1. Introduzione alla Crittografia

La crittografia è la scienza che studia come proteggere le informazioni tramite tecniche di cifratura, rendendo i dati incomprensibili a chi non possiede le chiavi appropriate per decifrare.

1.1 Concetti fondamentali

- **Testo in chiaro** (plaintext): informazione originale leggibile
- **Testo cifrato** (ciphertext): informazione cifrata, non leggibile
- **Cifratura** (encryption): processo di trasformazione da testo in chiaro a testo cifrato
- **Decifratura** (decryption): processo inverso che ripristina il testo originale
- **Algoritmo di cifratura**: regole matematiche per la cifratura/decifratura
- **Chiave**: parametro dell'algoritmo che determina il risultato specifico della cifratura

1.2 Obiettivi della crittografia

- **Confidenzialità**: proteggere le informazioni da accessi non autorizzati
- **Integrità**: garantire che le informazioni non siano state alterate
- **Autenticità**: verificare l'identità della fonte dell'informazione
- **Non ripudio**: impedire a un mittente di negare l'invio di un messaggio

1.3 Principi di Kerckhoffs

Principi fondamentali della crittografia moderna:

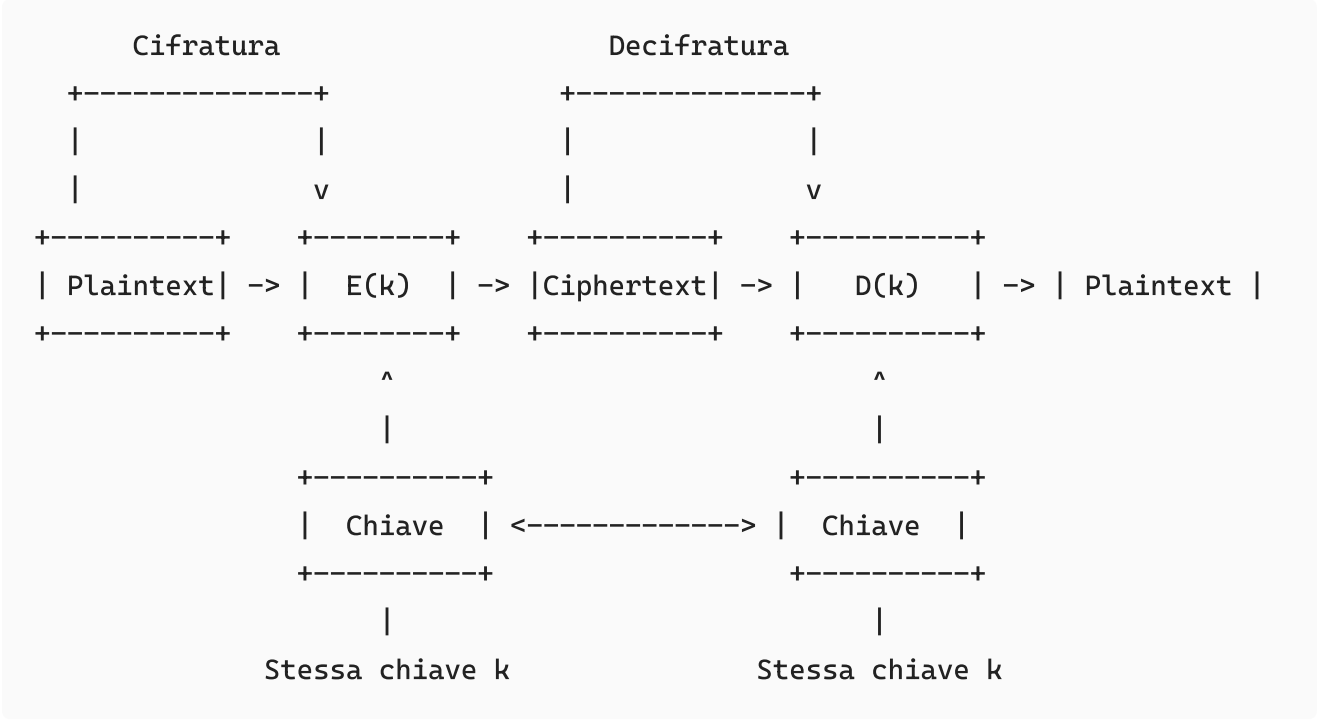
1. Il sistema deve essere **praticamente, se non matematicamente, indecifrabile**
2. Il sistema non deve richiedere la segretezza e deve poter cadere nelle mani del nemico
3. La **chiave deve essere memorizzabile senza annotazioni** e può essere cambiata a volontà
4. Il sistema deve essere applicabile alle **comunicazioni telegrafiche**
5. Il sistema deve essere **portatile** e non richiedere più persone
6. Il sistema deve essere **facile da utilizzare**, non richiedendo complesse regole o concentrazione mentale

