Copy protection & Statistics

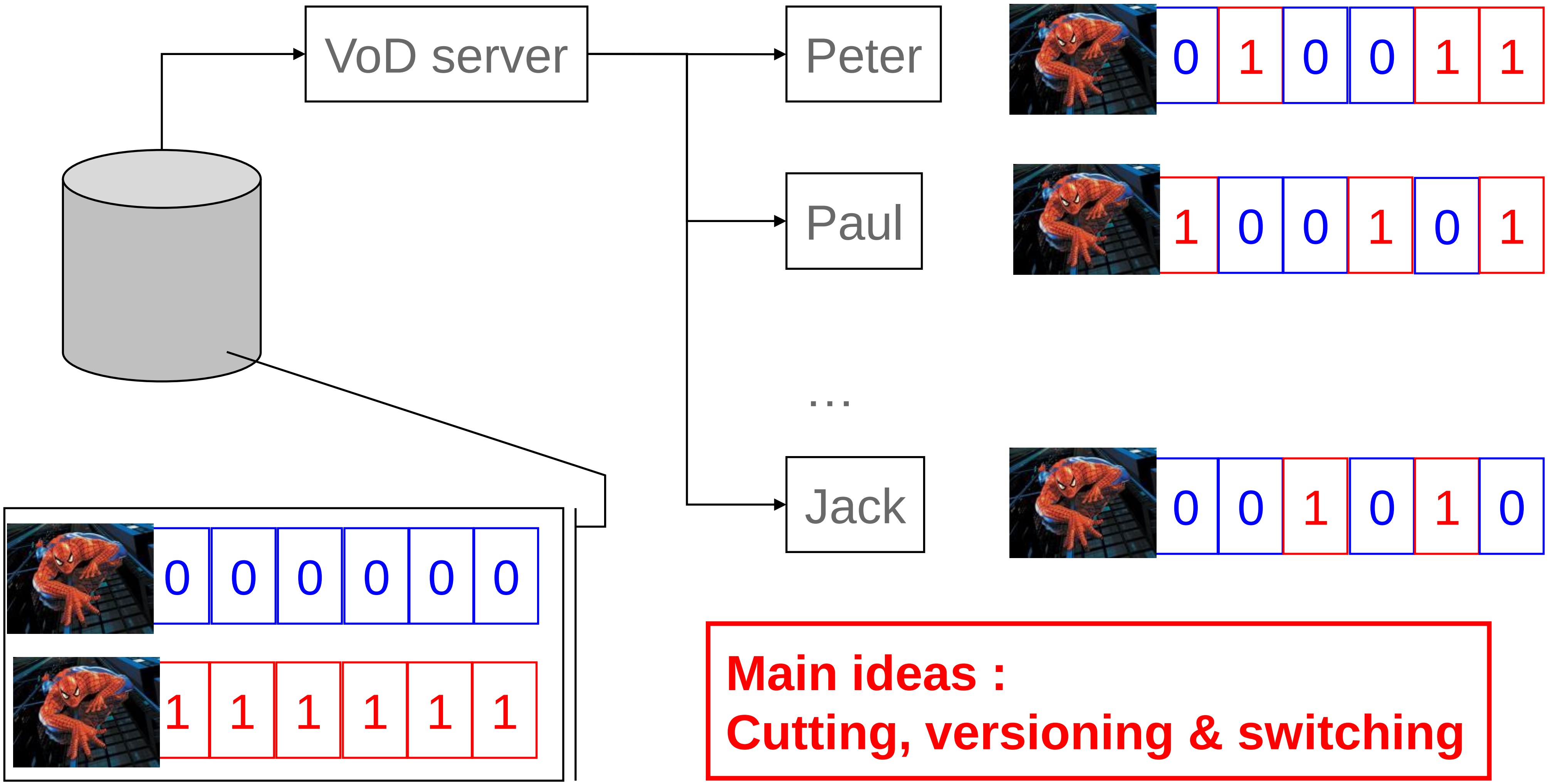
teddy.furon@thomson.net

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…*

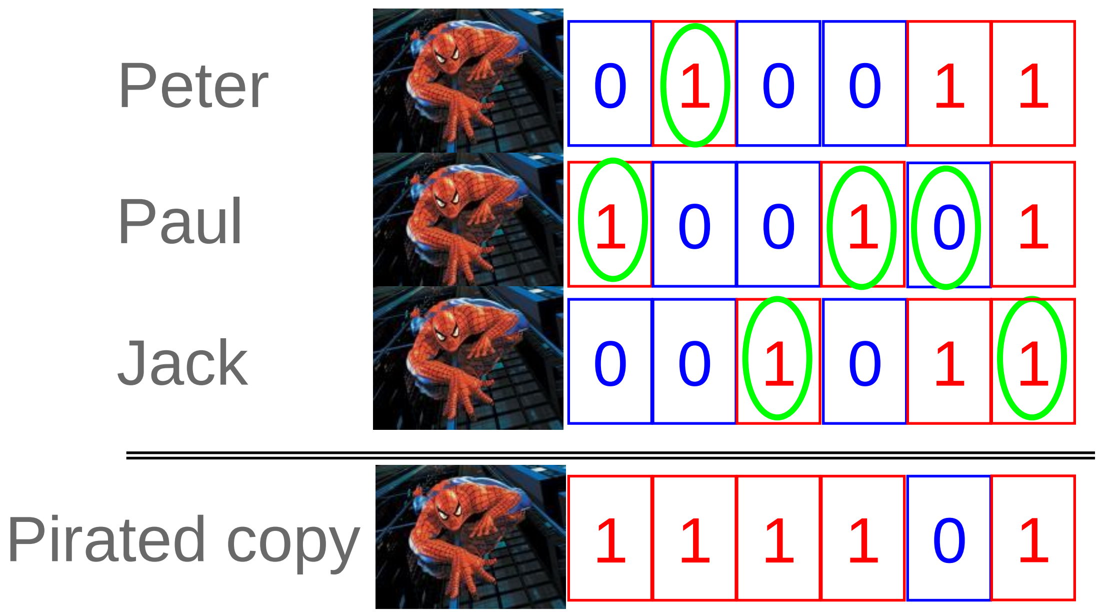
Introduction: typical scenario



Main ideas :
Cutting, versioning & switching

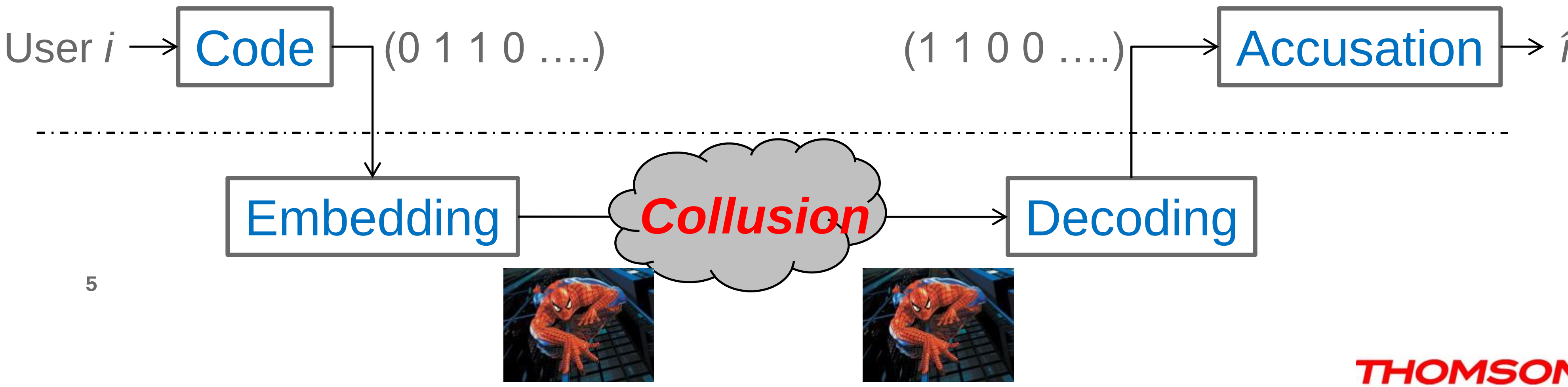
Introduction: the collusion

- **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 »)



