Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea Triennale in Matematica

Crittografia Funzionale

Tesi di Laurea in: Crittografia

Relatore:
Chiar.mo Prof.
Davide Aliffi

Presentata da:
Davide Leonardi

III Sessione
Anno Accademico 2016/2017

Danzavamo
come Neve
nel Castagneto
quel fine Luglio

Introduzione

La crittografia è un metodo utilizzato dagli utenti per condividere in modo sicuro dei dati attraverso un network insicuro. Fino a qualche decennio fa, era largamente condivisa l'idea che per compiere un tale scambio di informazioni, fosse necessario che gli utenti interessati fossero a conoscenza di una chiave o password segreta condivisa a priori. E' evidente il conflitto che un tale metodo pone fra sicurezza e praticità, poiché il metodo sicuro per scambiarsi informazioni ne richiede a sua volta un altro a priori. Circa trent'anni fa furono introdotti i concetti di crittografia a chiave pubblica, dove la conoscenza della chiave segreta è necessaria ad una sola delle due parti interessate. Ad ogni modo, la crittografia a chiave pubblica rimaneva ancorata a varie limitazioni: ad esempio il fatto che l'accesso al dato cifrato fosse del tipo "tutto o niente", decifrarlo completamente o non ottenere alcuna informazione. E' per queste ed altre necessità dell'epoca contemporanea che si è giunti a parlare di *crittografia funzionale*. L'idea fondamentale consiste nel fatto che chiunque possiede una chiave segreta relativa ad una certa *funzionalità* possa ricavare una particolare informazione sul dato, a partire dalla sua cifratura, senza arrivare a conoscere il messaggio nella sua interezza.
Per esempio, consideriamo il processo di filtraggio della spam che opera sulle caselle di posta elettronica: l'utente sfrutta un programma delegato esterno (fidato o certificato) per determinare se un messaggio in arrivo è spam, senza però far trapelare altre informazioni sul messaggio. Si ottiene ciò impostando uno schema di cifratura funzionale per una funzione (creata dall'utente) che restituisce un valore binario positivo o nullo a seconda che riconosca o meno

un messaggio indesiderato.

Oppure si pensi al cosiddetto *data mining* largamente utilizzato nell'ambito medico e finanziario. Nello specifico consideriamo un ricercatore medico che vuole effettuare delle indagini riguardo alla connessione fra una certa malattia e una particolare caratteristica genetica: poiché sarà di suo interesse conoscere solo alcune informazioni particolari di ogni cartella clinica, attraverso un sistema di crittografia funzionale sarà possibile rendere visibile solo queste nascondendo il resto della storia clinica del paziente.

Notiamo infine che tipicamente non si ha la conoscenza a priori di quali possano essere le richieste che gli utenti possano avanzare su dei dati raccolti. La crittografia funzionale risponde elegantemente a questa necessità: quando i dati vengono creati essi vengono cifrati. Più tardi un utente può fare richiesta di conoscere alcune informazioni parziali. Se autorizzato, gli verrà fornita una chiave segreta relativa all'informazione (*funzionalità*) che desidera e che gli permetterà di accedere anche ai nuovi dati immessi successivamente. Questi esempi dunque motivano gli obbiettivi di ricerca in questo ambito: creare un ambiente di crittografia funzionale che supporti la più vasta famiglia di funzionalità e capirne i limiti legati alla realizzazione.

Indice

Capitolo 1

Definizioni di crittografia funzionale

Iniziamo dando la definizione sintattica di una criptazione funzionale (FE) per la funzionalità F. La funzionalità F descrive o calcola informazioni riguardanti il testo in chiaro (*plaintext*) a partire dal testo cifrato (*ciphertext*). Formalmente, una funzionalità è definita come segue:

Definizione 1.1. *Una funzionalitá F definita su (K, X) è una funzione $F : K \times X \to \{0,1\}^*$ descritta come una macchina di Turing deterministica.*
L'insieme K è detto spazio delle chiavi e l'insieme X è detto spazio dei messaggi in chiaro.
Si richiede che lo spazio K contenga la chiave vuota denotata con ε.

Diamo anche la definizione di funzione trascurabile che servirà più avanti:

Definizione 1.2. Una funzione $\eta : \mathbb{N} \to \mathbb{R}$ si dice *trascurabile* se $\forall c > 0$, $\exists$ una costante n_0 tale che per ogni $n > n_0$ si ha $\eta(n) < n^{-c}$

Uno schema di cifratura funzionale (FES) per la funzionalità F permette di calcolare $F(k, x)$ conoscendo la criptazione c di x e una chiave segreta sk_k per k. L'algoritmo che calcola $F(k, x)$ usando sk_k è detto di $de-criptazione$.