2. Crittografia Simmetrica

2.1 Definizione e caratteristiche

- Usa la **stessa chiave** per cifrare e decifrare
- Veloce ed efficiente
- Richiede uno scambio sicuro della chiave

- Algoritmi più leggeri in termini di risorse



2.2 Algoritmi principali

2.2.1 DES (Data Encryption Standard)

- Sviluppato negli anni '70 da IBM, adottato come standard dal governo USA
- Chiave di 56 bit (considerata molto debole oggi)
- Blocchi di 64 bit
- Ora considerato obsoleto e vulnerabile a brute force
- Sostituito da algoritmi più recenti e sicuri

2.2.2 3DES (Triple DES)

- Evoluzione di DES
- Applica DES tre volte con chiavi diverse
- Chiave effettiva di 168 bit (3 x 56)
- Più sicuro di DES ma lento e ormai deprecato
- Superato da AES in termini di sicurezza ed efficienza

2.2.3 AES (Advanced Encryption Standard)

- Standard attuale, selezionato dal NIST nel 2001
- Algoritmo Rijndael
- Chiavi di 128, 192 o 256 bit
- Blocchi di 128 bit
- Bilancia sicurezza e performance

- Ampiamente utilizzato (HTTPS, VPN, archivi cifrati)

2.4 Vantaggi e svantaggi della crittografia simmetrica

2.4.1 Vantaggi

- Velocità di esecuzione
- Efficienza per grandi volumi di dati
- Minore complessità computazionale
- Chiavi più corte per lo stesso livello di sicurezza

