Copy protection & Statistics

teddy.furon@thomson.net
teddy.furon@inria.fr
Introduction: problem

- New trend in copy protection
  - Fight against illegal redistribution of content.
    - content = Hollywood movies
  - Find the identity of the hackers amongst $n$ users.
  - Dissuasive weapon.
  - a.k.a. : fingerprinting, content serialization, user forensics, transactional watermarking...
Main ideas:
Cutting, versioning & switching

VoD server → Peter → Paul → Jack
Introduction: the collusion

- **Block exchange:**
  - Colluders cannot create version they don’t have.
  - The \( i \)-th block in the pirated copy is one of the \( i \)-th blocks from the colluders (« marking assumption »)

\[
\begin{array}{l}
\text{Peter} & 0 & 1 & 0 & 0 & 1 & 1 \\
\text{Paul} & 1 & 0 & 0 & 1 & 0 & 1 \\
\text{Jack} & 0 & 0 & 1 & 0 & 1 & 1 \\
\text{Pirated copy} & 1 & 1 & 1 & 1 & 0 & 1 \\
\end{array}
\]
Outlines

- **Introduction**
- **Traitor tracing**
  - How to design the code?
  - How to accuse guilty people?
- **Digital watermarking**
  - How to create two versions of a block?
- **False positive probability estimation**
Traitor tracing

● Requirements
  – $n$ users, $c$ colluders, $m$ binary code length

● Code construction
  – $X$ binary matrix $n \times m$ (set $X \subset \mathcal{B}^m$)
  – $x_j$ codeword given to user #j
  – $x_{ji}$ $i$-th bit of the $j$-th codeword

● Collusion
  – Input: $C = \{x_1, \ldots, x_c\} \subset X$
  – Output: $y \subset \mathcal{B}^m$ pirate sequence

● Accusation
  – Input: $y$ pirate sequence
  – Output: $G$ set of guilty users
Cryptographic contributions

- Problem statement [Fiat&Naor]
- Terminology (old) [Pfitzmann]
  - Ex.: Frameproof code \( \text{\#} C \mid y \in X \setminus C \)
- Relationship with Error Correcting Codes [Stinson]
  - Pirate sequence = codeword + noise
  - Accusation = correcting errors
  - Not efficient: very very very long code
- Relaxation of the constraint [Boneh&Shaw]
  - \( P_{fa} \) Probability of accusing at least one innocent
  - \( P_{mi} \) Probability of missing all colluders
- Non constructive theorem [Peikert03]
  \[ m \geq O( c^2 \log( n.P_{fa}^{-1}) ) \]
A revolution coming from the Statistics

- Probabilistic codes [Gabor Tardos]

- The first exhibition of a code achieving the Peikert bound

\[ m = 100 \ c^2.\log( n.P_{fa}^{-1}) \]

- An unknown genius… work rediscovered 2 years after.
- No rationale, no clue, no intuition except

  « the full power of randomization »

- Extremely simple: 10 lines of matlab
- Extremely flexible: \( n, m \) loosely tightened
- Constant ‘100’ raised suspicion
Tardos codes

- **Initialization**
  - Draw randomly: \( p_i \in [0,1] \), \( i=1:m \), i.i.d., \( p \sim f(p) \)
  - \( p = (p_1, \ldots, p_m) \) is the secret of the code
  - \( f(p) = 1 / \pi(p(1-p))^{1/2} \)
Tardos accusation

- **Accusation**
  - Accuse user \( j \) if \( S_j > T \)

\[
S_j = \sum_i g(y_i, x_{ji}, p_i)
\]

- \( g(0,0,p) = + \ ( p/(1-p) )^{1/2} \)
- \( g(0,1,p) = - \ ( (1-p)/p )^{1/2} \)
- \( g(1,0,p) = - \ ( p/(1-p) )^{1/2} \)
- \( g(1,1,p) = + \ ( (1-p)/p )^{1/2} \)
Tardos code

Why does it work?
Mathematical model of the collusion

- **Assumptions about the collusion**
  - **Memoryless**
    \[ y_i = F_i(x_{1i}, \ldots, x_{ci}) \]
  - **Stationary**
    \[ y_i = F(x_{1i}, \ldots, x_{ci}) \]
  - **Permutation Invariant**
    \[ y_i = F(s_i) \quad s_i = \sum_j x_{ji} \]
  - **Probabilistic**
    \[ \theta_s = \text{Prob}(y = 1 \mid s) \]

