Privacy

- Protect confidential information from non-authorized use

- Protection Mechanisms: Crypto, Access control, ...

- Problem: information leakage in many applications is unavoidable

---

I. Leakage that happens intentionally

- eg: extract statistics from a dataset

- Problem: inference of personal information

- eg: “what is the median age of cancer patients”

---

II. Leakage due to side channels

- ge: OpenSSL timing attack [BonehBrumley03]

- Also: cache misses, power, radiation, faults, ...

- Completely preventing such channels is impossible
III. Leakage in exchange to **efficiency**
- eg: Anonymous Communication Systems (Crowds, Onion Routing, ...)
- Strong anonymity is achievable (eg Dining Cryptographers)
- But such systems are not efficient!

IV. Leakage in exchange to a **service**
- eg: Location Based Services
  - Points Of Interest / Dating / social networks / games / ...
- Privacy issues: tracking, profiling, identification

---

**Quantitative Information Flow**

How can we measure information leakage?

---

**Quantitative Information Flow**

Plan of the course
- Motivation, application examples
- Secrets and vulnerability
- Channels and leakage
- Robustness and capacity
- Comparing systems, the lattice of information
- Applications and exercises
Example: analysis of a password checker

**Password checker 1**

Password: $K_1 K_2 \ldots K_N$
Input by the user: $x_1 x_2 \ldots x_N$
Output: $\text{out}$ (Fail or OK)

**Intrinsic leakage**

By learning the result of the check the adversary learns something about the secret

```plaintext
for i = 1, ..., N do
    if $x_i \neq K_i$ then
        $\text{out} := \text{FAIL}$
    end if
end for
```

**Password checker 2**

Password: $K_1 K_2 \ldots K_N$
Input by the user: $x_1 x_2 \ldots x_N$
Output: $\text{out}$ (Fail or OK)

**Side channel attack**

If the adversary can measure the execution time, then he can also learn the longest correct prefix of the password

```plaintext
for i = 1, ..., N do
    if $x_i \neq K_i$ then
        $\text{out} := \text{FAIL}$
    end if
end for
```

---

The Dining Cryptographers protocol

**Example of Anonymity Protocol:**

DC Nets [Chaum’88]

- A set of nodes with some communication channels (edges).
- One of the nodes (source) wants to broadcast one bit $b$ of information
- The source (broadcaster) must remain anonymous

![Example of Anonymity Protocol](image)

**Chaum’s solution**

- Associate to each edge a fair binary coin
- Toss the coins
- Each node computes the binary sum of the incident edges. The source adds $b$. They all broadcast their results
- Achievement of the goal: Compute the total binary sum: it coincides with $b$

![Chaum’s solution](image)
Observables: An (external) attacker can only see the declarations of the nodes

Question: Does the protocol protects the anonymity of the source?

The Crowds protocol

The initiator:
- Forwards the message

A forwarder:
- With pb \( p_f \) forwards
- With pb \( 1 - p_f \) delivers
- The path is used in the opposite direction for the reply
- The same path is used in future requests

The Crowds protocol: anonymity

- We consider sender anonymity
- Attacker model
  - Cannot see the whole network
  - Only messages sent to him
- The server:
  - only sees the last user
  - Strong anonymity is satisfied

DC is not practical for a large number of users
In practice we might want to trade anonymity for efficiency
Crowds offers a weaker notion of anonymity called probable innocence
Designed for anonymous web surfing
The Crowds protocol: anonymity

Corrupted users:

- They can see forwarding requests and "detect" a user $i$
- User $i$ can still claim that he was forwarding the message for user $j$
- Is strong anonymity satisfied?
- Compare the probab. to detect $i$:
  - when $i$ is the payer
  - when $j$ is the payer
- They are different: strong anonymity is violated

Location-Based Systems

A location-based system is a system that uses geographical information in order to provide a service.

- Retrieval of Points of Interest (POIs).
- Mapping Applications.
- Deals and discounts applications.
- Location-Aware Social Networks.

Solution: obfuscation

- Find restaurants within 300 meters.
- Hide location, not identity.
- Provide approximate location.
Issues to study

- How can we generate the noise?
- What kind of formal privacy guarantees do we get?
- Which mechanism gives optimal utility?
- What if we use the service repeatedly?

Timing Attacks in Cryptosystems

Remote timing attack [BonehBrumley03]

1024-bit RSA key recovered in 2 hours from standard OpenSSL implementation across LAN

Response time depends on the key!
Timing Attacks in Cryptosystems

What counter-measures can we use?

Make the decryption time constant
Too slow!

Force the set of possible decryption times to be small
Is it enough?
Must be combined with blinding
Careful analysis of the privacy guarantees is required

Plan of the course

Quantitative Information Flow

• Motivation, application examples
• Secrets and vulnerability
• Channels and leakage
• Multiplicative Bayes-Capacity
• Comparing systems, the lattice of information
• Applications and exercises

Location-Based Systems

A location-based system is a system that uses geographical information in order to provide a service.

› Retrieval of Points of Interest (POIs).
› Mapping Applications.
› Deals and discounts applications.
› Location-Aware Social Networks.

MPRI 2.3.2 - Foundations of privacy

Lecture 2

Kostas Chatzikokolakis

Jan 18, 2018
Location-Based Systems

- Find restaurants within 300 meters.
- Hide location, not identity.
- Provide approximate location.

Solution: obfuscation
Issues to study

• How can we generate the noise?
• What kind of formal privacy guarantees do we get?
• Which mechanism gives optimal utility?
• What if we use the service repeatedly?

Timing Attacks in Cryptosystems

• Remote timing attack [BonehBrumley03]
• 1024-bit RSA key recovered in 2 hours from standard OpenSSL implementation across LAN
  
  SSL Handshake (simplified)
  
  
  • Response time depends on the key!

Timing Attacks in Cryptosystems

• What counter-measures can we use?
• Make the decryption time constant
  - Too slow!
• Force the set of possible decryption times to be small
  - Is it enough?
  - Must be combined with blinding
  - Careful analysis of the privacy guarantees is required

Quantitative Information Flow

How can we measure information leakage?
What is a secret?