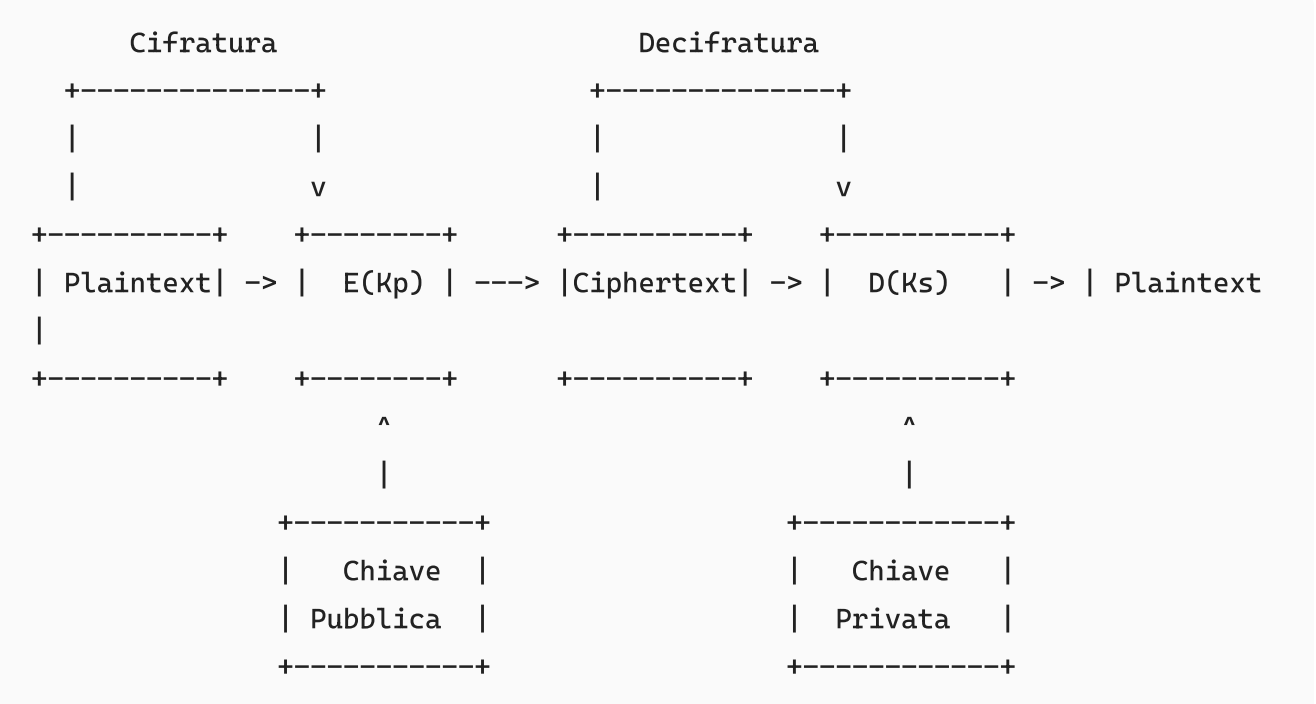
2.4.2 Svantaggi

- Problema della distribuzione sicura della chiave
- Scalabilità limitata: n*(n-1)/2 chiavi per n utenti
- Mancanza di supporto nativo per firma digitale
- Difficoltà nella gestione delle chiavi in sistemi grandi

3. Crittografia Asimmetrica

3.1 Definizione e caratteristiche

- Utilizza una **coppia di chiavi**: pubblica e privata
- La chiave pubblica (per cifrare) può essere condivisa
- La chiave privata (per decifrare) resta segreta
- Risolve il problema della distribuzione delle chiavi
- Computazionalmente più intensiva della simmetrica



3.2 Algoritmi principali

3.2.1 RSA (Rivest-Shamir-Adleman)

- Uno dei primi e più diffusi algoritmi a chiave pubblica
- Sicurezza basata sulla fattorizzazione di numeri primi grandi
- Chiavi tipicamente da 2048 o 4096 bit
- Utilizzato per cifratura e firma digitale
- Funzionamento sintetico:
  1. Generazione di due numeri primi grandi (p, q)
  2. Calcolo di n = p*q
  3. Calcolo di φ(n) = (p-1)*(q-1)
  4. Scelta di un esponente e coprimo con φ(n)
  5. Calcolo di d tale che e*d ≡ 1 mod φ(n)
  6. Chiave pubblica: (n, e)
  7. Chiave privata: (n, d)

Esempio semplificato di RSA

1. **Generazione chiavi**:
   - Scegliere p=3, q=11
   - n = p×q = 33
   - φ(n) = (p-1)×(q-1) = 2×10 = 20
   - e = 7 (coprimo con 20)
   - d = 3 (7×3 ≡ 1 mod 20)
   - Chiave pubblica: (33, 7)
   - Chiave privata: (33, 3)
2. **Cifratura** (c = m^e mod n):
   - Messaggio m = 2
   - c = 2^7 mod 33 = 128 mod 33 = 29
3. **Decifratura** (m = c^d mod n):
   - Cifrato c = 29
   - m = 29^3 mod 33 = 24389 mod 33 = 2

3.2.2 Diffie-Hellman

- Protocollo per lo scambio sicuro di chiavi su canale insicuro
- Non per cifratura/decifratura diretta
- Permette a due parti di generare una chiave segreta condivisa
- Sicurezza basata sul problema del logaritmo discreto
- Usato in TLS/SSL, IPSec, SSH

