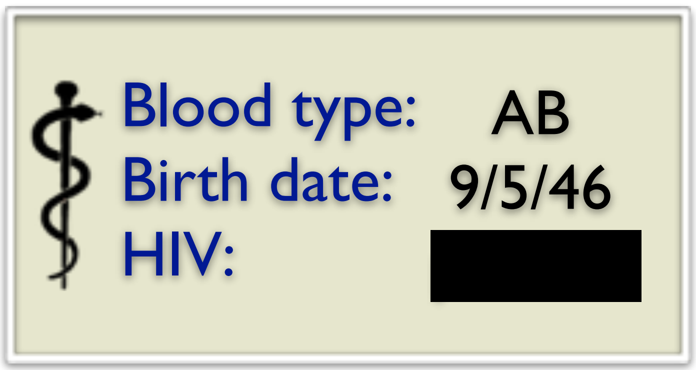
Quantitative Information Flow

Lecture 6

Protection of sensitive information

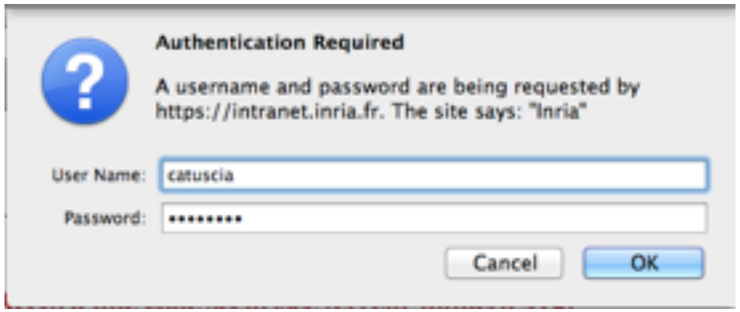
- This part of the course is dedicated to the problem of protecting secret information when it is collected, stored, processed and communicated by computer systems. This is the central issue in Security



- Typical counter-measures are encryption and access-control. However, they are not always sufficient! Systems could leak secret information through correlated observables.
  - The notion of "observable" depends on the adversary
  - Often, secret-leaking observables are public, and therefore available to any adversary

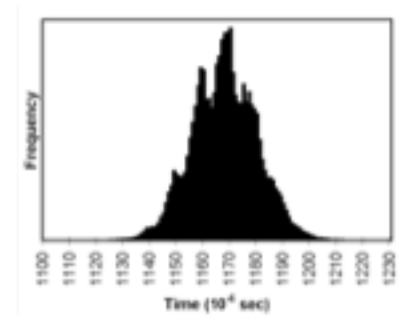
Leakage through correlated observables

Password checking



Election tabulation

Timings of decryptions



Plan of the lecture

1. Information leakage: motivation for quantitative approaches.
2. The information-theoretic approach to quantify the leakage of information:
3. Channel matrix
4. Prior and posterior probablity

Quantitative Information Flow

Information Flow: Leakage of secret information via correlated observables

Ideally: No leak

- No interference [Goguen & Meseguer'82]

In practice: There is almost always some leak

- Intrinsic to the system (public observables, part of the design)
- Side channels

➪ **need quantitative ways to measure the leak**

Example 1

Password checker 1

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

Intrinsic leakage

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

out := OK
for $i = 1, \ldots, N$ **do**
  if $x_i \neq K_i$ **then**
    out := FAIL

  end if
end for

Example 1

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, ..., N do
  if x_i ≠ K_i then
  { out := FAIL
    exit() }
  end if
end for
```

More efficient, but what about security?

Example 1

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, ..., N do
  if x_i ≠ K_i then
  { out := FAIL
    exit()      }
  end if
end for
```