- My password: $x = "dme3@21!SDFm12"
- What does it mean for this password to be secret?
- The adversary should not know it, i.e. $x$ comes from a set $\mathcal{X}$ of possible passwords
- Is $x' = 123$ an equally good password? Why?
- Passwords are drawn randomly from a probability distribution

Vulnerability

- How vulnerable is our secret under prior $\pi$?
- The answer highly depends on the application
- Eg: assume uniformly distributed secrets but the adversary knows the first 4 bytes (this can be expressed by a prior $\pi$)
- Is this threat substantial?
  - No: if the secrets are long passwords
  - Yes: if the first 4 bytes is a credit card pin

What is a secret?

- A secret $x$ is something about which the adversary knows only a probability distribution $\pi$
- $\pi$ is called the adversary's prior knowledge
  - $\pi$ could be the distribution from which $x$ is generated
  - or it could model the adversary's knowledge on the population the user comes from
- How vulnerable is $x$?
- It's a property of $\pi$, not of $x$
First approach

- Assume the adversary can ask for properties of $x$
  i.e. questions of the form “is $x \in P$?”

- **Goal**: completely reveal the secret as quickly as possible

- Measure of success: **expected number of steps**

Best strategy: at each step split the search space in sets of equal probability mass

1. At step $i$ the total probability mass is $2^{-i}$.
2. If the probability of $x$ is $\pi_x = 2^{-i}$, then at step $i$ the search space will only contain $x$. So it will take $i = -\log_2 \pi_x$ steps to reveal $x$.
3. So the **expected number of steps** is:
   \[
   \sum_x -\pi_x \log_2 \pi_x
   \]
   (if $\pi$ is not powers of 2 this will be a lower bound)
Shannon Entropy

• This is the well known formula of Shannon Entropy:

\[ H(\pi) = \sum_x -\pi_x \log_2 \pi_x \]

• It’s a measure of the adversary's uncertainty about \( x \)

• Minimum value: \( H(\pi) = 0 \) iff \( \pi_x = 1 \) for some \( x \)

• Maximum value: \( H(\pi) = \log_2 |\mathcal{X}| \) iff \( \pi \) is uniform

Shannon Entropy

• Very widely used to measure information flow

• Is it always a good uncertainty measure for privacy?

• Example: \( \mathcal{X} = \{0, \ldots, 2^{32}\} \), \( \pi = (\frac{1}{8}, \frac{7}{8}2^{-32}, \ldots, \frac{7}{8}2^{-32}) \)

\[ H(\pi) = 28.543525 \]

• But the secret can be guessed with probability \( 1/8! \)

• Undesired in may practical scenarios

Bayes Vulnerability

• Adversary's goal: correctly guess the secret in one try

• Measure of success: probability of a correct guess

• Optimal strategy: guess the \( x \) with the highest \( \pi_x \)

• Bayes Vulnerability:

\[ V_b(\pi) = \max_x \pi_x \]
Bayes Vulnerability

- **Maximum value**: $V_b(\pi) = 1$ iff $\pi_x = 1$ for some $x$
- **Minimum value**: $V_b(\pi) = 1/|X|$ iff $\pi$ is uniform
- **Min-entropy**: $\log_2(V_b(\pi))$
  Bayes risk: $1 - V_b(\pi)$
- **Previous example**: $\pi = (\frac{1}{8}, \frac{7}{8}, 2^{-32}, \ldots, 2^{-32})$
  $V_b(\pi) = 1/8$

Guessing Entropy

- **Adversary’s goal**: correctly guess the secret in many tries
- **Measure of success**: expected number of tries
- **Optimal strategy**: try secrets in decreasing order of probability
- **Guessing entropy**: $G(\pi) = \sum_{i=1}^{|X|} i \cdot \pi_{x_i}$
  $x_i$: indexing of $X$ in decreasing probability order

The binary case $X = \{0, 1\}$:
Still not completely satisfied

- What if the adversary wants to reveal part of a secret?
- Or is satisfied with an approximative value?
- Or we are interested in the probability of guessing after multiple tries?

Example

- **Secret**: database of 10-bit passwords for 1000 users: pwd_0, pwd_1, ..., pwd_999
- The adversary knows that the password of some user is z, but does not know which one (all are equally likely)
- \( A_1 \): guess the complete database
  - \( V_g(\pi) = 2^{-9990} \)
- \( A_2 \): guess the password of a particular user i
  - Create distribution \( \pi_i \) for that user
  - \( V_g(\pi_i) = 0.001 \cdot 1 + 0.999 \cdot 2^{10} \approx 0.00198 \)
- \( A_3 \): guess the password of any user
  - intuitively, the secret is completely vulnerable

Abstract operational scenario

- \( \mathcal{A} \) makes a guess \( w \in \mathcal{W} \) about the secret
- The benefit provided by guessing \( w \) when the secret is \( x \) is given by a gain function:
  \[
g : \mathcal{W} \times \mathcal{X} \to \mathbb{R}
\]
- **Success measure**: the expected gain of a best guess

\( g \)-vulnerability

- **Expected gain** under guess \( w \): \( \sum_x \pi_x g(w, x) \)
- Choose the one that maximizes the gain
- \( g \)-vulnerability:
  \[
  V_g(\pi) = \max_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x g(w, x)
  \]
  (\( \sup \) if \( \mathcal{W} \) is infinite)
- **Uncertainty** measures can be also defined using loss functions:
  \[
  U_l(\pi) = \min_{w \in \mathcal{W}} \sum_{x \in \mathcal{X}} \pi_x l(w, x)
  \]
The power of gain functions

Guessing a secret **approximately**.
\[ g(w, x) = 1 - \text{dist}(w, x) \]

Guessing a **property** of a secret.
\[ g(w, x) = \text{is } x \text{ of gender } w? \]

Guessing a **part** of a secret.
\[ g(w, x) = \text{Does } w \text{ match the high-order bits of } x? \]

Guessing a secret in **3 tries**.
\[ g_3(w, x) = \text{Is } x \text{ an element of set } w \text{ of size } 3? \]

