Sicurezza

Alessio Marini, 2122855

Appunti presi durante il corso di **Sicurezza** nell'anno **2025/2026** del professore Emiliano Casalicchio.
Gli appunti li scrivo principalmente per rendere il corso più comprensibile **a me** e anche per imparare il linguaggio Typst. Se li usate per studiare verificate sempre le informazioni 🙏.

Contatti:
alem1105
marini.2122855@studenti.uniroma1.it

September 27, 2025

Indice

1. CIA - Key Objective / Requirements 3
    1.1. Levels of Impact 3
    1.2. Computer Security Challenges 3
    1.3. Un modello per la Sicurezza 4
2. Classificazione degli Attacchi 5
    2.1. Requisiti di Sicurezza 5
    2.2. Principi fondamentali di Design per la sicurezza 6
    2.3. Superfici e Alberi di Attacco 6
    2.4. Computer Security Strategies 7
3. Cryptographic Tools 8
    3.1. Symmetric Encryption 8
    3.2. Type of Symmetric Encryption 9
    3.3. Message Authentication 9
    3.4. Funzioni di Hash 10
    3.5. Crittografia a Chiave Pubblica (Asimmetrica) 11

1. CIA - Key Objective / Requirements

Sicurezza Informatica - NIST

Le misure che garantiscono **confidenzialità, integrità e disponibilità** di un sistema informatico nel sua interezza quindi hardware, software, firmware, le informazioni che memorizza, processa e comunica

Queste tre proprietà vengono indicate nella **triade CIA** e significano:
- **Confidenzialità**: Garantire che le informazioni private non vengano divulgate a persone non autorizzate e garantire anche che ogni utente abbia il controllo su quali dei suoi dati personali vengono memorizzati nel sistema o a chi vengono inviati. L'utente deve quindi essere in grado di gestire la sua **privacy**.
- **Integrità**: Assicurare che i dati o il sistema vengano modificati solo in modo autorizzato.
- **Disponibilità**: Garantire l'accesso affidabile alle informazioni e ai servizi, non devono esserci interruzioni del servizio.

La triade è in costante conflitto con usabilità e costi del sistema, il lavoro di un esperto di sicurezza è quindi quello di trovare il giusto compresso fra queste variabili.

Nei sistemi moderni, ai tre pilastri della triade, si aggiungono:
- **Autenticità**: La garanzia che un messaggio o un utente provengano effettivamente dalla fonte dichiarata.
- **Accountability**: La capacità di tracciare in modo univoco le azioni di un'entità, è utile per indagini forensi, isolare i guasti e la **nonrepudiation** ovvero quando qualcuno nega di aver compiuto un'azione.

1.1. Levels of Impact

Possiamo classificare l'impatto di eventuali violazioni:
- *Basso*: Danni minori o perdite finanziare limitate, il sistema continua comunque a funzionare anche se magari in modo ridotto.
- *Moderato*: Danni significativi alle capacità operative e perdite finanziare importanti.
- *Alto*: Il sistema non funziona più, danni finanziari enormi e ci sono rischi catastrofici o letali per le persone.

