Quantitative Information Flow

Catuscia Palamidessi
INRIA Saclay & Ecole Polytechnique

These lectures are based on work done in collaboration with:
Mario Alvim, **Miguel Andrés**, and **Konstantinos Chatzikokolakis**

Plan of the lectures

- **Lecture 1 (Monday)**
  - Secure information flow. Motivations and examples
  - Information-theoretic framework
- **Lecture 2 (Tuesday)**
  - Quantification of leakage: models of adversaries
  - Focus on: Shannon entropy and Rényi min-entropy
  - Bayes risk
- **Lecture 3 (Friday)**
  - Differential privacy
  - Relation between QIF and differential privacy

Information Flow

Problem: Avoid the leakage of information in computer systems

Leakage of secret information via public observables

Ideally: No leak

- No interference [Goguen & Meseguer'82]

In practice: There is almost always some leak

- Intrinsic to the problem
- Side channels

Example

Password checker 1

Password: $K_1K_2 \ldots K_N$
Input by the user: $x_1x_2 \ldots x_N$
Output: out (Fail or OK)

Intrinsic leakage

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

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

   end if
end for
```

Example

Password checker 2

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

out := OK
for $i = 1, \ldots, N$ **do**
  if $x_i \neq K_i$ **then**
    { out := FAIL
    exit() }
  end if
end for

More efficient, but what about security?

Example

Password checker 2

Password: $K_1K_2 \ldots K_N$
Input by the user: $x_1x_2 \ldots x_N$
Output: 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

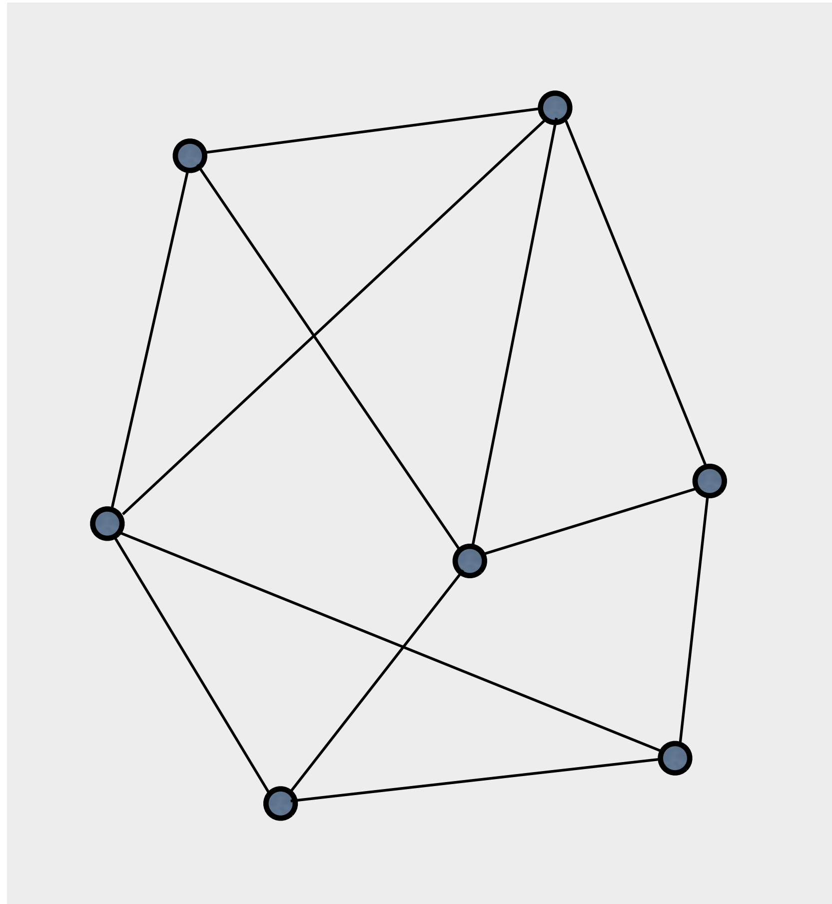
$out :=$ OK
for $i = 1, ..., N$ **do**
  if $x_i \neq K_i$ **then**
    $\left\{ \begin{array}{l} out := \text{FAIL} \\ \text{exit()} \end{array} \right\}$
  end if
end for

Quantitative Information flow

- It is necessary to **quantify** the notion of **Information Leakage**
- To this purpose, most of the recent proposals use **information-theoretic approaches**
  - Suitable also for **probabilistic** programs
- Convergence with other fields, in particular that of **anonymity protocols** which typically use randomization to hide the secrets
  - **secret = culprit's identity**

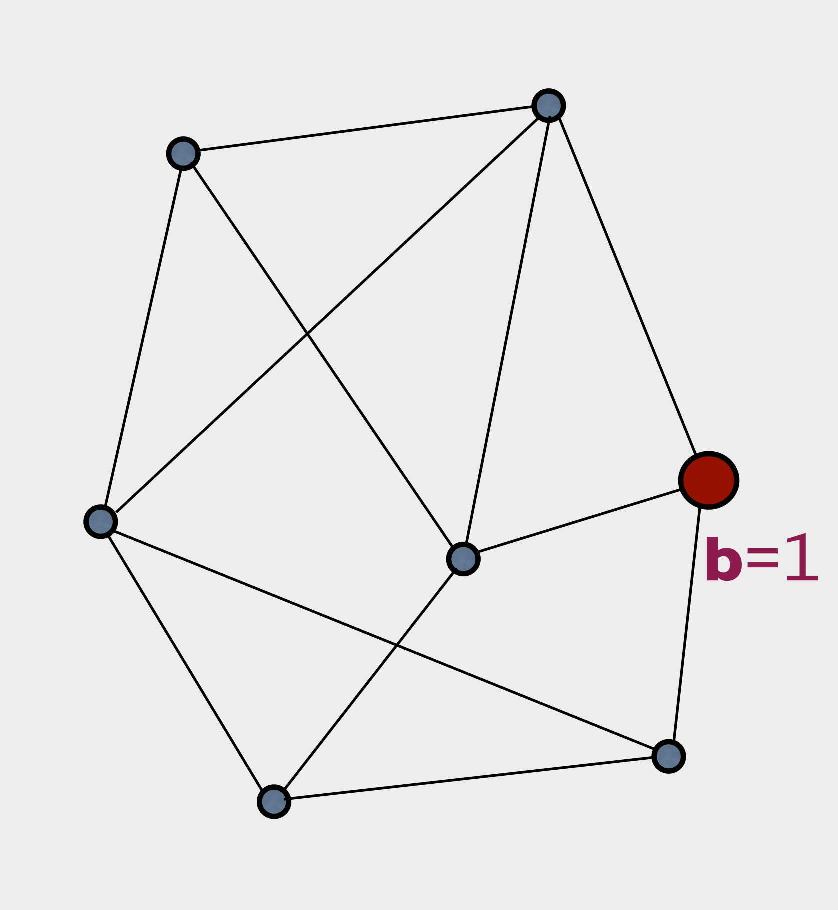
Example of Anonymity Protocol: DC Nets [Chaum'88]