Outlines

- Introduction
- **Traitor tracing**
  - How to design the code?
  - How to accuse gilty 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**
  - $\mathbf{X}$ binary matrix n x m (set $\mathcal{X} \subset \mathcal{B}^m$)
  - $\mathbf{x}_j$ codeword given to user #j
  - x_{ji} *i-th bit of the j-th codeword*
- **Collusion**
  - Input: $\mathcal{C} = \{\mathbf{x}_1, \ldots, \mathbf{x}_c\} \subset \mathcal{X}$
  - Ouput: $\mathbf{y} \subset \mathcal{B}^m$ pirate sequence
- **Accusation**
  - Input: $\mathbf{y}$ pirate sequence
  - Output: $\mathcal{G}$ set of guilty users

Cryptographic contributions

- **Problem statement** **[Fiat&Naor]**
- **Terminology (old)** **[Pfitzmann]**
  - Ex.: Frameproof code $\nexists \mathcal{C} \mid \mathbf{y} \in \mathcal{X} \setminus \mathcal{C}$
- **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]**

$$m \gtrsim O(c^2.\log(n.P_{fa}^{-1}))$$

A revolution coming from the Statistics

- **Probabilistic codes** **[Gabor Tardos]**
  - The first exhibition of a code achieving the Peikert bound

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

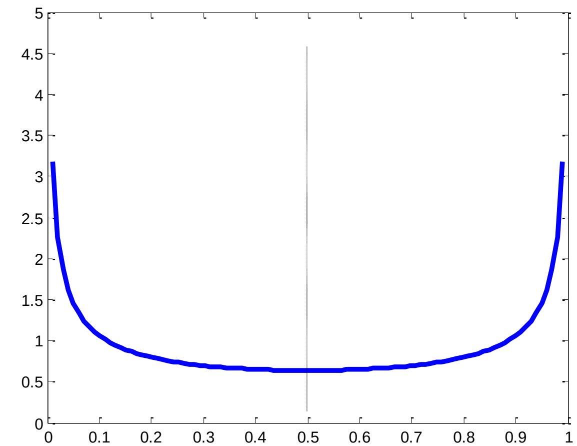
  - An unknown genius… work rediscovered 2 years after.
  - No rationale, no clue, no intuition except

    « *the full power of randomization* »
  - Extremely simple : 10 lines of matlab
  - Extremely flexible : n, m loosely tightened
  - Constant ‘100’ raised suspicion

THOMSON

Tardos codes

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



- **Code construction**
  - Draw randomly: $x_{ji} \in \{0,1\}$, $\text{Prob}[x_{ji} = 1] = p_i$
  - If $p_i \sim 0^+$, then almost all user have a '0' sparse code
  - If $p_i \sim 1/2$, then as many '0' as '1' dense code

THOMSON

Tardos accusation

- **Accusation**
  - Accuse user #j if $S_j > T$

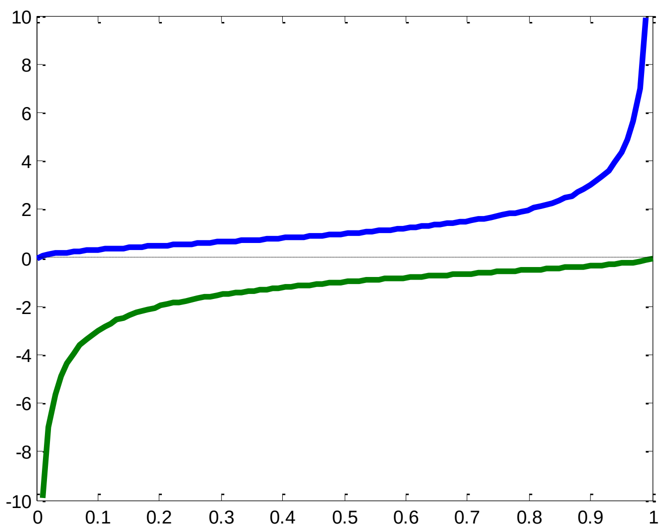
$$S_j = \Sigma_i \; g(y_i, x_{ji}, p_i)$$

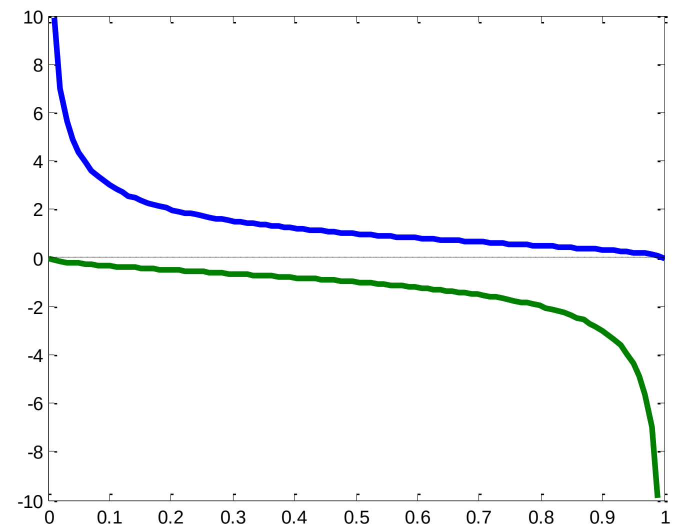
  - $g(0,0,p) = + \left(p/(1-p) \right)^{1/2}$
  - $g(0,1,p) = - \left((1-p)/p \right)^{1/2}$

$g(1,1,p) = + \left((1-p)/p \right)^{1/2}$

$g(1,0,p) = - \left(p/(1-p) \right)^{1/2}$

(?!?)





THOMSON

Tardos code

Why does it work?

