1 Lezione del 18-11-25

1.1 Riassunto di Internet

Cerchiamo di mettere insieme quanto visto sull'infrastruttura Internet osservando la vita di un messaggio relativo ad un applicazione che conosciamo bene, cioè una *richiesta Web*.

Poniamo di avere un host, come ad esempio un portatile o un telefono cellulare, collegato alla rete locale (magari una LAN istituzionale su Ethernet).

1. Appena arrivato sulla rete, l'host non avrà indirizzo IP, per cui dovrà inviare un messaggio di *Discovery* **DHCP** sulla sua rete locale. Ricordiamo che DHCP è un protocollo di livello application, per cui questo messaggio verrà incapsulato in un segmento UDP, incapsulato in un datagramma IP, incapsulato in un frame Ethernet 802.3 con indrizzo di *broadcast*.

   Il server DHCP riceve il messaggio, demultiplexando il datagramma IP e il segmento UDP, e risponde. Questo processo si ripete per il cosiddetto *4-way handshake* DHCP (Discovery, Offer, Request, Acknowledge).

   Al termine della trasmissione di *Acknowledge*, quindi, l'host conosce l'indirizzo del server DHCP, del DNS locale, il gateway predefinito e il suo indirizzo IP all'interno della rete;

2. Il secondo passaggio è quello della risoluzione del nome di dominio attraverso il **DNS**. Verrà quindi creata una query DNS (che è nuovamente un protocollo di livello application), che verrà incapsulata in un segmento UDP, incapsulato in un datagramma IP, incapsulato nuovamente in un frame Ethernet 802.3 inviato al server DNS della rete locale.

   Il server DNS riceve tale messaggio, e in maniera iterativa o ricorsiva interroga i server DNS dal top-level (o più in basso se ha cache), fino al server autoritativo associato al dominio cercato. Da questo ottiene un record contenente l'indirizzo IP del Web server con cui l'host desidera comunicare. Questo indirizzo viene inserito in una risposta DNS inviata all'host.

   A questo punto l'host riceve la richiesta: conosce l'indirizzo IP del Web server con cui deve parlare;

3. Si stabilisce una connessione **TCP** col Web server, attraverso il classico *3-way handshake* TCP sul socket di ascolto.

   Il Web server riceve l'handshake, e da lato suo provvede ad aprire una nuova connessione TCP con relativo socket. Le modalità con cui gestisce questo nuovo socket potrebero essere variegate (processi, thread, server iterativi, ecc...).

   Al termine di questa fase sia host che Web server hanno dei socket TCP attivi e sono pronti a scambiarsi dati;

4. Infine, l'host, che a questo punto si comporterà da client **HTTP**, inviando una richiesta HTTP ad esempio per la pagina `index.html` del Web server contattato, sulla porta 80. La richiesta verrà incapsulata su TCP, e quindi in datagrammi IP e frame Ethernet che raggiungeranno il gateway di rete. Da lì in poi attraverso il protocollo IP la richiesta raggiungerà il server.

Il server riceverà la richiesta, che verrà demultiplexata da TCP sulla porta 80, cercherà la risorsa desiderata e invierà indietro una risposta HTTP che la contiene. Questo processo si svolge analogamente a prima, ma nella direzione opposta.

Quando il client riceve la pagina, il browser ivi installato potrà accedere al messaggio di risposta HTTP inviato dal server, accedere ai dati HTML che contiene, e visualizzare la pagina sullo schermo.

1.2 Sicurezza in rete

Un problema che abbiamo finora ignorato delle reti è la **sicurezza**. La sicurezza di rete si pone di assicurare:

- **Confidenzialità**: solo sorgente e destinatario dei messaggi in rete dovrebbero essere in grado di decodificare il contenuto del messaggio. In questo si vuole che il *trasmettitore* **cifri** il messaggio, e il *ricevitore* lo **decifri**.

  Questo permette di evitare attachi del tipo *man-in-the-middle* o di *sniffing*, cioè a eventuali malintenzionati di *"spiare"* la conversazione fra 2 o più host;

- **Autenticazione**: si vuole che ricevitore e trasmettitore possano confrmare l'identità l'uno dell'altro.