Precisamente uno schema di criptazione funzionale lavora così:

Definizione 1.3. *Uno schema di criptazione funzionale per la funzionalitá* F *definita su* (K, X) *é una* $n-pla$ *di quattro algoritmi ppt* $(setup, keygen, enc, dec)$ *che soddisfa la seguente condizione di correttezza* :

1. $(pp, mk) \leftarrow setup(1^\lambda)$ $(genera\ le\ chiavi\ principali\ pubblica\ e\ segreta)$
2. $sk \leftarrow keygen(mk, k)$ $(genera\ una\ chiave\ segreta\ per\ la\ chiave\ k)$
3. $c \leftarrow enc(pp, x)$ $(cripta\ il\ messaggio\ x)$
4. $y \leftarrow dec(sk, c)$ $(usa\ sk\ per\ calcolare\ F(k,x)\ conoscendo\ c)$

Richiediamo (condizione di correttezza) che y=F(k,x) con probabilità 1

In questi termini possiamo già mostrare brevemente che il sistema standard di cifratura a chiave pubblica, seppur in modo piuttosto banale, é un esempio di crittografia funzionale.
Sia $K := \{1, \varepsilon\}$ e consideriamo la funzionalità F definita su (K,X) per un generico spazio X di messaggi:

$$F(k,x) := \begin{cases} x & \text{se } k = 1 \\ len(x) & \text{se } k = \varepsilon \end{cases}$$

Una chiave segreta per $k = 1$ decifra pienamente il testo cifrato, mentre la chiave vuota restituisce semplicemente la lunghezza del testo in chiaro. E' chiaro che questa funzionalità descrive sintatticamente la cifratura standard a chiave pubblica.

Osservazione: La chiave vuota permette di conoscere tutte le informazioni sul messaggio in chiaro, che il testo cifrato fa intenzionalmente trapelare, come ad esempio la lunghezza del testo. La chiave segreta per ε è anch'essa vuota e denotata con ε. Dunque chiunque può calcolare dec(c,ε) e ottenere tutte le informazioni sul testo in chiaro che trapelano dalla sua cifratura.

1.1 Sottoclassi della crittografia funzionale

Date le definizioni di un FES, è conveniente, per le applicazioni che vogliamo presentare, introdurre due sottoclassi di crittografia funzionale su uno spazio dei messaggi X che ammette struttura additiva:

Crittografia Predicativa In molte applicazioni il messaggio $x \in X$ e' a sua volta una coppia $(ind, m) \in I \times M$ dove ind e' detto *indice* e m e' detto *messaggio utile* (payload message). Per esempio in un sistema di posta elettronica l'indice potrebbe essere il nome del mittente, mentre il messaggio utile il vero e proprio testo comunicato.

In questo contesto, un FES su $(K, (I \times M))$ è definito a partire da un *predicato* (algoritmo di valutazione a tempo polinomiale) $P : K \times I \to \{0, 1\}$ e la funzionalita' F lavora come segue:

$$F(k \in K, (ind, m) \in X) := \begin{cases} m & \text{se } P(k, ind) = 1 \\ \perp & \text{se } P(k, ind) = 0 \end{cases}$$

Il termine $\perp$ è usato in informatica per indicare un arresto o un output indefinito.

Di conseguenza, sia c la cifratura di (ind,m) e sia sk_k una chiave segreta per k. Allora dec(sk_k,c) rivela il messaggio utile m quando P(k,ind)=1, altrimenti si interrompe senza rivelare ulteriori informazioni riguardanti m.

Crittografia Predicativa a indice pubblico E' un caso particolare della crittografia predicativa in cui l'indice (che potrebbe essere, per esempio,

il mittente di una email) è pubblico. Precisamente in questo tipo di FE la chiave vuota rivela esplicitamente l'indice *ind* in questo modo:

$$F(\varepsilon, (ind, m)) = (ind, len(m)).$$

Dunque, $\mathrm{dec}(\varepsilon, c)$ restituisce a chiunque l'indice e la lunghezza in bit del messaggio in chiaro.

Capitolo 2

Esempi di crittosistemi funzionali

Molti recenti modelli e concetti crittografici possono essere visti come casi particolari di FE. In questo capitolo diamo alcuni esempi di come la FE comprenda questi modelli e concetti pre-esistenti e gia' dimostrati sicuri secondo definizioni di sicurezza che affronteremo piu' avanti.

2.1 FES predicativi a indice pubblico

La prima famiglia di sistemi che consideriamo sono quelli a indice pubblico, iniziando dal caso più semplice Identity-Based, per poi esaminarne altri più sofisticati, usando la notazione introdotta nel capitolo precedente e ricordando che $F(\varepsilon, (ind, m)) = (ind, len(m))$

Identity-Based Enryption In questo schema (IBE) il testo cifrato e la chiave privata sono associate a stringhe (anche note come *identità*) e una chiave può decifrare il messaggio solo se le stringhe corrispondenti sono uguali. IBE è formalmente presentato come un FES predicativo dove:

1. Lo spazio delle chiavi K è: $K := \{0,1\}^* \cup \{\varepsilon\}$

2. Il messaggio in chiaro è la coppia (ind,m) dove lo spazio degli indici è $I := \{0,1\}^*$

3. Il predicato P su $K \times I$ è definito nel modo seguente:
$$P(k \in K, ind \in I) := \begin{cases} 1 & \text{se } k = ind \\ 0 & \text{altrimenti} \end{cases}$$

4. la funzionalita' F lavora come segue:
$$F(k \in K, (ind, m) \in X) := \begin{cases} m & \text{se } P(k, ind) = 1 \\ \perp & \text{se } P(k, ind) = 0 \end{cases}$$

IBE rappresenta la prima funzionalità che non è direttamente realizzabile nei termini della crittografia a chiave pubblica.
Boneh,Franklin e Cocks costruirono il primo sistema pratico di IBE che fu dimostrato sicuro secondo la definizione basata sull'indistinguibilità e nel modello dell'oracolo casuale.