**Lab location:**
N 39.95185
W 75.18749

Password database example

**Secret:** database of 10-bit passwords for 1000 users:
pwd_0, pwd_1, \ldots, pwd_999

**A**: guess the password of any user
\[ W = \{ p | p \in \{0 \ldots 1023\} \} \]

**W**
\[ g(p, x) = \begin{cases} 
1, & \text{if } x[u] = p \text{ for some } u \\
0, & \text{otherwise.} 
\end{cases} \]

**V_g(\pi) = 1**

Expressiveness of \( V_g \)

- Can we **express** Bayes-vulnerability using \( g \)-vulnerability?
- \( A \) guesses the **exact** secret \( x \) in **one try**
- Guesses \( W = X \)
- Gain function:
\[
g_{id}(w, x) = \begin{cases} 
1, & \text{if } w = x \\
0, & \text{if } w \neq x. 
\end{cases} \]

- \( V_{gd} \) coincides with \( V_b \)

Expressiveness of \( V_g \)

- **What about guessing entropy?**
- It’s an uncertainly measure, so we need **loss functions**
- Guesses \( \mathcal{W} = \text{permutations of } X \)
  \( \text{eg } W_1 = (x_3, x_1, x_2) \)
  - Think of \( w \) as the order of guesses
- Loss function: \( l_G(w, x) = i \)
  where \( i \) is **the position** of \( x \) in the permutation \( w \)
- \( U_{l_g}(\pi) \) coincides with \( G(\pi) \)
### Expressiveness of $V_g$

- What about Shannon entropy?
- Again we need loss functions, and infinitely many guesses
- Guesses $\mathcal{W} = \text{probability distributions over } \mathcal{X}$
  - Think of $w$ as a way to construct the search tree
- Loss function: $l_S(w, x) = -\log_2 w_x$
  (number of questions to find $x$ using this tree)
- Because of Gibb’s inequality: $U_G(\pi) = H(\pi)$
- We can restrict to countably many guesses

### Plan of the course

**MPRI 2.3.2 - Foundations of privacy**

**Lecture 3**

Kostas Chatzikokolakis

Jan 25, 2018

Quantitative Information Flow

- Motivation, application examples
- Secrets and vulnerability
- Channels and leakage
- Multiplicative Bayes-Capacity
- Comparing systems, the lattice of information
- Applications and exercises
Expressiveness of $V_g$

- Bayes-vulnerability
  
  $$W = \mathcal{X}$$
  
  $$g_{sd}(w, x) = \begin{cases} 
  1, & \text{if } w = x, \\
  0, & \text{if } w \neq x.
  \end{cases}$$

- Guessing-entropy
  
  $$W = \text{permutations of } \mathcal{X}$$
  
  $$l_G(w, x) = \text{index of } w \text{ in } x$$

- Shannon-entropy
  
  $$W = \mathcal{P} \mathcal{X}$$
  
  $$l_S(w, x) = -\log_2 w_x$$

Desired properties of uncertainty measures

- **Domain and range**: $f : \mathcal{P}(\mathcal{X}) \rightarrow [0, \infty)$

- **Continuity**: a small change in $\pi$ should have a small effect in $f(\pi)$

- **Concavity**
  - We flip a coin give to the adversary the prior $\pi^1$ with pb $c$ and prior $\pi^2$ with pb $1 - c$
  - His uncertainty on average is $cf(\pi^1) + (1 - c)f(\pi^2)$
  - If we give him a **single prior** $\pi = c\pi^1 + (1 - c)\pi^2$ his uncertainly should be at least as big
    
    $$f(\sum_i c_i\pi^i) \geq \sum_i c_if(\pi^i) \quad \text{where } \sum_i c_i = 1$$

Implies continuity everywhere except on the boundary

Shannon-entropy, Bayes-vulnerability and Guessing-entropy satisfy these properties
Desired properties of uncertainty measures

- Let $\mathcal{U}$ denote the set of all uncertainty measures (i.e. non-negative continuous concave functions of $\mathcal{P}$).
- Let $\mathcal{L}$ denote the set of $l$-uncertainty functions $U_l$.
- What's the relationship between $\mathcal{U}$ and $\mathcal{L}$?

Direction $\mathcal{U} \supseteq \mathcal{L}$

- We might have an infinite number of guess vectors $w$.
- $U_l(\pi) = \inf_w l_w(\pi)$ where $l_w(\pi) = \sum x \pi_x l(w, x)$.
- Is $U_l(\pi)$ concave?
  - Yes as $\inf$ of concave functions.
- Is $U_l(\pi)$ continuous?
  - Upper semi-continuous as $\inf$ of continuous functions.
  - Lower semi-continuous due to the Gale-Klee-Rockafellar theorem.
- So $\mathcal{L} \subseteq \mathcal{U}$.
  - What about the converse?

Geometric view of $U_l$

- A guess $w$ can be though as a vector in $\mathbb{R}^n$ containing the loss for each secret $x$:
  $$(l(w, x_1), \ldots, l(w, x_n))$$
- Loss for a fixed $w$:
  $$l_w(\pi) = \sum x \pi_x w_x = \pi \cdot w$$
- The graph of $l_w$ is a hyperplane with parameters $a = (-w, 1)$, $b = 0$.
- Any hyperplane $(a, 1) \cdot (\pi, y) = b$ is the graph of $l_w$ for a guess vector $w = b1 - a$.

Direction $\mathcal{U} \subseteq \mathcal{L}$

Supporting hyperplane theorem

If $S$ is a convex set and $x$ is a point on the boundary of $S$, then there is a supporting hyperplane that contains $x$ (i.e. all points lie on one side of the hyperplane and $x$ lies on the hyperplane).
Direction $\mathcal{U} \subseteq \mathcal{L}$

**Theorem**
Let $f \in \mathcal{U}$. There exist a loss function $l$ with a countable number of guesses $s.t. f = U$. Hence $\mathcal{U} = \mathcal{L}$.