  Questo è chiaramente importante nel caso di sistemi che devono verificare l'identità degli utenti (pagine di login, ecc...);

- **Integrità dei messaggi**: vogliamo assicurarci che i messaggi in transito non vengano modificati, o comunque che non vengano modificati senza la possibilità di rilevare le modifiche.

  Anche questo torna utile nel caso di attacchi del tipo *man-in-the-middle* (quando non *"spiamo"* soltanto i messaggi, ma li manomettiamo), o semplicemente per combattere le imprecisioni date dalle caratteristiche di inaffidabilità dei mezzi di comunicazione stessi;

- **Accesso e disponibilità**: vogliamo assicurare che i servizi Internet che offriamo siano accessibili e disponibili agli utenti, anche di fronte a elementi di disturbo o carico consistente.

  Questo è particolarmente importante per il paradigma client-server, dove vogliamo che il server sia disponibile in qualsiasi momento, e si può incorrere in attacchi **DoS** (*Denial of Service*) mirati appunto ad impedire l'accesso ai server sovraccaricandoli.

  Altri attacchi di questo tipo sono quelli di tipo **ransomware**, dove si mira a crittografare i dati degli utenti, chiedendo riscatto per la decifrazione (in questo caso l'accesso che sottraiamo è quello ai dati stessi). Secondo l'Anastasi questo è un buon modo per fare soldi facili.

1.2.1 Un semplice modello

Riprendiamo i personaggi di Francesca e Maurizio, assunto che fra i due sia presente un canale di comunicazione non sicuro, e introduciamo un nuovo personaggio, *Giuseppina*, che ha come obiettivo quello di spiare e manomettere la conversazione fra Francesca e Maurizio.

Chiaramente, Francesca e Maurizio saranno un modello per due host che trasmettono informazioni sensibili, come client e server di un servizio Web di transazioni online, server DNS o router che si scambiano tabelle di routing, ecc...

Ci sono molte azioni dannose che Giuseppina potrebbe intraprendere sulla comunicazione fra Francesca e Maurizio, come ad esempio:

- **Sniffing** dei messaggi;
- **Inserimento** di nuovi messaggi sul canale;
- **Impersonazione** di Francesca o Maurizio, replicando il loro indirizzi IP;
- **Manomissione** della comunicazione fra Francesca e Maurizio, ad esempio rimuovendo uno dei due dal canale di comunicazione;
- **Denial of service**, cioè prevenire che il servizio venga usato (eliminando i messaggi sul canale o sovraccaricando le risorse di Francesca o Maurizio).

1.3 Introduzione alla crittografia

Il primo meccanismo di difesa che abbiamo a disposizione è la **crittografia**.

Definiamo un tipo di crittografia generale, basata su un dato *algoritmo crittografico*. Un algoritmo crittografico è un algoritmo che prende in ingresso una stringa comprensibile all'uomo (cioè in *chiaro*) e restituisce una stringa crittografata, cioè incomprensibile.

L'algoritmo crittografico prende come argomento una chiave, che usa per crittografare il messaggio in chiaro. Fissiamo l'algoritmo e diciamo che la sua applicazione al messaggio in chiaro m con chiave K si può semplicemente indicare come $K(m)$.

Tornando ad Francesca e Maurizio, diciamo che Francesca possiede la chiave crittografica K_A e Maurizio la chiave K_B. Vogliamo come condizione che per qualsiasi m:

$$m = K_B(K_A(m))$$

cioè K_B *decifra* i messaggi cifrati con K_A.

A questo punto quello che Francesca invierà sulla linea sarà $K_A(m)$, e Maurizio potrà successivamente usare la sua chiave K_B per ottenere il messaggio m secondo la stessa equazione di prima.

