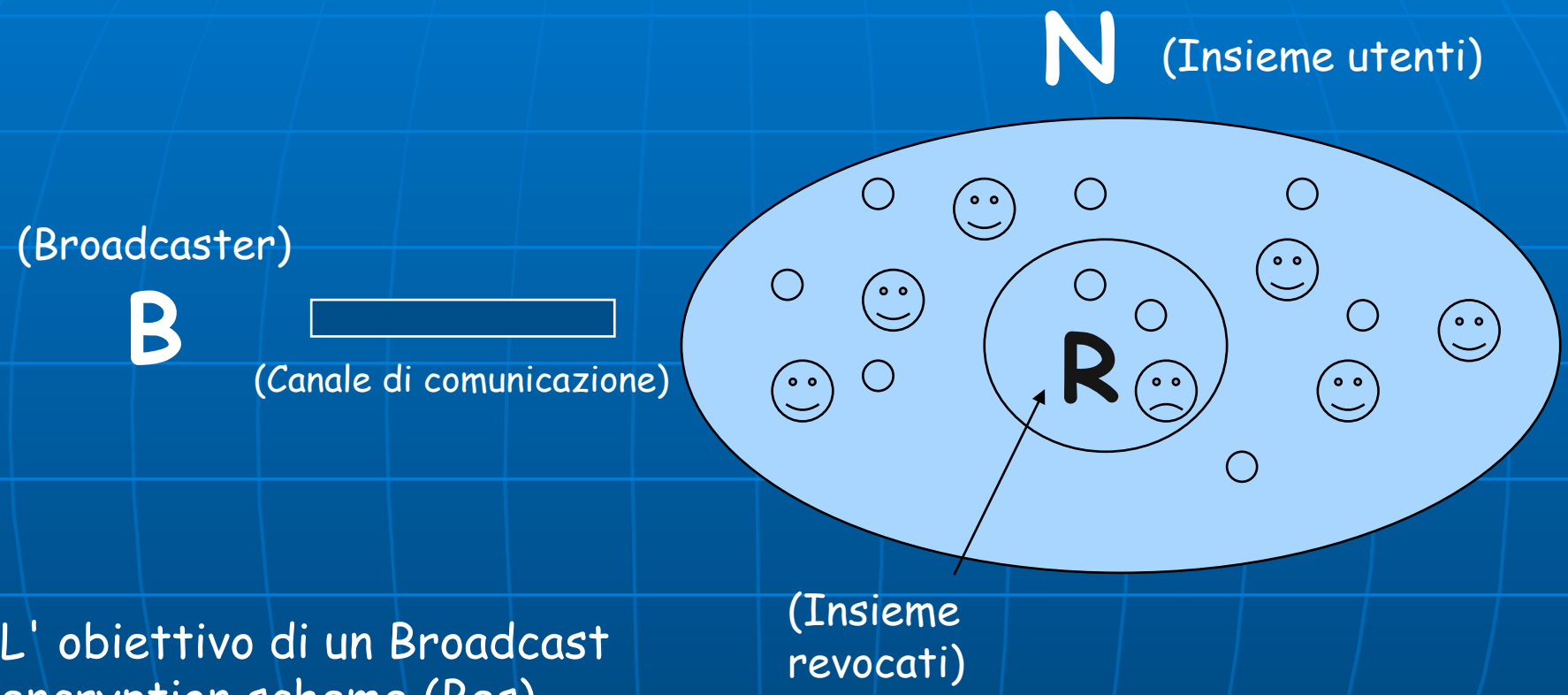
Complete Subtree

Lezione tenuta dal
Prof. P. D'Arco

Presentazione di:

Alfonso Sabato Siciliano

Broadcast Encryption



L' obiettivo di un Broadcast encryption scheme (Bes) è permettere a B di inviare un messaggio M in modo tale che:

- Un utente u appartenente a N\R possa decrittare M.
- Un utente u appartenente a R non ottenga alcuna informazione su M.