- One guess for each $\pi$ in the interior of $\mathcal{P} \mathcal{X}$
- Apply the supporting hyperplane thm on the hypo-graph of $f$
- From the hyperplane construct a guess vector $w_\pi$ s.t.$$
  w_\pi \cdot \pi = f(\pi) \quad \text{for } \pi \text{ in particular, and}
  w_\pi \cdot \pi' \geq f(\pi') \quad \text{for all (other) } \pi'$
- Conclude that $U = f$
- Restrict to $\pi$ with rational coordinates

---

### Plan of the course

Quantitative Information Flow
- Motivation, application examples
- Secrets and vulnerability
- Channels and leakage
- Multiplicative Bayes-Capacity
- Comparing systems, the lattice of information
- Applications and exercises

---

### Channels

- Basic model from information theory to capture the behaviour of a system
- **Inputs** $\mathcal{X}$: secret events
- **Outputs** $\mathcal{Y}$: observable events

---

Probabilistic systems are **noisy** channels: an output can correspond to different inputs, and an input can generate different outputs, according to a prob. distribution

$p(o|s)$: the conditional probability to observe $o_i$ given the secret $s_i$
Channel Matrix

\[
\begin{pmatrix}
y_1 & \cdots & y_n \\
x_1 & C_{11} & \cdots & C_{1n} \\
\vdots & \ddots & \ddots & \vdots \\
x_m & C_{m1} & \cdots & C_{mn}
\end{pmatrix}
\]

- Rows are probability distributions over the observations \(Y\).
- Prior \(\pi\): distribution over the secrets \(X\).
- Given \(\pi, C\) we can construct a joint distribution over \(X \times Y\):
  \[p(x, y) = \pi_x C_{xy}\]
- For this distribution we have:
  \[p(x) = \sum_y p(x, y) = \pi_x\]
  \[p(y|x) = \frac{p(x, y)}{p(x)} = C_{xy}\]

**DC Nets**

Example: DC nets (ring of 3 nodes, \(b=1\))

**Channel Matrix**

Particular case: **Deterministic systems**
In these systems an input generates only one output
Still interesting: the problem is how to retrieve the input from the output

The entries of the channel matrix can be only 0 or 1

**DC Nets**

Example: DC nets (ring of 3 nodes, \(b=1\))
**Password Checker I**

Let us construct the channel matrix

```
out := OK
for i = 1, ..., N do
    if x_i ≠ K_i then
        out := FAIL
    end if
end for
```

**Note:** The string $x_1x_2x_3$ typed by the user is a parameter, and $K_1K_2K_3$ is the channel input.

The standard view is that the input represents the secret. Hence we should take $K_1K_2K_3$ as the channel input.

---

**Example: DC nets (ring of 3 nodes, $b=1$)**

<table>
<thead>
<tr>
<th>Secret Information</th>
<th>Observables</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_0$</td>
<td>001</td>
</tr>
<tr>
<td>$n_1$</td>
<td>010</td>
</tr>
<tr>
<td>$n_2$</td>
<td>100</td>
</tr>
<tr>
<td>$n_3$</td>
<td>111</td>
</tr>
</tbody>
</table>

**Example: DC nets (ring of 3 nodes, $b=1$)**

<table>
<thead>
<tr>
<th>$n_0$</th>
<th>001</th>
<th>010</th>
<th>100</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_1$</td>
<td>010</td>
<td>010</td>
<td>010</td>
<td>010</td>
</tr>
<tr>
<td>$n_2$</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

fair coins: $Pr(0) = Pr(1) = \frac{1}{2}$

**strong anonymity**

biased coins: $Pr(0) = \frac{2}{3}$, $Pr(1) = \frac{1}{3}$

The source is more likely to declare 1 than 0.

---

**Assume the user string is $x_1x_2x_3 = 110$**

Let us construct the channel matrix

```
Input: $K_1K_2K_3 \in \{000, 001, \ldots, 111\}$
Output: $out \in \{OK, FAIL\}$
```

**Different values of $x_1x_2x_3$ give different channel matrices, but they all have this kind of shape (seven inputs map to Fail, one maps to OK)**
Assume the user string is \( x_1x_2x_3 = 110 \)

Let us construct the channel matrix

\[
\begin{array}{cccc}
(Fail, 1) & (Fail, 2) & (Fail, 3) & OK \\
\hline
000 & 1 & 0 & 0 \\
001 & 1 & 0 & 0 \\
010 & 1 & 0 & 0 \\
011 & 1 & 0 & 0 \\
100 & 0 & 1 & 0 \\
101 & 0 & 1 & 0 \\
110 & 0 & 0 & 1 \\
111 & 0 & 0 & 1 \\
\end{array}
\]

Assume the adversary can measure the execution time

Input: \( K_1K_2K_3 \in \{000, 001, \ldots, 111\} \)

Output: \( out \in \{OK, (FAIL, 1), (FAIL, 2), (FAIL, 3)\} \)

Assume the user string is \( x_1x_2x_3 = 110 \)

Posterior distributions

\[
\begin{align*}
& y_1 & y_2 \\
& x_1 & 2/4 & 2/4 \\
& x_2 & 1/4 & 3/4 \\
\end{align*}
\]

- Each observation \( y \) provides evidence for some secret(s)
- Starting from \( \pi \), it gives a posterior dist. \( \sigma^y \in \mathcal{P} \mathcal{X} \):
  \[
  \sigma^y_x = \frac{p(x,y)}{p(y)} \quad \text{Bayes thm}
  \]
- Eg. from \( \pi = (1/2, 1/2) \) we get:
  \[
  \sigma^{y_1} = (2/3, 1/3) \quad \sigma^{y_2} = (2/5, 3/5)
  \]

Hyper distributions

\[
\pi \quad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix} \quad C \quad x_1 \quad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 2/4 \\ 2/4 \end{bmatrix} \quad \text{joint} \quad x_1 \quad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 2/8 \\ 2/8 \end{bmatrix}
\]

- Posterior: \( \sigma^{y_1} = (2/3, 1/3) \), \( \sigma^{y_2} = (2/5, 3/5) \)
- Output distribution \( \delta \in \mathcal{P} \mathcal{Y} \):
  \[
  \delta_y = p(y) = \sum_x p(x,y) = \sum_x \pi_x c_{xy} \quad \text{hence}
  \delta = \pi C = (3/8, 5/8)
  \]
- Hyper distribution \( \delta \in \mathcal{P}^2 \mathcal{X} \):
  \[
  \begin{bmatrix} 3/8 & 5/8 \\
  x_1 & 2/3 & 2/5 \\
  x_2 & 1/3 & 3/5 \\
  \end{bmatrix}
  \]