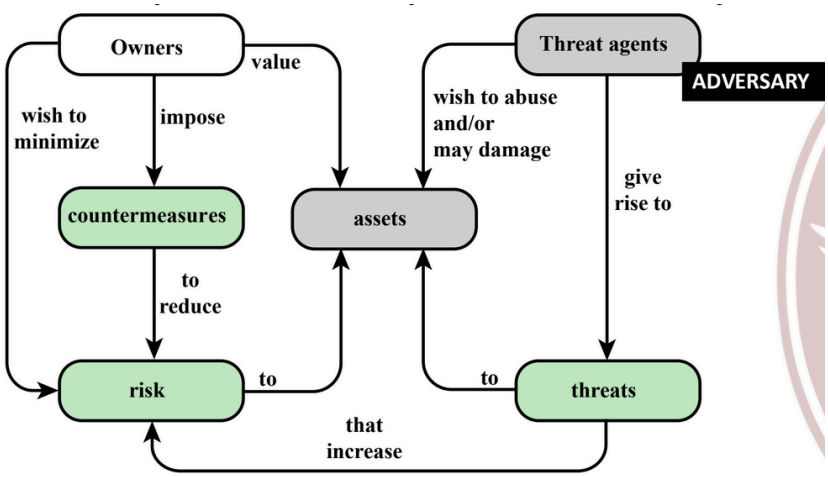
1.2. Computer Security Challenges

La sicurezza informatica presenta diverse sfide che la rendono complessa:
1. Non è semplice per chi si trova alla prime armi e spesso le procedura di sicurezza risultano controintuitive
2. Richiede una gestione di **informazioni riservate** e anche una decisione su dove posizionare le difese.
3. Dagli utenti viene vista come **un ostacolo** e dai manager come uno spreco di risorse fino a quando non subiscono un attacco.
4. Spesso viene aggiunta alla fine dell'implementazione di un sistema come un «ripensamento» invece di essere implementata dall'inizio.

Dilemma del Difensore

Chi difende deve eliminare **tutte le debolezze** di un sistema (praticamente impossibile), mentre a chi attacca ne basta una sola.

1.3. Un modello per la Sicurezza



Nel diagramma sono indicati:

- **Asset (Risorsa)**: È quello cha ha un valore nel sistema (Hardware, software, Dati ecc...)
- **Vulnerabilità (di un asset)**: Una debolezza nel sistema che può essere sfruttata. Ne esistono di diversi tipi:
  - **Corrupted**: Dati alterati
  - **Leaky**: Dati trapelati
  - **Unavailable**: Sistema non disponibile
- **Threat**: Una qualsiasi «situazione» che potrebbe causare danni di qualsiasi tipo.
- **Attack**: Quando un *Threat Agent* ovvero un avversario sfrutta attivamente una vulnerabilità.
- **Countermeasure**: Tecniche o dispositivi utili a prevenire gli attacchi o a ridurne gli effetti.

Possiamo anche calcolare il rischio grazie alla formula:

$$R = P \times D$$

Dove:

- P: Probabilità che l'evento si verifichi
- D: Impatto dell'evento

Nota

L'obiettivo delle contromisure **non è azzerare il rischio** dato che sarebbe troppo costoso o comunque impossibile, ma è quello di portarlo ad un livello accettabile definito **Rischio Residuo**.

2. Classificazione degli Attacchi

La prima distinzione che possiamo fare è quella della provenienza dell'attacco, o dall'interno (**insider**) o dall'esterno (**outsider**). Gli attacchi si dividono poi in:

- **Passivi**: L'attaccante osserva e raccoglie informazioni senza alterare il sistema, questi sono molto difficile da rilevare e l'unica difesa è la **prevenzione**. Ne esistono di due tipi:
  - *Release of Message Contents*: Leggere il contenuto di un messaggio
  - *Traffic Analysis*: Anche se il messaggio è cifrato, l'attaccante osserva chi sta parlando con chi e guardando anche la dimensione del messaggio, la frequeunza e altri dati può ricavare qualche informazione.
- **Attivi**: L'attaccante altera i dati o il funzionamento del sistema:
  - *Replay*: Catturare un dato e reinviarlo successivamente per ottenere accesso.
  - *Masquerade*: Fingersi qualcun altro
  - *Modification of Messages*: Alterare parti legittime di un messaggio o ritardarlo.
  - *Denial of Service (DoS)*: Interrompere il servizio di un sistema sovraccaricandolo di richieste.