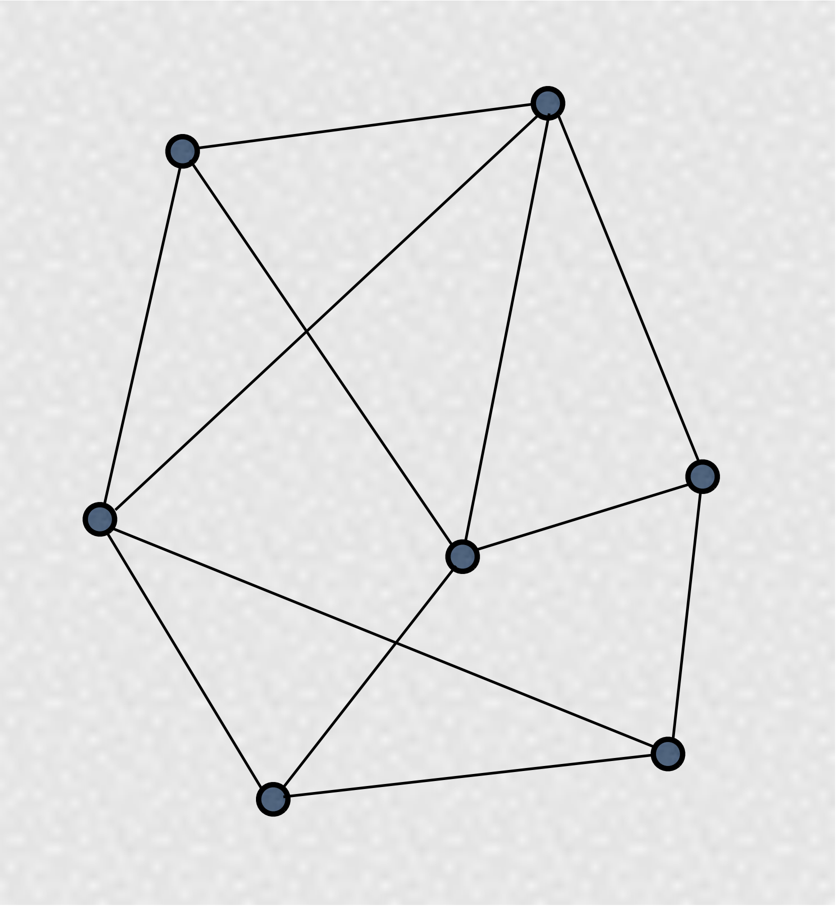
Side channel attack

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

Example 2

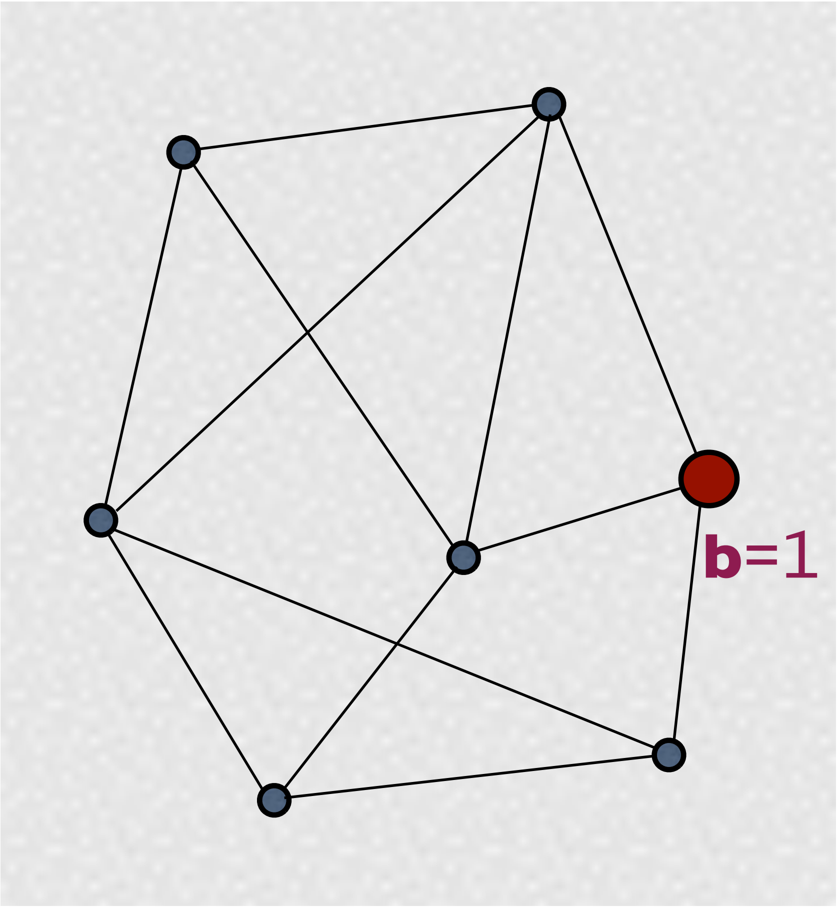
Example of Anonymity Protocol: DC Nets [Chaum'88]