Procedura Diffie-Hellman

1. Alice e Bob concordano su due numeri primi pubblici g e p
2. Alice sceglie un numero casuale a, calcola A = g^a mod p e invia A a Bob
3. Bob sceglie un numero casuale b, calcola B = g^b mod p e invia B ad Alice
4. Alice calcola la chiave K = B^a mod p
5. Bob calcola la chiave K = A^b mod p
6. Entrambi ottengono la stessa chiave segreta K = g^(a*b) mod p

3.3 Vantaggi e svantaggi della crittografia asimmetrica

3.3.1 Vantaggi

- Risolve il problema della distribuzione delle chiavi
- Supporta la firma digitale
- Migliore scalabilità (solo 2n chiavi per n utenti)
- Supporta il non-ripudio

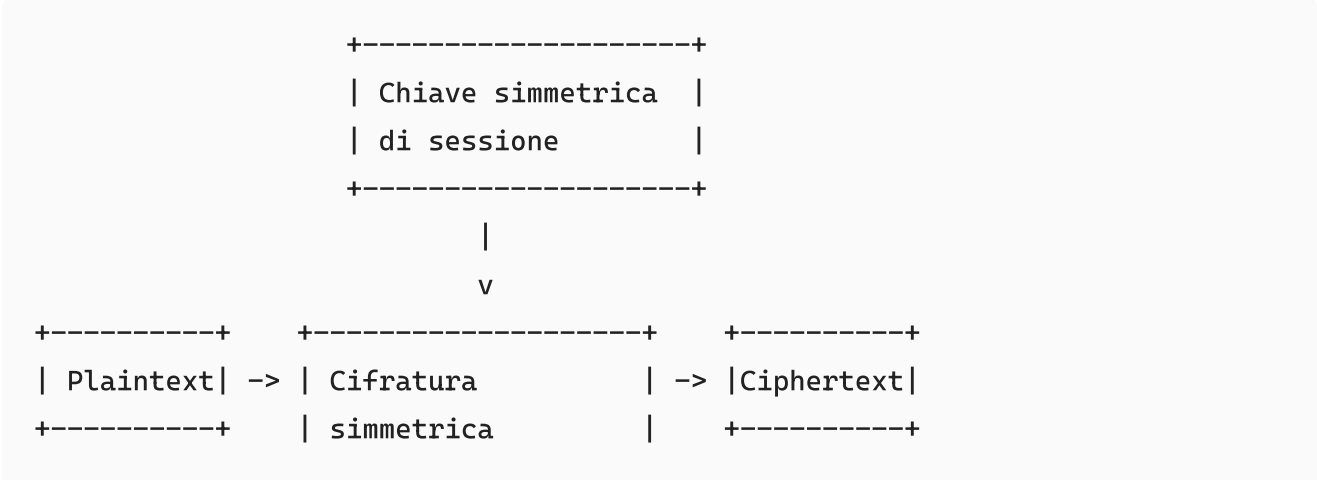
3.3.2 Svantaggi