Diciamo anche che K_A è la chiave *pubblica*, cioè quella che può essere nota a tutti (e lo sarà ad Francesca come a Giuseppina), in quanto viene usata per crittografare i messaggi destinati a Maurizio. K_B sarà invece la chiave *privata*, cioè quella che solo Maurizio deve avere, e che egli può usare per decifrare i messaggi rivolti a sé. Questo è sicuro in quanto, una volta crittografati, nessuno (né Francesca ne Giuseppina) può decifrare i messaggi rivolti a Maurizio, se non Maurizio stesso.

1.3.1 Approcci alla decifratura

Vediamo quindi come Giuseppina, entrata in possesso di un messaggio crittografato da Francesca per Maurizio, potrebbe procedere alla decodifica del messaggio.

- Un primo approccio è chiaramente quello di ottenere in qualche modo la chiave di Maurizio. Questo non è un approccio attuabile nella pratica (ci aspettiamo che Maurizio custodisca attentamente la sua chiave), ma se non altro ci dimostra che non esistono approcci *"a prova di idiota"*, e cioè che non facendo attenzione con le chiavi si possono invalidare anche gli schemi crittografici più complessi;

- Un altro approccio è quello a **forza bruta**: si può provare a decrittografare raccogliendo a priori un insieme di chiavi possibili, ecc... il messaggio di Francesca applicando l'algoritmo crittografico (che abbiamo detto è prefissato e noto) con tutte le chiavi possibili: questo però richiede grandi capacità computazionali e non è quasi mai conveniente;
- Un approccio più intelligente è quello di tipo statistico, cioè si può pensare di fare un'**analisi statistica** prima di applicare la forza bruta, magari raccogliendo a priori un insieme di chiavi possibili, ecc...

1.4 Crittografia a chiave simmetrica

Il metodo di crittografia più semplice è quello a **chiave simmetrica**.

In questo caso sia Francesca che Maurizio sono in possesso della stessa chiave, detta K_S (da *chiave Simmetrica*). Assumiamo allora che la doppia applicazione della chiave porta alla decifratura del messaggio, cioè:

$$m = K_S(K_S(m))$$

per cui basta che Francesca cifri il messaggio con la sua chiave, e Maurizio (che ha una sua copia della stessa chiave) potrà decifrare il messaggio una volta ricevuto.

Il problema è chiaramente quello di come Francesca e Maurizio possono scambiarsi la chiave, in particolare se l'unico mezzo di comunicazione che hanno a disposizione è lo stesso mezzo inaffidabile su vogliono cifrare i loro messaggi.

1.4.1 Metodi di sostituzione

Uno dei più semplici metodi per la cifratura è il cosiddetto **cifrario di sostituzione** (noto anche come *cifrario di Cesare*).

- L'esempio più semplice è il cifrario per sostituzione *monoalfabetico*, dove si sostituisce ogni lettera con un'altra lettera dello stesso alfabeto, equivalente alla lettera originale spostata avanti secondo la chiave (che sarà quindi un intero). Se assumiamo l'alfabeto a 26 lettere, questo metodo crea una mappatura da 26 lettere a 26 lettere, per cui per ogni lettera abbiamo 25 possibili altri lettere con cui sostituire (tralasciando la cifratura banale, cioè l'identità).

  Ciò porta a 25 possibili chiavi da testare, che sui moderni calcolatori è sostanzialmente banale. Abbiamo bisogno di un metodo migliore;
- Un miglioramento del cifrario per sostituzione monoalfabetico si ha assumendo non di spostare ogni lettera in avanti, ma di mappare in maniera arbitraria ogni lettera ad un altra lettera. Assumendo sempre 26 possibili lettere nell'alfabeto, questo porta a $26!$ possibili chiavi, che è già migliore rispetto all'approccio precedente.
- Un altro miglioramento che possiamo attuare è quello di prevedere n chiavi di sostituzione cicliche, e assumere che ogni lettera successiva è cifrata con la prossima fra le chiavi cicliche.

  Poniamo ad esempio di avere M_0, M_1 e M_2 come chiavi di sostituzione. In questo caso una stringa come *"Rio"* sarà ottenuta:

  - Decifrando R col cifrario M_0;

  - Decifrando i col cifrario M_1;
  - Decifrando o col cifrario M_2;

  Dovrebbe essere apparente come la complessità dello schema di cifratura esplode in maniera esponenziale, per cui metodi di questo tipo risultano ancora più sicuri (metodi di forza bruta effettivamente impossibili da applicare, metodi statistici più inefficaci, ecc..). Il tradeoff è però che chiavi di questo tipo occupano dimensioni sempre più grandi man di mano che complichiamo lo schema.

