Università degli Studi di Modena e Reggio Emilia

FACOLTÀ DI INGEGNERIA
Corso di Laurea in Ingegneria Informatica

Prova Finale

Non so ancora il titolo

Candidato:
Luca Lumetti
Matricola 118447

Relatore:
Prof. Maurizio Casoni

Correlatore:
Dott. Martin Klapez

Anno Accademico 2019-2020

Indice

1 Introduzione **4**

2 Schemi a chiave privata **5**

2.1 Confidenzialità 6

2.1.1 Perfectly Secret 6

2.1.2 Computationally Secret 6

2.2 Integrità 7

2.2.1 Funzioni di Hash 7

2.2.2 Birthday Attack 8

2.3 Autenticazione 8

3 Conclusione **9**

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Introduzione

Capitolo 2

Schemi a chiave privata

Gli schemi crittografici a chiave privata (anche detti simmetrici) sono stati i primi schemi ad essere creati e sono tutt'ora molto usati. Sono caratterizzati dalla presenza di una sola chiave che viene utilizzata sia per criptare un messaggio sia per decriptarlo.
Più precisamente, uno schema crittografico a chiave privata è una tupla $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ dove:

- $\mathsf{Gen}(\cdot)$**:** è un algoritmo randomizzato che a fronte di un input 1^n genera una chiave k tale che $|k| \geq n$. E' quindi definito nel seguente modo: $k \leftarrow \mathsf{Gen}(1^n)$. Il parametro n viene detto *parametro di sicurezza*

- $\mathsf{Enc}(\cdot)$**:** è un algoritmo polinomiale che puo' essere di tipo deterministico o probabilistico. Ha come input una chiave k e un messaggio m e restituisce un messaggio cifrato c. Potendo essere un algoritmo probabilistico, scriveremo: $c \leftarrow \mathsf{Enc}_k(m)$.

- $\mathsf{Dec}(\cdot)$**:** è un algoritmo polinomiale deterministico che a ha come input una chiave k e un messaggio cifrato c e restituisce un messaggio m. Essendo un algoritmo deterministico scriveremo: $m := \mathsf{Dec}_k(c)$.

Si definisce inoltre con $\mathcal{M}$ lo spazio di tutti i messaggi che possono essere inviati $\mathcal{K}$ lo spazio di tutte le chiavi. Questi due insiemi variano dipendentemente da come vengono implementate 3 funzioni dello schema.
Inoltre, per ogni n, per ogni $k \in \mathcal{K}$ e per ogni $m \in \mathcal{M}$ deve valere la seguente relazione:

$$m = \mathsf{Dec}_k(\mathsf{Enc}_k(m))$$

2.1 Confidenzialità

La confidenzialità assicura che in una comunicazione, un *eavesdropper*, ovvero una persona esterna che intercetta i messaggi scambiati tra le due parti di una comunicazione, non possa ottenere alcune informazione utile. In altre parole significa che il testo cifrato non fa trasparire nessuna informazione riguardante il messaggio originale o la chiave.

2.1.1 Perfectly Secret

Dalla definizione di confidenzialità deriva direttamente quella di schema crittografico "perfectly secret". Riprendendo la notazione utilizzata per definire uno schema crittografico a chiave privata, definiamo con $\mathsf{Pr}[M = m]$ la probabilità che il messaggio m venga trasmesso e con $\mathsf{Pr}[C = c]$ la probabilità che $\mathsf{Enc}_k(m)$ sia c. Allora uno schema è "perfectly secret" se:

$$\mathsf{Pr}[M = m|C = c] = \mathsf{Pr}[M = m]$$

Ovvero la distribuzione dei messaggi in chiaro è indipendente dalla distrubuzione dei messaggi cifrati.
Questa definizione però ha delle conseguenze sulla cardinalità dello spazio dei messaggi $\mathcal{M}$ e delle chiavi $\mathcal{K}$ in quanto $|\mathcal{K}| \geq |\mathcal{M}|$ è condizione necessaria affinchè uno schema sia "perfectly secret". per questo motivo nella pratica non è una caratteristica molto usata.