Mathematical model of the collusion

- **Assumptions about the collusion**
  - Memoryless $y_i = F_i(x_{1i},\dots, x_{ci})$
  - Stationary $y_i = F(x_{1i},\dots, x_{ci})$
  - Permutation Invariant $y_i = F(s_i)$ $s_i = \Sigma_j x_{ji}$
  - Probabilistic $\theta_s = \mathrm{Prob}(y = 1 | s)$
  - Model $\boldsymbol{\theta} = (\theta_0, \theta_1, \dots, \theta_c)$

Marking assumption $\theta_0 = 0$ and $\theta_c = 1$

Therefore, the collusion indeed lies in $[0,1]^{c-1}$

THOMSON

Mathematical model of the collusion

- **Using the model**

  $\mathrm{Prob}[y=1 \mid p] = \Sigma_s \, \mathrm{Prob}[y=1,s|p] = \Sigma_s \, \theta_s \,.\, \binom{c}{s} \, p^s \, (1-p)^{c-s}$

  $\mathrm{Prob}[y=1 \mid x=1, p] = \Sigma_s \, \theta_s \,.\, \binom{c-1}{s-1} \, p^{s-1} \, (1-p)^{c-s}$

- **1st and 2nd order statistics**
  - Innocent: $\mathbb{E}[\,S_j\,] = 0$ $\mathbb{E}[\,S_j^2\,] = \mathrm{m}$
  - Colluder: $\mathbb{E}[\,Sj\,] = 2m/\pi c$ $\mathbb{E}[\,S_j^2\,] = \mathrm{m}$

(here: $\mathbb{E}[\,.\,] = \mathbb{E}_P[\mathbb{E}_X[\mathbb{E}_Y[.]\,]\,]$)

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

THOMSON

Statistical interpretations (I)

- The collusion process is a <u>nuisance parameter</u>
- The hypothesis test is based on a <u>pivotal quantity</u>…
- … at least up to 1st and 2nd order statistics.

$$\boxed{\begin{array}{l} f(p) = 1/\pi(p(1-p))^{1/2} \\ g(0,0,p) = +\left(p/(1-p)\right)^{1/2} \\ g(1,1,p) = \ldots \end{array}} \begin{array}{c} \Longrightarrow \\ \Longleftarrow \end{array} \boxed{\begin{array}{ll} \text{Innocent} & \mathbb{E}[S_j] = 0;\ \mathbb{E}[S_j^2] = m \\ \text{Colluder} & \mathbb{E}[Sj] = 2m/\pi c;\ \mathbb{E}[S_j^2] = m \end{array}}$$

THOMSON

Statistical interpretation (II)

- **One collusion process – Many code densities**
  - Dense (p=1/2): worst attack: minority vote 3 colluders
  - Sparse ($p=0^+$ 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.

THOMSON

Statistical interpretation (III)

- **Hypothesis test**
  - H_0: Innocent $\quad$ $\mathrm{Prob}(\mathbf{y},\mathbf{x}|\mathbf{p}) = \mathrm{Prob}(\mathbf{y}|\theta,\mathbf{p}) \,.\, \mathrm{Prob}(\mathbf{x}|\mathbf{p})$
  - H_1: Colluder $\quad$ $\mathrm{Prob}(\mathbf{y},\mathbf{x}|\mathbf{p}) = \mathrm{Prob}(\mathbf{y}|\mathbf{x},\theta,\mathbf{p}).\mathrm{Prob}(\mathbf{x}|\mathbf{p})$
  - Performance criterion

$$R\,(\,f\,;\,\theta\,) = \mathbb{E}_P\,[\,D_{KL}(\,H_1\,;\,H_0\,|\,P,\,\theta\,)\,]$$

- **Game theory**
  - Between designer and colluders
  - MaxMin game

$$R\,(\,f^*\,;\,\theta^*\,) = \max_f \min_\theta R\,(\,f\,;\,\theta\,)$$

- **Asymptotically, $c \longrightarrow \infty$:**
  - 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 θ and use Likelihood ratio test to accuse (E.-M.)

- **K-uplets scores**
  - Inf. theory:

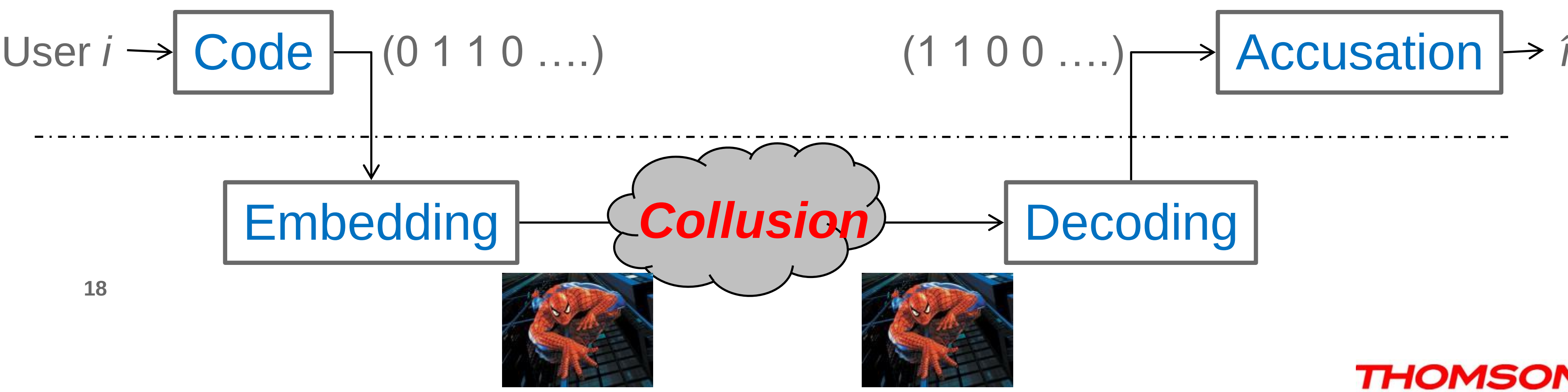
    $$\mathrm{I}(\mathbf{y}\,;\,\mathbf{x}|\,\theta) \leqslant \mathrm{I}(\mathbf{y}\,;\,\{\mathbf{x}_1, \mathbf{x}_2\}\,|\,\theta) \leqslant \ldots \leqslant \mathrm{I}(\mathbf{y}\,;\,\{\,\mathbf{x}_1,\ldots,\mathbf{x}_c\,\}|\,\theta)$$
  - Practical? Complexity in ~ O(n^c)

- **Q-ary alphabet**

- **Other collusion models**
  - Erasures (cut movie scenes)

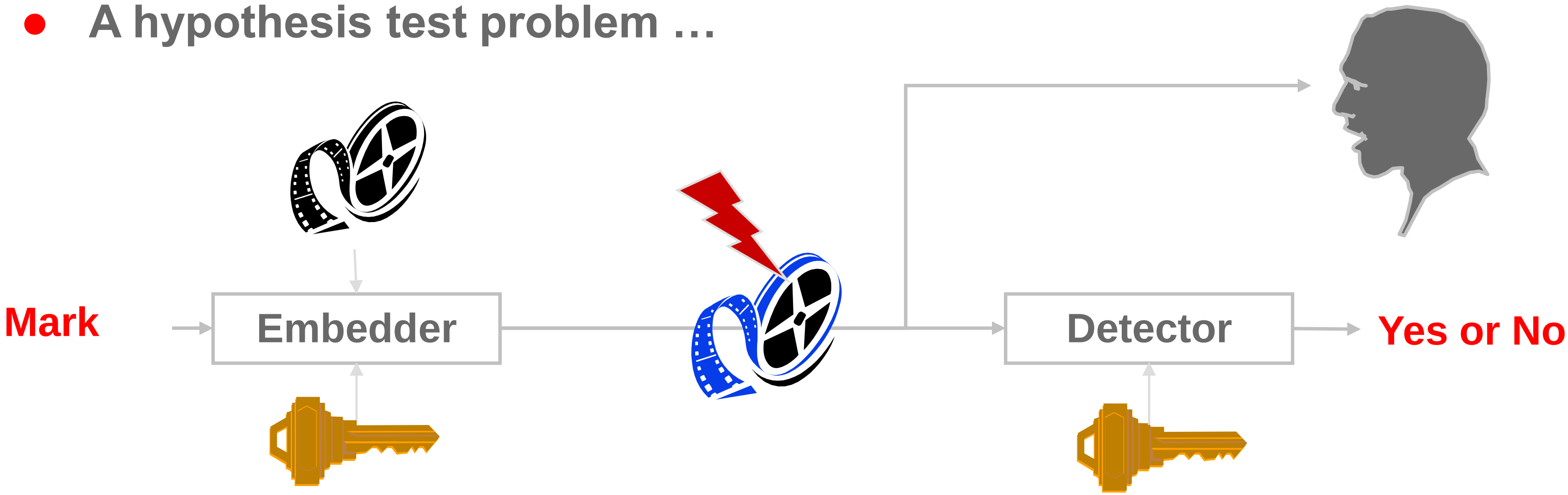
Outlines

- **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**



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

$$\boldsymbol{s} = \mathrm{Ext}(B), \quad \text{with } \boldsymbol{s} \in \mathbb{R}^L$$