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



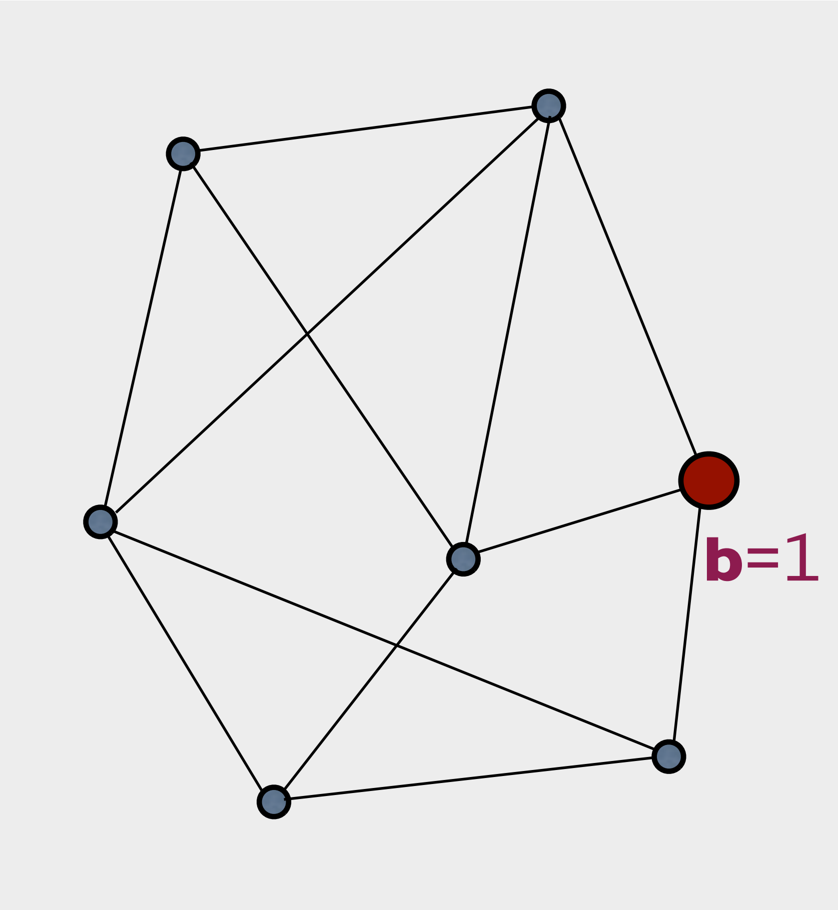
Example of Anonymity Protocol: DC Nets [Chaum'88]

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



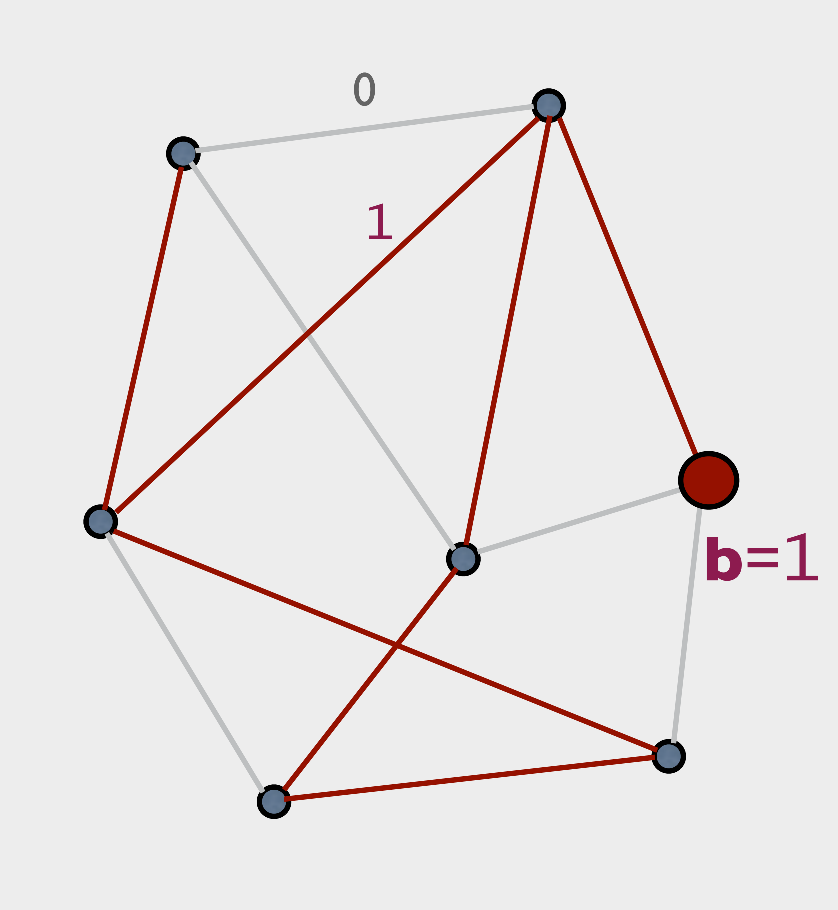
Chaum’s solution

- Associate to each edge a fair coin



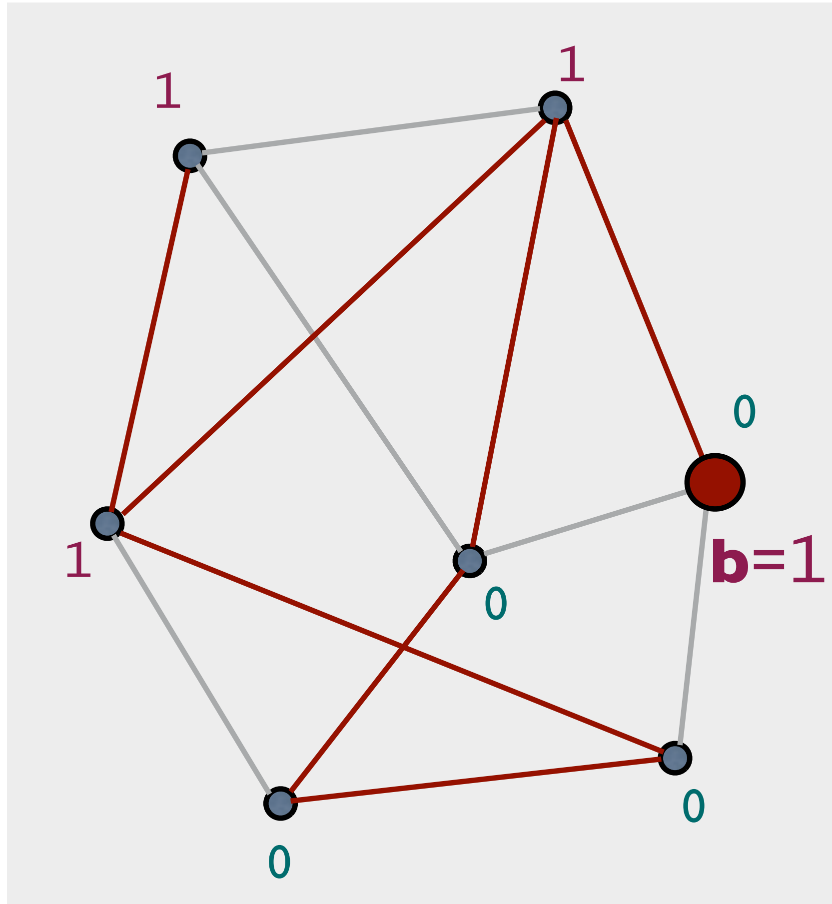
Chaum's solution

- Associate to each edge a fair coin
- Toss the coins



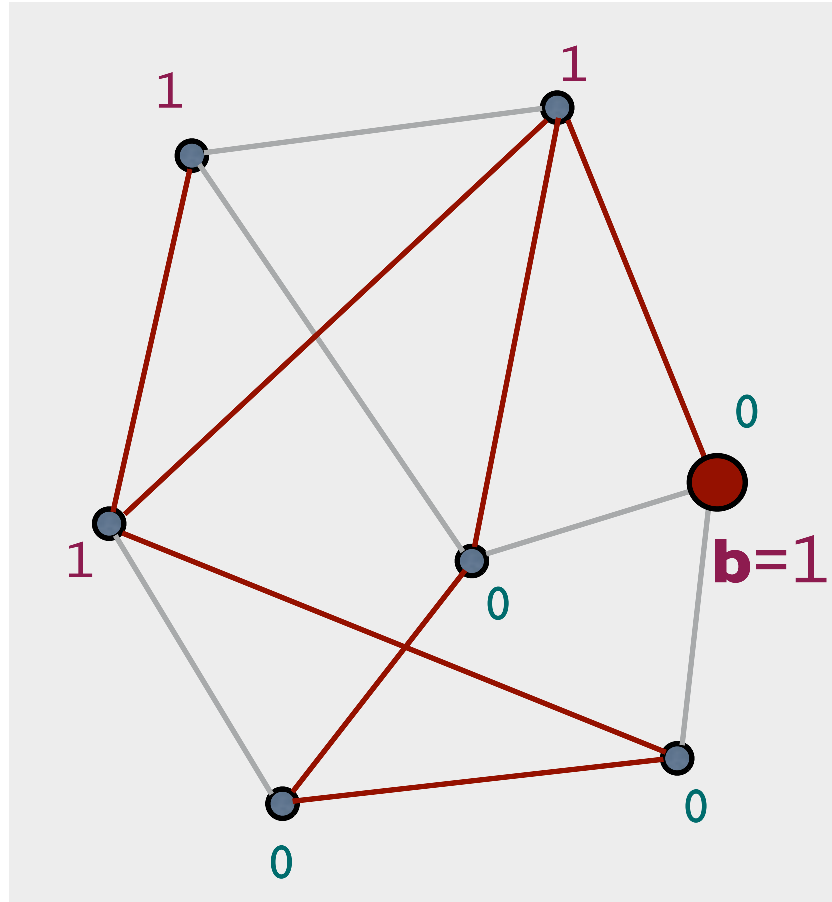
Chaum’s solution

- Associate to each edge a fair coin
- Toss the coins
- Each node computes the binary sum of the incident edges. The source adds **b**. They all broadcast their results



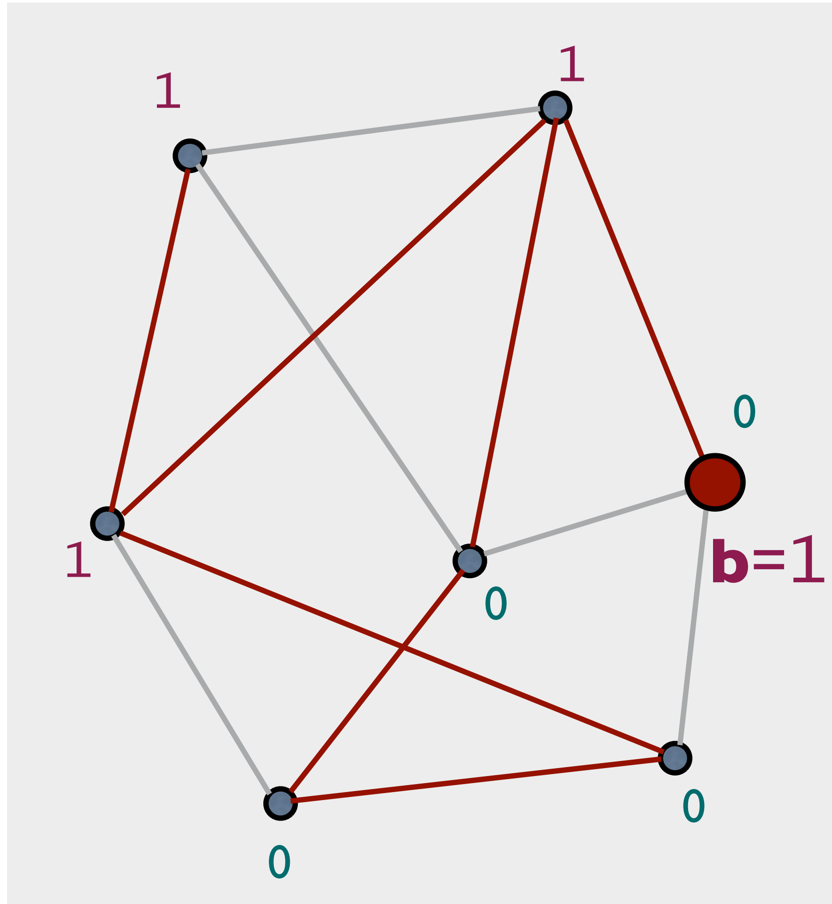
Chaum's solution

- Associate to each edge a fair 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

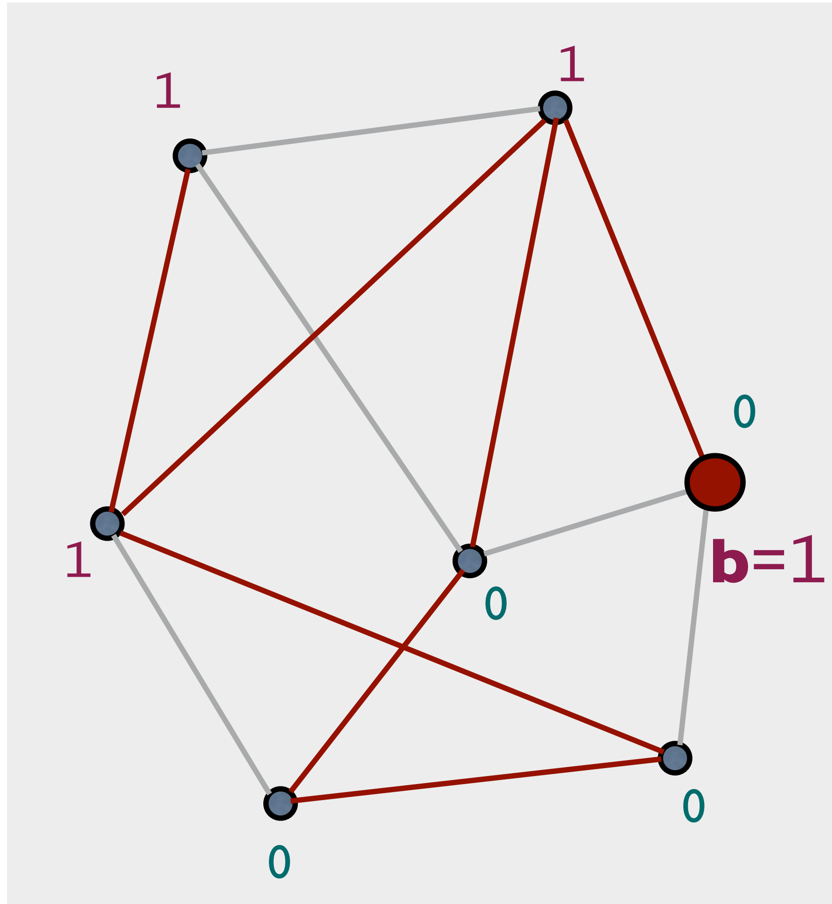
- Associate to each edge a fair coin
- Toss the coins
- Each node computes the binary sum of the incident edges. The source adds **b**. They all broadcast their results
- Achievement of the goal: Compute the total binary sum: it coincides with **b**



Question: why is that?

Chaum's solution