- **Model**
  \[ \theta = (\theta_0, \theta_1, \ldots, \theta_c) \]

**Marking assumption**
\[ \theta_0 = 0 \quad \text{and} \quad \theta_c = 1 \]

Therefore, the collusion indeed lies in \([0,1]^{c-1}\)
Mathematical model of the collusion

- **Using the model**

\[
\text{Prob}[y=1 \mid p] = \sum_s \text{Prob}[y=1,s\mid p] = \sum_s \theta_s \cdot (s^c) \ p^s \ (1-p)^{c-s}
\]

\[
\text{Prob}[y=1 \mid x=1, p] = \sum_s \theta_s \cdot (s^{-1} c^1) \ p^{s-1} \ (1-p)^{c-s}
\]

- **1\textsuperscript{st} and 2\textsuperscript{nd} order statistics**
  - Innocent: \( \mathbb{E} [ S_j ] = 0 \quad \mathbb{E} [ S_j^2 ] = m \)
  - Colluder: \( \mathbb{E} [ S_j ] = 2m/\pi c \quad \mathbb{E} [ S_j^2 ] = m \)

( here: \( \mathbb{E} [ . ] = \mathbb{E} \ p[\mathbb{E} \ x[\mathbb{E} \ y[.] ] ] ] \))

- **Miracle**: independent from \( \theta \)
- **Markov bound**: \( m = 100 \ c^2.\log( n. P_{fa}^{-1}) \)
- **Asymptotics**: CLT – Scores are Gaussian distributed
Statistical interpretations (I)

- The collusion process is a **nuisance parameter**
- The hypothesis test is based on a **pivotal quantity**...
- ... at least up to 1\textsuperscript{st} and 2\textsuperscript{nd} order statistics.

\[
\begin{align*}
  f(p) &= \frac{1}{\pi(p(1-p))^{1/2}} \\
  g(0,0,p) &= + \left( \frac{p}{1-p} \right)^{1/2} \\
  g(1,1,p) &= \ldots
\end{align*}
\]

\[
\begin{align*}
  \mathbb{E}[ S_j ] &= 0; \quad \mathbb{E}[ S_j^2 ] = m \\
  \mathbb{E}[ S_j ] &= 2m/\pi c; \quad \mathbb{E}[ S_j^2 ] = m
\end{align*}
\]
One collusion process – Many code densities

- Dense ($p=1/2$): worst attack: minority vote 3 colluders
- Sparse ($p=0^+ \text{ or } 1^-$): worst attack: majority vote

The sequence $p$ is THE secret!

« How secret is this secret? »

- The colluders could estimate it: $p_i = s_i / c$
- The colluders know $f(p)$: a priori distribution.

Wrong idea: already captured by our model.

Shed more light on Tardos choice:

- $f(p)$ is the Jeffreys prior, the less informative prior distribution
- The less useful for the colluders.
Statistical interpretation (III)

- **Hypothesis test**
  
  - $H_0$: Innocent  \( \text{Prob}(y,x|p) = \text{Prob}(y|\theta,p) \cdot \text{Prob}(x|p) \)
  
  - $H_1$: Colluder  \( \text{Prob}(y,x|p) = \text{Prob}(y|x,\theta,p) \cdot \text{Prob}(x|p) \)
  
  - Performance criterion
    \[
    R ( f ; \theta ) = \mathbb{E}_P \left[ \text{D}_{KL}( H_1 ; H_0 | P, \theta ) \right]
    \]

- **Game theory**
  
  - Between designer and colluders
  
  - MaxMin game
    \[
    R ( f^* ; \theta^* ) = \max_f \min_{\theta} R ( f ; \theta )
    \]

- **Asymptotically, \( c \to \infty \):**
  
  - Equilibrium: \( f^*(p) = (\pi^2 p(1-p))^{-1/2} \) and \( \theta^* = (0, c^{-1}, 2c^{-1}, \ldots, 1) \).
Trends

● New accusation strategy
  – « Learn and Match »
  – Estimate $\theta$ and use Likelihood ratio test to accuse (E.-M.)