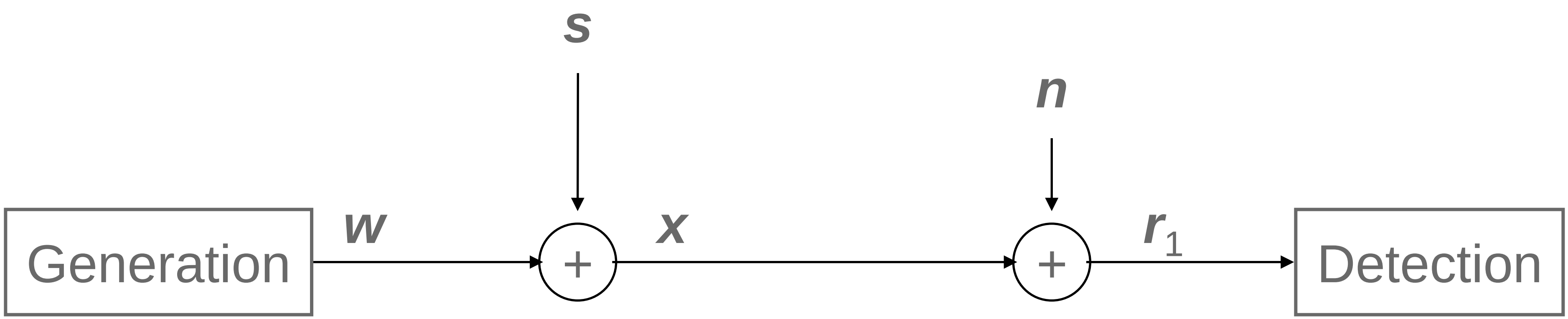
  - Modify $\boldsymbol{s}$ into $\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{w}$ s.t. $\|\boldsymbol{w}\|^2 \leqslant L.P_w$
  - The vector $\boldsymbol{w}$ is the secret of the watermarking scheme
  - Put back the features into the content

$$B_w = \mathrm{Ext}^{-1}(\boldsymbol{x}, B)$$

- **Attack on the watermarked image**
  - Distort $\boldsymbol{x}$ into $\boldsymbol{z} = \boldsymbol{x} + \boldsymbol{n}$ s.t. $\mathbb{E}\,\|\boldsymbol{n}\|^2 \leqslant L.P_n$
- **Detection**
  - H_0: $\boldsymbol{r}_0 = \boldsymbol{s} + \boldsymbol{n}$ (given by Nature)
  - H_1: $\boldsymbol{r}_1 = \boldsymbol{s} + \boldsymbol{w} + \boldsymbol{n}$

THOMSON

Naïve idea



$\|\boldsymbol{w}\|^2 \leqslant L.P_w$

- **Gaussian setup**

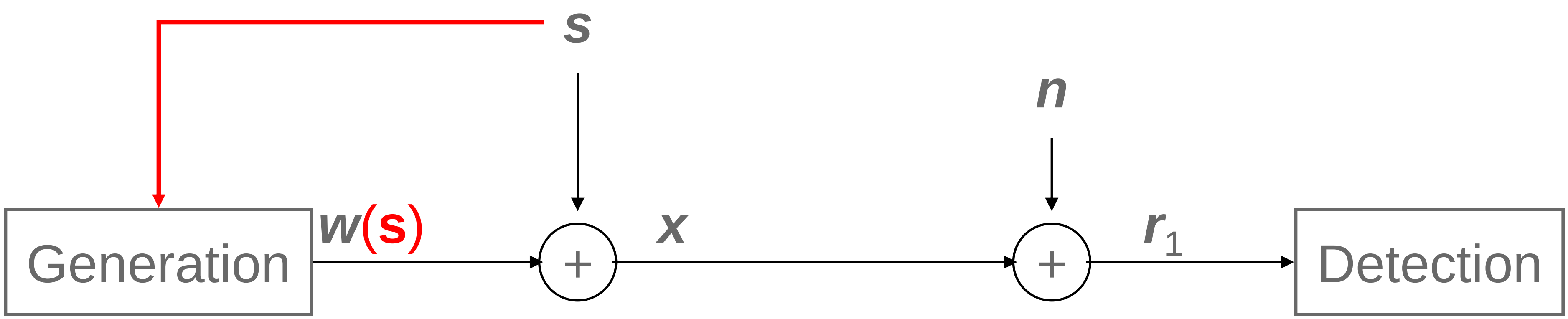
  $\boldsymbol{r}_0 = \boldsymbol{s} + \boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, P_s+P_\mathrm{n})$ vs. $\boldsymbol{r}_1 = \boldsymbol{s} + \boldsymbol{w} + \boldsymbol{n} \sim \mathcal{N}(\mathbf{w}, P_s+P_\mathrm{n})$

  – Performances limited by the Kullback-Leibler distance

  $$\mathrm{D}_{\mathrm{KL}}(\boldsymbol{r}_0\|\boldsymbol{r}_1) = L.P_w / 2\,(P_n+P_s)$$

  – Data processing theorem, Stein Lemma.

An Information theoretic revolution [Costa]



$$\mathbb{E}_{\mathbf{s}}\|\boldsymbol{w}(\mathbf{s})\|^2 \leqslant L.P_w$$

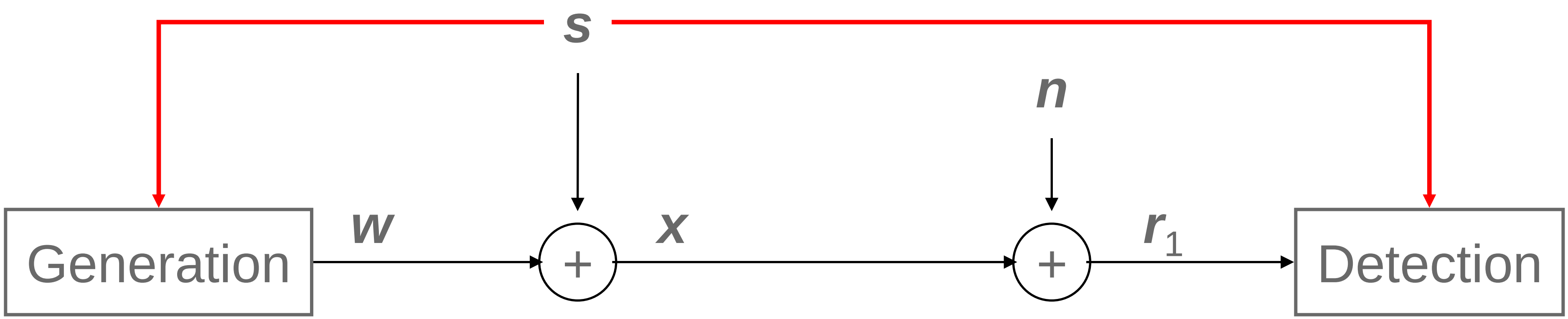
- **Gaussian setup**

  $\boldsymbol{r}_0 = \boldsymbol{s} + \boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, P_s + P_n)$ vs. $\boldsymbol{r}_1 = \boldsymbol{s} + \boldsymbol{w}(\boldsymbol{s}) + \boldsymbol{n}$

