Broadcast Encryption

Implementing the canonical Fiat, Naor [94] Paper

Liav Elisha

Advisor: Assaf Barak, Benny Pinkas

CS@BIU
המחלקה למדעי המחשב
אוניברסיטת בר-אילן

BIU Center for Research in Applied Cryptography and Cyber Security

אוניברסיטת בר-אילן
Bar-Ilan University

Preface

Paid Media Broadcast Services need to transmit on one channel to millions of parties efficiently, and be able to remove or add subscribers without distributing keys to the whole network. This was solved in 1994 by Amos Fiat, Moni Naor and is a cornerstone solution in industry.

Research Process and Sub-Goals

Understand and Implement the protocol, comparing Naïve, 1-Reilient Scheme, Low-Memory K-Resilient Scheme.

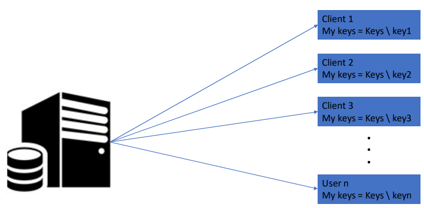
Scheme details

Scheme	Client Keys	Server Encryptions	Comm.
Naïve	1	N	M*N
1-Resilient	logN	1	M*1
LMK-R	l*logN	l*m	M*l*m

M – Message Size
N – # Clients
k – Resilient Level
l – # Functions Of One Level Schemes
m – # 1-Resilient Scheme For Each Function

1-Resilient detail

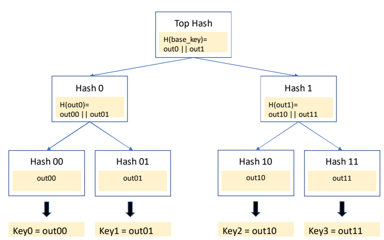
The basic scheme we define allows users to determine a common key for every subset, resilient to any set S of size = 1. The idea is very simple. For every user i , define a key K_i and give K_i to every user except i. Clearly, every user x will be missing key K_x and will therefore be unable to compute the common key for any privileged set T such that $x \notin T$.



Keys	{key1, key2, … , keyn}
key_i	Key which missing to client i
key	$key_{i_1} \oplus key_{i_2} \oplus \dots \oplus key_{i_l}$
broadcast	Msg ⊕ Key

Low-Memory by 1-way function

By the 1-Resilient scheme, every user x should get all the keys except the one associated with the singleton set B = {x}. To meet this goal remove the path from the leaf associated with the user x to the root. We are left with a forest of O(logn) trees. Give the user x the labels associated with the roots of these trees. The user can compute the all leaf labels (except K_B) without additional help.

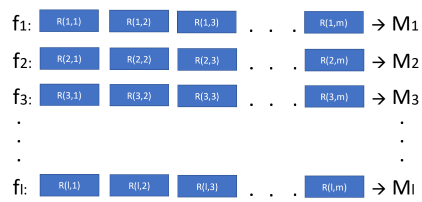


H	hash function:128bit → 256bit
base_key	leads to all keys

For example, user 1 will get <u>out 1</u> & <u>out 00</u> and compute <u>out10</u> & <u>out11</u> by himself.

K-Resilient detail

Consider a family of functions f1, f2, …, fl, fi : U→{1,2, …,m}, with the following property : $\forall_{|S|=k} S \subset U: \exists_{1 \le i \le l}\, i: \forall x, y \in S: f_i(x) \neq f_i(y)$.
This is equivalent to the statement that the family of functions {fi} contains a perfect hash function for all size k subsets of U when mapped to the range {1,. . . , m}. Such a family can be used to obtain a k-resilient scheme from a 1-resilient scheme. For every 1 ≤ i ≤ l and 1 ≤ j ≤ m use an independent 1-resilient scheme R(i, j). Every user x∈U receives the keys associated with schemes R(i, fi(x)) for all 1 ≤ i ≤ l. In order to send a secret message M to a subset T⊂U the center generates random strings $M^1, \dots, M^l$ such that $\bigoplus_{i=1}^{l} M_i = M$.
The center broadcasts for all 1 ≤ i ≤ l and 1 ≤ j ≤ m the message Mi to the privileged subset {x∈T | fi(x) = j} using scheme R(i,j). Every user x∈T can obtain all the messages $M^1, \dots M^l$ and by Xoring them get M.



R(i,j)	1 resilient scheme
F1, F2, … , Fl	Fi: U→{1,2, …,m} - a family of functions with the following property: $\forall_{|S|=k} S \subset U: \exists_{1 \le i \le l}\, i: \forall x, y \in S: f_i(x) \neq f_i(y)$
Message	$M_1 \oplus M_2 \oplus M_3 \oplus \dots \oplus M_l$

Performance

/Msg/ = 1024 bytes | # Clients = 10

<u>Naïve scheme</u> –
Encryption time - 0.002589 seconds
Transmission time & Decryption time - 0.008747 seconds

<u>1-resilient scheme –</u>
Encryption time - 0.000979 seconds
Transmission time & Decryption time - 0.001484 seconds

<u>Low memory 1-resilient scheme –</u>
Encryption time - 0.001035 seconds
Transmission time & Decryption time - 0.001183 seconds

<u>Low memory k-resilient scheme –</u>
Encryption time - R * 0.001035 seconds
Transmission time & Decryption time - R * 0.001183 seconds

R # 1-Resilient Schemes

Implementation

- Demo for N=10 for all schemes
- C++11. Linux Ubuntu
- Socket Programming
- SHA-256, AES-128-ECB
- MBED TLS library

System operation