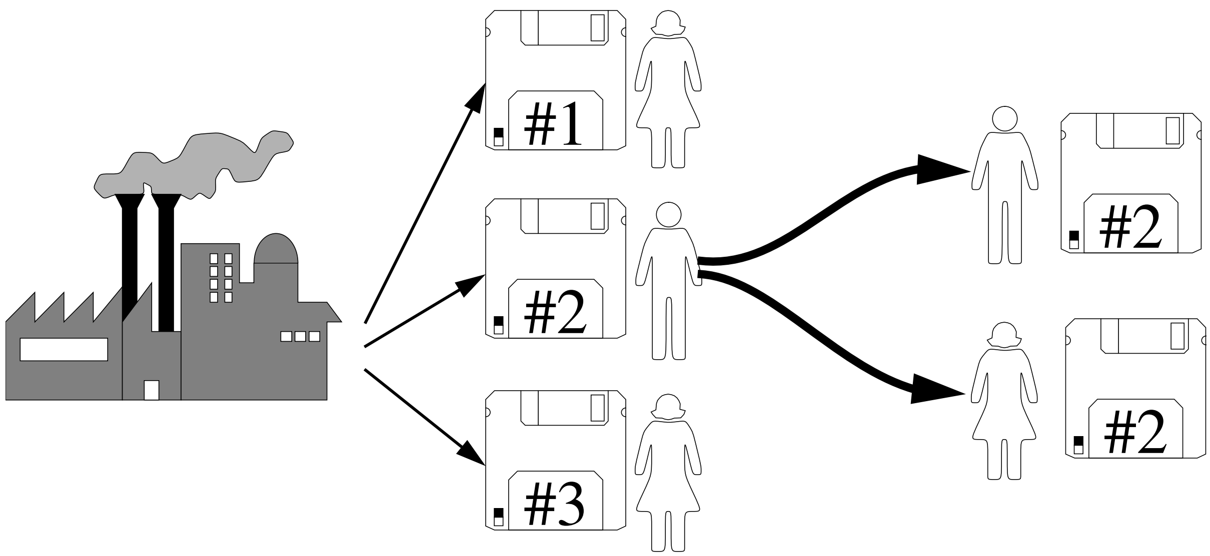
Collusion Secure Fingerprinting for Digital Data