  – Performances limited by the Kullback-Leibler distance

    $$\max_{\boldsymbol{w}(.)} D_{KL}(\boldsymbol{r}_0 \| \boldsymbol{r}_1) = \textbf{???}$$

The informed setup

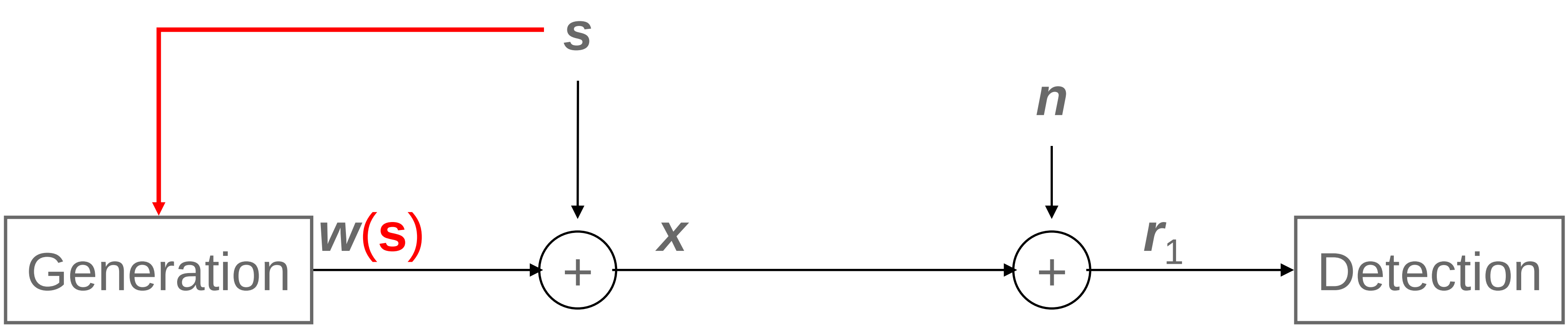


$||\boldsymbol{w}||^2 \leqslant L.P_w$

- **Gaussian setup**
  - $\boldsymbol{r}_0 - \boldsymbol{s} = \boldsymbol{n} \sim \mathcal{N}(\mathbf{0}, P_n)$ vs. $\boldsymbol{r}_1 - \boldsymbol{s} = \boldsymbol{w} + \boldsymbol{n} \sim \mathcal{N}(\mathbf{w}, P_n)$
  - Performances limited by the Kullback-Leibler distance

$$D_{KL}(\boldsymbol{r}_0||\boldsymbol{r}_1) = P_w / 2.P_n$$

The side informed setup [Costa]



$\mathbb{E}_s||\boldsymbol{w}(\boldsymbol{s})||^2 \leqslant L.P_w$

- **Gaussian setup**
  - Performances limited by the Kullback-Leibler distance

$$P_w / 2.(P_n+P_s) \quad \leqslant \quad \max D_{KL}(\boldsymbol{r}_0||\boldsymbol{r}_1) \quad \leqslant \quad P_w / 2.P_n$$

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

Detection *Generation*

Asymptotical Gaussian case

- **Suppose the following watermark embedding**

$$\boldsymbol{x} = \boldsymbol{s} + \boldsymbol{w}(\boldsymbol{s}) = \boldsymbol{s} + (\alpha - \lambda.\boldsymbol{s}^T\boldsymbol{u}).\boldsymbol{u} \qquad \text{with } \|\boldsymbol{u}\|^2=1$$

  - $\lambda \in [0,1]$, $\alpha > 0$
  - "Push and cancel" mixed strategy
  - Power constraint:

$$\alpha^2 + \lambda^2.P_s = L.P_w$$

- **KL- distance**
  - $D_{KL}(\boldsymbol{r}_0\|\boldsymbol{r}_1) = F(\lambda, \alpha) = F(\lambda)$.
  - Optimize : $\lambda^* = \arg\max_\lambda F(\lambda)$
  - Take to the limit

$$\lim_{L\longrightarrow\infty} F(\lambda^*) = P_w / 2.P_n$$

THOMSON

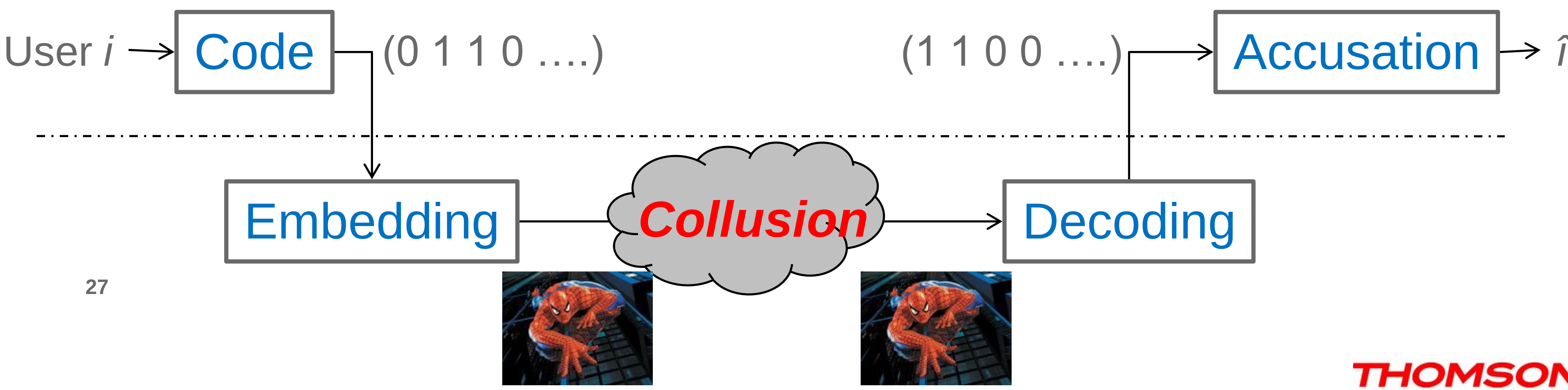
How to put this into practice?