1.4.2 Tipi di cifrari simmetrici

Esistono 2 tipi particolari di applicazione dei metodi a chiave simmetrica nelle reti informatiche:

- Cifrari di **stream**: dove ogni byte viene cifrato. Questo approccio viene usato ad esempio per le connessioni **SSL/TLS** (*Secure Sockets Layer / Transport Layer Security*), le connessioni **Bluetooth**, e le reti cellulari;
- Cifrari di **blocco**: dove i messaggi in chiaro vengono divisi in blocchi di dimensione uguale, ed ogni blocco viene cifrato a sé. Questo metodo è usato ad esempio per gli standard **DES** (*Data Encryption Standard*), **3DES** (*Triple DES*), **AES** (*Advanced Encryption Standard*) e **IDEA** (*International Data Encryption Algorithm*).

Vediamo nel dettaglio alcuni di questi algoritmi:

- Il **DES** (*Data Encryption Standard*) è un metodo con chiavi simmetriche a 56 bit, e input a blocchi di 64 bit (con *block chaining*). Prove sperimentali hanno dimostrato che una stringa crittografata con DES può essere decifrata in meno di un giorno.
- Lo stesso algoritmo è stato quindi riproposto sotto la forma di **3DES** (*Triple DES*), che consiste nel crittografare 3 volte con DES usando chiavi diverse;
- L'**AES** (*Advanced Encryption Standard*) è un altro approccio a chiave simmetrica, con chiavi su 128, 192 o 256 bit e input a blocchi di 128 bit. Risulta già migliore del DES in quanto a forza bruta si richiede qualcosa nell'ordine dei 149 migliaia di miliardi di anni, contro circa un secondo del DES (forse meno con la potenza di calcolo odierna).

1.4.3 Scambio di chiavi simmetriche

Torniamo al problema che ci eravamo posti in subito in 23.3.2: come possono Francesca e Maurizio, se vogliono usare un approccio a chiave simmetrica, scambiarsi tale chiave prima di iniziare la comunicazione?

- L'approccio più sicuro (ideale) è chiaramente quello di scambiarsi la chiave su un canale diverso da quello inaffidabile, come ad esempio di persona. Questo è impossibile nel caso non si abbiano altri canali di comunicazione a disposizione;
- Si può usare la crittografia a chiave pubblica, che abbiamo già introdotto in 23.3;
- Si può pensare di usare un centro di distribuzione di chiavi, cioè un **KDC** (*Key Distribution Center*). Questo dovrà essere un ente terzo, fidato, a cui Francesca e Maurizio potranno iscriversi per ottenere una chiave fidata nota solo a loro. Compito del KDC sarà assicurarsi che solo Francesca e Maurizio possano entrare in

possesso della chiave, e non terzi (come ad esempio Giuseppina). Nella pratica, ciò che abbiamo ottenuto è lo spostamento del problema, da Francesca e Maurizio e il loro canale inaffidabile, al KDC.

Dettagliamo questo processo:

1. Quando vuole iniziare una comunicazione, Francesca contatta il KDC, con cui inizia una comunicazione ottenendo una chiave detta K_{A-KDC}. Come si ottiene la chiave per il KDC non ci è di interesse (diciamo che se la sono scambiata di persona);
2. Maurizio, di lato suo, avrà a disposizione una sua chiave per il suo KDC locale, detta K_{B-KDC};
3. Visto che lo scopo di Francesca non è comunicare col KDC, ma con Maurizio, questa invia una richiesta cifrata al KDC che segnala l'intenzione di comunicare con Maurizio, cioè invia il messaggio:

   $$K_{A-KDC}(A, B)$$

   Il KDC risponderà quindi con una *chiave di sessione* (la chiave R_1), che avrà una certa scadenza prefissata. Oltre alla chiave di sessione, il KDC invierà ad Francesca la chiave di sessione di Maurizio, cifrata usando la chiave $K_(B - KDC)$ (cioè la chiave simmetrica di Maurizio con il KDC), a cui avrà annesso l'informazione riguardo a *chi* sta richiedendo la comunicazione (cioè Francesca).
   In simboli, ciò che il KDC invierà a Francesca sarà:

   $$K_{A-KDC}\left(R_1, K_{B-KDC}\left(A, R_1\right)\right)$$

   Notiamo che il messaggio destinato a Maurizio, cioè $K_{B-KDC}\left(A, R_1\right)$, sarà inviato a Francesca ma risulterà a lei effettivamente indecifrabile.
4. A questo punto Francesca potrà inviare il messaggio cifrato secondo la chiave di Maurizio a Maurizio, e da qui in poi questo conoscerà la chiave di sessione e chi richiede la comunicazione (Francesca). Da qui in poi la comunicazione può avere inizio.

1.5 Crittografia a chiave pubblica

Le limitazioni della crittografia a chiave simmetrica dovrebbero ormai esserci chiare: si richiede che trasmettitore e ricevitore conoscano a priori la chiave simmetrica, senza che nessun'altro possa entrarne in possesso, e questo non è banale.

Potrebbe quindi essere utile sfruttare un approccio a **chiave pubblica**, che abbiamo già accennato in 23.3. Questo è un approccio completamente diverso, dove ricevitore e trasmettitore non devono condividere una chiave condivisa, ma esistono 2 chiavi:

- Una chiave **pubblica** di cifratura, nota a tutti;
- Una chiave **privata** di decifratura, nota solo al ricevitore.

Il funzionamento è semplice: una volta che il ricevitore ha generato la coppia di chiavi, tiene sotto custodia la chiave privata e diffonde la chiave pubblica a tutti quelli che vogliono parlare con lui in maniera cifrata. Questi si occuperanno quindi di cifrare i messaggi destinati al ricevitore (messaggi che, una volta cifrati, non potranno più leggere),

e inviarli. Il ricevitore sarà quindi il solo che potrà usare la chiave privata per decifrare i messaggi ricevuti. Gli unici che necessitano di conoscere il messaggio in questo processo sono trasmettitore e ricevitore.

In simboli, diciamo che K_B^+ è la chiave pubblica di Maurizio, mentre K_B^- è la sua chiave privata. Varrà l'equazione, per un qualsiasi messaggio in chiaro m:

$$m = K_B^- \left(K_B^+(m) \right)$$

Quello che Francesca metterà sul canale sarà quindi:

$$K_B^+(m)$$

cioè il messaggio in chiaro crittografato con la chiave pubblica di Maurizio, da cui Maurizio potrà ricavare il significato originale sfruttando l'equazione scritta sopra.

1.5.1 Algoritmi di crittografia a chiave pubblica

Nel descrivere la crittografia a chiave pubblica, abbiamo fatto la pretesa (non banale) di avere a disposizione un **algoritmo** di cifratura (assunto fisso) e una coppia di chiavi K_B^+, K_B^- tali che:

$$m = K_B^- \left(K_B^+(m) \right)$$

Un'altra proprietà, che aggiungiamo ora (e ci tornerà utile in seguito), è la seguente:

$$m = K_B^- \left(K_B^+(m) \right) = K_B^+ \left(K_B^-(m) \right)$$

Un'algoritmo tipico di questo tipo è l'**RSA** (da Ron Rivest, Adi Shamir e Leonard Adleman, che lo pubblicarono nel 1977).

1.5.2 Chiavi di sessione

Il problema dell'RSA è che la cifratura è molto complessa dal punto di vista computazionale: ad esempio, DSA è circa 100 volte più veloce dell'RSA.

Un approccio migliore è quindi quello dove l'RSA viene usato per stabilire una prima connessione sicura, sulla quale si stabilisce una seconda chiave, la cosiddetta **chaive di sessione** (simbolo K_S), con cui la comunicazione vera e propria