- \( [\pi, C] \): hyper distribution obtained from \( \pi, C \)
  - \( \delta \in \mathcal{P} \mathcal{Y} \): outer distribution (output)
  - \( \sigma^y \in \mathcal{P} \mathcal{X} \): inner distributions (posteriors)
- \( \pi \) can be obtained by averaging the inners
  \[
  \pi = \sum_y \delta_y \sigma^y
  \]
- Abstract channel \( C \): mapping \( \mathcal{P} \mathcal{X} \rightarrow \mathcal{P}^2 \mathcal{X} \):
  \[
  C(\pi) = [\pi, C]
  \]
Posterior vulnerability

- After running $C$ on $\pi$ we get $[\pi, C]$
- How vulnerable is $[\pi, C]$?
- We have vulnerability measures on $\mathcal{P}X$, we need to extend them to $\mathcal{PP}X$.
- Natural choice: averaging:
  $$V[\pi, C] = \sum_y \delta_y V(\sigma^y)$$
- Natural geometric view
- Low probability observations are not considered important!

Plan of the course

Quantitative Information Flow
- Motivation, application examples
- Secrets and vulnerability
- Channels and leakage
- Multiplicative Bayes-Capacity
- Comparing systems, the lattice of information
- Applications and exercises

Hyper distributions

$$\pi \begin{bmatrix} x_1 \ \ 1/2 \\ x_2 \ \ 1/2 \end{bmatrix} \quad \mathcal{C} \begin{bmatrix} x_1 \ \ 2/4 \\ x_2 \ \ 1/4 \end{bmatrix} \quad \text{joint} \begin{bmatrix} y_1 \ \ 2/8 \\ y_2 \ \ 2/8 \end{bmatrix}$$

Posters: $\sigma^h = (2/3, 1/3), \sigma^{\mathcal{Y}} = (2/5, 3/5)$

Output distribution $\delta \in \mathcal{PY}$:
  $$\delta_y = \rho(y) = \sum_x \rho(x, y) = \sum_x \pi_x c_{xy} \quad \text{hence}$$
  $$\delta = \pi c = (3/8, 5/8)$$

Hyper distribution $\delta \in \mathcal{P}^2 X$:

$$\begin{bmatrix} x_1 \ \ 2/3 \\ x_2 \ \ 1/3 \end{bmatrix} \quad \begin{bmatrix} y_1 \ \ 3/8 \\ y_2 \ \ 5/8 \end{bmatrix}$$
Posterior vulnerability

After running \( C \) on \( \pi \) we get \([\pi, C]\)  
How vulnerable is \([\pi, C]\)?  
We have vulnerability measures on \( \mathcal{P}\mathcal{X} \), we need to extend them to \( \mathcal{P}\mathcal{P}\mathcal{X} \)  
Natural choice: averaging:  
\[
V[\pi, C] = \sum y \delta_y V(\sigma^y)
\]
Natural geometric view  
Low probability observations are not considered important!

Leakage

\( V[\pi, C] \) might be high because of \( \pi \), not because of \( C \)  
eg easy to guess password  
What we care about is how much more vulnerable our secret becomes because of \( C \)  
So we compare \( V[\pi] \) and \( V[\pi, C] \)  
Additive case:  
\[
\mathcal{L}^+ (\pi, C) = V[\pi, C] - V[\pi]
\]
Multiplicative case:  
\[
\mathcal{L}^\times (\pi, C) = \frac{V[\pi, C]}{V[\pi]}
\]

Example

Prior vulnerability: \( V[\pi] = 1/2 \)  
Individual posteriors: \( V[\sigma^1] = 2/3, V[\sigma^2] = 3/5 \)  
Posterior vuln.: \( V[\pi, C] = 3/8 \cdot 2/3 + 5/8 \cdot 3/5 = 5/8 \)  
Leakage: \( \mathcal{L}^+ (\pi, C) = 1/8, \mathcal{L}^\times = 5/4 \)
Leakage

**Question**
Can $\mathcal{L}^+ (\pi, C)$ be negative?

Zero Leakage

- **Non-interfering** channel: all rows are the same, i.e. $C_{xy} = C_{x'y'}$ for all $x, x', y$
- Fix some arbitrary **full-support** prior $\pi$
- NI iff input and output are independent
- If NI then $\mathcal{L}^+ (\pi, C) = 0$
- Is the converse true?

Imperfect Cancer Test

<table>
<thead>
<tr>
<th>$\pi$</th>
<th>$Y$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>0.008</td>
<td>0.992</td>
</tr>
<tr>
<td>$N$</td>
<td>0.07</td>
<td>0.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$C$</th>
<th>$Y$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>0.90</td>
<td>0.10</td>
</tr>
<tr>
<td>$N$</td>
<td>0.07</td>
<td>0.93</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>joint $Y$</th>
<th>$Y$</th>
<th>$N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>0.00720</td>
<td>0.00080</td>
</tr>
<tr>
<td>$N$</td>
<td>0.06944</td>
<td>0.92256</td>
</tr>
</tbody>
</table>

Assuming a “generally healthy” population, the **pb of cancer is low**

The probability of getting $Y$ as a **false positive** is higher than that of a **true positive**!

Hence the **best guess** is always to guess no cancer! (the test by itself is useless)

**Geometric view:** posteriors fall on the same hyperplane!

Capacity

- Often we want to **abstract from a specific prior**
- **Capacity:** maximize leakage over $\pi$
  - $\mathcal{M} \mathcal{L}^+ (C) = \max_\pi \mathcal{L}^+ (\pi, C)$
  - $\mathcal{M} \mathcal{L}^\times (C) = \max_\pi \mathcal{L}^\times (\pi, C)$
Multiplicative Bayes-Capacity

\[ \mathcal{M}L_b^\times (C) = \max_{\pi} \frac{V_b[\pi, C]}{V_b[\pi]} \]

Given for a uniform prior

Theorem: for any channel \( C \):

\[ \mathcal{M}L_b^\times (C) = L_b^\times (\pi_u, C) = \sum_y \max_x C_{xy} \]

Geometric view: the posterior must be the same as the prior

Password Checker I

A universal upper bound for leakage!