- **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 probabilty estimation**



False positive estimation

- **2 techniques have a common issue: very very small** P_{fa}
  - Traitor tracing: probability to accuse an innocent.

$$P_{fa} = \text{Prob}\,[\, d(\mathbf{x}) > T \,]$$

    - with $d(.)$ Tardos scoring
    - $\mathbf{x}$ the sequence of a user, $\text{Prob}(\mathbf{x}_j) = \Pi_i\, p_i^{x_{ji}} . (1-p_i)^{(1-x_{ji})}$

  - Watermarking: probability of detecting the mark in a non watermarked content

$$P_{fa} = \text{Prob}\,[\, d(\mathbf{x}) > T \,]$$

    - with $d(.)$ likelihood of being watermarked
    - $\mathbf{x}$ features extracted from a content block, $\mathbf{x} \sim p_{\mathbf{X}}$

Monte Carlo Method estimation

- **Structure of the detector**

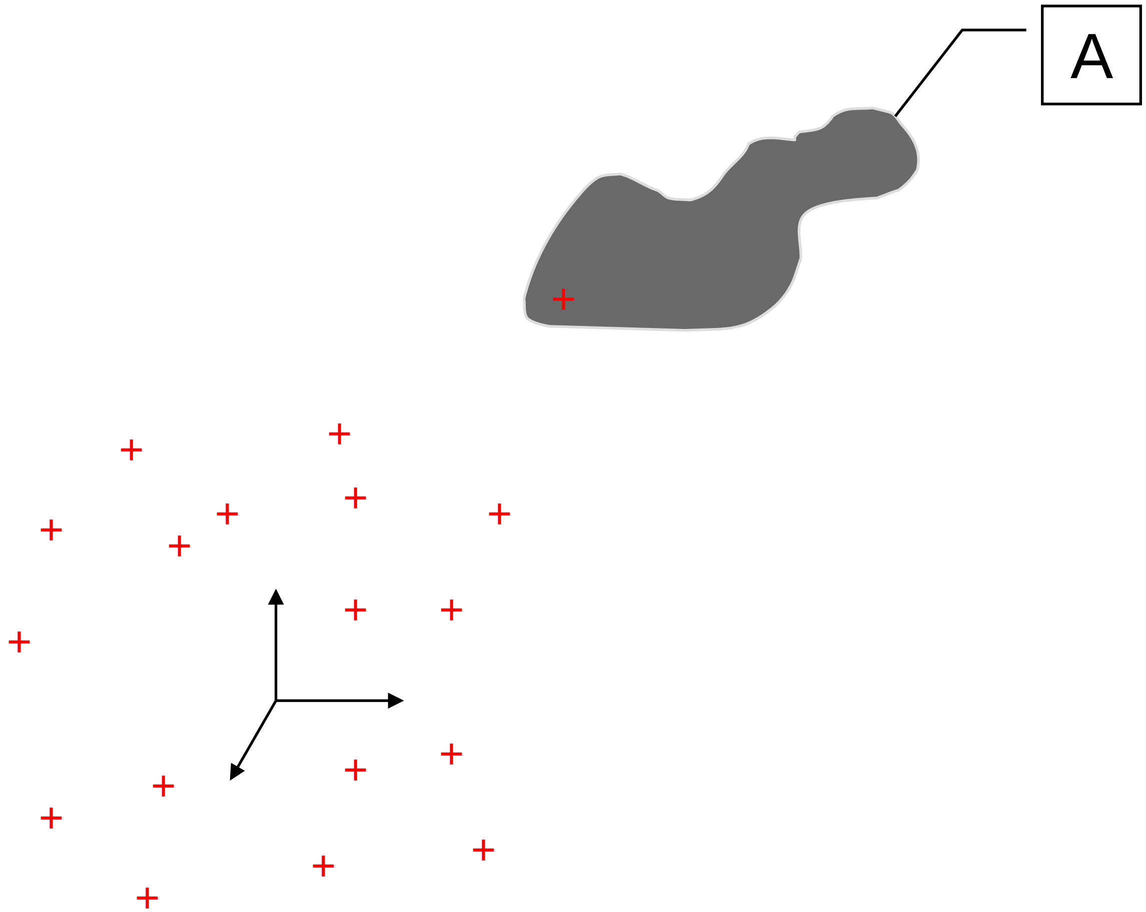
$$\mathbf{x} \in \mathbb{R}^L \sim p_X \longrightarrow \boxed{\text{score}} \xrightarrow{d(\mathbf{x}) \in \mathbb{R} \ \sim\ ?} \boxed{\text{threshold } T} \longrightarrow \text{Y / N}$$

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

- **Issues**
  - $k \neq 0$ $\qquad n = O(1/P_{fa})$
  - Relative std of $\hat{P}_{fa} = (P_{fa}.n)^{-1/2}$ $\qquad n \sim 100/P_{fa}$

$\Longrightarrow$ The smaller the probability, the harder its estimation

Geometric interpretation



Key idea of our algorithm

Divide and Conquer

P_{fa} $= \Pr(A) = \Pr(A, A_{N-1})$ if A implies A_{N-1}

$= \Pr(A | A_{N-1}) \,.\, \Pr(A_{N-1})$ 2 estimations, less difficult

$= \Pr(A | A_{N-1}) \,.\, \Pr(A_{N-1} | A_{N-2}) \,.\, \Pr(A_{N-2} | A_{N-3}) \ldots \Pr(A_1)$

$A \Rightarrow A_{N-1} \Rightarrow A_{N-2} \Rightarrow \ldots \Rightarrow A_1$

P_{fa} $= \Pr(d(\mathbf{x}) \geq T)$

$= \Pr(d(\mathbf{x}) \geq T \,|\, d(\mathbf{x}) \geq T_{N-1}).\Pr(d(\mathbf{x}) \geq T_{N-1} | d(\mathbf{x}) \geq T_{N-2}) \ldots \Pr(d(\mathbf{x}) \geq T_1)$

$T > T_{N-1} > T_{N-2} > \ldots. > T_1$

$\hat{P}_{fa} = \hat{a}_N \,.\, \hat{a}_{N-1} \,.\, \ldots\, \hat{a}_1$

THOMSON

Our estimator

- **Inputs**
  - Distribution of input data p_X, score $d(.)$, threshold T
- **Outputs**
  - Estimation $\hat{P}_{fa}$ of $\Pr(d(\mathbf{x})>T)$ for $\mathbf{x} \sim p_X$
- **Ingredients: 3 subroutines**
  - SCORE
    - Function $d(.): \mathbb{R}^L \rightarrow \mathbb{R}$
    - 'Smooth'
  - GENERATE
    - Generate input data **x** distributed $\sim p_X$
  - MODIFY
    - $\mathbf{y} = f(\mathbf{x})$, random function
    - such that $\mathbf{y} \sim p_X$ and **y** is 'close' to **x**

Our estimator

- **Divide and conquer**

$$P_{fa} = \Pr(d(\mathbf{x}) \geq T \mid d(\mathbf{x}) \geq T_{N-1}) \ldots \Pr(d(\mathbf{x}) \geq T_{j+1} \mid d(\mathbf{x}) \geq T_j) \ldots \Pr(d(\mathbf{x}) \geq T_1)$$

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

$$k_1 = |\{\mathbf{x}_i^{(1)} \mid d(\mathbf{x}_i^{(1)}) > T_1\}|$$

$$\hat{a}_1 = k_1/n$$

Our estimator

- **Divide and conquer**

  $P_{fa} = \Pr(d(\mathbf{x}) \geq T \,|d(\mathbf{x}) \geq T_{N-1}) \ldots \Pr(d(\mathbf{x}) \geq T_{j+1} | d(\mathbf{x}) \geq T_j) \ldots \Pr(d(\mathbf{x}) \geq T_1)$