Attribute-Based Encryption Il concetto di Attribute-Based Encryption (ABE) fu inizialmente introdotto da Sahai e Waters per definire una "politica di accessi" a un certo dato, complessa e selettiva, che regoli l'accesso alle diverse funzionalità di uno stesso dato. Successivamente, insieme a Goyal e Pandey, si ridefinirono questi schemi in due differenti formulazioni: Key-Policy ABE e Ciphertext-Policy ABE. Iniziamo col descrivere il primo:

Key-Policy ABE Descriviamo questo schema per formule booleane in n variabili, come fu inizialmente descritto da Goyal. Un sistema di Key-Policy ABE su n variabili puo' essere descritto come un FES predicativo a indice pubblico per il predicato $P_n : K \times I \to \{0,1\}$ dove:

1. Lo spazio delle chiavi K è l'insieme di tutte le formule booleane ϕ su n variabili e posto $\overrightarrow{z} = (z_1, \ldots, z_n) \in \{0,1\}^n$, indichiamo con $\phi(\overrightarrow{z})$ il valore di ϕ valutata su $\overrightarrow{z}$.

2. Il messaggio in chiaro è la coppia ($ind = \overrightarrow{z}$,m) dove lo spazio degli indici è $I := \{0,1\}^n$ nel quale interpretiamo $\overrightarrow{z}$ come un vettore di bit rappresentante i valori booleani $z_1, \ldots, z_n$
3. Il predicato P_n su $K \times I$ è definito nel modo seguente:
$$P_n(\phi \in K, ind = \overrightarrow{z} \in I) := \begin{cases} 1 & \text{se } \phi(\overrightarrow{z}) = 1 \\ 0 & \text{altrimenti} \end{cases}$$

In questo tipo di sistemi una chiave fornisce una formula d'accesso, che opera su un insieme di n attributi, la quale deve essere soddisfatta per poter decifrare e conoscere m.

Ciphertext-Policy ABE Un concetto duale di ABE è Ciphertext Policy ABE, dove i ruoli del testo cifrato e della chiave sono essenzialmente invertiti . Come prima, questo sistema su n variabili può essere visto come un FES predicativo a indice pubblico per il predicato P_n definito su $K \times I \to \{0,1\}$ dove:

1. Lo spazio delle chiavi è: $K = \{0,1\}^n$
2. Il messaggio in chiaro è la coppia ($ind = \phi$, m) dove lo spazio degli indici I è l'insieme di tutte le formule booleane ϕ su n variabili
3. Il predicato P_n su $K \times I$ è definito come segue:
$$P_n(\overrightarrow{z} \in K, ind = \phi \in I) := \begin{cases} 1 & \text{se } \phi(\overrightarrow{z}) = 1 \\ 0 & \text{altrimenti} \end{cases}$$

2.2 FES predicativi a indice privato

Nonostante i modelli sopra presentati siano efficienti per gestire un sistema di accessi, sono comunque limitati per due ragioni. L'indice è un dato pubblico e ciò per alcuni fini in ambito crittografico è considerata una perdita di informazioni sensibile; in secondo luogo il sistema non permette di eseguire molti compiti, come ad esempio una ricerca, sui dati cifrati.
Diamo qui esempi di FES predicativo che nascondono la componente *ind*.

Anonymous Identity-Based Encryption La funzionalità di Anonymous IBE è simile al precedente IBE fatta eccezione per la stringa rappresentante l'indice che può essere vista solo se in possesso di una chiave segreta adatta. Dunque l'unica differenza sta nelle informazioni che ε lascia trapelare:

$$F(\varepsilon, (ind, m)) = len(m)$$

Hidden Vector Encryption Un sistema di questo tipo prevede che il testo cifrato sia un vettore di n elementi, dove ogni elemento è un vettore in $\{0,1\}^n$ e una chiave privata sia anch'essa un vettore di n elementi in $\{*\} \cup \{0,1\}^n$ dove con $*$ intendiamo un elemento speciale (detto *wildcard*). Formalmente:

1. Lo spazio delle chiavi è composto di tutti i vettori $(v_1, \ldots, v_n)$ dove ogni $v_i \in \{*\} \cup \{0,1\}^n$
2. Il messaggio in chiaro è la coppia $(ind = (w_1, \ldots, w_n), m)$ dove ogni $w_i \in \{0,1\}^n$. Lo spazio degli indici lo denotiamo con $(\{0,1\}^n)^n$
3. Il predicato P_n su $K \times I$ è definito così:

$$P_n((v_1, \ldots, v_n) \in K, ind = (w_1, \ldots, w_n)) := \begin{cases} 1 & \text{se } v_i = w_i \, per \, ogni \, v_i \neq * \\ 0 & \text{altrimenti} \end{cases}$$

Notiamo che il test del predicato richiede verifiche non banali, come una ricerca sul dato, che sono quindi permessi in questo modello.