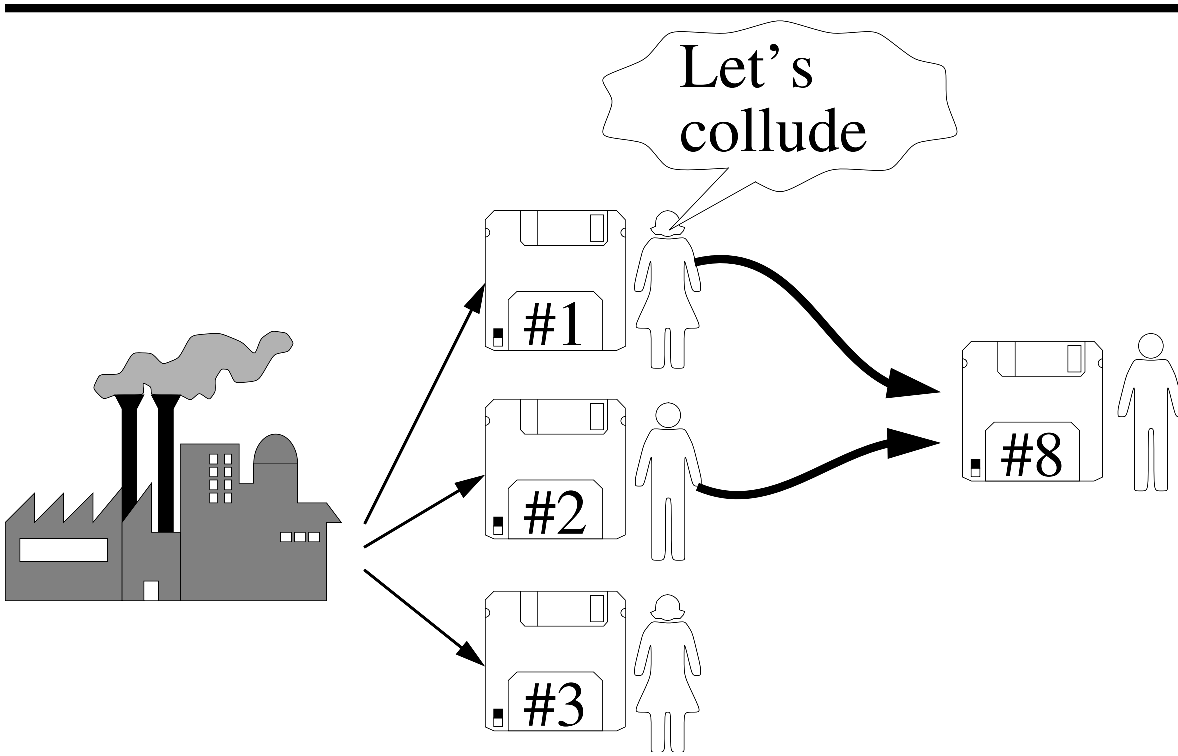
Dan Boneh and James Shaw

Princeton University

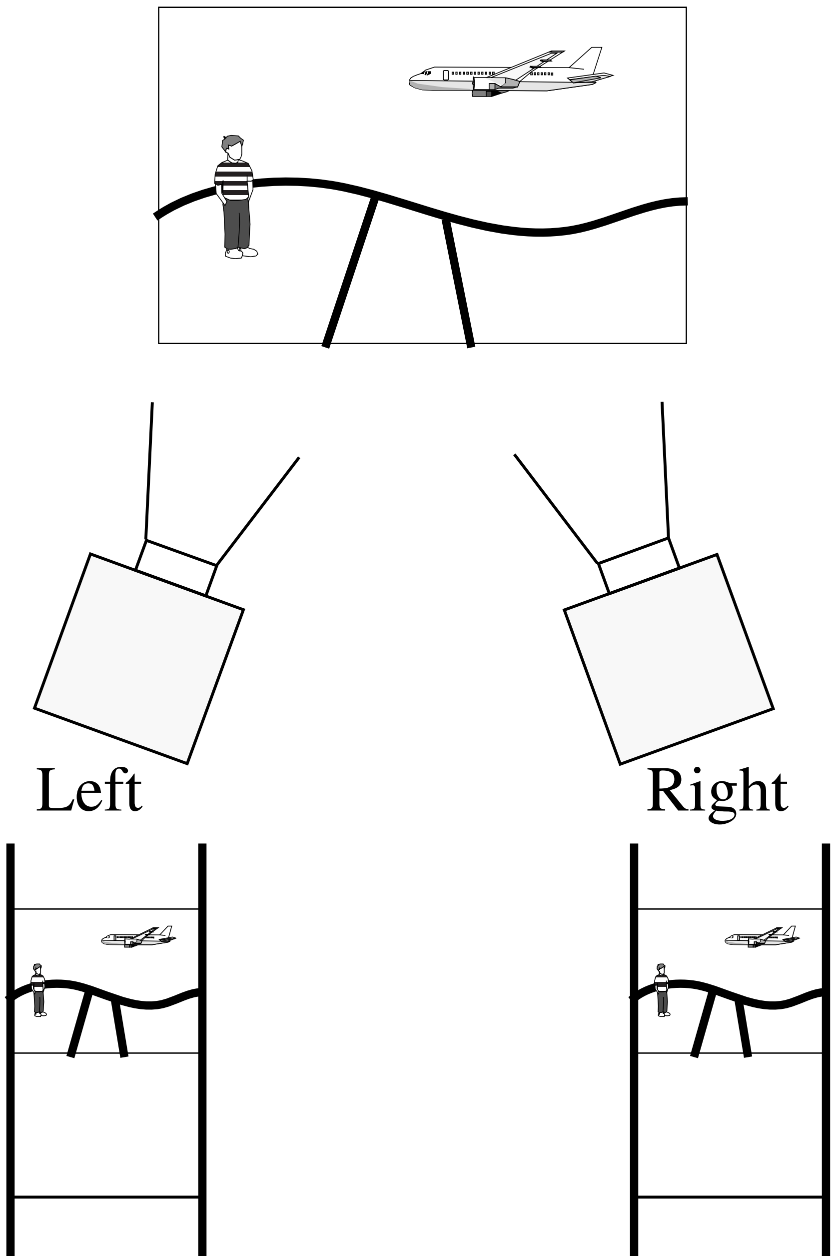
(In proceedings Crypto 95)

Fingerprinting





Marking movies



A fingerprint is a collection of marks.

Challenges

Embedding fingerprints in digital data introduces new challenges:

➢Hide fingerprint in data.

➢Secure against collusion.