Theorem

For any channel \( C \), prior \( \pi \) and gain function \( g \):

\[ L_b^\times (\pi, C) \leq \mathcal{M}L_b^\times (C) = \sum_y \max_x C_{xy} \]

Assume the user string is \( x_1x_2x_3 = 110 \)

Let us construct the channel matrix

Input: \( K_1K_2K_3 \in \{000, 001, \ldots, 111\} \)

Output: \( \text{out} \in \{\text{OK, FAIL}\} \)
Password Checker II

Assume the user string is $x_1x_2x_3 = 110$

Assume the adversary can measure the execution time

Let us construct the channel matrix

Input: $K_1, K_2, K_3 \in \{000, 001, \ldots, 111\}$

Output: $out \in \{OK, (FAIL, 1), (FAIL, 2), (FAIL, 3)\}$

Input:

<table>
<thead>
<tr>
<th>out</th>
<th>K1</th>
<th>K2</th>
<th>K3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>000</td>
<td>001</td>
<td>010</td>
</tr>
<tr>
<td>OK</td>
<td>001</td>
<td>001</td>
<td>011</td>
</tr>
<tr>
<td>OK</td>
<td>010</td>
<td>010</td>
<td>011</td>
</tr>
<tr>
<td>FAIL</td>
<td>011</td>
<td>011</td>
<td>100</td>
</tr>
<tr>
<td>FAIL</td>
<td>100</td>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>FAIL</td>
<td>101</td>
<td>101</td>
<td>110</td>
</tr>
<tr>
<td>FAIL</td>
<td>110</td>
<td>110</td>
<td>111</td>
</tr>
</tbody>
</table>

Output:

<table>
<thead>
<tr>
<th>out</th>
<th>K1</th>
<th>K2</th>
<th>K3</th>
</tr>
</thead>
<tbody>
<tr>
<td>OK</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OK</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>OK</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FAIL</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FAIL</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FAIL</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FAIL</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>FAIL</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>FAIL</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FAIL</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>FAIL</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Exam 2015-16, Question 4 (20%)

Let $C$ be a channel from $\mathcal{X}$ to $\mathcal{Y}$.

5.1 Show that for any prior $\pi$ and gain function $g$:

$\mathcal{L}_g^\pi(\pi, C) \leq |\mathcal{Y}|$ and $\mathcal{L}_g^\pi(\pi, C) \leq |\mathcal{X}|$

5.2 Let $\pi_u$ be the uniform prior. Show that

$(\forall g: \mathcal{L}_g^\pi(\pi_u, C) = 1)$

if and only if $C$ is non-interfering.

Comparing channels

When can we say that $C_1$ is better than $C_2$?

We could check $\mathcal{L}_g(\pi, C_1) \leq \mathcal{L}_g(\pi, C_2)$

What about different $\pi, g$’s? We want robustness!

Partition refinement

Any deterministic channel $C$ induces a partition on $\mathcal{X}$.

$x_1$ and $x_2$ are in the same block iff they map to the same output.

$C_{\text{country}}$

person $\rightarrow$ country of birth

$C_{\text{state}}$

person $\rightarrow$ state of birth

Partition refinement ☐: Subdivide zero or more of the blocks.
Leakage ordering

\[ C_1 \leq_m C_2 \quad m \in \{ \text{Shannon, min-entropy, guessing entropy} \} \]
the leakage of \( C_1 \) is no greater than that of \( C_2 \) for all priors

**Theorem (Yasuoka, Terauchi, Malacaria)**
On deterministic channels, the relations below coincide:

- \( \leq \) Shannon entropy
- \( \leq \) min-entropy
- \( \leq \) guessing entropy
- \( \subseteq \)

Can we apply this to any of the examples seen so far?

**Password Checker I**

Assume the user string is \( x_1x_2x_3 = 110 \)
Let us construct the channel matrix

**Input:** \( K_1K_2K_3 \in \{000, 001, \ldots, 111\} \)
**Output:** \( \text{out} \in \{\text{OK, FAIL}\} \)

Different values of \( x_1x_2x_3 \) give different channel matrices, but they all have this kind of shape (seven inputs map to Fall, one maps to OK)

**Password Checker II**

Assume the user string is \( x_1x_2x_3 = 110 \)
Assume the adversary can measure the execution time
Let us construct the channel matrix

**Input:** \( K_1K_2K_3 \in \{000, 001, \ldots, 111\} \)
**Output:** \( \text{out} \in \{\text{OK, (FAIL, 1), (FAIL, 2), (FAIL, 3)}\} \)

Composition refinement

How can we generalize this to probabilistic channels?

First issue: \( \subseteq \) is not defined for probabilistic channels

\( C_{\text{merge}}: \text{state} \rightarrow \text{country} \)
\( C_{\text{country}} = C_{\text{state}}C_{\text{merge}} \)

**Definition (composition-refinement)**
\( C_1 \subseteq_C C_2 \iff C_1 = C_2C_3 \text{ for some } C_3, \)

**Theorem**
For deterministic channels, \( \subseteq \) and \( \subseteq_C \) coincide.
\( \subseteq_C \) is a promising generalization of \( \subseteq \) to probabilistic channels.
Refinement and leakage ordering

composition refinement $\overset{?}{\iff}$ leakage order

for probabilistic channels?

---

Plan of the course

Quantitative Information Flow
- Motivation, application examples
- Secrets and vulnerability
- Channels and leakage
- Multiplicative Bayes-Capacity
- Comparing systems, the lattice of information
- Applications and exercises

---

Leakage ordering

$C_1 \leq_m C_2 \quad m \in \{\text{Shannon, min-entropy, guessing entropy}\}$
the leakage of $C_1$ is no greater than that of $C_2$ for all priors

**Theorem (Yasuoka, Terauchi, Malacaria)**

On deterministic channels, the relations below coincide:

- $\leq$ Shannon entropy
- $\leq$ min-entropy
- $\leq$ guessing entropy
- $\subseteq$

Directly applicable to the password example!
Composition refinement

How can we generalize this to probabilistic channels?