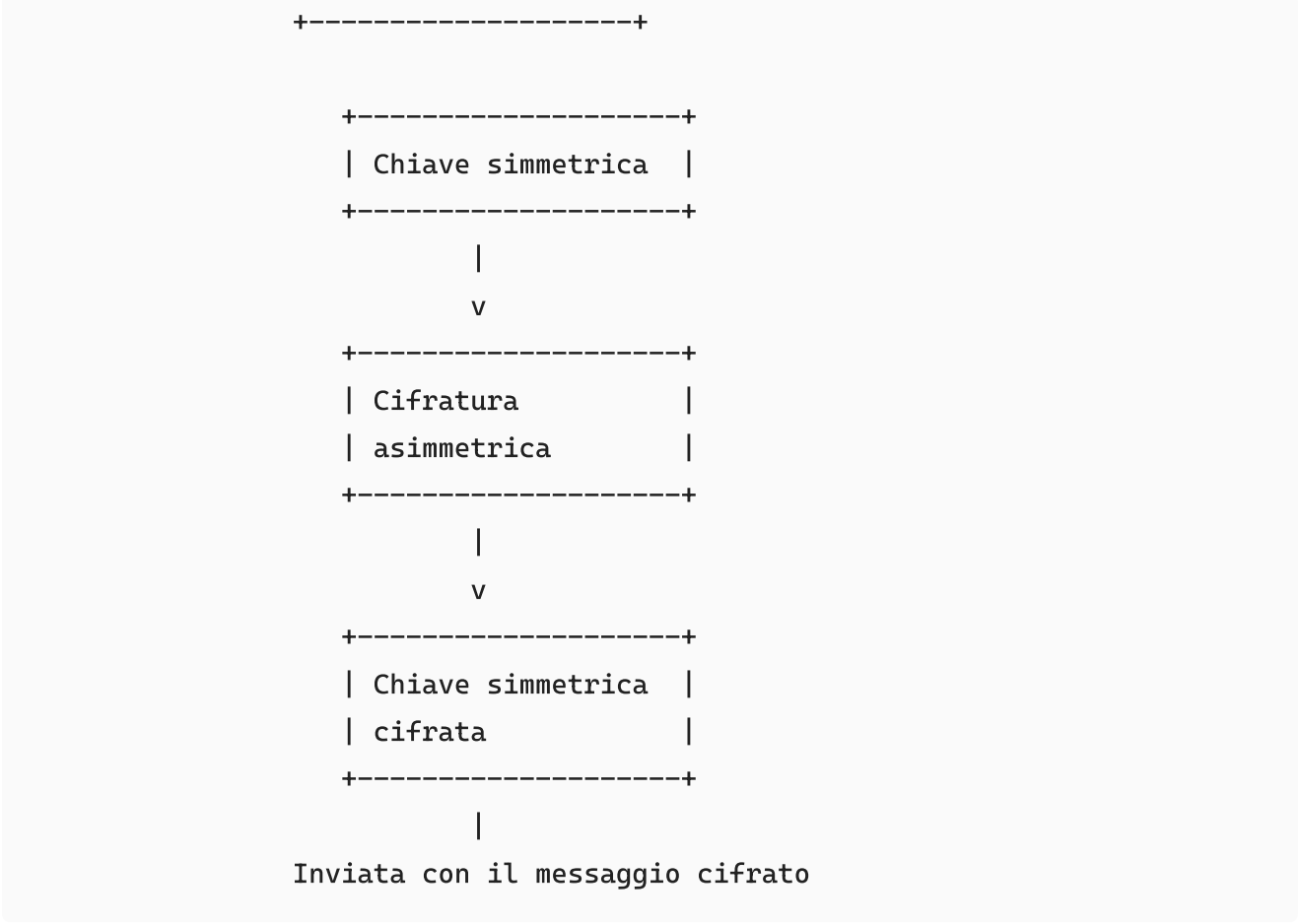
- Molto più lenta della crittografia simmetrica (100-1000 volte)
- Richiede chiavi più lunghe
- Maggiore consumo di risorse computazionali
- Non adatta per cifrare grandi volumi di dati

4. Crittografia Ibrida

4.1 Concetto e motivazione

- Combina i vantaggi di crittografia simmetrica e asimmetrica
- Usa crittografia asimmetrica per scambiare una chiave di sessione simmetrica
- Usa crittografia simmetrica per cifrare i dati effettivi
- Ottimizza sicurezza e performance





4.2 Applicazioni comuni

- **TLS/SSL**: Usato in HTTPS
- **PGP/GPG**: Email sicura
- **VPN**: Comunicazioni di rete sicure
- **Messaggistica sicura**: Signal, WhatsApp