● K-uplets scores
  – Inf. theory:
    \[ l(y ; x|\theta) \leq l(y ; \{x_1, x_2\} | \theta) \leq \ldots \leq l(y ; \{x_1, \ldots, x_c\} | \theta) \]
  – Practical? Complexity in $\sim O(n^c)$

● Q-ary alphabet

● Other collusion models
  – Erasures (cut movie scenes)
Outlines

- **Introduction**
- **Traitor tracing**
  - How to design the code?
  - How to accuse people?
- **Digital watermarking**
  - How to create two versions of a block?
- **False positive probability estimation**

```
User i → Code (0 1 1 0 ....) → Collusion (1 1 0 0 ....) → Accusation → i
```

Embedding → **Collusion** → Decoding
Definition of digital watermarking

- Data hiding: art and science of hiding data in multimedia digital contents.
- A hypothesis test problem ...

... under some special constraints:
- non perceptibility (watermark, filigrane in French)
- robustness (tatouage in French)
- security
Assumptions: feature extraction

- **Watermark Embedding**
  - From a content block $B$, extract some meaningful features
    $$s = \text{Ext}(B), \quad \text{with } s \in \mathbb{R}^L$$
  - Modify $s$ into $x = s + w$ \quad s.t. $||w||^2 \leq L.P_w$
  - The vector $w$ is the secret of the watermarking scheme
  - Put back the features into the content
    $$B_w = \text{Ext}^{-1}(x, B)$$

- **Attack on the watermarked image**
  - Distort $x$ into $z = x + n$ \quad s.t. $\mathbb{E} ||n||^2 \leq L.P_n$

- **Detection**
  - $H_0$: $r_0 = s + n$ (given by Nature)
  - $H_1$: $r_1 = s + w + n$
Naïve idea

**Gaussian setup**

\[ r_0 = s + n \sim \mathcal{N}(0, P_s+P_n) \quad \text{vs.} \quad r_1 = s + w + n \sim \mathcal{N}(w, P_s+P_n) \]

- Performances limited by the Kullback-Leibler distance

\[ D_{KL}(r_0 \| r_1) = L.P_w / 2 (P_n+P_s) \]

- Data processing theorem, Stein Lemma.

\[ ||w||^2 \leq L.P_w \]
An Information theoretic revolution

**Gaussian setup**

\[ r_0 = s + n \sim \mathcal{N}(0, P_s + P_n) \quad \text{vs.} \quad r_1 = s + w(s) + n \]

- Performances limited by the Kullback-Leibler distance

\[ \max_{w(.)} D_{KL}(r_0 \| r_1) = ??? \]
The informed setup

- **Gaussian setup**
  - $r_0 - s = n \sim \mathcal{N}(0, P_n)$ vs. $r_1 - s = w + n \sim \mathcal{N}(w, P_n)$
  - Performances limited by the Kullback-Leibler distance
    \[
    D_{KL}(r_0 || r_1) = P_w / 2.P_n
    \]
The side informed setup

- **Gaussian setup**
  - Performances limited by the Kullback-Leibler distance
    \[ \frac{P_w}{2(P_n + P_s)} \leq \max D_{KL}(r_0 \| r_1) \leq \frac{P_w}{2P_n} \]

- **Philosophical question**
  - Is \( s \) a channel noise or a channel state?

\[ E_s \| w(s) \|^2 \leq L.P_w \]
Suppose the following watermark embedding

\[ x = s + w(s) = s + (\alpha - \lambda s^T u)u \]

with \( ||u||^2 = 1 \)

- \( \lambda \in [0,1], \alpha > 0 \)
- “Push and cancel” mixed strategy
- Power constraint:

\[ \alpha^2 + \lambda^2 P_s = L P_w \]

**KL-distance**

- \( D_{KL}(r_0 || r_1) = F(\lambda, \alpha) = F(\lambda) \).
- Optimize : \( \lambda^* = \arg \max_\lambda F(\lambda) \)
- Take to the limit

\[ \lim_{L \to \infty} F(\lambda^*) = P_w / 2 P_n \]
How to put this into practice?