**Definition (composition-refinement)**

\[ C_1 \sqsubseteq_\circ C_2 \text{ iff } C_1 = C_2 C_3 \text{ for some } C_3, \]

\[ X \xrightarrow{c_1} Y \xrightarrow{c_2} Z \xrightarrow{c_3} \]

**Theorem**

For deterministic channels, \( \sqsubseteq \) and \( \sqsubseteq_\circ \) coincide.

\( \sqsubseteq_\circ \) is a promising generalization of \( \sqsubseteq \) to probabilistic channels.

Refinement and leakage ordering

composition refinement \( \overset{?}{\leftrightarrow} \) leakage order

for probabilistic channels?

\( V_g \) and optimal strategies

- **Strategy** \( S \): mapping outputs to guesses
- Represented as a channel \( S: Y \to W \) (possibly deterministic)
- \( CS \): composed
- **Composition** \( CS \): channel from \( Y \) to \( W \)

\[ X \xrightarrow{c_1} Y \xrightarrow{c_2} W \]

- Expected gain: \( E_{\pi, CS}(g) = \sum_{x, w} \pi_x CS_{xw} g(w, x) \)
- \( V_g(\pi, C) = \max_S E_{\pi, CS}(g) \)

Composition refinement

**Definition (composition-refinement)**

\[ C_1 \sqsubseteq_\circ C_2 \text{ iff } C_1 = C_2 C_3 \text{ for some } C_3, \]

\[ X \xrightarrow{c_1} Y \xrightarrow{c_2} Z \xrightarrow{c_3} \]

**Theorem**

\( C_1 \sqsubseteq_\circ C_2 \) implies \( L_g(\pi, C_1) \leq L_g(\pi, C_2) \) for all \( \pi, g \)

**Proof:** if \( S_1 \) is a strategy for \( C_1 \) then \( S_2 = C_3 S_1 \) is a strategy for \( C_2 \) such that \( C_1 S_1 = C_2 S_2 \).
Refinement and leakage ordering

For probabilistic channels?

Definition

\[ C_1 \leq_g C_2 \quad \text{iff} \quad \mathcal{L}_g(\pi, C_1) \leq \mathcal{L}_g(\pi, C_2) \text{ for all } \pi, g \]

Theorem

\[ C_1 \subseteq C_2 \Rightarrow C_1 \leq_g C_2 \]

What about the converse?

It turns out that \( \leq_{\text{min-entropy}} \nRightarrow \subseteq \)

On the other hand \( \leq_g \) is strong enough:

**Theorem ("Coriaceous")**

\[ C_1 \leq_g C_2 \Rightarrow C_1 \subseteq C_2 \]

The proof uses the separating hyperplane theorem!

Applications

- Dining Cryptographers
- Crowds
  - exam 2015-16, Question 6
- Timing Attacks in Cryptosystems

Dining Cryptographers

Example: DC nets (ring of 3 nodes, b=1)
Dining Cryptographers

Example: DC nets (ring of 3 nodes, b=1)

<table>
<thead>
<tr>
<th></th>
<th>001</th>
<th>010</th>
<th>100</th>
<th>111</th>
</tr>
</thead>
<tbody>
<tr>
<td>n₀</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>n₁</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
<tr>
<td>n₂</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
</tr>
</tbody>
</table>

fair coins: \( \Pr(0) = \Pr(1) = \frac{1}{2} \)

biased coins: \( \Pr(0) = \frac{2}{3}, \Pr(1) = \frac{1}{3} \)
The source is more likely to declare 1 than 0

The Crowds protocol

- **Adversary**: group of corrupted users
- \( n \) honest, \( c \) corrupted, \( m = n + c \) total
- **Secrets**: \( \mathcal{X} = \{x_1, \ldots, x_n\} \)
- **Observations**: the adversary only sees messages passing forwarded to him
- \( \mathcal{Y} = \{y_1, \ldots, y_n, \perp\} \)
  - \( y_i \) means that user \( i \) forwarded a message to the adversary

\[ C_{x_i} = \alpha = 1 - (n - 1)\gamma - \beta = \frac{n - np}{m - np} \]

\[ C_{x_i y_i} = \beta = \frac{c}{m} + \frac{1}{m} P_{y_i} \beta + \frac{n - 1}{m} P_{\perp} \gamma = \frac{c(n - p)(n - 1)}{m(m - np)} \]

\[ C_{x_i y_j} = \gamma = \frac{1}{m} P_{y_i} \beta + \frac{n - 1}{m} P_{\perp} \gamma = \frac{cm}{m(m - np)} \]

\[ C = \begin{bmatrix} y_1 & \cdots & y_0 \end{bmatrix} \begin{bmatrix} \beta & \cdots & \gamma & \alpha \\ \vdots & \ddots & \vdots & \alpha \\ x_n & \gamma & \cdots & \beta \end{bmatrix} \]
The Crowds protocol

• Posteriors:

\[ \sigma^y = (k, \ldots, k, \frac{m - pf(n - 1)}{m}, k, \ldots, k) \]

\[ \sigma^\perp = (\frac{1}{n}, \ldots, \frac{1}{n}) \]

• Bayes-Capacity:

\[ M_{\beta}^x(C) = \beta + \alpha = \frac{n(c + 1)}{c + n} \]

independent from \( pf \)

Why is \( M_{\beta}^x(C) \) independent from \( pf \)?