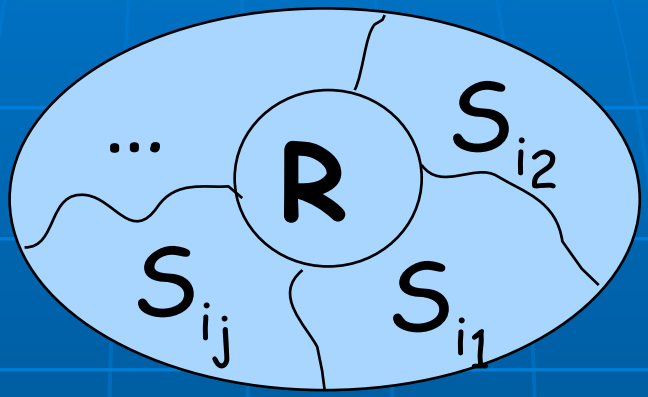
Subset Cover Framework

Algoritmi definiti da:

- Collezione di sottoinsiemi $S_1,...,S_u$.
- Chiavi L_j assegnate ai sottoinsiemi S_j , per j = 1,...,u .
- Metodo di partizionamento:

  dato R (insieme revocati), N/R viene partizionato tramite l' unione di sottoinsiemi S_j , con j=1,...,m. Il messaggio M viene poi cifrato m volte con $L_{i_1}, ... ,L_{i_m}$, dove L_{i_j} è la chiave corrispondente al sottoinsieme S_{i_j}.

Subset Cover Framework



- Metodo di recupero delle chiavi L_{ij} :
  ogni utente u appartenente a S_{ij} (per qualche j) può calcolare la propria chiave L_{ij} usando un' informazione privata I_u .

Fasi in un BES

1 Inizializzazione:

Ogni utente u riceve l' informazione privata I_u , grazie alla quale se u appartiene a S_i allora I_u permette di calcolare L_i.

2 Broadcast (B):

- B sceglie una chiave di sessione k.
- Dato R, partiziona N\R in S_{i_1} ,..., S_{i_m} (sottosistemi disgiunti) a cui corrispondono le chiavi L_{i_1} ,..., L_{i_m} .
- Cifra k con le chiavi L_{i_1} ,...,L_{i_m} e invia il messaggio M cifrato:

$$< [\, i_1,\ldots,i_m,\ E_{L_{i_1}}(k),\ldots,\ E_{L_{i_m}}(k)\,],\ F_k(M) >$$

header body

Fasi in un BES

3 Decrifratura:

- l' utente u riceve $<[i_1,\ldots,i_m, E_{L\,i_1}(k),\ldots, E_{L_{i_m}}(k)], F_k(M)>$
- Individua i_j tale che u appartiene a S_{i_j}.
- Calcola L_{i_j} tramite I_u.
- Calcola $D_{L_{i_j}}(E_{L_{i_j}}(k)) = k$.
- Calcola $F_k^{-1}(F_k(M)) = M$.

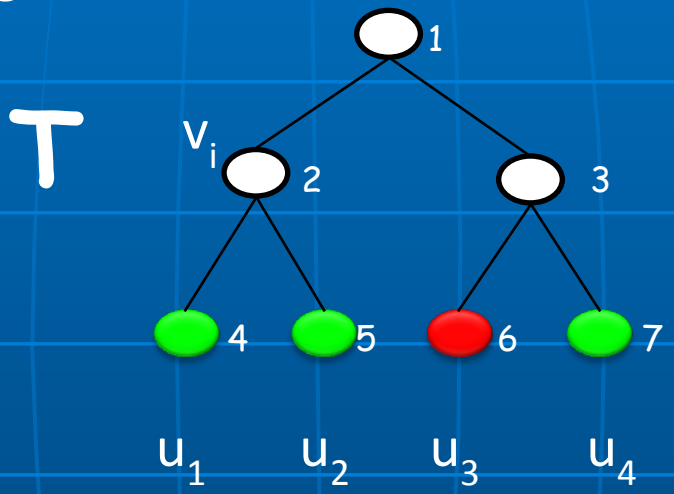
Implementazione di Schemi Crittografici per Trasmissioni Broadcast Sicure

- *Complete Subtree*
- Subset Difference
- Layer Subset Difference

Complete Subtree

Fase 1 - Inizializzazione

Ogni utente è una foglia di un albero binario completo.