Vediamo adesso le conseguenze e i tipi di attacco che le producono (secondo lo standard RFC4949):

- **Unauthorized Disclosure**: Vengono divulgati dati da persone non autorizzate, attacchi possibili:
  - *Exposure*: Esposizione diretta
  - *Interception*
  - *Inference*: Deduzione di dati da altri segnali
  - *Intrusion*
- **Deception**: Vengono fornite informazioni ad un utente non autorizzato
  - *Masquerade*: Fingersi un'altra persona
  - *Falsification*: Alterari i dati per ingannare
  - *Repudiation*: Negare di aver compiuto un'azione
- **Disruption**: Interruzione delle funzionalità del sistema:
  - *Incapacitation*: Disabilitare un componente
  - *Corruption*: Alterare i dati che permettono il funzionamento del sistema
  - *Obstruction*: Ostacolare i servizi
- **Usurpation**: Viene preso il controllo del sistema da parte di utenti non autorizzati.
  - *Misappropriation*: Prendere il controllo logico o fisico di una risorsa
  - *Misuse*: Utilizzare un componenete per svolgere operazioni dannose.

2.1. Requisiti di Sicurezza

Le contromisure alle vulnerabilità e minacce possono essere classificate e categorizzate in differenti modi, possiamo classificarle in funzione dei requisiti funzionali, il FIPS 200 le divide in:

- **Requisiti di Sicurezza Tecnici**: Ad esempio controllo degli accessi, autenticazione e integrità del sistema.
- **Funzionali**: Formazione del personale, valutazione dei rischi, sicurezza fisica
- **Misti**: Gestione della configurazione, risposta agli incidenti e protezione dei media.

2.2. Principi fondamentali di Design per la sicurezza

Questi principi sono fondamentali per chi progetta architetture di rete o sviluppa software:

- **Economy of Mechanism**: I sistemi di sicurezza devono essere piccoli e semplici, più codice scrivi e più bug introduci.
- **Fail-safe default**: L'accesso deve basarsi sui permessi e non sulle esclusioni, se qualcosa va storto il sistema deve bloccarsi in uno stato sicuro negando l'accesso.
- **Complete Mediation**: Ogni singolo accesso a una risorsa deve essere verificato senza fidarsi ciecamente della cache.
- **Open Design**: La sicurezza non deve basarsi sulla segretezza del codice ma sulla forza dell'algoritmo o delle chiavi.
- **Separation of Privilige**: Richiedere più condizioni per sblcocare un'azione (2FA).
- **Least Privilege**: Utenti e processi devono avere solo i permessi strettamente necessari per fare il loro lavoro.
- **Layering**: Usare ostacoli multipli e sovrapposti, se l'attaccante buca il firewall deve trovare altri blocchi.

Esempio Fail-Safe

Prendiamo questo esempio in pseudocodice:

```c
DWORD dwRet = IsAccessAllowed(...);
if (dwRet == ERROR_ACCESS_DENIED) {
  // Security check failed
} else {
  // Security check OK
}
```

Questo codice non va bene perché se per qualsiasi motivo la funzione `IsAccessAllowed()` fallisce e ritorna un dato non previsto noi stiamo garantendo l'accesso al sistema.

2.3. Superfici e Alberi di Attacco

Per difendere un sistema è utile capire come possiamo attaccarlo.

Definiamo come **Attack Surface** l'insieme di tutte le vulnerabilità raggiungibili in un sistema, si divide in:

- *Rete*: Porte aperte e server esposti
- *Software*: Bug nel codice
- *Umana*: Social engineering o errore del personale

Il nostro obiettivo è quello di ridurre il più possibile questa superficie.

Per analizzare le debolezze del sistema possiamo costrutire un **albero di attacco** ovvero un albero che rappresenta le minacce al nostro sistema:

- Il nodo radice è l'obiettivo dell'attaccante
- I nodi intermedi sono i sotto-obiettivi
- Le foglie sono gli attacchi che può fare per colpire un sotto-obiettivo.

I sotto-obiettivi possono essere in AND o OR fra di loro.

Questi alberi sono utilizzati dai team di sicurezza per capire qual è il percorso «più economico» o «più facile» per un attaccante, questo permette quindi di posizionare le difese (layering) dove sono più necessarie.

Esempio di albero:

2.4. Computer Security Strategies

Come si gestisce quindi la sicurezza a livello aziendale?