4.3 Processo tipico di crittografia ibrida

1. Il mittente genera una chiave simmetrica casuale (chiave di sessione)
2. Il mittente cifra il messaggio con la chiave simmetrica
3. Il mittente cifra la chiave simmetrica con la chiave pubblica del destinatario
4. Il mittente invia il messaggio cifrato e la chiave simmetrica cifrata
5. Il destinatario decifra la chiave simmetrica con la propria chiave privata
6. Il destinatario usa la chiave simmetrica per decifrare il messaggio

5. Funzioni di Hash Crittografiche

5.1 Definizione e proprietà

- Trasformano input di lunghezza arbitraria in output di lunghezza fissa
- Proprietà fondamentali:
  - **One-way** (non invertibile): impossibile risalire all'input dall'output

- **Deterministica**: stesso input produce sempre stesso output
- **Effetto valanga**: piccole modifiche nell'input causano grandi cambiamenti nell'output
- **Resistenza alle collisioni**: difficile trovare due input diversi con stesso output
- **Resistenza alle collisioni di secondo tipo**: dato un input, difficile trovare un altro input con stesso hash

5.2 Algoritmi principali

5.2.1 MD5 (Message Digest Algorithm 5)

- Output di 128 bit
- Sviluppato da Ron Rivest nel 1991
- Oggi considerato obsoleto e vulnerabile
- Collisioni trovate facilmente
- Non usare per sicurezza, solo per checksum non critici

5.2.2 SHA-1 (Secure Hash Algorithm 1)

- Output di 160 bit
- Sviluppato dalla NSA
- Considerato non sicuro dal 2005
- Collisione pratica dimostrata nel 2017
- Deprecato per usi di sicurezza

5.2.3 SHA-2 (SHA-256, SHA-512)

- Famiglia di funzioni hash
- Output da 224 a 512 bit
- Standard attuale per molte applicazioni
- SHA-256: 32 byte, comune in blockchain/Bitcoin
- SHA-512: 64 byte, maggiore sicurezza

5.3 Utilizzi comuni

- **Verifica dell'integrità dei file**: verifica che un file non sia stato alterato
- **Memorizzazione sicura delle password**: mai salvare password in chiaro
- **Firme digitali**: hashing del documento prima della firma
- **Blockchain e cryptocurrencies**: Proof of Work, merkle trees
- **Certificati digitali**: impronta dell'identità digitale
- **Protocolli di autenticazione**: challenge-response
- **Identificazione univoca dei dati**: deduplica, content-addressable storage

5.4 Salt e Pepper

Tecniche per migliorare la sicurezza dell'hashing delle password:

5.4.1 Salt

- Valore casuale unico aggiunto alla password prima dell'hashing
- Protegge contro attacchi con tabelle rainbow e precompilate
- Il salt è pubblico, salvato insieme all'hash
- Esempio: `hash = SHA256(password + salt)`
- Ogni utente ha un salt diverso

5.4.2 Pepper

- Valore segreto aggiunto alla password prima dell'hashing
- Non memorizzato insieme all'hash (conservato separatamente)
- Aggiunge un ulteriore livello di sicurezza
- Esempio: `hash = SHA256(password + salt + pepper)`
- Lo stesso pepper usato per tutti gli utenti