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

Our estimator

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

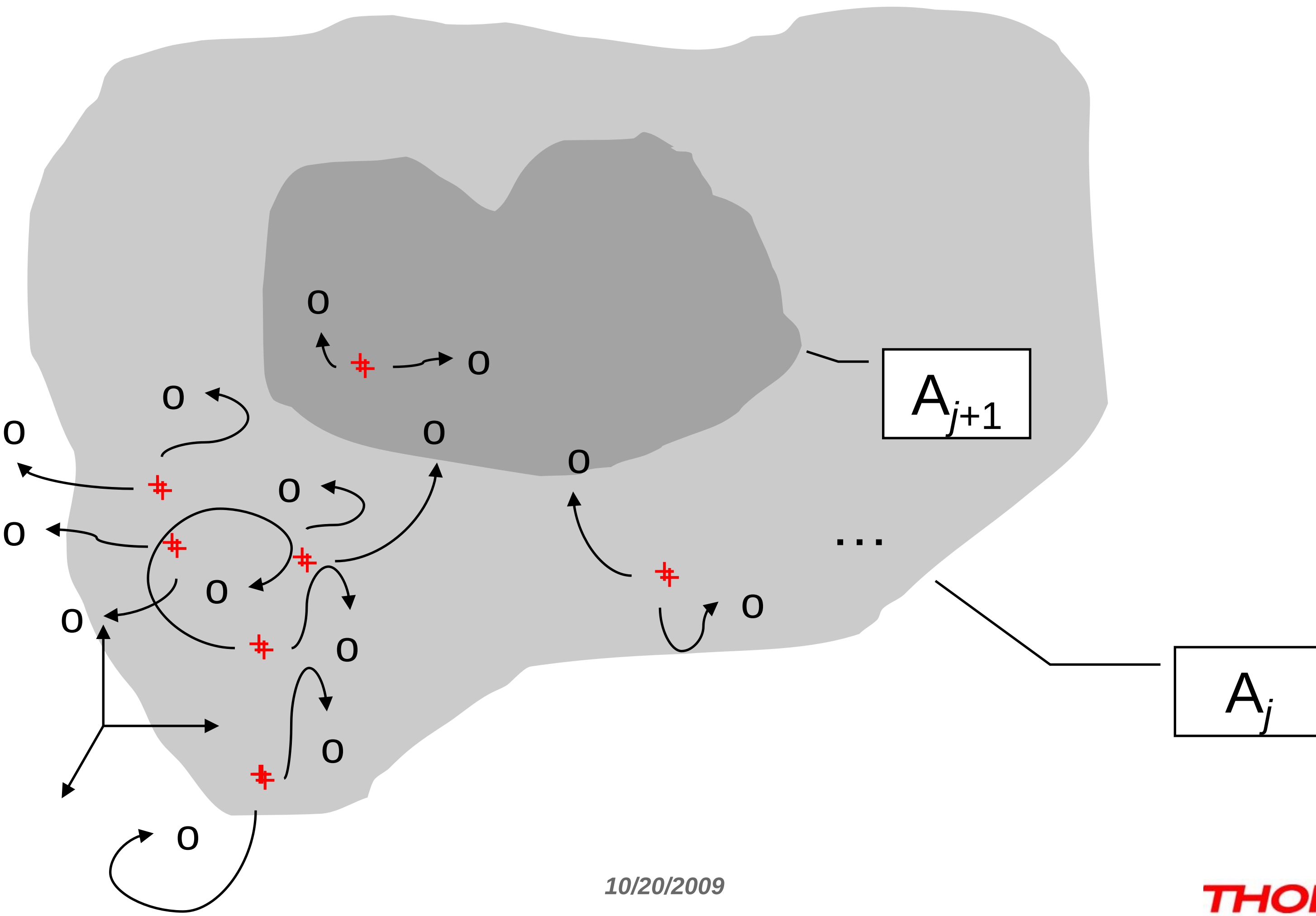
  – THRESHOLD (or MCM estimator)
    - Count the number of times the score is above the $(j+1)^{th}$ threshold

$$k_{j+1} = |\{\mathbf{x}_i^{(j+1)} \mid d(\mathbf{x}_i^{(j+1)}) > T_{j+1}\}|$$

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

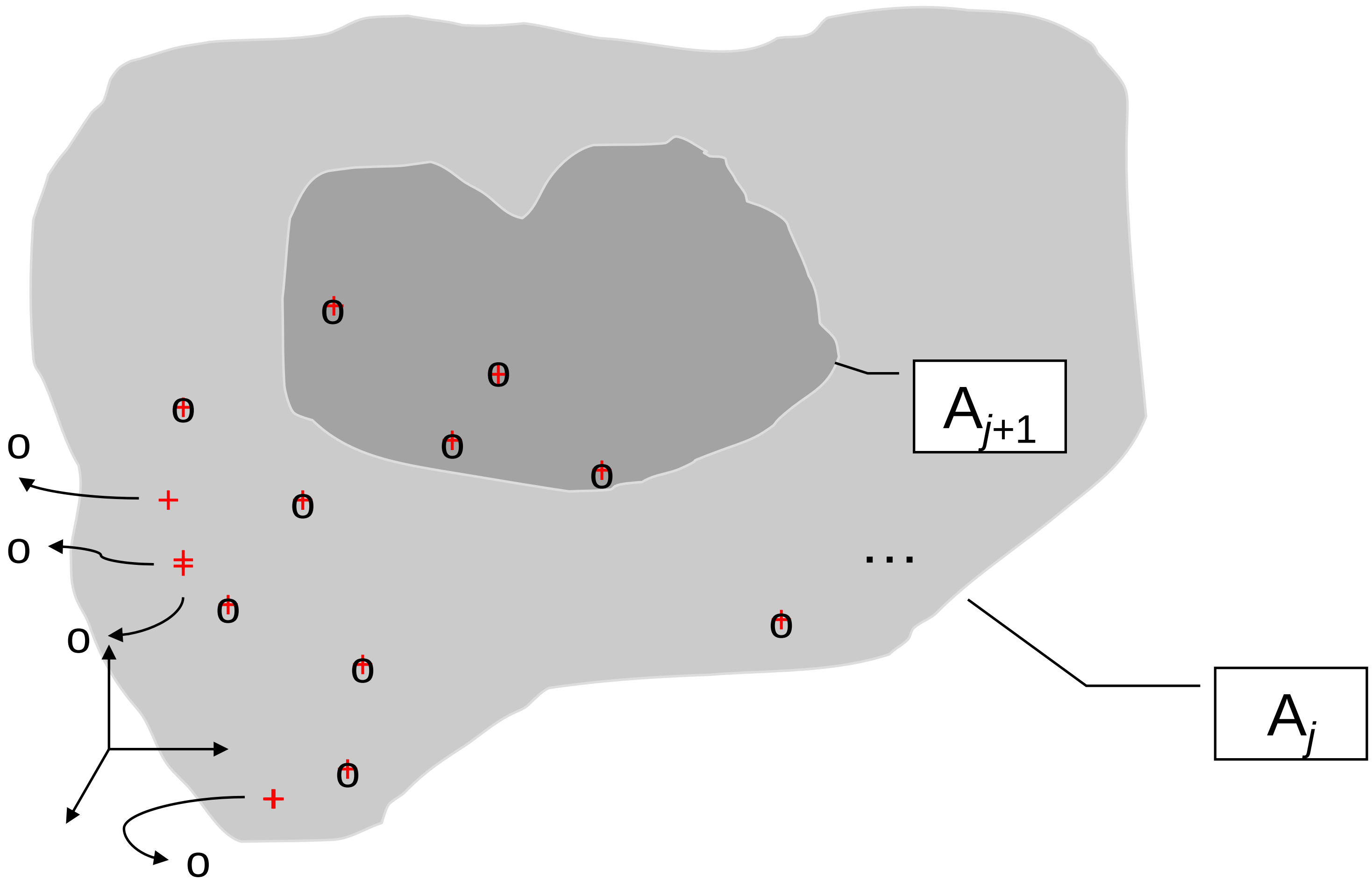
Our estimator

- **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 = cte
  - Inverse: T_j is the k-th biggest score out of n: $\hat{a}_j = k/n$