- **Specification / Policy (Cosa Fare?)**: Le regole che indicano cosa deve fare il sistema (es. lunghezza minima di una password). Queste devono bilanciare il costo della sicurezza col costo di un eventuale attacco.
- **Implementation / Mechanisms (Come Farlo?)**: Le tecnologie usate per prevenire, rilevare, rispondere e ruperare i dati.
- **Assurance ed Evaluation (Funziona?)**: Il livello di confidenza che i meccanismi implementati rispettino effettivamente le policy. Questo si ottiene tramite test, certificazioni e analisi.

3. Cryptographic Tools

3.1. Symmetric Encryption

È la tecnica di crittografia universale per garantire la **confidenzialità** dei dati. Il mittente e il destinatario, per scambiarsi messaggi, devono condividere la stessa chiave segreta K. Questa tecnica, per essere sicura, richiede:

- Un algoritmo crittografico forte.
- Che la condivisione della chiave sia avvenuta in modo sicuro.

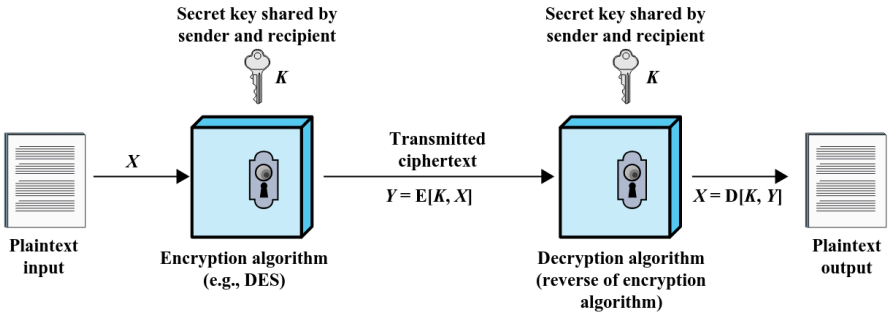


Figure 2.1 Simplified Model of Symmetric Encryption

I principali tipi di attacco a queste tecniche sono:

- **Crittoanalisi**: Sfruttare la natura matematica dell'algoritmo o se si conoscono alcune caratteristiche del testo non cifrato per dedurre la chiave.
- **Brute-Force**: Si provano tutte le chiavi possibili finché non si ottiene un testo sensato. In media è necessario provare almeno la metà di tutte le chiavi possibili.

La distrubuzione delle chiavi

Il vero problema di questa tecnica è la distribuzione della chiave, infatti, come fanno due persone a scambiarsi la chiave in modo sicuro se il canale su cui parlano non è protetto? Questo problema verrà risolto dalla **crittografia asimmetrica**.

Tra i principali algoritmi per la crittografia simmetrica troviamo:

- **DES (Data Encryption Standard)**: Utilizza blocchi da 64-bit e una chiave da 56-bit. Anche se è l'algoritmo più studiato la sua chiave da 56-bit lo rende molto debole contro i moderni processori di oggi.
- **3DES (Triple DES)**: Ripete il DES tre volte usando 2 o 3 chiavi diverse portando quindi la lunghezza effettiva della chiave a 112 o 168 bit. I problemi principali sono che è molto lento e usa ancora blocchi piccoli da 64-bit.
- **AES (Advanced Encryption Standard)**: Creato per sostituire il 3DES. È molto più efficiente, utilizza blocchi da 128-bit e chiavi da 128, 192 o 256-bit.

L'AES è più efficiente perchè utilizza un'architettura chiamata **Rijndael** che applica le sue trasformazioni all'intero blocco di 128-bit simultaneamente organizzandolo in una matrice, questa struttura è ottimizzata per eseguire calcoli paralleli ed è perfetta per come lavorano i

processori moderni. Questo rende l'algoritmo più veloce rispetto ad un DES o 3DES che invece utilizzano la **Rete di Feistel**, questa divide il blocco a metà e, a ogni passaggio, cripta solo una metà alla volta incrociandola con l'altra.