- Associate to each edge a fair 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**



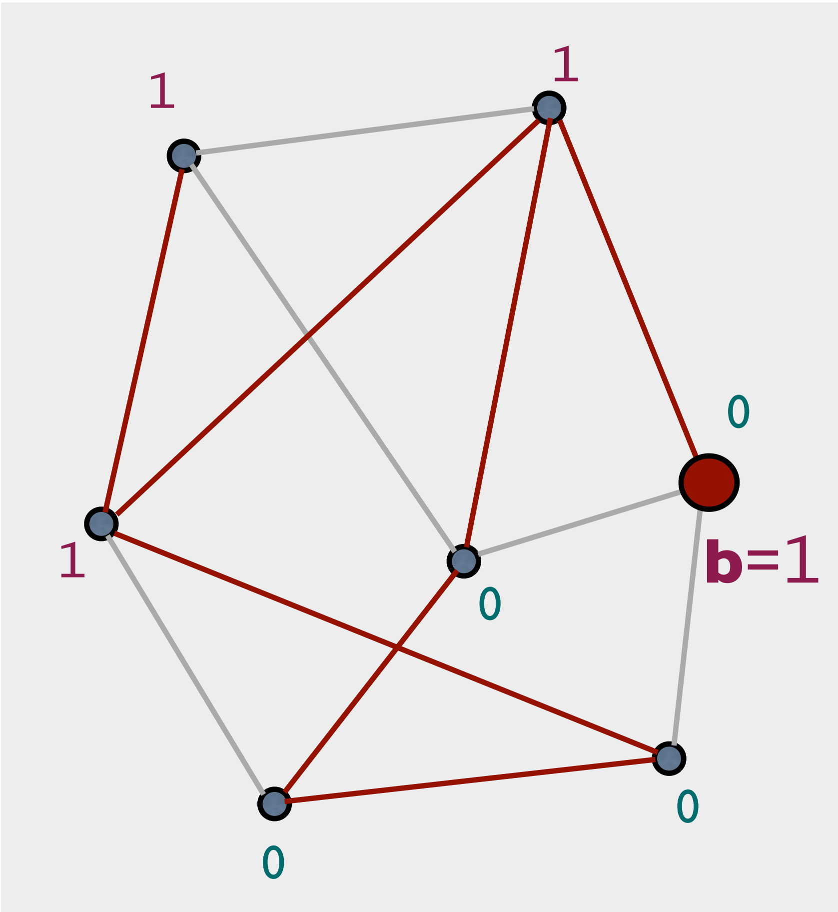
Answer: each coin is counted twice!

Strong anonymity (Chaum)

- If the graph is connected and the coins are fair, then for an external observer, the protocol satisfies **strong anonymity**:

the *a posteriori* probability that a certain node is the source is equal to its *a priori* probability

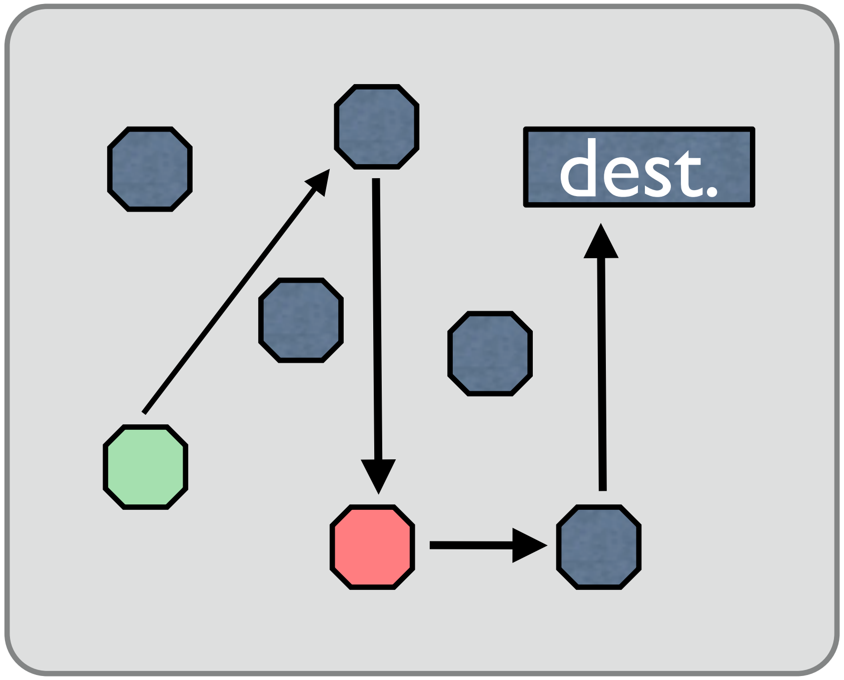
  - A priori / a posteriori = before / after observing the declarations



- Question: what about the internal nodes?

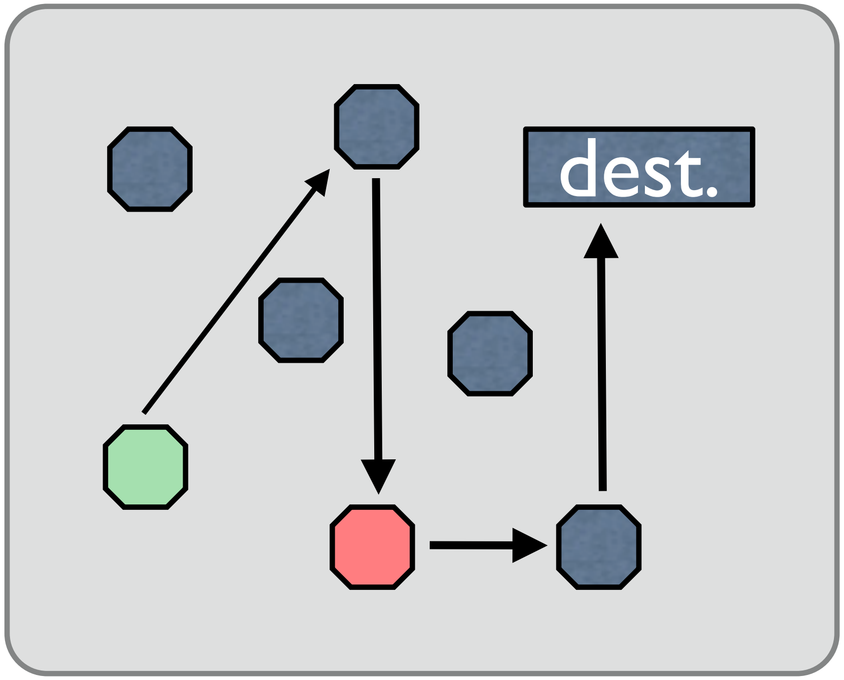
Another example: Crowds [Rubin and Reiter'98]

- Problem: A user (initiator) wants to send a message anonymously to another user (dest.)
- Crowds: A group of n users who agree to participate in the protocol.
- The initiator selects randomly another user (forwarder) and forwards the request to her
- A forwarder randomly decides whether to send the message to another forwarder or to dest.
- ... and so on



Another example: Crowds [Rubin and Reiter'98]

- Problem: A user (initiator) wants to send a message anonymously to another user (dest.)