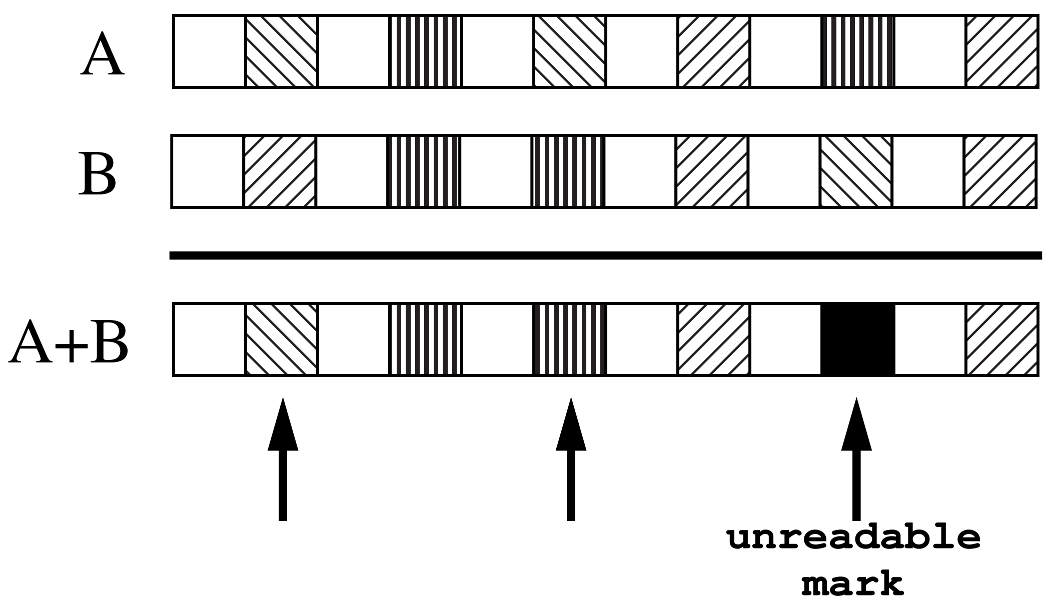
➢Efficient distribution.

Fingerprinting Domains

- ➢Video - movies
- ➢Video - images: Digimarc, NEC, ...
- ➢Audio: Digimarc
- ➢Documents: A T&T
- ➢Executables (?)
- ➢Raw data - lists, financial data, ...

Collusion

By colluding users can create
unregistered copies.



Detectable marks: Those marks which a coalition of users can detect.

Fingerprinting codes

Suppose each mark has k different states.

Each user is a assigned a codeword over the alphabet $\{1, \ldots, k\}$.

Users generate new words by colluding:

A:	1	2	1	3	2	3
B:	3	2	2	3	1	3
Example:	$\underline{\boldsymbol{1}}$	2	$\underline{\boldsymbol{2}}$	3	$\underline{\mathbf{?}}$	3
General:	$\overset{13}{?}$	2	$\overset{12}{?}$	3	$\overset{12}{?}$	3

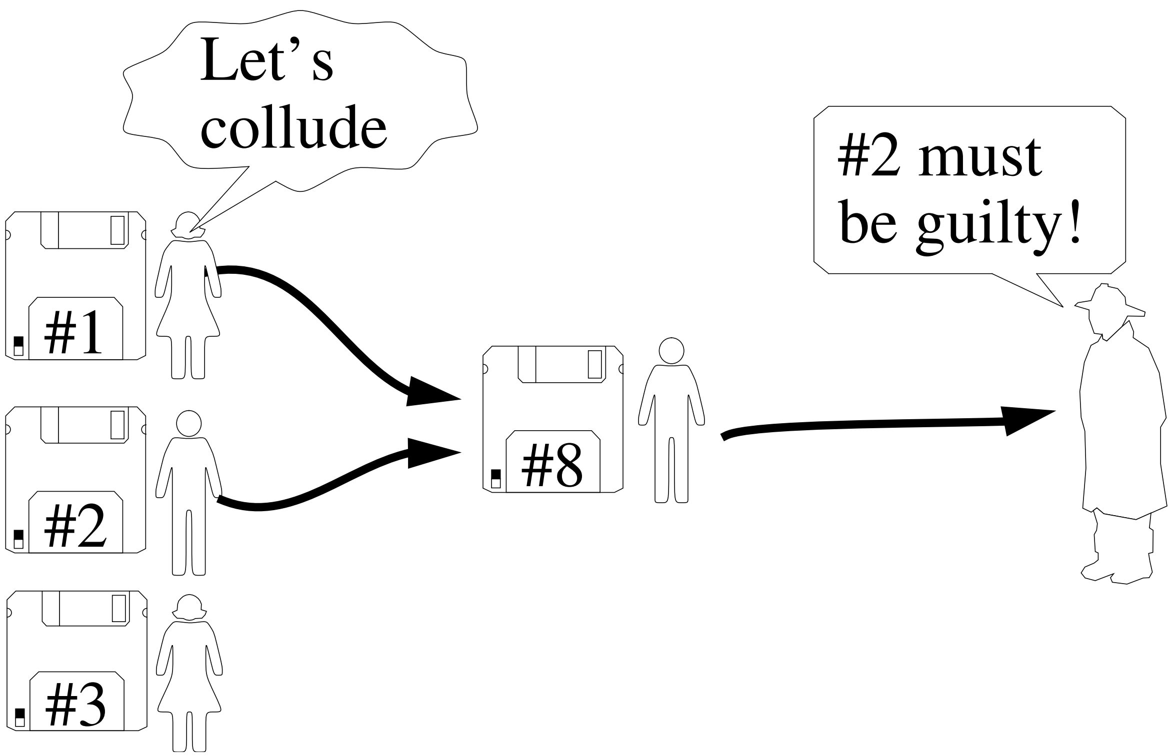
Note: users can make marks unreadable.

