

# Fault Tolerant and Distributed Broadcast Encryption

Paolo D Arco and Douglas R Stinson  
Univ. of Waterloo

Presented By:  
Karthik Narayan  
[narayanc@cs.fsu.edu](mailto:narayanc@cs.fsu.edu)

1

## Abstract

- Reliability and trust are the main concerns of this paper
- Improves reliability by looking at other schemes and extending them
- Objective to gain fault tolerance and to remove the need for trust in the broadcaster

2

## The practical scenario

- A movie company wants to sell a movie to a broadcaster
- Broadcaster wants to make money and hence decides to illegally copy the movie and later use it
- The question : Can the Movie company efficiently use a pay per view scheme without trusting the broadcaster ?
- One solution is for the movie company to directly broadcast the film, which is impractical.

3

## A likely solution

- The movie company creates  $n$  servers that are placed at the broadcaster's site.
- Each server stores a part of the movie and a key
- Run an algorithm with the privileged set of users and broadcast the movie
- The privileged users get parts of the movie from each server using the decryption key

4

## Notations used

- Let  $U$  be the set of users  $U_1 \dots U_m$  be the set of  $m$  users
- Let  $S_1 \dots S_n$  be a set of  $n$  servers
- $N$  can be relatively small  $< 10$
- $M$  will be relatively large like a million
- Every user has a secure point to point channel with the server
- Servers and users have access to a common broadcast channel

5

## The setup phase

- The movie company chooses a random key  $k$
- It uses a  $(t,n)$  secret sharing scheme to compute  $n$  shares of key  $K$  named  $y_1, y_2 \dots y_n$
- The server runs an independent setup phase of a BES  $\Sigma$  with the users.

6

## Broadcast Phase

- The movie is split into  $n$  parts  $stream_1, stream_2 \dots stream_n$
- Each part is encrypted by computing  $E_k(stream_i)$   $i=1$  to  $n$
- This is sent to the various servers where  $S_i$  gets  $E_k(stream_i)$   $i=1$  to  $n$
- Every server  $S_i$  using  $\Sigma$  broadcasts the value  $y_i$

7

## Broadcast contd.

From any subset  $t$  out of  $n$  values of  $y_i$ , each user recovers the decryption key  $k$

Every server  $S_i$  broadcasts  $E_k(stream_i)$

Every privileged user collects the  $n$  parts rearranges them and decrypts them to get the whole movie

8

## Problems to the scheme

- If a server crashes, the movie will never be complete
- The unavailability of one server could stall the whole process

9

## Solution

- Split the movie into packets and let at least two servers broadcast the same packet at the same time.
- The author assumes that there will never be an opportunity when both the mirror servers will crash.

10

## What follows ?

Takes an in depth look at other schemes

Identifies a few schemes that could be applied to this scenario (like traitor tracing, broadcast key distribution)

Speaks of the advantages of adding them to modify this scheme.

11

## Questions ?

12