Tempi medi richiesti per rompere un algoritmo:

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/s	Time Required at 10^{13} decryptions/s
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	2^{55} ns = 1.125 years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	2^{127} ns = 5.3×10^{21} years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	2^{167} ns = 5.8×10^{33} years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	2^{191} ns = 9.8×10^{40} years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	2^{255} ns = 1.8×10^{60} years	1.8×10^{56} years

Oggi però i **computer quantistici** minacciano gravemente gli algoritmi con chiave pubblica, infatti questi si basano su problemi matematici come la fattorizzazione che un QC può risolvere facilmente. Oggi si cercano quindi algoritmi **Quantum - resistant**.

3.2. Type of Symmetric Encryption

Ovviamente dobbiamo cifrare file più grandi di un qualsiasi blocco, per questo esistono diverse tipologie di cifratura:

- **Block Cipher**: Cifra il testo elaborando un blocco alla volta, può riutilizzare le chiavi ed è il più comune.
- **Stream Cipher**: Cifra il flusso di dati continuamente, byte alla volta, generando un flusso pseudocasuale, è molto più veloce e usa meno codice.

Un metodo di cifratura che utilizza il cifrario a blocchi è il **ECB**:

- **Electronic Codebook (ECB)**: È la modalità base per usare i cifrari a blocchi su messaggi lunghi. Ogni blocco viene cifrato singolarmente con la stessa chiave, il problema è che se il testo in chiaro ha delle regolarità allora queste appariranno anche nel testo cifrato.

3.3. Message Authentication

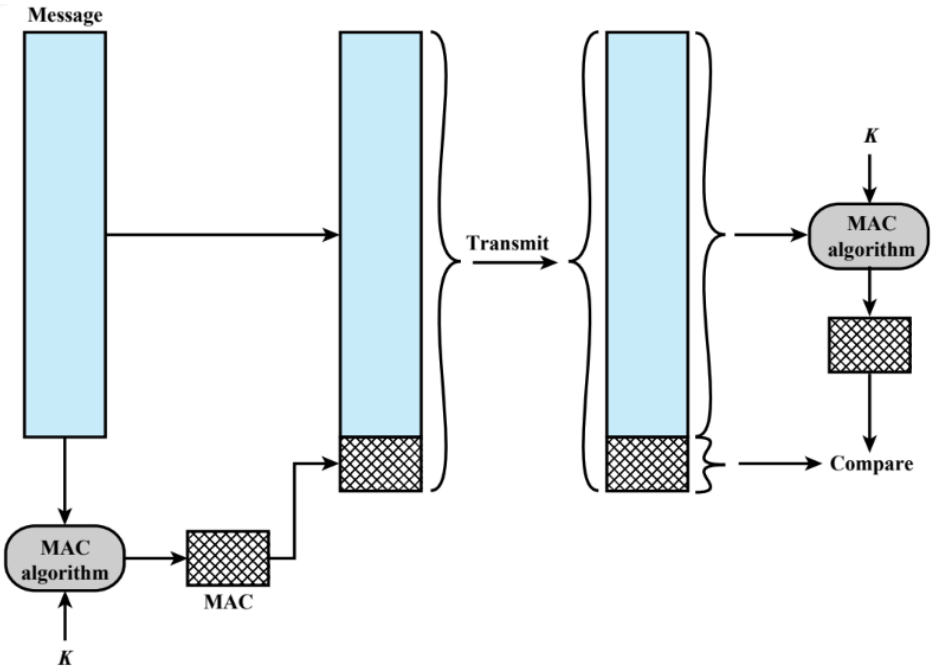
La cifratura dei messaggi protegge dalla lettura non autorizzata ma non dalla manomissione del messaggio ovvero gli attacchi attivi. Con l'autenticazione dei messaggi garantiamo:

- Il contenuto non è stato alterato
- Il messaggio proviene da una fonte legittima

Per garantire che il messaggio non sia stato alterato si utilizza un **MAC (Message Authentication Code)**. Attraverso la chiave K e il messaggio M si calcola un «tag»:

$$\text{MAC} = F(K, M)$$

Se il ricevente, ricalcolando il MAC con la stessa chiave segreta, ottiene lo stesso risultato, ha la garnazie che il messaggio non sia stato modificato.