Inner Product Predicate Katz, Sahai e Waters inizialmente proposero questo sistema dove si va a testare se un prodotto interno sull'anello $\mathbb{Z}_{\mathbb{N}}$ è nullo, con N prodotto di tre numeri primi casuali determinati dall'algoritmo Setup. Successivamente si è adattato a una versione più compatta sul campo F_p che quindi usa un solo primo. Lo descriviamo nella seconda costruzione che opera su vettori di lunghezza n:

1. L'algoritmo Setup genera un primo p casualmente, di lunghezza k, dove k è il parametro di sicurezza.
2. Lo spazio delle chiavi è K è composto dei vettori di $(F_p)^n$
3. Il testo in chiaro è la coppia $(ind = (w_1, \ldots, w_n), m)$ con $w_i \in F_p$. Dunque I=K.
4. Il predicato $P_{n,p}$ su $K \times I$ è definito così:

$$P_{n,p}((v_1, \ldots, v_n) \in K, ind = (w_1, \ldots, w_n)) := \begin{cases} 1 & \text{se } \sum_{i=1,\ldots,n} v_i \cdot w_i = 0 \\ 0 & \text{altrimenti} \end{cases}$$

Tutti questi sistemi e combinazioni di essi sono spesso utilizzati e vengono raccolti sotto l'unico modello della crittografia funzionale. Questi sistemi permettono di limitare la decifrazione di informazioni solo quando sono soddisfatte condizioni complesse, che possono riguardare identità, livelli di autorità gerarchici, permessi ecc.. implementando quella che si usa chiamare "politica di accesso". Presentiamo ora i concetti di sicurezza relativi a questo ambito e i risultati finora ottenuti.

Capitolo 3

Sicurezza Game-Based e Simulation-Based

Sia Ω un FES per la funzionalità F definita su (K,X). Il nostro scopo è definire un concetto di sicurezza rispetto a un avversario che chiede ripetutamente delle chiavi segrete sk_k corrispondenti a chiavi $k \in K$ scelte dall'attaccante. Il problema consiste nel determinare dei parametri sui messaggi cifrati da testare in modo da creare una simulazione semanticamente corretta. Più precisamente, l'avversario ottiene tutte le chiavi segrete che desidera, poi sceglie due messaggi $m_0, m_1 \in X$ e ottiene la cifratura c di uno dei due. Chiaramente se l'attaccante possiede una chiave segreta per un qualche $k \in K$ per cui $F(k, m_0) \neq F(k, m_1)$, può facilmente rispondere al quesito su c dando in output:

$$\begin{cases} 0 & \text{se } dec(sk_k, c) = F(k, m_o) \\ 1 & \text{altrimenti} \end{cases}$$

Dunque, ai nostri fini, dobbiamo richiedere che: $\forall k$ per cui l'avversario conosce sk_k si abbia:

$$F(k, m_0) = F(k, m_1) \tag{3.1}$$ A

E poiché la chiave vuota rivela la lunghezza del messaggio si vuole

$$len(m_0) = len(m_1)$$

Definizione 3.1. Tenendo conto della condizione (A) possiamo definire la *sicurezza Game − Based* per Ω. Per b=0,1 definiamo l'esperimento b per un avversario A come segue:

- Setup: calcola $(pp, mk) \leftarrow Setup(1^\lambda)$ e dà pp a A
- Query: A seleziona $k_i \in K$ con i generico e ottiene $sk_i \leftarrow keygen(mk, k_i)$
- Challenge: A sceglie due messaggi $m_1, m_0 \in X$ che soddisfano (A) e gli viene data $c = enc(pp, m_b)$
- A continua a chiedere e ottenere chiavi, rispettando la condizione (A), infine restituisce un output in $\{0, 1\}$

Sia W_b per b=0,1 l'evento in cui l'avversario dà in output 1 nell'esperimento b e sia $FE_{adv}[\Omega, A](\lambda) := |Pr[W_0] - Pr[W_1]|$

Ω è sicuro Game-Based se per ogni algoritmo ppt A la funzione $FE_{adv}[\Omega, A](\lambda)$ è trascurabile.

La funzione $FE_{adv}[\Omega, A](\lambda)$ è chiamata *vantaggio dell'avversario* poiché è la differenza fra la probabilità di successo avversario (ossia che riesca a indovinare il messaggio la cui cifratura è c) e quella del suo fallimento. Usando questo concetto, richiamiamo brevemente il concetto di sicurezza semantica per un cifrario a chiave pubblica (G,E,D)=(Gen, Enc, Dec). E' del tutto analoga alla definizione per FES: si stabiliscono le regole del "gioco", in cui l'avversario riceve la cifratura di un messaggio scelto dallo sfidato fra due noti a entrambi, l'unica differenza sta nel fatto che lo sfidante chiede chiavi e ottiene con esse risultati dell'algoritmo di decifrazione e non quelli del calcolo della funzionalità.