- Crowds: A group of n users who agree to participate in the protocol.
- The initiator selects randomly another user (forwarder) and forwards the request to her
- A forwarder randomly decides whether to send the message to another forwarder or to dest.
- ... and so on



Probable innocence: under certain conditions, an attacker who intercepts the message from x cannot attribute more than 0.5 probability to x to be the initiator

Common features

- Secret information
  - the values of the high variables
  - DC: the identity of the broadcaster
  - Crowds: the identity of the initiator
- Public information (Observables)
  - the values of the low variables
  - DC: the declarations
  - Crowds: the interception of a forwarder by a corrupted user
- The system may be probabilistic
  - often the system uses randomization to obfuscate the relation between secrets and observables
  - DC: coin tossing
  - Crowds: random forwarding to another user

Simplifying assumptions

In this lectures we assume:

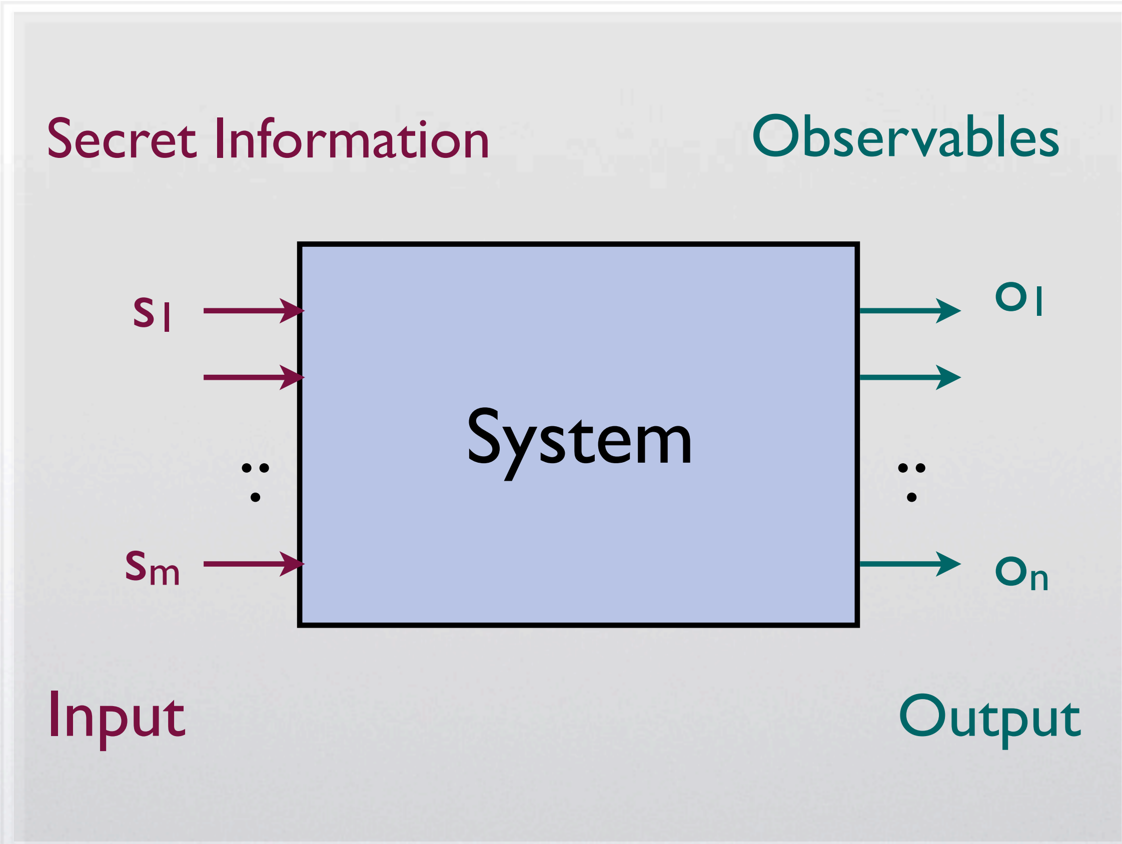
- Secrets: elements of a random variable S
- Observables: elements of a random variable O