Ovviamente, anche qui, la chiave deve essere in possesso solo delle persone fidate.

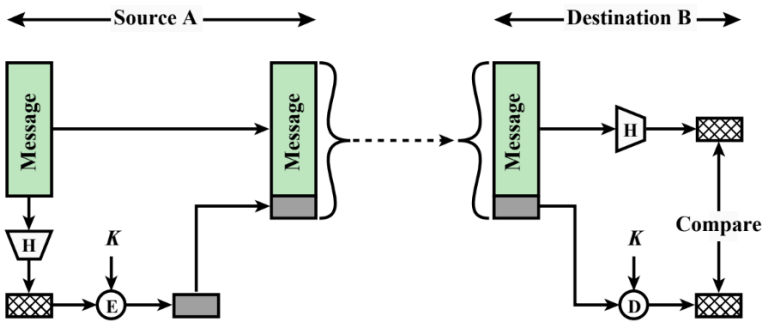
Possiamo inoltre distinguere due casi:
- **Without Confidentiality**: Il messaggio è autenticato ma non cifrato, viene soltanto aggiunto il TAG al messaggio.
- **With Confidentiality**: Il messaggio è autenticato e cifrato, quindi anche il tag viene cifrato.

Quando è utile avere autenticazione ma non confidenzialità?
- Sistemi broadcast o molto carichi
- Protocolli di rete, l'header del pacchetto IP non deve essere alterato ma non abbiamo bisogno di nasconderlo dato che i router devono leggerlo.

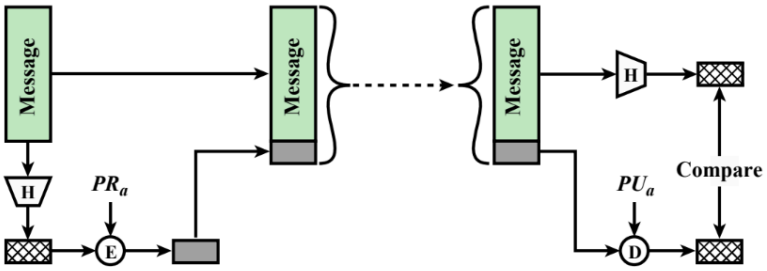
3.4. Funzioni di Hash

Una funzione Hash $H(M)$ è un'alternativa al *MAC* che non richiede una chiave in input, prende un messaggio di dimensione variabile e restituisce un'impronta di dimensione fissa (*digest*). Per essere sicura, una funzione Hash deve possedere:
- **One-Way or preimage resistant**: Dato un hash, deve essere impossibile risalire al messaggio originale.
- **Second preimage or Weak Collision resistant**: Dato un messaggio X, deve essere impossibile trovare un altro messaggio Y diverso che produca lo stesso hash.
- **Strong collision resistant**: Deve essere impossibile trovare qualsiasi coppia di messaggi X e Y che abbiano lo stesso Hash

Possiamo utilizzare l'hash a scopo di autenticazione in diversi modi:
- **One-Way Hash function + Symmetric Encryption**

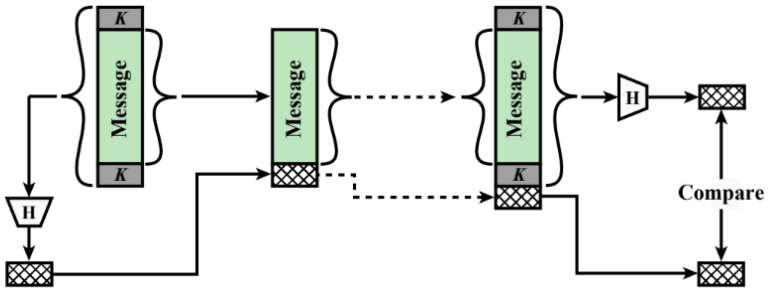


- **One-Way Hash function + Public Key Encryption**



Qui il vantaggio è che non è richiesta la distribuzione delle chiavi in modo sicuro.