3.1 Costruzione di un sistema sicuro "Brute Force"

Mostriamo in questa sezione che ogni funzionalità F definita su uno spazio delle chiavi K a dimensione polinomiale è realizzabile mediante un sistema

ad-hoc. Sia $s = |K| - 1$ dove $K = \{\varepsilon, k_1, \ldots, k_s\}$. In questa costruzione le dimensioni delle chiavi e del testo sono proporzionali a s. Questo sistema sfrutta un cifrario a chiave pubblico semanticamente sicuro (G,E,D) e lavora come segue:

1. $setup(1^\lambda)$: $per\, i = 1, ..., s$ calcola$(pp_i, mk_i) \leftarrow G(1^\lambda)$
   e restituisce: pp:=$(pp_1, \ldots, pp_s)$ e mk:=$(mk_1, \ldots, mk_s)$
2. $keygen(mk, k_i)$: restituisce $sk_i := mk_i$
3. $enc(pp, x)$: restituisce $\overrightarrow{c}$:=$(F(\varepsilon, x), E(pp_1, F(k_1, x), \ldots, E(pp_s, F(k_s, x)))$
4. $dec(sk_i, \overrightarrow{c})$: restituisce c_0 (ossia la lunghezza di x) se $sk_i = \varepsilon$, e $D(sk_i, c_i)$ altrimenti

E' evidente che il testo cifrato $\overrightarrow{c}$ lascia trapelare la lunghezza in bit di $F(k_i, x)$ per ogni i, perciò richiediamo che queste informazioni siano già contenute nel calcolo di $F(\varepsilon, \cdot)$. In questo caso diciamo che F rivela la lunghezza bit funzionale. Notiamo che conoscere la chiave segreta particolare sk_k equivale a conoscere la chiave segreta principale con la quale è possibile ricavare ogni chiave segreta particolare. Questa implicazione evidenzia una grande debolezza di questo sistema, non a caso detto Brute Force.

Teorema 1. Sia F una funzionalità che rivela la lunghezza in bit. Se (G,E,D) è un sistema a chiave pubblica semanticamente sicuro, allora il FES "Brute Force" che implementa F è sicuro.
Dimostrazione: Si basa sull'argomento ibrido sulle s componenti del testo cifrato. Per maggiori dettagli si veda [1]

3.2 Insufficienza della definizione Game Based

Mostriamo ora come alcune funzionalità possano essere implementate in un sistema sicuro secondo la definizione Game Based, dove però è evidente

una non-sicurezza dettata da aspetti pratici. Per chiarire, sia π una permutazione one-way e consideriamo F che ammette solo la chiave vuota definita così:

$$F(\varepsilon, x) = \pi(x)$$

Chiaramente un modo corretto di implementare F sarebbe quello di usare un algoritmo di cifratura che dia in output semplicemente $\pi(x)$. Questo schema soddisferebbe anche la definizione di sicurezza che introdurremo più avanti. Ad ogni modo, consideriamo la seguente implementazione. Sia $enc(pp, x) = x$. Chiaramente il sistema lascia trapelare parecchie informazioni su x, tuttavia è facile verificare che questo modello è sicuro secondo la definizione Game Based, infatti la probabilità che ha l'avversario di indovinare tra i due messaggi è la stessa di pescarne a caso due uguali tra tutti quelli di lunghezza fissata (perché l'avversario può usare solo la chiave vuota, e ottenere come informazioni due immagini dalla funzione π, delle quali, per definizione, è pressoché impossibile trovare le retro immagini).

3.3 Sicurezza Simulation Based

In questa sezione, introduciamo e commentiamo un nuovo concetto di sicurezza che sorge dal *paradigma della simulazione*, un modello che risulta particolarmente efficace nell'ambito dei protocolli sicuri per lo scambio di informazioni.
Iniziamo col dare una "definizione" intuitiva del concetto, cioè che data la chiave segreta sk_k relativa a una certa $k \in K$, essa riveli solamente $F(k, x)$ data una cifratura di x. La definizione Simulation Based richiama i concetti di indistinguibilità. Per continuare l'analisi, dobbiamo introdurre il *modello dell'oracolo casuale.*

3.3.1 Modello dell'oracolo casuale (ROM)

In crittografia un oracolo casuale è una "scatola nera" teorica che risponde ad ogni domanda ricevuta in input con una risposta casuale, scelta uniforme-