- For each secret s, the probability that we obtain an observable o is given by p(o | s)

- No feedback: the secret is not influenced by the observables

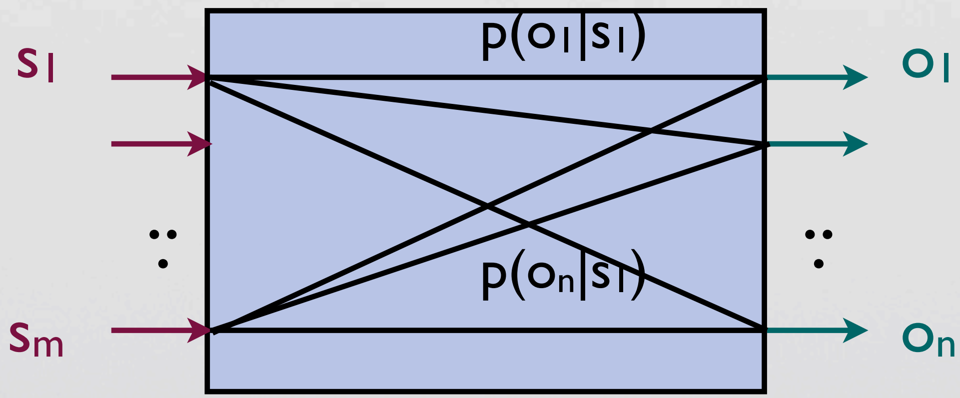
- No nondeterminism: everything is (either deterministic or) probabilistic, although we may not know the distribution

An intriguing analogy:

Systems as Information-Theoretic channels



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



$p(o_j|s_i)$: the conditional probability to observe o_j given the secret s_i

	o_1	...	o_n
s_1	$p(o_1 \mid s_1)$	...	$p(o_n \mid s_1)$
$\vdots$	$\vdots$		
s_m	$p(o_1 \mid s_m)$		$p(o_n \mid s_m)$

$$p(o|s) = \frac{p(o \ and \ s)}{p(s)}$$

A channel is characterized by its matrix: the array of conditional probabilities

In a information-theoretic channel these conditional probabilities must be independent from the input distribution

This means that we can apply the i.t. approach only to systems whose behavior may depend on the secret values, but not on their distribution

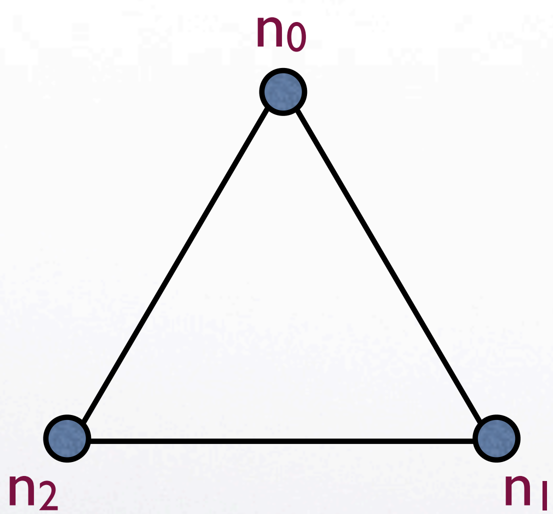
Information theory: useful concepts

- **Entropy** H(X) of a random variable X