- **One-Way Hash function + Secret Value (no encryption)**



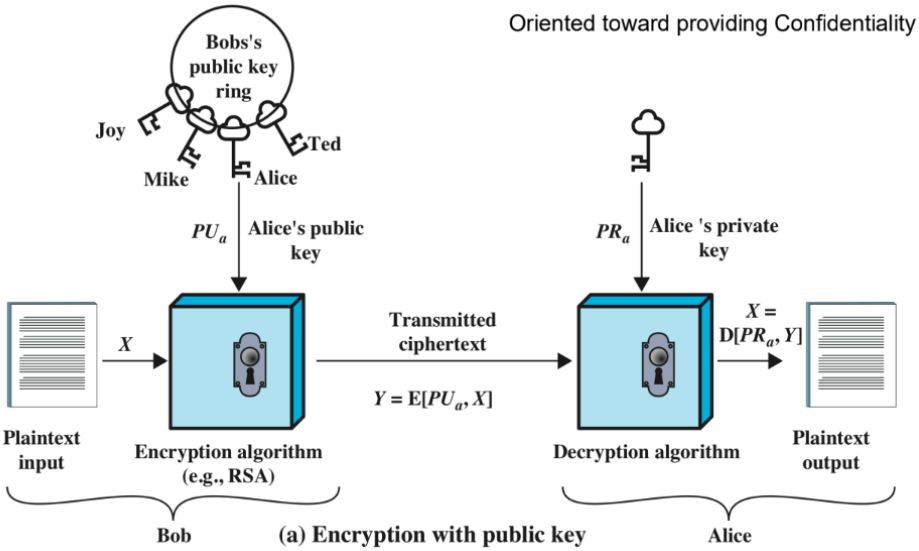
Non abbiamo bisogno di crittografia, risparmiando quindi computazione. Chi invia il messaggio deve calcolare $\mathrm{MD}_M = H(K \mid M \mid K)$ e inviare $\mathrm{MD}_M \mid M$, a questo punto il destinatario, che conosce K, deve calcolare $H(K \mid M \mid K)$ e verificare MD_M

Lo standard più utilizzato è lo **SHA** anche se oggi è considerato debole e si utilizza **SHA-2 o SHA-3**.

3.5. Crittografia a Chiave Pubblica (Asimmetrica)

È un algoritmo di crittografia proposto da **Diffie e Hellman** nel 1976, risolve principalmente il problema della distribuzione della chiave. Viene utilizzata una coppia di chiavi collegate fra loro matematicamente, una **pubblica** e una **privata**. Queste garantiscono:

- **Confidenzialità**: Io cifro con la chiave pubblica di Bob e solo Bob con la sua chiave privata sarà in grado di decifrare il messaggio.
- **Autenticazione**: Cifro l'hash con la mia chiave privata, adesso tutti possono decifrarlo con la mia chiave pubblica garantendo quindi che sono stato io a generare quel messaggio.



Tra i principali algoritmi a chiave asimmetrica troviamo:

- **RSA**: Utilizzato per la cifratura, firme digitali e scambio di chiavi.
- **Diffie-Hellman**: Serve *solo* per permettere a due entità di generare una chiave simmetrica condivisa attraverso un canale sicuro.
- **DSS (Digital Signature Standard)**: Garantisce firma digitale con SHA-1
- **ECC (Elliptic Curve)**: Sicuro quanto RSA ma usa chiavi più corte, è quindi più veloce e leggero.

Algorithm	Digital Signature	Symmetric Key Distribution	Encryption of Secret Keys
RSA	Yes	Yes	Yes
Diffie-Hellman	No	Yes	No
DSS	Yes	No	No
Elliptic Curve	Yes	Yes	Yes

Vero vantaggio della chiave asimmetrica

La crittografia asimmetrica è **computazionalmente lentissima** rispetto a quella simmetrica, nel mondo reale infatti viene si utilizzata ma soltanto all'inizio delle comunicazioni per scambiare in modo sicuro una chiave segreta temporanea simmetrica, dopo questo scambio il resto della comunicazione avviene con cifratura simmetrica tramite AES.