- Gaussian setup with fixed length $L$
  - How to maximize $D_{KL}$?
  - How to design the detector? [Merhav]

- Real world
  - Nuisance parameters: $P_s$, $P_w$, $P_n$, type of attack
  - No longer Gaussian r.v.
  - No longer Euclidean distance, but perceptual distance
Outlines

- **Introduction**
- **Traitor tracing**
  - How to design the code?
  - How to accuse people?
- **Digital watermarking**
  - How to create two versions of a block?
- **False positive probability estimation**

![Diagram](image_url)
False positive estimation

2 techniques have a common issue: very very small $P_{fa}$

- Traitor tracing: probability to accuse an innocent.
  \[ P_{fa} = \text{Prob} [ d(x) > T ] \]
  - with $d(.)$ Tardos scoring
  - $x$ the sequence of a user, $\text{Prob}(x_j) = \prod_i p_i^{x_{ji}} \cdot (1-p_i)^{(1-x_{ji})}$

- Watermarking: probability of detecting the mark in a non watermarked content
  \[ P_{fa} = \text{Prob} [ d(x) > T ] \]
  - with $d(.)$ likelihood of being watermarked
  - $x$ features extracted from a content block, $x \sim p_x$
Monte Carlo Method estimation

● Structure of the detector
\[ x \in \mathbb{R}^L \xrightarrow{\sim p_X} \text{score} \xrightarrow{d(x) \in \mathbb{R}} \text{threshold } T \xrightarrow{Y / N} d(x) > T \]

● MCM estimation
  – Run \( n \) experiments
  – Increment \( k \) when \( d(x) > T \)
  – Estimate \( \hat{P}_{fa} = k/n \)

● Issues
  – \( k \neq 0 \)
  – Relative std of \( \hat{P}_{fa} = (P_{fa} \cdot n)^{-1/2} \)

\[ n = O(1/P_{fa}) \quad n \sim 100/P_{fa} \]

\[ \Rightarrow \text{The smaller the probability, the harder its estimation} \]
Geometric interpretation
Key idea of our algorithm

Divide and Conquer

\[ P_{fa} = \Pr(A) = \Pr(A, A_{N-1}) \]
\[ = \Pr(A|A_{N-1}) \cdot \Pr(A_{N-1}) \quad \text{if } A \text{ implies } A_{N-1} \]
\[ = \Pr(A|A_{N-1}) \cdot \Pr(A_{N-1}|A_{N-2}) \cdot \Pr(A_{N-2}|A_{N-3}) \ldots \Pr(A_1) \]
\[ A \Rightarrow A_{N-1} \Rightarrow A_{N-2} \Rightarrow \ldots \Rightarrow A_1 \]

\[ P_{fa} = \Pr(d(x) \geq T) \]
\[ = \Pr(d(x) \geq T|d(x) \geq T_{N-1}).\Pr(d(x) \geq T_{N-1}|d(x) \geq T_{N-2})\ldots\Pr(d(x) \geq T_1) \]
\[ T > T_{N-1} > T_{N-2} > \ldots > T_1 \]

\[ \hat{P}_{fa} = \hat{a}_N \cdot \hat{a}_{N-1} \ldots \hat{a}_1 \]
Our estimator

● Inputs
  – Distribution of input data $p_X$, score $d(.)$, threshold $T$

● Outputs
  – Estimation $\hat{P}_{fa}$ of $\Pr(d(x)>T)$ for $x \sim p_X$

● Ingredients: 3 subroutines
  – SCORE
    ● Function $d(.)$: $\mathbb{R}^L \rightarrow \mathbb{R}$
    ● ‘Smooth’
  – GENERATE
    ● Generate input data $x$ distributed $\sim p_X$
  – MODIFY
    ● $y = f(x)$, random function
    ● such that $y \sim p_X$ and $y$ is ‘close’ to $x$
Our estimator

- **Divide and conquer**
  \[ P_{fa} = \Pr(d(x) \geq T | d(x) \geq T_{N-1}) \cdots \Pr(d(x) \geq T_{j+1} | d(x) \geq T_j) \cdots \Pr(d(x) \geq T_1) \]