- 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



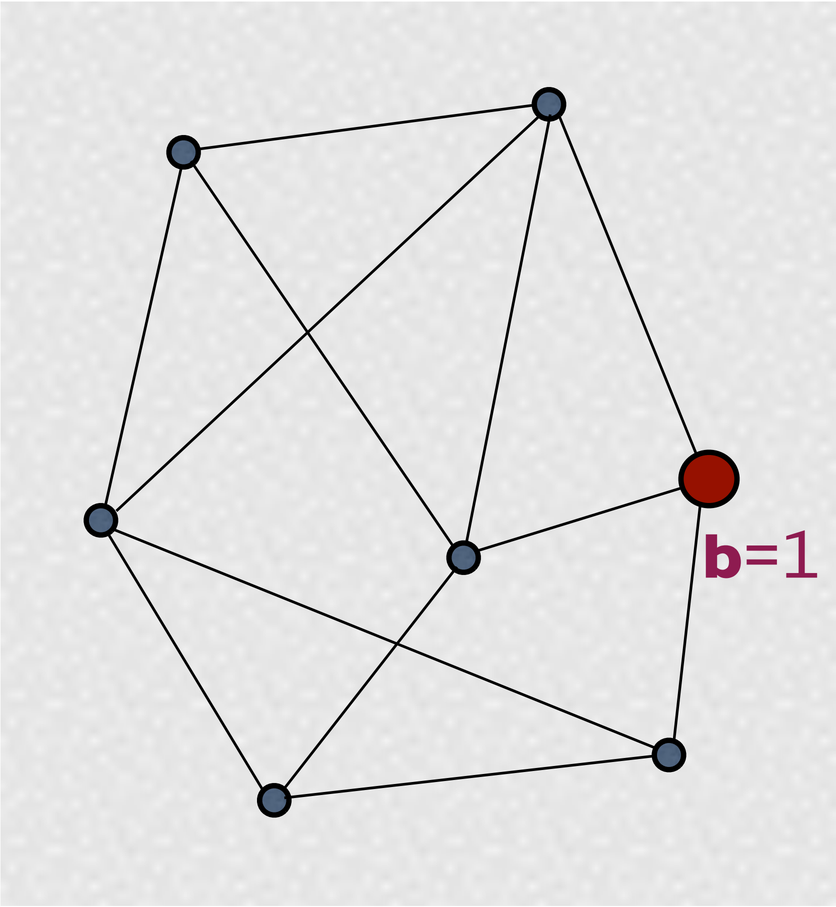
Example of Anonymity Protocol: DC Nets [Chaum'88]