mente all'interno del suo dominio di output. Per ogni domanda specifica la risposta, una volta assegnata dall'oracolo, è sempre uguale. In altri termini, un oracolo casuale è una *funzione* matematica che associa ogni possibile input ad un valore random scelto all'interno del suo dominio di output. Gli oracoli random sono usati per modellare funzioni crittografiche hash (unidirezionali, resistenti alle collisioni) all'interno di schemi dove sono necessarie delle forti assunzioni di casualità uniforme sull'output. Nel ROM tutte le parti in gioco hanno accesso al medesimo oracolo e chiamiamo "query" la generica richiesta sottoposta ad esso. Una dimostrazione nel ROM non equivale sempre al medesimo risultato nel modello standard, rimane comunque un buon punto di partenza. La questione più controversa riguardante un oracolo rimane quella della *programmabilità*. Teoricamente non richiede troppa fantasia concepire un oracolo casuale ideale che svolga il suo lavoro coerentemente alle proprietà richieste, il problema sorge nel momento in cui esso deve essere simulato (ossia programmato) per verificare concretamente ciò che in teoria sappiamo costruire. In altre parole la realizzazione concreta (che quindi richiede la *programmabilità*), allo stato attuale delle conoscenze, è un processo ineluttabile di indebolimento della casualità teorica(cioè quella *non-programmabile*). Nello specifico vedremo come nel modello del ROM non-programmabile sia valido un risultato di *non-esistenza* di FES sicuri Simulation Based, mentre nel modello del ROM *programmabile*(o come viene recentemente chiamato, *standard*) è possibile costruire un FES sicuro Simulation Based. Nella sezione che segue useremo una sintassi non-standard per descrivere il funzionamento delle interazioni con un generico oracolo con memoria (*stateful oracle*). Quando scriviamo $A^{B(\cdot)[[x]]}$ intendiamo che l'algoritmo A può chiedere con successo una query q al suo oracolo, a quel punto $B(q, x)$ viene eseguito e restituisce la coppia (y, x'). Il valore y è dato ad A come risposta alla query e la variabile x viene aggiornata in x' e questo valore aggiornato viene usato come input per B alla prossima chiamata come oracolo, e per qualunque altro algoritmo chiamato nell'esperimento che vedrebbe x come input. Inoltre, quando scriviamo $A^{B^o(\cdot)}$, significa che A invia una query al suo oracolo, a

quel punto $B^o(q)$ è eseguita, e ogni volta che B fa una richiesta oracolare, essa è risposta da A.

Definizione 3.2. Un FES Ω è *sicuro simulation based* (o anche Simulation-sicuro) se esiste un (oracolo) algoritmo ppt $Sim = (Sim_1, Sim_0, Sim_2)$ tale che per ogni (oracolo) algoritmi ppt $Message$ e Adv, si ha che le seguenti due distribuzioni sono indistinguibili rispetto al parametro di sicurezza λ:

Distribuzione Reale

1. $(pp, mk) \leftarrow setup(1^\lambda)$
2. $(\overrightarrow{x}, \tau) \leftarrow Message^{keygen(mk,\cdot)}(pp)$
3. $\overrightarrow{c} \leftarrow enc(pp, \overrightarrow{x})$
4. $\alpha \leftarrow Adv^{keygen(mk,\cdot)}(pp, \overrightarrow{c}, \tau)$
5. Siano $y_1, \ldots, y_l$ le queries fatte a keygen da $Message$ e Adv nei passi precedenti
6. Output:$(pp, \overrightarrow{x}, \tau, \alpha, y_1, \ldots, y_l)$

Distribuzione Ideale

1. $(pp, \sigma) \leftarrow Sim_1(1^\lambda)$
2. $(\overrightarrow{x}, \tau) \leftarrow Message^{Sim_0(\cdot)[[\sigma]]}(pp)$
3. $\alpha \leftarrow Sim_2^{F(\cdot,\overrightarrow{x}),Adv^o(pp,\cdot,\tau)}(\sigma, F(\varepsilon, \overrightarrow{x}))$
4. Siano $y_1, \ldots, y_l$ le queries fatte a F da Sim nei passi precedenti
5. Output:$(pp, \overrightarrow{x}, \tau, \alpha, y_1, \ldots, y_l)$

L'idea dietro la definizione, è che quando un avversario intercetta dei dati non ne trae informazioni, poiché lo stesso tipo di dati può crearli da sé senza conoscere nulla sulle sorgenti dalle quali provengono L'ultima notazione, precisata nel paragrafo precedente alla definizione, è da tenere in considerazione quando si deve costruire l'algoritmo *Sim* (*Simulatore*) che utilizza un oracolo interno, anch'esso simulato. Questa richiesta serve per assicurarsi che il *Simulatore* riesca effettivamente a creare una distribuzione simile a quella reale, rimanendone totalmente estraneo.

3.4 Impossibilità di FES Simulation-sicuri nel modello del ROM non programmabile

Vogliamo mostrare l'impossibilità di un FES simulation-sicuro e nello specifico lo mostriamo (a grandi linee) per un sistema semplice come IBE. Notiamo inoltre che il risultato che stiamo per presentare sussiste anche per formulazioni meno restrittive della definizione di sicurezza Simulation Based, in particolare per la seguente:

Definizione 3.3. Un FES Ω è *debolmente sicuro simulation based* (o anche debolmente Simulation-sicuro) se esiste un (oracolo) algoritmo ppt *Sim* tale che per ogni (oracolo) coppia di algoritmi ppt $Message$ e Adv, si ha che le seguenti due distribuzioni sono indistinguibili rispetto al parametro di sicurezza λ:

Distribuzione Reale

1. $(pp, mk) \leftarrow setup(1^\lambda)$
2. $(\overrightarrow{x}, \tau) \leftarrow Message(1^\lambda)$
3. $\overrightarrow{c} \leftarrow enc(pp, \overrightarrow{x})$
4. $\alpha \leftarrow Adv^{keygen(mk,\cdot)}(pp, \overrightarrow{c}, \tau)$