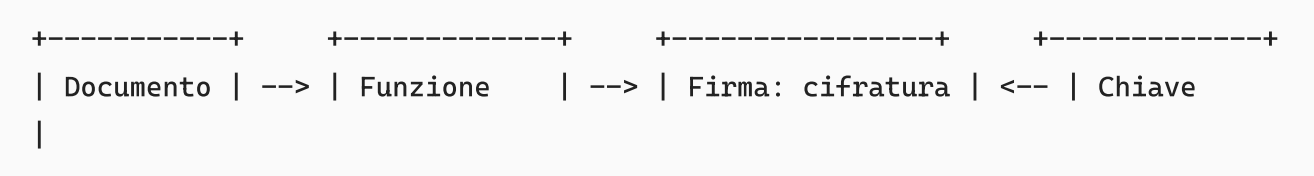
6. Firma Digitale

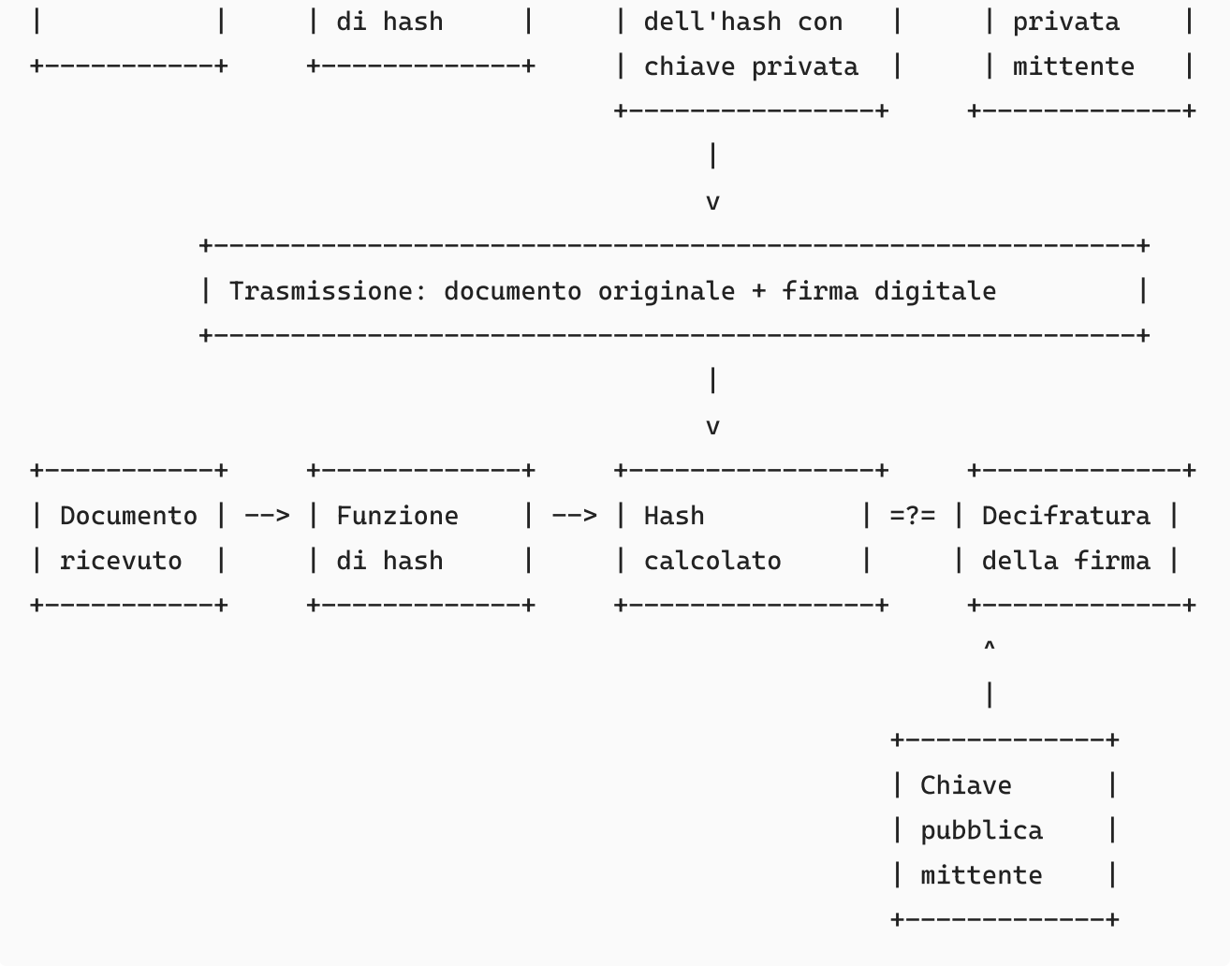
6.1 Concetto e scopo

- **Definizione**: Meccanismo crittografico che assicura autenticità, integrità e non ripudio
- **Analogia**: Equivalente elettronico di una firma autografa
- **Scopo**:
  - Verificare l'identità del mittente (autenticazione)
  - Garantire che il documento non sia stato alterato (integrità)
  - Impedire al mittente di negare di aver firmato (non ripudio)