Is it always true for other \( \pi \) or \( g \)?

\[ n = 2, c = 1 \]

\( p \): probability of user 1

Bayes-leakage as a function of \( pf \)

\[ p = 0.3 \]
\[ p = 0.4 \]
\[ p = 0.5 \]

\[ pf = 0.0 \]
\[ pf = 0.5 \]
\[ pf = 1.0 \]
Modified Crowds

- **Modification**: the adversary can somehow know whether the forwarding is happening in the first round or not
- $n$ honest, $c$ corrupted, $m = n + c$ total
- **Secrets**: $X = \{x_1, \ldots, x_n\}$
- **Observations**: $Y = \{(y_1, 1), (y_1, 2+), \ldots, (y_n, 1), (y_n, 2+), \bot\}$
  1: first round, 2+: second round or higher

*Modified Crowds: channel*

- $C_{X,1} = \alpha = \frac{n - np_f}{m - np_f}$
- $C_{X, (y_1, 1)} = \beta = \frac{c}{m}$
- $C_{X, (y_i, 1)} = 0$ if $i \neq j$
- $C_{X, (y_i, 2)} = \gamma = \frac{1 - \alpha - \beta}{n}$

\[
C_{\text{mod}} = \begin{bmatrix}
(y_1, 1) & \cdots & (y_n, 1) & (y_1, 2) & \cdots & (y_n, 2) & \bot \\
\vdots & & \vdots & \ddots & & \ddots & \\
(x_1 \beta) & \cdots & 0 & (y_1, 2) & \cdots & (y_n, 2) & \gamma \\
(x_n \beta) & \cdots & 0 & \gamma & \cdots & \gamma & \alpha
\end{bmatrix}
\]

*Modified Crowds: leakage*

- $C_p$: original Cowds, $C_{\text{mod}}^p$: modified, for given $p_f$
- Does this channel leak more or less than the real C?
  \[ C_p \subseteq_{o} C_{\text{mod}}^p \]
- Does $V_g(\pi, C)$ depend on $p_f$?
  \[ C_{p_f}^{\text{mod}} = o C_{p_f}^{\text{mod}} \quad \forall p_f, p_f' \]

\[ = o \text{ means } \subseteq_{o} \cap \supseteq_{o} \]

*Modified Crowds: strategies*

- What is an **optimal strategy** for Bayes-vulnerability?
- What is an optimal strategy for the original C?
- Why is Bayes-leakage the same for uniform prior?
  - $(y_i, 2)$ offers no evidence, the posterior is the same as the prior. Best guess: $x$ with max $\pi_x$ (i.e. any $x$)
  - both $(y_i, 1)$ and $(y_i, 2)$ can be mapped to $x_i$ by an optimal strategy $S_{\text{mod}}$ for $C_{p_f}^{\text{mod}}$
    - The optimal strategy $S$ for $C_{p_f}$ maps $y_i$ to $x_i$
    - For the optimal strategies: $C_{p_f} S = C_{p_f}^{\text{mod}} S_{\text{mod}}$, so $V_g(\pi_{\text{uni}}, C_{p_f}) = V_g(\pi_{\text{uni}}, C_{p_f}^{\text{mod}})$
    - And hence independent from $p_f$
- What about other priors?
In the Crowds protocol, due to the probabilistic routing, each request could pass through corrupted users multiple times before arriving to the server, as shown in the figure below. However, in the security analysis, we only considered as “detected” the first user who forwards the request to a corrupted one.

To perform a more precise analysis, let us consider the first two detected users, instead of only the first one.

Let \( n, m \) be the number of honest and total users respectively. The set of secrets is still \( \mathcal{X} = \{1, \ldots, n\} \) (we are only interested in the privacy of honest users).

On the other hand, the information available to the adversary is now more detailed. Observations are of the form \( y = (d_1, d_2) \) where \( d_1 \in \{1, \ldots, n, \perp\} \) (the first detected user, similarly to the original analysis) and \( d_2 \in \{1, \ldots, m, \perp\} \) (the second detected user, who might be corrupted himself).

Show that this extra information is in fact useless to the adversary. More precisely, show that for any prior \( \pi \) and gain function \( g \):

\[
V_g(\pi, \mathcal{C}_1) = V_g(\pi, \mathcal{C}_2)
\]

where \( \mathcal{C}_2 \) is the channel obtained by the detailed analysis, considering two detected users, and \( \mathcal{C}_1 \) is the channel of the original analysis, considering a single detected user.

Timing Attacks in Cryptosystems

- Remote timing attack [BonehBrumley03]
- 1024-bit RSA key recovered in 2 hours from standard OpenSSL implementation across LAN

Timing Attacks in Cryptosystems

- Time of \( \text{Dec}(sk, ct) \) depends on both \( sk, ct \)
- Channel \( \mathcal{C}_{ct} \) for each ciphertext \( ct \)
- Input: secret key \( sk \)
- Output: time \( t \)
Timing Attacks in Cryptosystems

• First counter measure: bucketing
• Limit time to at most $b$ buckets
  - Run the decryption then wait until the next bucket
  - $b$ should be small, e.g., 5 or 6
  - much more efficient than always waiting until the max time
• What can we say about the leakage of $C_{ct}$?
  - $L^2_{\mathbb{F}}(\pi, C) \leq b$
• Is this enough?
  - No! we need to know how different $C_{ct}$ relate to each other.

Timing Attacks in Cryptosystems

• Second counter measure: blinding
• Randomize $ct$ before decryption: $ct \otimes r$
• De-randomize after decryption: obtain $\text{Dec}(sk, ct)$ from $\text{Dec}(sk, ct \otimes r)$
• Possible because of properties of RSA encryption
• Makes time independent from $ct$
• $n$ decryptions: repeated independent runs of $C$

Repeated independent runs

• Run $C$ multiple times with the same secret $x$
  
  \[
  \begin{array}{c}
  X \\
  \downarrow \\
  C \\
  \uparrow \\
  Y
  \end{array}
  \]

  \[
  \begin{array}{c}
  C \\
  Y
  \end{array}
  \]

• Output: $(a_1, \ldots, a_n) \in \mathcal{Y}^n$
• Channel $C^n$: $C^n_{x,(a_1,\ldots,a_n)} = \prod_i C_{x,a_i}$
Repeated independent runs

- Channel $T_n : \mathcal{X} \rightarrow \mathcal{T}_n$ from secrets to types
- $C_{x,t} = \prod_i C_{xy_i}$
- How does $T_n$ relate to $C^n$?

\[ C^n, t = \prod_i C_{xy_i} t_i \]

Timing Attacks in Cryptosystems

Combine bucketing (with $b$ buckets) and blinding

\[ \mathcal{ML}_b^\chi (C^n) \leq \binom{n+b-1}{n} \]

eg $n = 2^{40}$, $b = 5$, leakage at most $2^{155}$. 2048 bit key becomes as guessable as a 1893 bit key