  - a measure of the degree of uncertainty of the events
  - It can be used to measure the vulnerability of the secret, i.e. how "easy" is for the adversary to obtain the secret
- **Mutual information** I(S;O)
  - degree of correlation between the input S and the output O
  - formally defined as difference between the entropy of S ***before*** knowing O, and the entropy of S ***after*** knowing O
  - aka the difference between the a priori and the a posteriori entropy of S
  - It can be used to measure the leakage:

$$\text{Leakage} = I(S;O) = H(S) - H(S|O)$$

  - H(S|O) can be computed using the distribution of S and the matrix

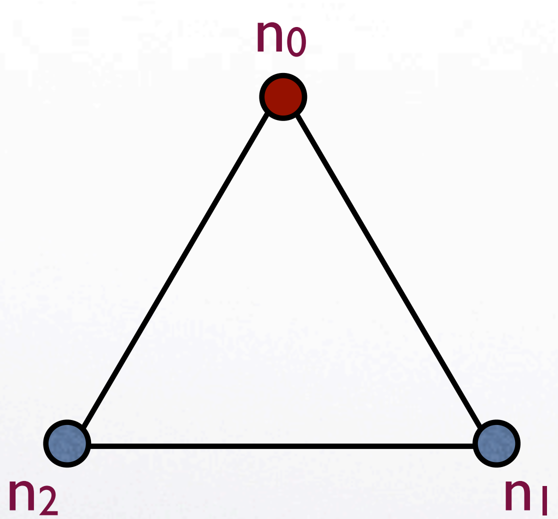
Example: DC nets (ring of 3 nodes, b=1)

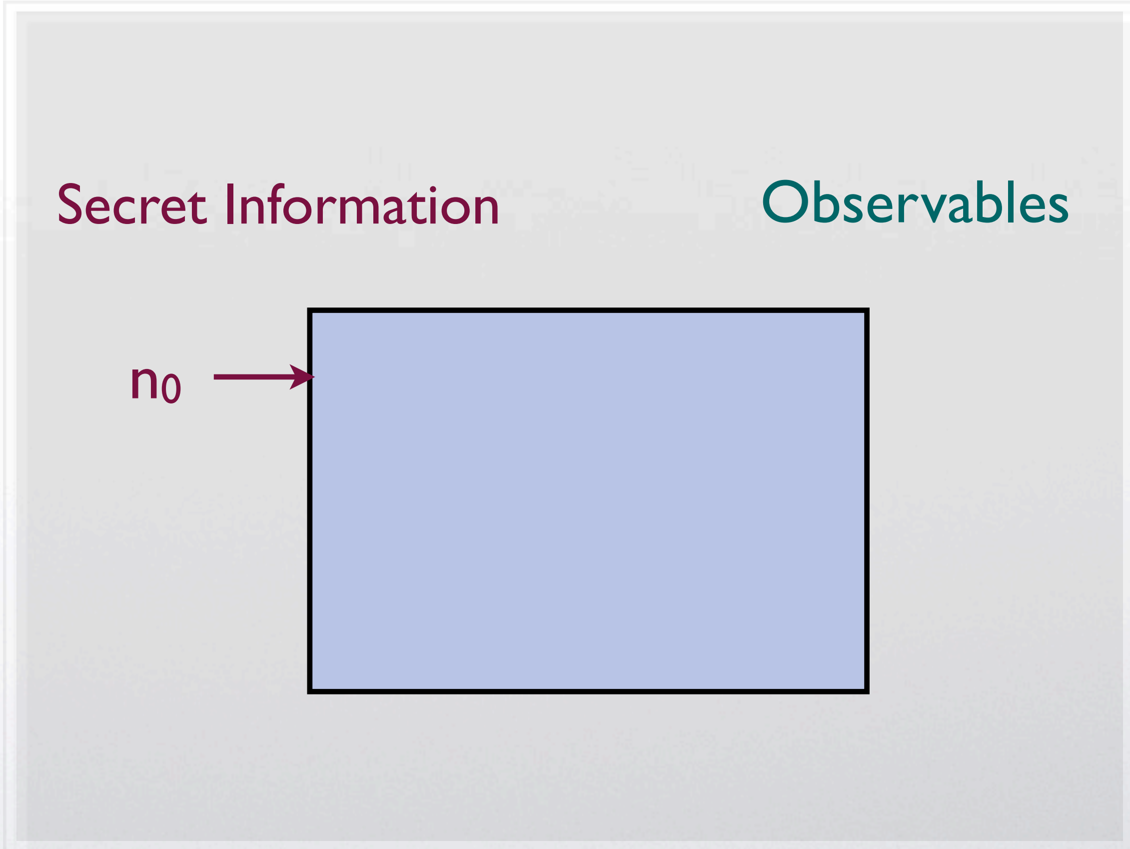


Secret Information

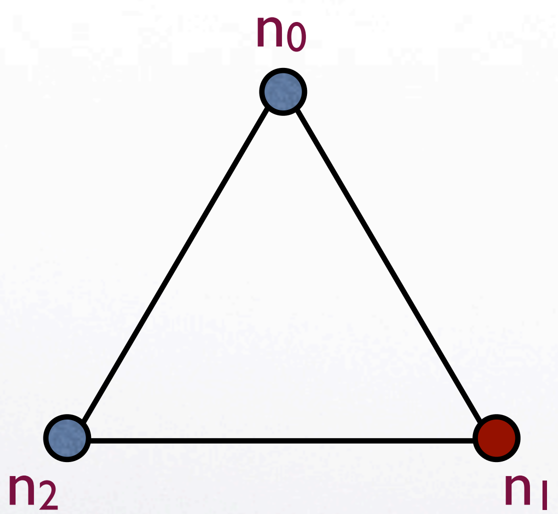
Observables

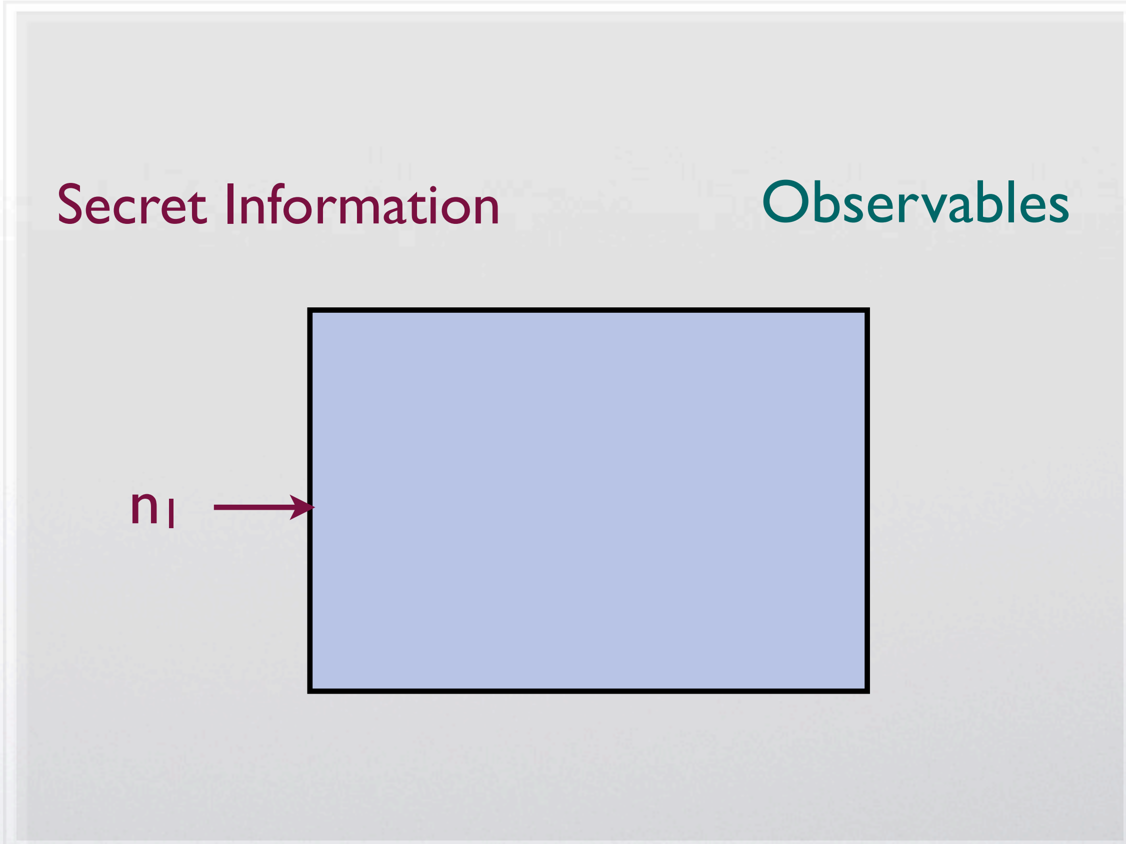
Example: DC nets (ring of 3 nodes, b=1)



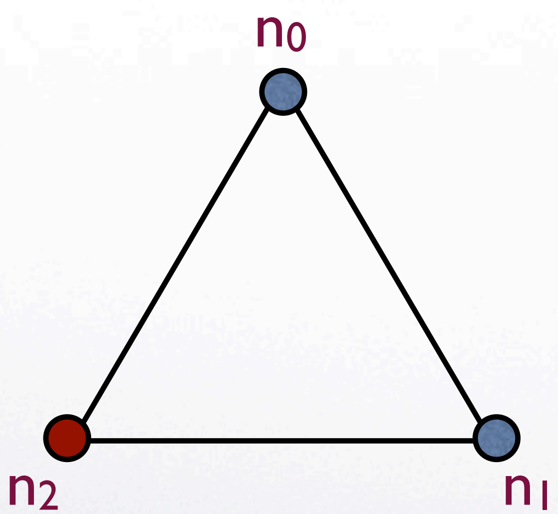


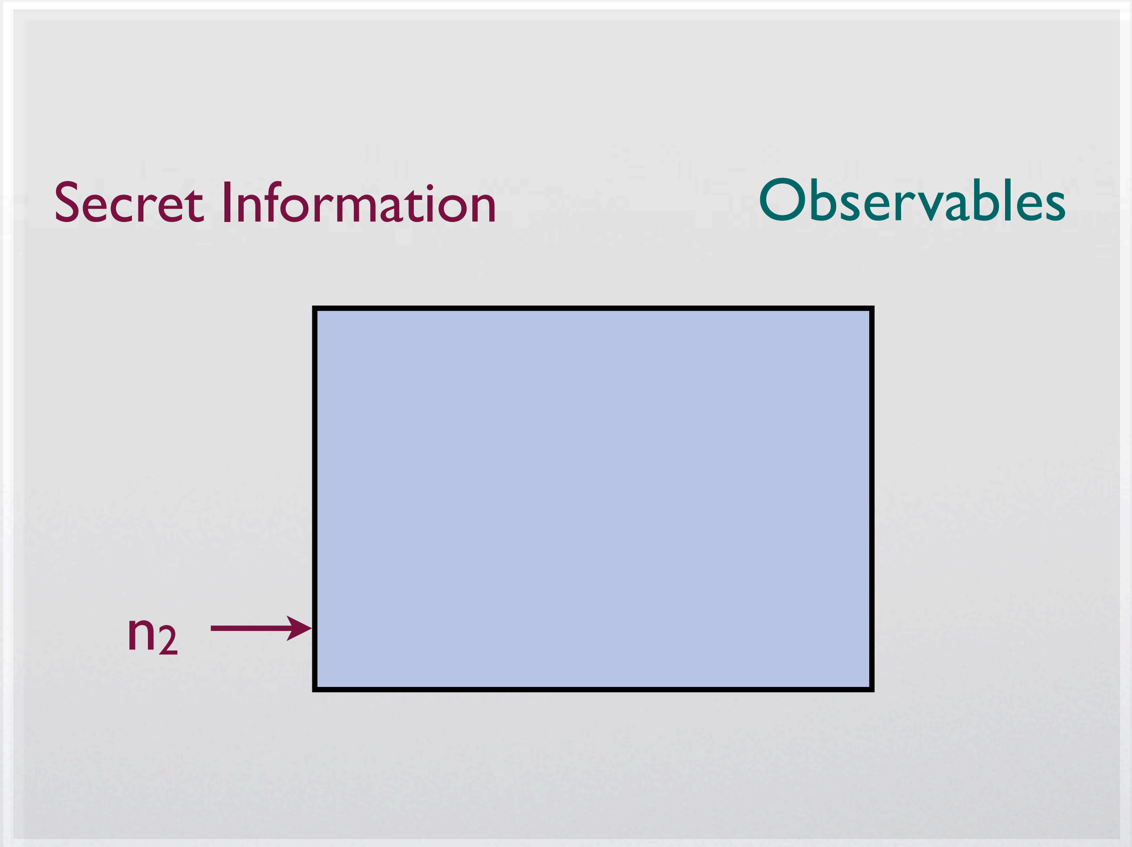
Example: DC nets (ring of 3 nodes, b=1)



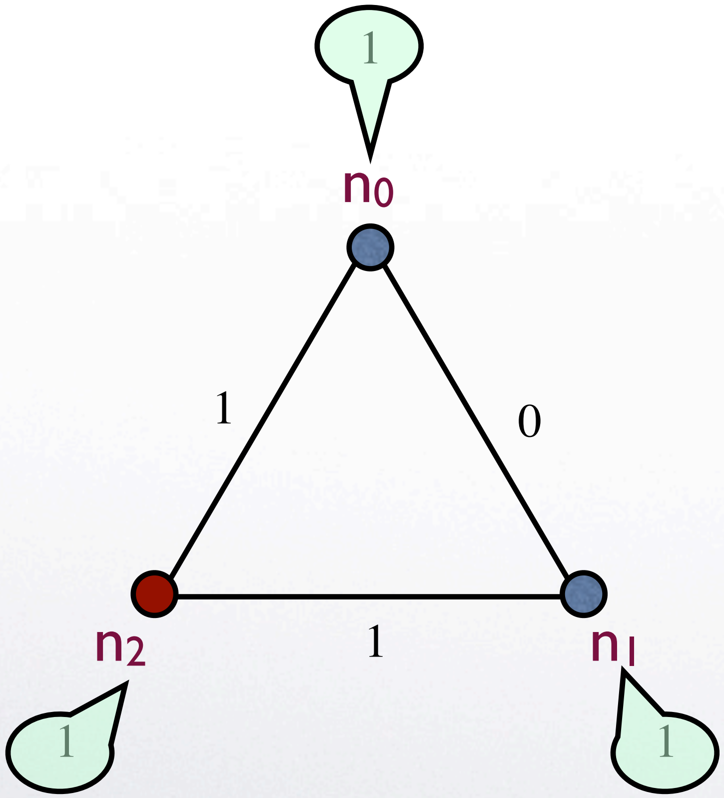


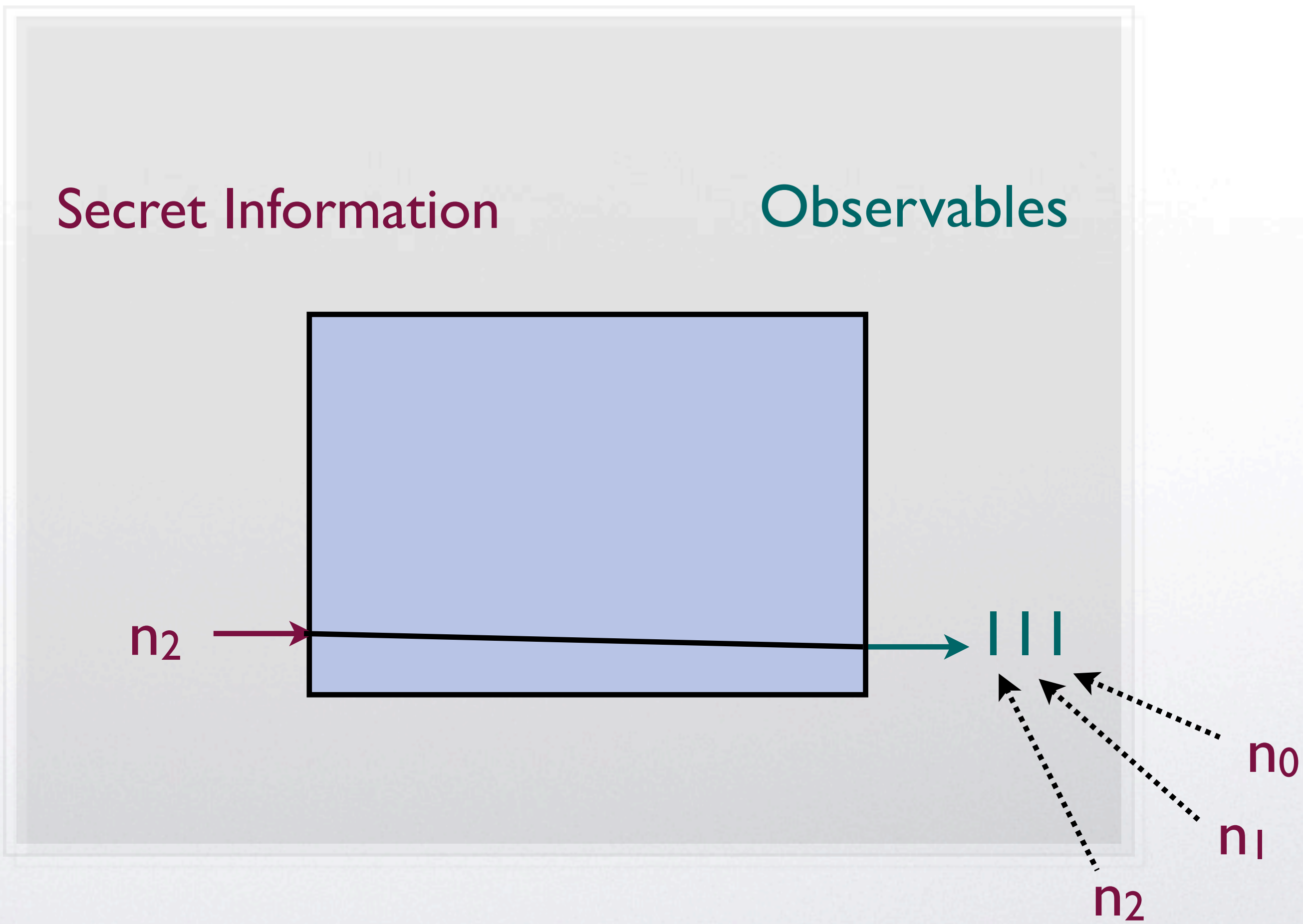