- **Initialization**
  - ‘small’ Monte Carlo Method Estimator
  - Generate \( n \) input data (particles) \( x^{(1)} \sim p_x \)
  - Count the number of times the score is above the 1\(^{st} \) threshold

\[
k_1 = \left| \{ x_{i}^{(1)} \mid d(x_{i}^{(1)}) > T_1 \} \right|
\]

\[
\hat{a}_1 = \frac{k_1}{n}
\]
Our estimator

- **Divide and conquer**

  \[ P_{fa} = \Pr(d(x) \geq T | d(x) \geq T_{N-1}) \cdots \Pr(d(x) \geq T_{j+1} | d(x) \geq T_j) \cdots \Pr(d(x) \geq T_1) \]

- **Iteration \( j \rightarrow j+1 \)**
  - We start with \( k_j = |\{x_i^{(j)} | d(x_i^{(j)}) > T_j\}| \) particles in region \( A_j \).
  - DUPLICATE: We randomly select \( n \) particles in this set
  - MODIFY: \( z = f(x_i^{(j)}) \)
  - SELECTION:
    - If \( d(z) > T_j \) then \( x_i^{(j+1)} = z \)
    - Else \( x_i^{(j+1)} = x_i^{(j)} \)
  - We now have \( n \) particles \( \sim p_x \) in \( A_j \).
Our estimator

- **Iteration** $j \rightarrow j+1$

- **Threshold (or MCM estimator)**

  - Count the number of times the score is above the $(j+1)^{th}$ threshold

  $$
  k_{j+1} = \left| \{ x_{i, (j+1)} \mid d(x_{i, (j+1)}) > T_{j+1} \} \right|
  $$

  $$
  \hat{a}_{j+1} = k_{j+1}/n
  $$
Our estimator

- Geometric interpretation
Our estimator

- Geometric interpretation
Our estimator

- **Last trick**
  - How to define the thresholds $T_j$?
  - Variance of the estimation $P_{fa}$ is minimized if $a_j = \text{cte}$
  - Inverse: $T_j$ is the $k$-th biggest score out of $n$: $\hat{a}_j = k/n$

- **Divide and conquer**
  \[ P_{fa} = \Pr(d(x) \geq T \mid d(x) \geq T_{N-1}) \ldots \Pr(d(x) \geq T_{j+1} \mid d(x) \geq T_j) \ldots \Pr(d(x) \geq T_1) \]

- **Stopping condition**
  - When $T_N > T$.
  - Count the number $k'$ of scores really above $T$
  - $\hat{P}_{fa} = (k/n)^{N-1} \cdot k' / n$
Properties

- **Small number of trials**
  \[ nN = n[\log P_{fa}^{-1} / \log (k/n)^{-1}] \]

- **Asymptotic consistency**
  \[ \hat{P}_{fa} \rightarrow P_{fa} \quad \text{as} \quad n \rightarrow \infty \]
  (almost surely)

- **Asymptotic Normality**
  \[ n^{1/2} (\hat{P}_{fa} - P_{fa}) \rightarrow \mathcal{N}(b, \sigma^2) \quad \text{as} \quad n \rightarrow \infty \]
  (in law)

- **Asymptotic statistics**
  - Relative std in \( O( (\log(P_{fa}^{-1})/n)^{1/2} ) \)
  - Fast vanishing relative bias in \( O(1/n) \)
Experiment #1: Digital Watermarking

**Toy example**
- $x \sim p_x$ isotropic (for instance, white Gaussian noise $\mathcal{N}(0, I_L)$)
- $d(x) = x^T u / \|x\|$
- Ground truth: $P_{fa} = 1 - \text{IncBeta}(T^2, 1/2, (L-1)/2)$

**Ingredients**
- GENERATE: Matlab randn
- MODIFY:
  
  $$y = (x+qn)/(1+q^2)^{1/2}$$
  $$n \sim N(0, I_L)$$

  - $q$ fixes the strength of the modification.
Experiment #1: Asymptotic normality

$L = 20$
$T = 0.95$
$P_{fa} = 4.7 \cdot 10^{-11}$

$k/n = 3/4$
$n = 50,000$
200 runs
Experiment #2: Estimating minimum length

Tardos
\[ m = 100 \cdot c^2 \log(n/P_{fa}) \]

Experimental
\[ m \approx K_1 \cdot c^2 \log(n/P_{fa}) + K_0(c) \]

with
\[ K_1 \approx 7.6 \]
Conclusion