- **Divide and conquer**

$$P_{fa} = \Pr(d(\mathbf{x}) \geq T \,|d(\mathbf{x}) \geq T_{N-1}) \ldots \Pr(d(\mathbf{x}) \geq T_{j+1}|d(\mathbf{x}) \geq T_j) \ldots \Pr(d(\mathbf{x}) \geq T_1)$$

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

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**

$$n^{1/2}.(\hat{P}_{fa} - P_{fa}) \rightarrow \mathcal{N}(b,\sigma^2) \text{ as } n \rightarrow \infty$$

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

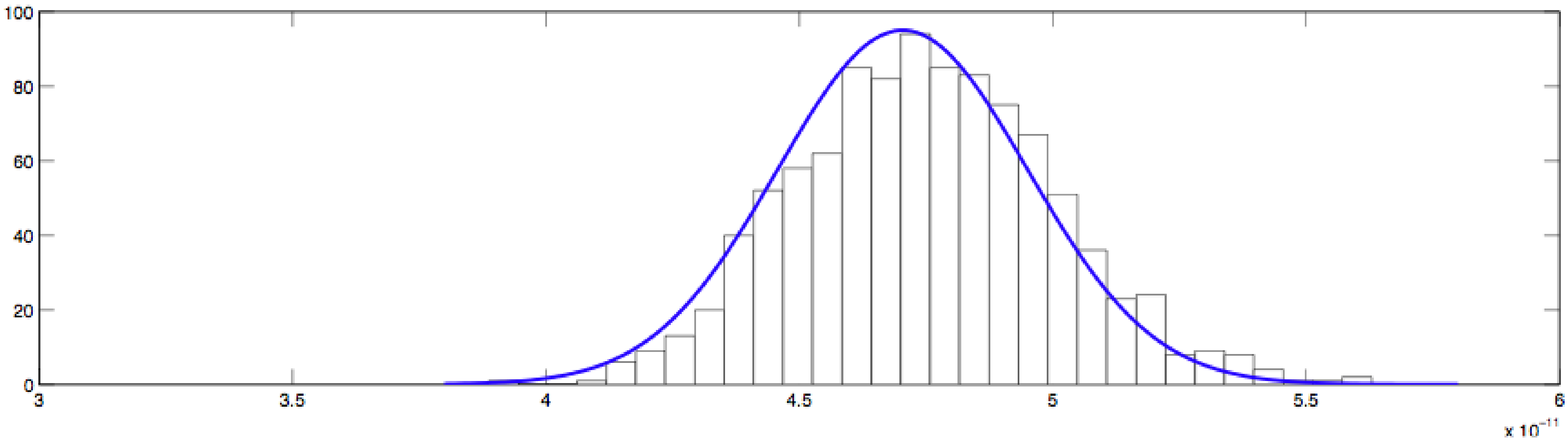
- **Ingredients**
  - GENERATE: Matlab randn
  - MODIFY:

$$\mathbf{y} = (\mathbf{x}+q.\mathbf{n})/(1+q^2)^{1/2}$$

$$\mathrm{n} \sim \mathrm{N}(\mathbf{0}, \mathbf{I}_L)$$

    - q fixes the strength of the modification.

Experiment #1: Asymptotic normality

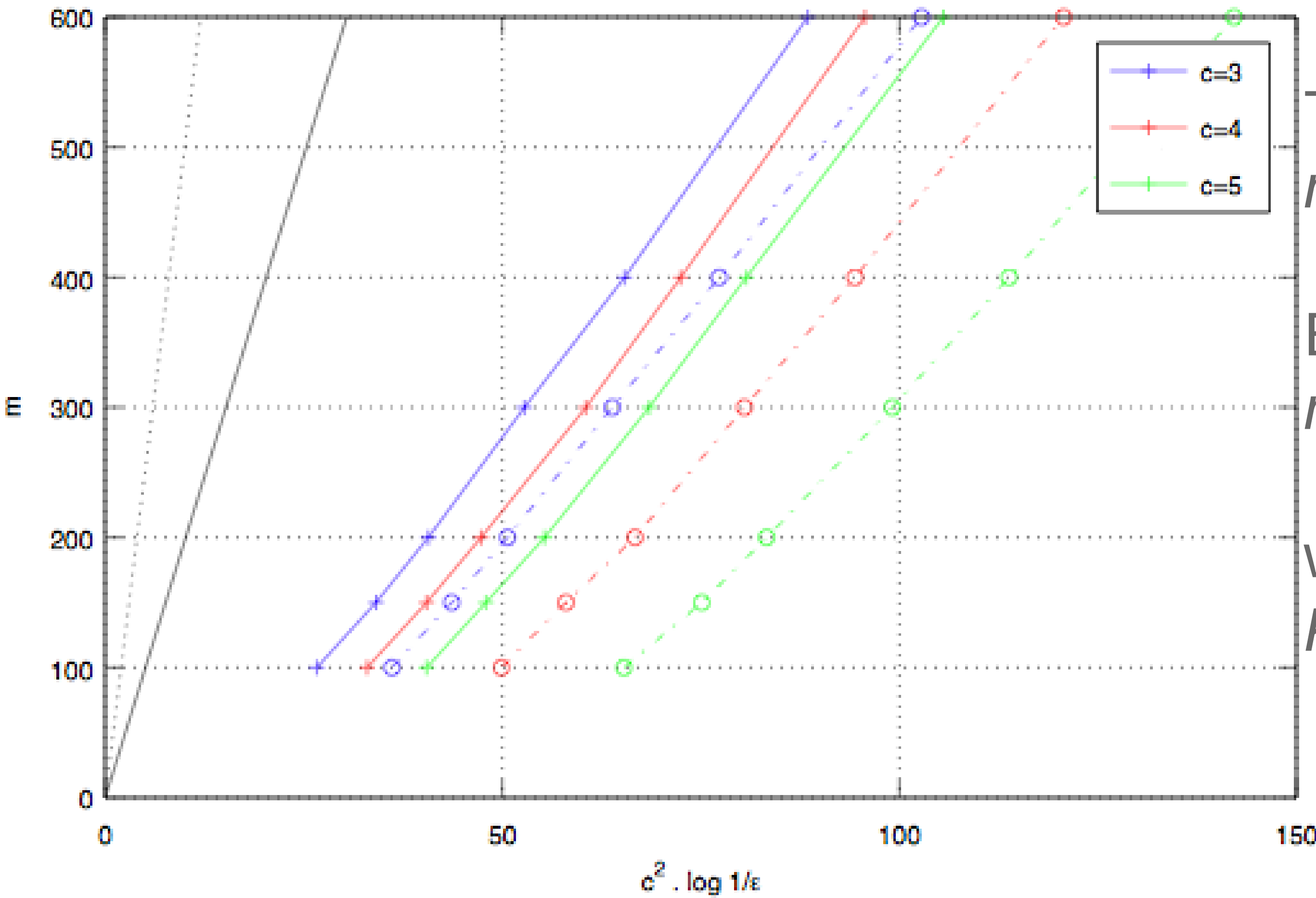


$L = 20$
$T = 0.95$
$P_{fa} = 4.7 \,.\, 10^{-11}$

$k/n = 3/4$
n=50,000
200 runs

THOMSON

Experiment #2: Estimating minimum length



Tardos

$m = 100.c^2\log(n/P_{fa})$

Experimental

$m \approx K_1.c^2\log(n/P_{fa}) + K_0(c)$

with

$K_1 \approx 7.6$

THOMSON

Conclusion

Thank you for your attention