5. Siano $y_1, \ldots, y_l$ le queries fatte a keygen da $Message$ e Adv nei passi precedenti
6. Output:$(\overrightarrow{x}, \tau, \alpha, y_1, \ldots, y_l)$

Distribuzione Ideale

1. $(\overrightarrow{x}, \tau) \leftarrow Message(1^\lambda)$
2. $\alpha \leftarrow Sim^{F(\cdot, \overrightarrow{x})}(1^\lambda, \tau, F(\varepsilon, \overrightarrow{x}))$
3. Siano $y_1, \ldots, y_l$ le queries fatte a F da Sim nei passi precedenti
4. Output:$(\overrightarrow{x}, \tau, \alpha, y_1, \ldots, y_l)$

Oss: Un altro indebolimento consisterebbe nel riportare $y_1, \ldots, y_l$ come insieme e non come tupla ordinata; il risultato di cui accenniamo la prova di seguito continuerebbe a valere.

Teorema 2 Sia F la funzionalità per IBE. Allora non esiste nessun FES debolmente simulation-sicuro che implementa F nel modello dell'oracolo casuale non-programmabile.
Cenni di Dimostrazione Denotiamo con H l'oracolo. Consideriamo il seguente schema di algoritmi:

- $Message(1^\lambda)$: sia len_{sk} il massimo fra le lunghezze in bit di una chiave segreta corrispondente alla chiave 0 e parametro λ. Sia $\overrightarrow{x}$ il vettore di lunghezza $len_{sk} + \lambda$, dove $\forall i$, $x_i = (r_i, 0)$ con r_i bit scelto casualmente e indipendentemente dai precedenti. τ è vuota.
- $Adv^{keygen(mk,\cdot)}(pp, \overrightarrow{c}, \tau)$: chiama H sull'input $(pp, \overrightarrow{c})$ e ottiene una stringa w di lunghezza λ. Ora richiede la chiave segreta per w e successivamente per la chiave 0 e usa quest'ultima per decifrare l'intero testo cifrato. Restituisce in output la trascrizione completa di tutto il

calcolo svolto da Adv, includendo le chiamate a H e le interazioni con l'oracolo $keygen$.

Ora consideriamo il compito che un simulatore dovrebbe svolgere per generare una distribuzione indistinguibile da quella soprastante. Poiche' Adv fa una sola richiesta all'oracolo per una chiave della forma w, ci troviamo nel caso in cui anche Sim può fare una solo chiamata al suo oracolo (la prima) F e sarà di questa forma. Dopo ciò, un distinguisher potrà indovinare se questo w è l'output di H applicato ad una qualche stringa $(pp, \overrightarrow{c})$. Dunque, il simulatore dovrà fare prima la richiesta a H e solo dopo le altre a F. A questo punto Sim non ha alcuna conoscenza sul testo in chiaro e gli r_i (che possono essere rivelati solo con la chiave privata corrispondente alla chiave 0 ossia solo se il simulatore la chiederà a F). Perciò la stringa fissata $z = (pp, \overrightarrow{c})$ dovrebbe possedere la proprietà (impossibile) per cui, dopo aver ricevuto solo len_{sk} bit di informazioni, essa può determinare se z sia un'arbitraria stringa di lunghezza $len_{sk} + \lambda$.

3.5 Costruzione dello schema Brute Force simulation-sicuro nel programmabile ROM

Consideriamo il seguente schema Brute Force, leggermente modificato con l'introduzione di un oracolo H intento a mascherare in modo casuale gli output di F.
Sia H:$\{0,1\}^n \to \{0,1\}$; per generare stringhe casuali più lunghe scriveremo anche $H(x)$ intendendo (sarà chiaro dal contesto) la stringa degli9 elementi concatenati $(H(x,1),...,H(x,m))$. Ricordiamo che $s = |K| - 1$. Lo schema Brute Force implementante F usa uno schema semanticamente sicuro a chiave pubblica (G,E,D) e lavora così:

1. $setup(1^\lambda)$: $per\ i = 1,...,s$ calcola $(pp_i, mk_i) \leftarrow G(1^\lambda)$
   e restituisce: pp:=$(pp_1, \ldots, pp_s)$ e mk:=$(mk_1, \ldots, mk_s)$

2. $keygen(mk, k_i)$: restituisce $sk_i := mk_i$

3. $enc(pp, x)$: sceglie valori random $r_1, \ldots, r_s \in \{0,1\}^\lambda$
   restituisce $\overrightarrow{c}$:=$(F(\varepsilon, x), E(pp_1, r_1), H(r_1) \oplus F(k_1, x), \ldots, E(pp_s, r_s), H(r_s) \oplus F(k_s, x))$

4. $dec(sk_i, \overrightarrow{c})$: restituisce c_0 se $sk_i = \varepsilon$, e $H(D(sk_i, c_{2i-1})) \oplus c_{2i}$ altrimenti