$N = \{u_1, u_2, u_3, u_4\}$

C = {sottoinsiemi associati a tutti i sottoalberi dell' albero binario completo}

$|C|=2N-1$

Sottoinsieme S_i = {foglie del sottoalbero radicato nel nodo v_i}

u_i appartiene a S_i <=> u_i discende da v_i.

Complete Subtree

Fase 2 – Assegnamento delle chiavi ai sottoinsiemi S_i

L' algoritmo sceglie indipendentemente ed uniformemente a caso una chiave L_i per il nodo v_i, per i= 1,..., 2n-1.

La chiave assegnata al nodo v_i è quella corrispondente al sottoinsieme S_i .

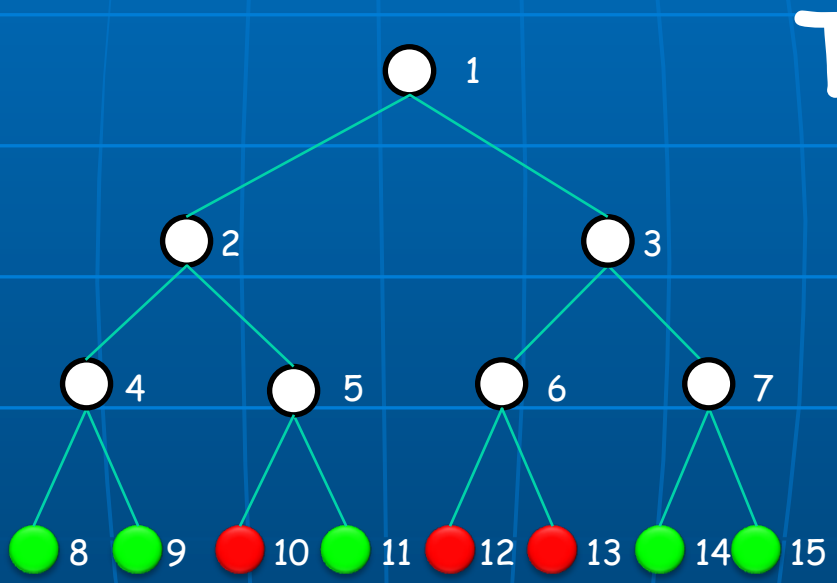
Ogni utente u riceve come informazione segreta I_u le chiavi assegnate ai nodi lungo la path u-root.

Ogni utente memorizza un numero di chiavi pari all' altezza dell' albero + 1, dunque $| I_u | = \log N + 1$.