2.1.2 Computationally Secret

Gli schemi di crittografia moderni e attualmente utilizzati non soddisfano la definizione di "perfectly secret", ovvero possono essere violati se si dispone di sufficente tempo e potenza computazionale, ma la quantità di tempo necessaria è dell'ordine di qualche decina di anni o più anche utilizzando i più grandi supercomputer oggi esistenti, in questo caso si parla quindi di schema "computationally secure". Questa definizione include due rilassamenti rispetto a quella di "perfectly secret":

- L'avversario è *efficiente*, ovvero un avversario che utilizza un algoritmo probabilistico polinomiale rispetto al valore n dello schema
- L'avversario ha sempre una probabilità di successo, ma sufficentemente bassa da considerarla trascurabile in n

Una funzione f viene considerata trascurabile (negligible) se per ogni polinomio $p(\cdot)$, esiste un valore N per cui:

$$f(n) < \frac{1}{p(n)}$$

E si incherà con: $f(n) = \mathsf{negl}(n)$.
Introduciamo ora un esperimento eseguito su uno schema Π a chiave privata (PrivK) in presenza di un avversario intercettatore (eavesdropper) $\mathcal{A}$ di tipo probabilistico polinomiale. L'esperimento viene eseguito su due messaggi m_0 e m_1 e solo uno dei due, scelto casualmente, viene cifrato.
Sia b un numero casuale tra 0 e 1, si ha: $c \leftarrow \mathsf{Enc}_k(m_b)$.
L'avversario $\mathcal{A}$ conosce m_0, m_1 e c. L'esperimento, che chiameremo $\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}$, ha valore 1 se $\mathcal{A}$ indovina il valore di b, altrimenti 0
Grazie a questo esperimento, è possibile definire uno schema "computationally secret" in presenza di un intercettatore, nel seguente modo:

$$\mathsf{Pr}[\mathsf{PrivK}^{\mathsf{eav}}_{\mathcal{A},\Pi}(n) = 1] \leq \frac{1}{2} + \mathsf{negl}(n)$$

E' importante notare che $\mathcal{A}$ conosce oltre a c anche m_0, m_1, ma comunque non è in grado di determinare quale dei due messaggi sia stato codificato.

2.2 Integrità

L'integrità assicura che un messaggio non venga modificato da una terza parte che si inserisce nella comunicazione. E' importante notare che in alcuni casi, un messaggio può essere modificato anche senza essere decodificato, ad esempio togliendo parti del ciphertext, scambiandone l'ordine oppure aggiungendo parti copiandone alcune dallo stesso ciphertext. Per assicurare l'integrità di uno schema a chiave privata si fa uso delle funzioni di *Hash*

2.2.1 Funzioni di Hash

Le funzioni di Hash sono funzioni che *comprimono* un messaggio di lunghezza arbitraria in una stringa generalmente più corta.
Più precisamente, una funzione di hash è una coppia $(\mathsf{Gen}, \mathsf{H})$ tale che:

- Gen è un algoritmo probabilistico e polinomiale che a fronte di un input 1^n restituisce come output una chiave s
- H è un algoritmo polinomiale deterministico con input una stringa $x \in \{0,1\}^*$ e output una stringa $\mathsf{H}^s(x) \in \{0,1\}^{l(n)}$ con l polinomio in n.

Dalla definizione emerge immediatamente un problema comune a tutte le funzioni di hash: l'esistenza di collisioni. Una collisione avviene quando due stringhe x_1 e x_2 diverse tra loro hanno lo stesso hash $\mathsf{H}^s(x_1) = \mathsf{H}^s(x_2)$. Essendo lo spazio di input illimitato a differenza di quello di output che è limitato, è ovvio che il numero di collisioni sia a sua volta infinito. Per questo motivo, un'importante caratteristica che le funzioni di hash devono avere è quella della resistenza a collisioni, ovvero rendere difficile il trovarle.
Nella pratica le funzioni di hash non utilizzano una chiave e quindi vengono definite solamente delle funzioni di hash In questo caso, volendo solo assicurare l'integrità del messaggio e nient'altro, il mittente del messaggio può inviare la coppia $(m, \mathsf{H}(m))$. In questo modo, il ricevente può verificare l'integrità del messaggio calcolando a sua volta $\mathsf{H}(m)$ e confrontando il risultato con quello ricevuto. Se i due hash corrispondono, allora il messaggio è considerabile valido.

2.2.2 Birthday Attack

2.3 Autenticazione

Capitolo 3

Conclusione

Conclusione

Bibliografia