Collusion secure codes

Definition:

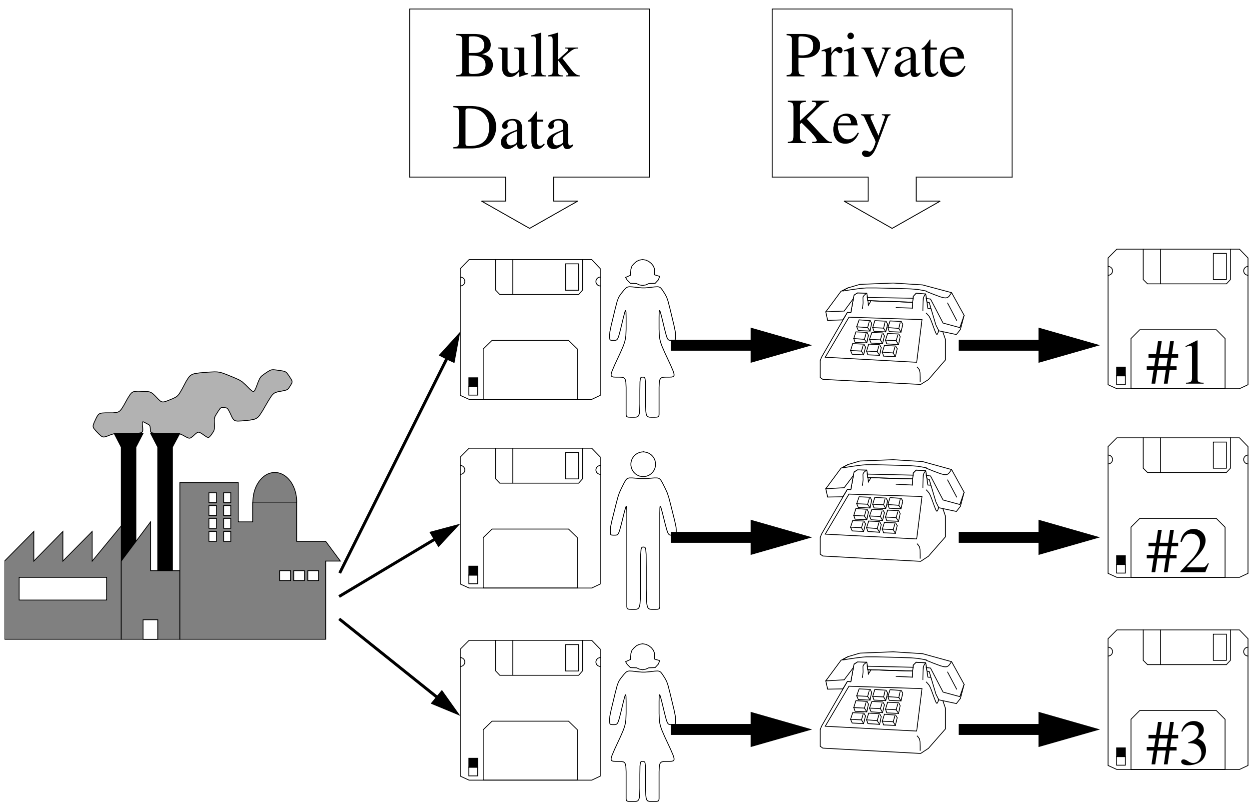
A code Γ is collusion secure if there is a tracing algorithm A such that: if a coalition C generates word x then

$$A(x) \in C$$



Distribution problem

How can fingerprinted data be distributed efficiently?



Bulk data and private key should give no information about marks.

A Distribution Scheme

Let $\Gamma \subseteq \{0, 1\}^n$ be a fingerprinting code.

Partition the data into n blocks.

Each block will receive one mark.

$B_{1,0}$	$B_{2,0}$	....	$B_{n,0}$
$B_{1,1}$	$B_{2,1}$	....	$B_{n,1}$

For each block $i \in [1, n]$ and $m \in \{0, 1\}$

choose a key $k_{i,m}$.

Bulk data: $E(B_{i,m}, k_{i,m})$ for all i, m.

Private key: for $x \in \Gamma$ set $P_x = \{k_{i, x_i}\}$.

An Improvement

Let L =size of data; n =length of code.

Previous scheme: $(2L, O(n))$

bulk = $2L$; Private = $O(n)$.

A better scheme gets: (3L, O(n/logL)).

Private key is shorter .