- uno per il webserver
- uno per il server imap
- uno per il server smtp
- uno per il server di autenticazione degli utenti
- uno per la VPN

Avete bisogno dei seguenti certificati per i client:

- uno per ogni macchina client
- uno per ogni utente
- uno per ogni roadwarrior

# Esempio: l'installazione

Dove mettere i certificati degli utenti?

- ogni macchina nella rete interna deve possedere un certificato non accessibile dall'utente. All'avvio la macchina dovrà produrre un'autenticazione con un server di autenticazione in modo da poter accedere allo switch a cui è collegata. C'è bisogno:
  - di uno switch compliant con IEEE 802.1X
  - di un server RADIUS
- ogni utente riceve un certificato (in una smartcard, o in un token di qualsiasi tipo), i terminali utenti devono avere un lettore hardware.
  - quando l'utente arriva alla postazione si autentica con il token allo stesso server RADIUS e ottiene il login.
  - lo stesso certificato verrà utilizzato dai client di posta elettronica e dal webserver.
- Volendo, oltre al certificato, il login può essere legato anche ad una password, utilizzabile direttamente dal token crittografico per *sbloccarsi*.

## Esempio: i token<sup>3</sup>



<sup>3</sup> Immagini in licenza CC da flickr.com

# Esempio: i token



# Esempio: l'installazione

Dove mettere i certificati dei server?

- nella configurazione di Apache
- nella configurazione del server IMAP
- nella configurazione del server SMTP
- nel server RADIUS
- nel server VPN

Ognuno di questi server deve possedere un suo certificato e deve poter accettare connessioni solo da utenti con certificati validi.

# Esempio: l'installazione

Sia gli utenti che i server devono avere accesso

- al certificato della CA,
- alla CRL.

# Esempio: la rete finale