Teorema 3 Sia F una funzionalità che rivela la lunghezza bit. Se (G,E,D) è un cripto-sistema a chiave pubblica semanticamente sicuro allora il modello Brute Force modificato qui sopra è simulation-sicuro nel programmabile ROM.
Dimostrazione : La si può trovare nell'appendice A dell'articolo [K1 1]

Capitolo 4

Esempio di delegazione e politica d'accesso

In uno schema crittografico a chiave pubblica standard, il testo cifrato (usando una chiave pubblica pk) è decifrabile solo usando una chiave segreta (sk corrispondente a pk). La crittografia funzionale fornisce relazioni più sofisticate e flessibili tra le chiavi: di seguito spieghiamo quali siano effettivamente le potenzialità raggiunte da uno schema del tipo Chiphertext-Policy ABE. Ogni chiave segreta sk_α è associata a un parametro α e il messaggio m viene criptato in $Enc(m, pk, \beta)$ nella modalità della crittografia a chiave pubblica, usando però un parametro β. La decifrazione di $Enc(m, pk, \beta)$ usando sk_α riesce solo se è verificata una relazione $R(\alpha, \beta) = Vero$. Il parametro β esprime una struttura d'accesso descritta da *attributi* e *porte di soglia*, mentre α funge da "documento d'identità", ossia esprime gli *attributi* posseduti da colui che detiene la chiave segreta. Un generico *attributo* si esprime in una categoria (per esempio "genere","ruolo" ecc..) e da un valore (per esempio "donna","manager" ecc..) rilevanti per lo schema che si vuole costruire. Concretamente "$genere = donna$" è usato come un *attributo* e β è un predicato del tipo "$genere = donna \wedge ruolo = manager$", mentre α è un insieme del tipo "$genere = donna, ruolo = manager$". Dunque il parametro pubblico β è sufficiente a regolare la politica d'accesso. Questo metodo è

particolarmente utile per controllare i diversi gradi di autorità in merito alle molteplici operazioni eseguibili su un dato.
Per chiarire, si pensi alla tematica del *cloud*: supponiamo che un utente A voglia condividere un album di foto con degli amici, rendendoli capaci di vederle, senza poterle modificare in alcun modo. Dunque l'utente A usa uno schema del tipo ABE, dove la relazione *"essere amico di A=Vero"* concede di accedere alla visualizzazione delle foto se verificata (*funzionalità parziale*). Supponiamo inoltre ci sia un utente B a cui A vuole concedere la completa gestione delle foto, per fare ciò ad A basta aggiungere una relazione del tipo *"identità=B"* e se verificata, questa permette di cancellare, modificare o aggiungere foto all'album (*funzionalità totale*).

Conclusioni

Abbiamo visto che in ogni schema di cifratura funzionale standard c'è un autorità principale (*master*) responsabile della distribuzione delle chiavi private particolari.
Questa impostazione può essere generalizzata, per regolare una politica di accessi dove sussistono vari domini di accesso, o in altre parole con più *master* (per esempio una forma di Ciphertext-Policy ABE può fornire un servizio di scambio di informazioni fra diversi organi governativi e militari). In queste situazioni ogni ente stabilirà la sua rete di permessi a cui ognuno potrà accedere con le dovute autorizzazioni.
Altrettanto significativo è il concetto di *delegazione*, ossia la capacità di trasferire ad un utente non delle informazioni, ma bensì la capacità di estrarre informazioni da una certa fonte (diffusissima necessità nell'organizzazione degli enti statali o in quasi tutti i settori di ricerca).
Non è inoltre da sottovalutare il fatto sorprendente che, questo metodo di approccio alla crittografia, sia molto vicino all'essere universale e coerente, dunque molto simile a quello che potremmo definire un settore vero e proprio della matematica, conquistando tutti i vantaggi di correttezza propri della scienza. Anche se ciò per ora è ancora lontano dall'essere vero (anche per il frenetico progresso tecnologico, che dona ogni anno nuovi "attacchi" da cui "difendersi").
Forse, in futuro, troveranno risposta anche le sfide aperte di questo interessante modello: Si possono costruire FES che implementano ogni funzionalità eseguibile in tempo polinomiale? Inoltre, questi schemi, sono realmente

implementabili?

Bibliografia

K1 [1] Dan Boneh, Amit Sahai, Brent Waters *Functional Encryption : Definitions and Challenges*, https://eprint.iacr.org/2010/543.pdf

K2 [2] Dan Boneh, Amit Sahai, Brent Waters *Functional Encryption : A New Vision for Public Key Cryptography*

K3 [3] Raphael Pass, Abhi Shelat: *A Course in Cryptography* https://www.cs.cornell.edu/courses/cs4830/2010fa/lecnotes.pdf

Ringraziamenti

Con grande affetto ringrazio tutti i miei amici, da quelli conosciuti grazie all'università a quelli che c'erano già prima. Alla mia famiglia riservo un pensiero speciale per la pazienza che loro, e anch'io, abbiamo scoperto di avere nei miei confronti. Ringrazio molto il relatore di questa tesi Davide Aliffi per la disponibilità e ogni mio altro professore, universitario e non, con cui ho avuto la fortuna di imparare qualcosa. E ringrazio la fortuna, per avermi dato tutto ciò.