6.2 Funzionamento

1. Il mittente crea un hash del documento (impronta digitale)
2. Il mittente cifra l'hash con la propria chiave privata (firma)
3. Il documento e la firma vengono inviati al destinatario
4. Il destinatario decifra la firma con la chiave pubblica del mittente
5. Il destinatario calcola l'hash del documento ricevuto
6. Se l'hash calcolato corrisponde all'hash decifrato, la firma è valida





6.4 Applicazioni pratiche

6.4.1 PEC (Posta Elettronica Certificata)

- Sistema di posta elettronica che garantisce valore legale alle comunicazioni
- Utilizza firme digitali per autenticare mittente, destinatario e contenuto
- Fornisce ricevute di accettazione e consegna
- Utilizza un'infrastruttura di gestione chiavi e certificati

6.4.2 Documenti PDF firmati

- Standard PAdES (PDF Advanced Electronic Signatures)
- Firma incorporata nel documento
- Verifica automatica con lettori PDF
- Possibilità di firme multiple e controfirme
- Timestamp per validità a lungo termine

6.4.3 Firma del codice

- Sicurezza per distribuzione software
- Protegge da manomissioni e malware

- Verifica dell'origine del software
- Usata per app, driver, aggiornamenti

6.4.4 Blockchain e smart contract

- Transazioni firmate digitalmente
- Immutabilità e non ripudio
- Esecuzione automatica di contratti
- Base tecnologica per cryptocurrencies

7. Gestione delle Chiavi e PKI

7.1 Infrastruttura a chiave pubblica (PKI)

- **Definizione**: Sistema che crea, gestisce, distribuisce, usa, memorizza e revoca certificati digitali
- **Componenti**:
  - CA (Certificate Authority): emette e firma certificati
  - RA (Registration Authority): verifica l'identità dei richiedenti
  - Archivi di certificati: dove i certificati sono memorizzati
  - Software di gestione: per emissione, revoca e gestione

7.2 Certificati digitali

- **Definizione**: Documento elettronico che collega una chiave pubblica a un'identità
- **Standard**: X.509
- **Contenuto tipico**:
  - Chiave pubblica del soggetto
  - Informazioni sul soggetto (nome, organizzazione)
  - Periodo di validità
  - Autorità di certificazione emittente
  - Firma digitale della CA
  - Algoritmi supportati
  - Limitazioni d'uso

7.2.1 Tipi di certificati

- **DV (Domain Validation)**: verifica solo il controllo del dominio
- **OV (Organization Validation)**: verifica anche l'organizzazione
- **EV (Extended Validation)**: verifica approfondita dell'identità legale
- **Self-signed**: firmati dal proprio creatore (non da CA esterna)
- **Code signing**: specifici per firma del codice

- **TLS/SSL**: per server web
- **Client**: per autenticazione client

7.3 Certificate Authority (CA)

- **Funzione**: Emette certificati firmandoli con la propria chiave privata
- **Gerarchia**:
  - Root CA: auto-firmata, offline, massima sicurezza
  - Intermediate CA: firmate dalla Root CA, emettono certificati utente
  - Cross-certification: trust tra CA diverse

7.3.1 CA pubbliche vs private

- **Pubbliche**:
  - Riconosciute da browser e sistemi operativi
  - Per servizi pubblici su Internet
  - Esempi: DigiCert, Let's Encrypt, Comodo
- **Private**:
  - Interne all'organizzazione
  - Per servizi interni
  - Non riconosciute pubblicamente