Complete Subtree

Fase 3 – Copertura / Partizionamento

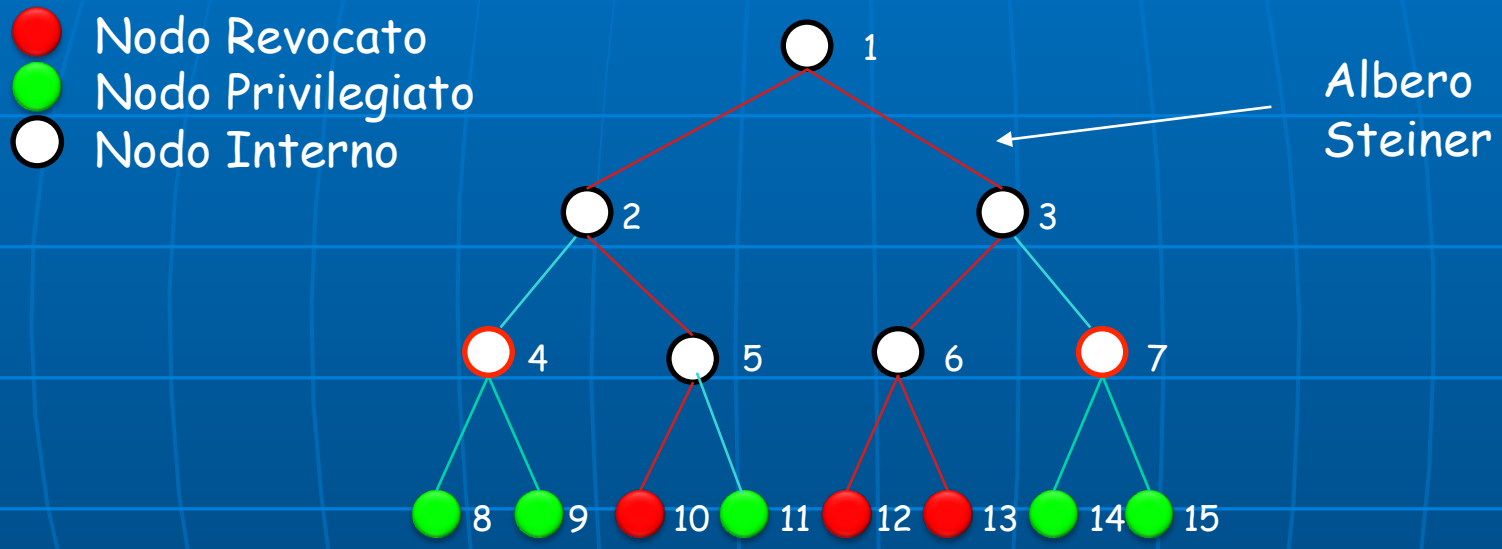
Nodo Revocato
Nodo Privilegiato
Nodo Interno



Si calcola l' albero di Steiner (cioè il sottoalbero di T minimale contenente tutte le r foglie di R).

Complete Subtree

Fase 3 – Copertura / Partizionamento



Per coprire l' insieme dei non revocati si prendono i sottoinsiemi associati ai sottoalberi le cui radici sono adiacenti a nodi dell' albero di Steiner che hanno grado uscente uguale a 1.

Complete Subtree

Fase 3 – Copertura / Partizionamento

Nodo Revocato
Nodo Privilegiato
Nodo Interno

Copertura Cs



Otteniamo: S_4 , S_{11} , S_7 .

Complete Subtree

Efficienza

Parametri su cui calcolare l' efficienza di CS:

1 Lunghezza dell' header.

2 Taglia di I_u.

3 Tempo di calcolo per u.

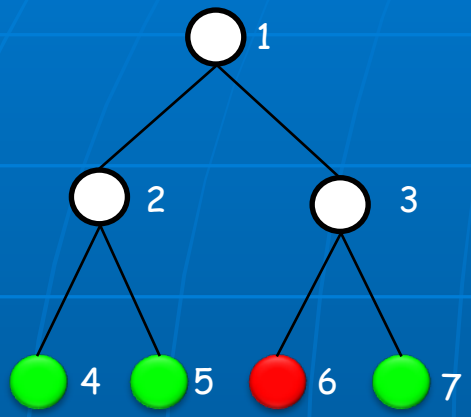
Gli ultimi 2 punti sono banali:

1 Le informazioni che ogni nodo riceve sono log N + 1 chiavi

2 Questo schema consente che un utente effettui una sola decifratura . Quindi il tempo richiesto a un utente è minimo.

Il primo punto richiede qualche considerazione in più...

Complete Subtree



...Quanto è lungo l' header?

Siccome ogni percorso da u a root è fatto da log N nodi e supponiamo ci siano r nodi revocati, l' header è al più

O(r * log N).

Una stima più precisa è

O(r * log N/r)

In particolare si può dimostrare che il # di nodi di grado uscente 1 dell' albero di Steiner è al più r * log N/r

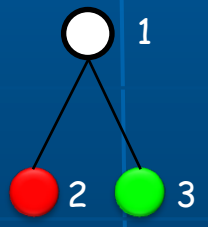
Complete Subtree

La dimostrazione avviene per induzione sull' altezza dell' albero.