Example: DC nets (ring of 3 nodes, b=1)



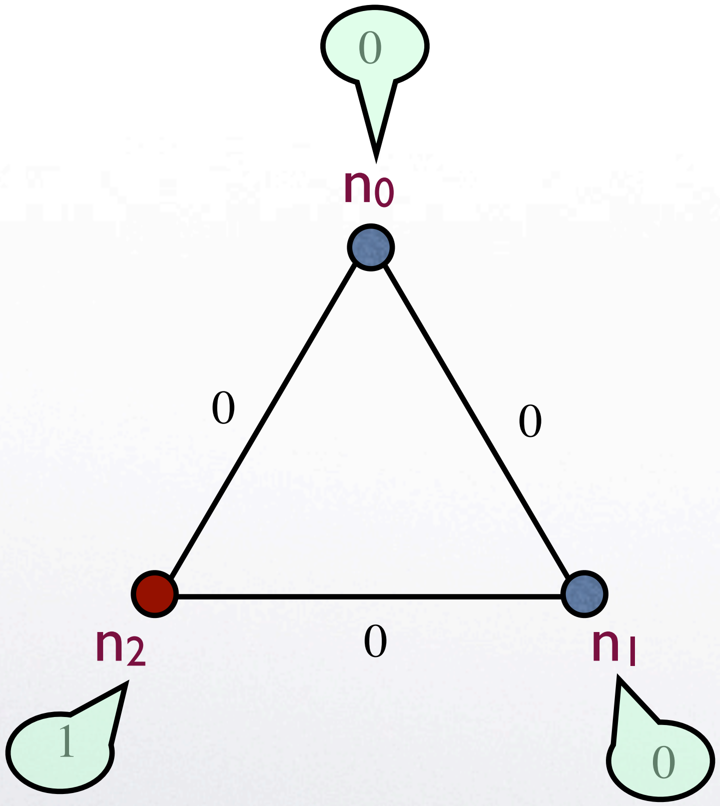


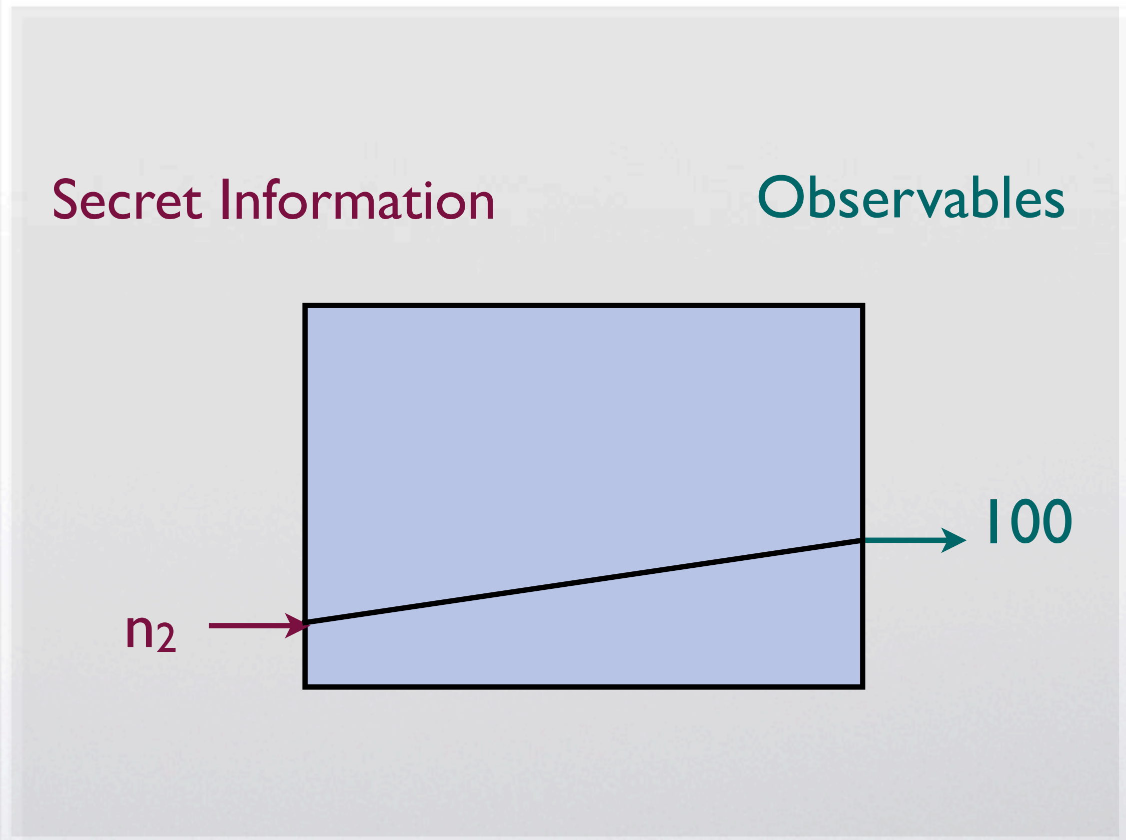
Example: DC nets (ring of 3 nodes, b=1)



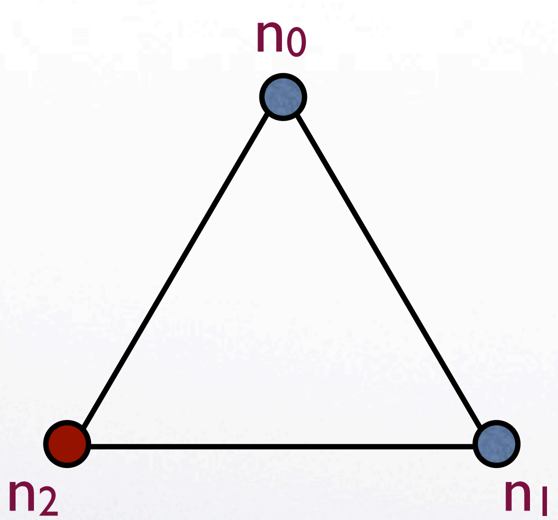


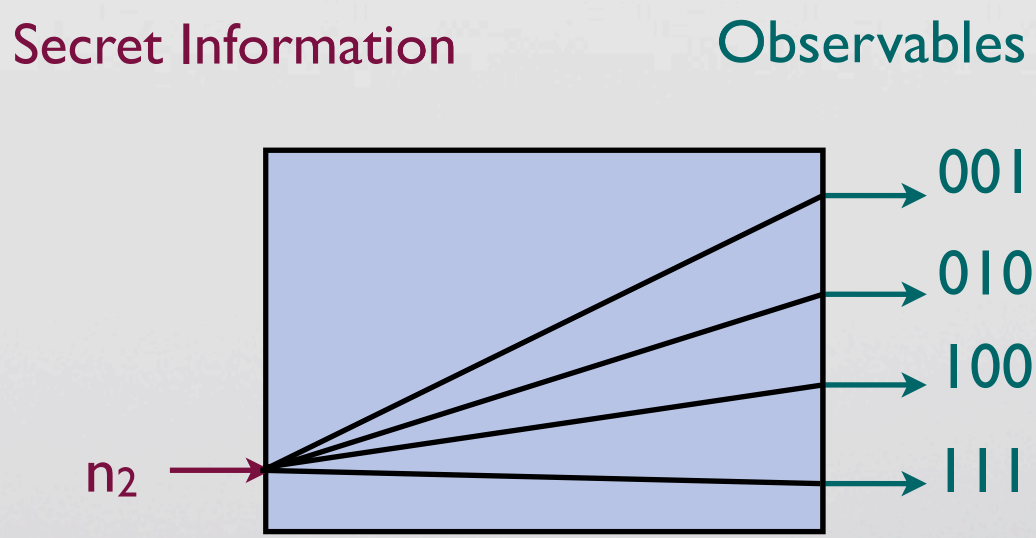
Example: DC nets (ring of 3 nodes, b=1)



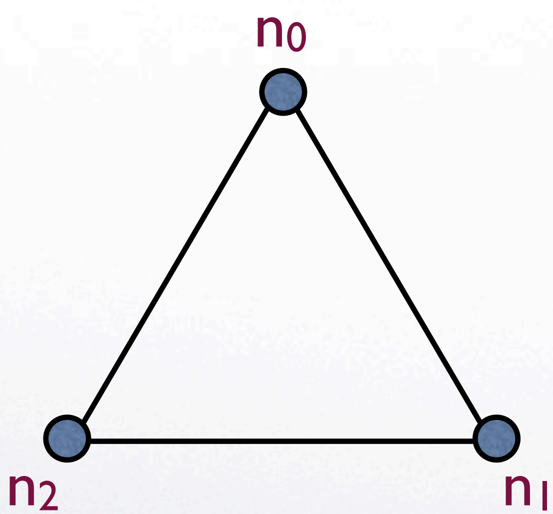


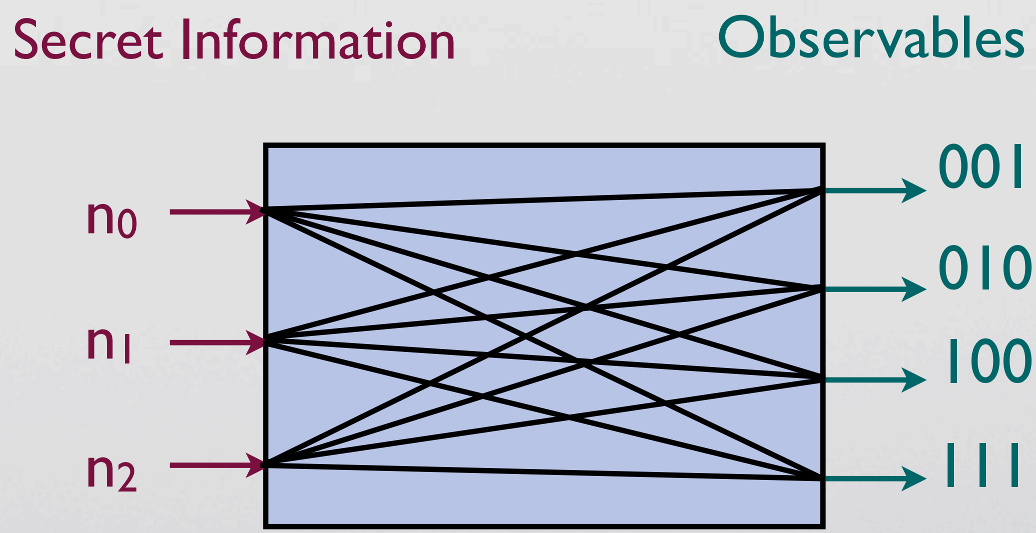
Example: DC nets (ring of 3 nodes, b=1)





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





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

	001	010	100	111
n_0	¼	¼	¼	¼
n_1	¼	¼	¼	¼
n_2	¼	¼	¼	¼

	001	010	100	111
n_0	⅓	$^2/_9$	$^2/_9$	$^2/_9$
n_1	$^2/_9$	⅓	$^2/_9$	$^2/_9$
n_2	$^2/_9$	$^2/_9$	⅓	$^2/_9$

fair coins: $\Pr(0) = \Pr(1) = ½$

strong anonymity

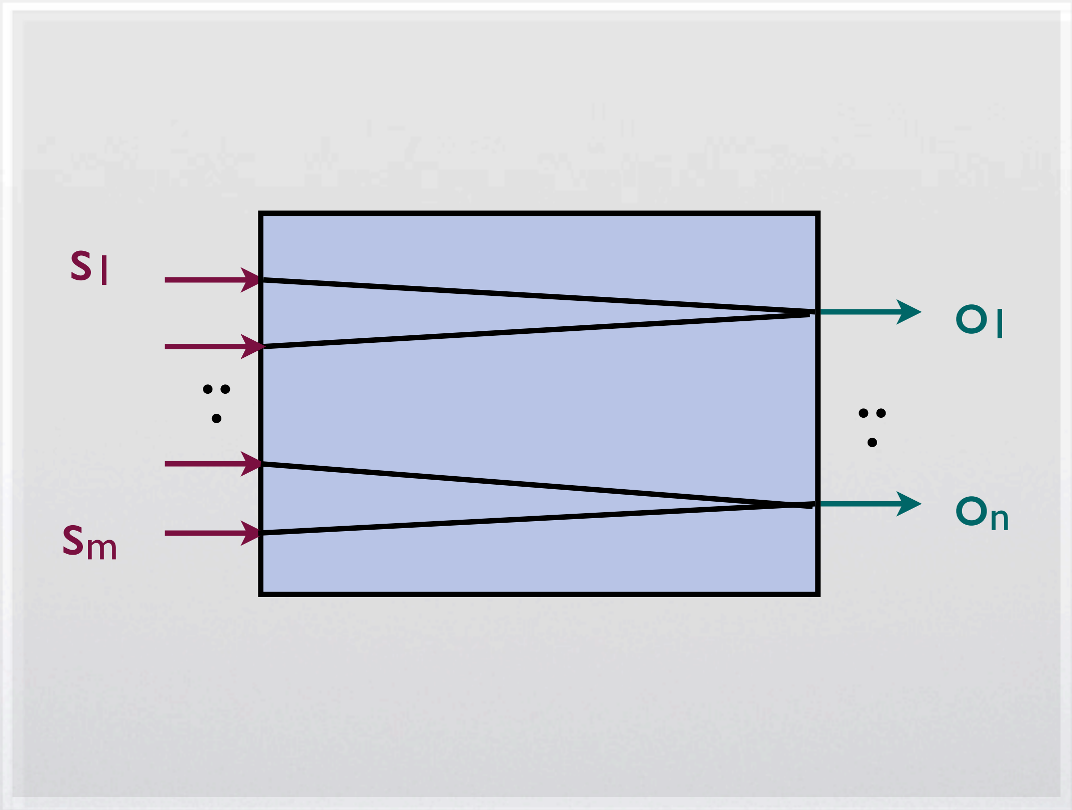
biased coins: $\Pr(0) = ⅔$, $\Pr(1) = ⅓$

The culprit is more likely to declare 1 than 0

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 conditional probabilities can be only 0 or 1

Exercises

- Compute the channel matrix for the two password-checker programs
- Compute the channel matrix for the DC nets with 3 nodes when the observer is one of the nodes