Assumendo che N sia una potenza di 2, i.e., $N = 2^i$, dove i è l' altezza dell' albero, si ha:

log N\r = log N – log r = log 2^i - log r = **i – log r**

Base d' induzione:



i = 1

r * log N\r = r(i – log r)= 1(1-0)=1 ***VERO***

Complete Subtree

Ipotesi induttiva:

Supponiamo che l' affermazione sia vera per un albero *di altezza i*, dunque il numero di nodi di grado uscente 1 nell' albero di Steiner con r foglie è al più:

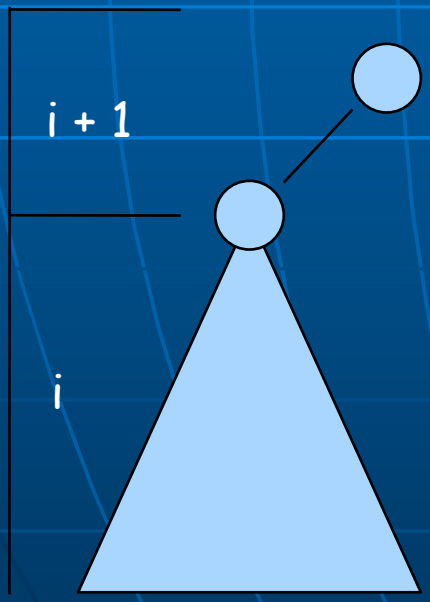
$$r * \log N/r$$

Complete Subtree

Passo induttivo (altezza i+1):

Ci sono 2 casi:

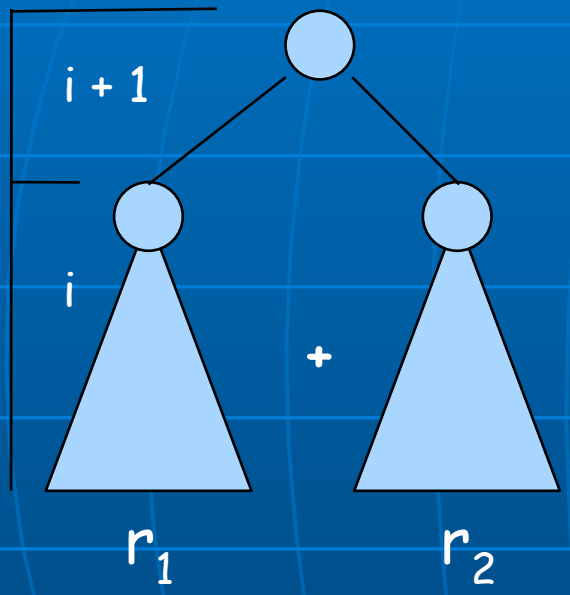
1° caso: tutte le foglie revocate sono nello stesso sottoalbero.



$$r\,(i - \log r) + 1 \leq r(i - \log r) + r$$
$$\leq r(i + 1 - \log r)$$

Complete Subtree

2° caso: sia $r = r_1 + r_2$:



Il # di nodi di grado uscente 1 nell' albero di Steiner di altezza i + 1 è uguale a

$$
\begin{aligned}
& r_1(i - \log r_1) + r_2(i - \log r_2) \\
&= r_1 i - r_1 \log r_1 + r_2 i - r_2 \log r_2 \\
&= (r_1 + r_2)i - (r_1 \log r_1 + r_2 \log r_2) \\
&\le r * i - r\,(\log r - 1) \\
&= r * i - r \log r + r \\
&= r\,(i + 1) - r \log r \\
&= r\,(i + 1 - \log r).
\end{aligned}
$$

Poichè risulta (log-sum inequality):

$$r_1 \log r_1 + r_2 \log r_2 \ge r\,(\log r - 1)$$

Riferimenti

D. Naor, M. Naor and J. Lotspiech.
Revocation and Tracing Schemes for Stateless Receivers.
Advances in Cryptology, Proceedings of Crypto'01, Lecture Notes in Computer Science, Vol. 2139, pp. 41--62, 2001.