- 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



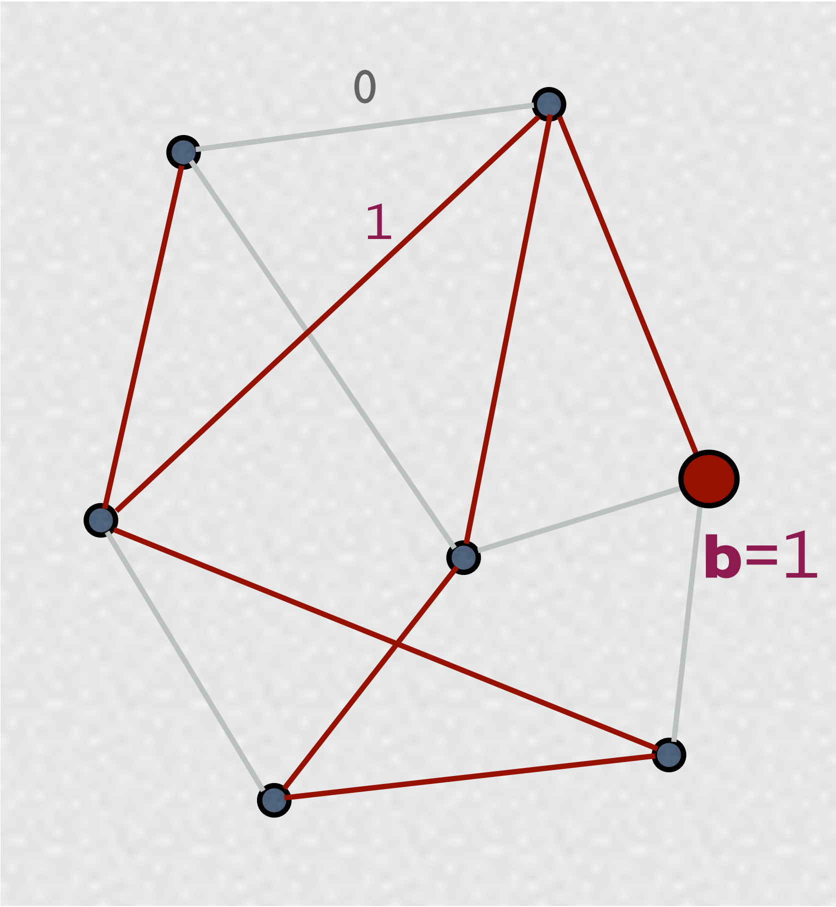
Chaum’s solution

- Associate to each edge a fair binary coin



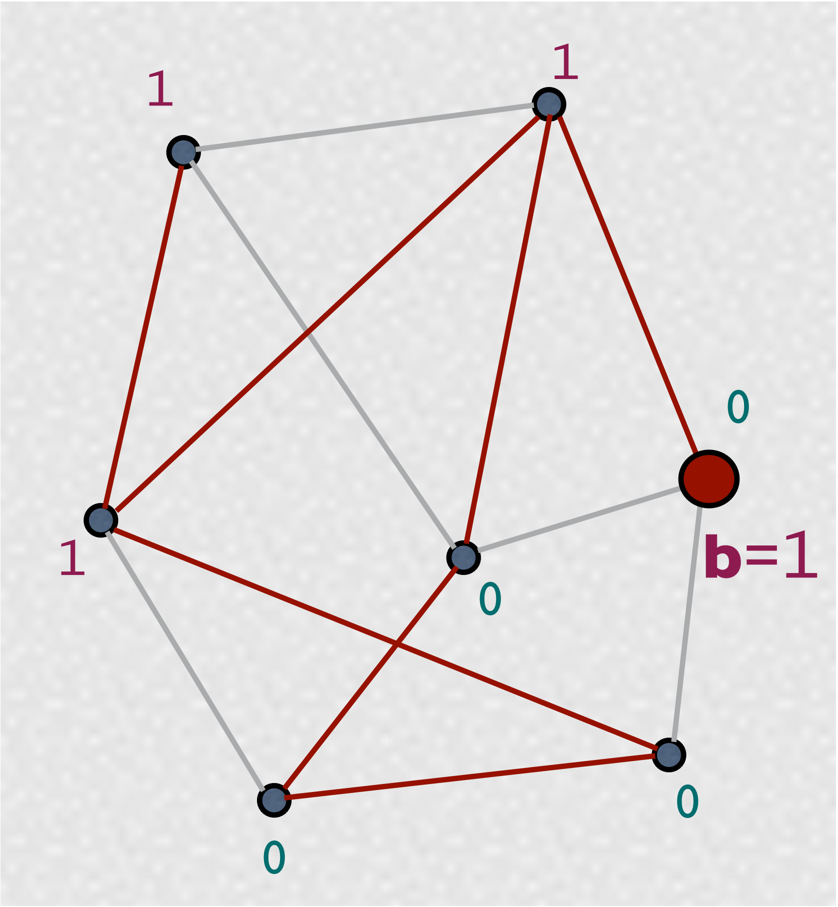
Chaum's solution

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



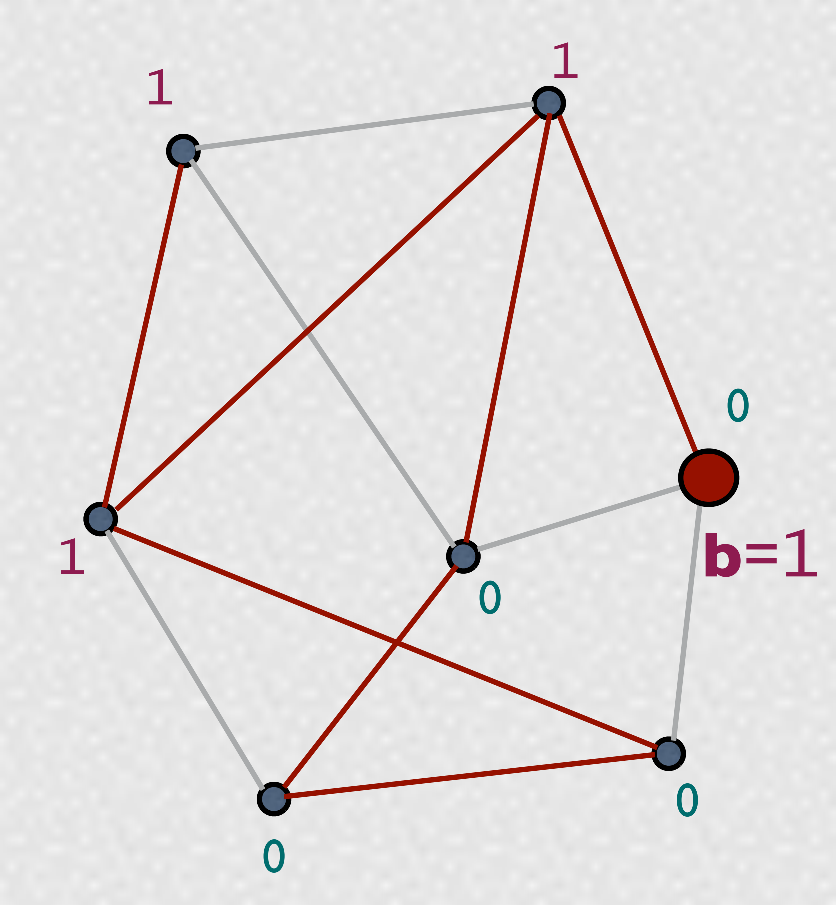
Chaum's solution

- 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



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**



Anonymity of DC Nets

Observables: An (external) attacker can only see the declarations of the nodes

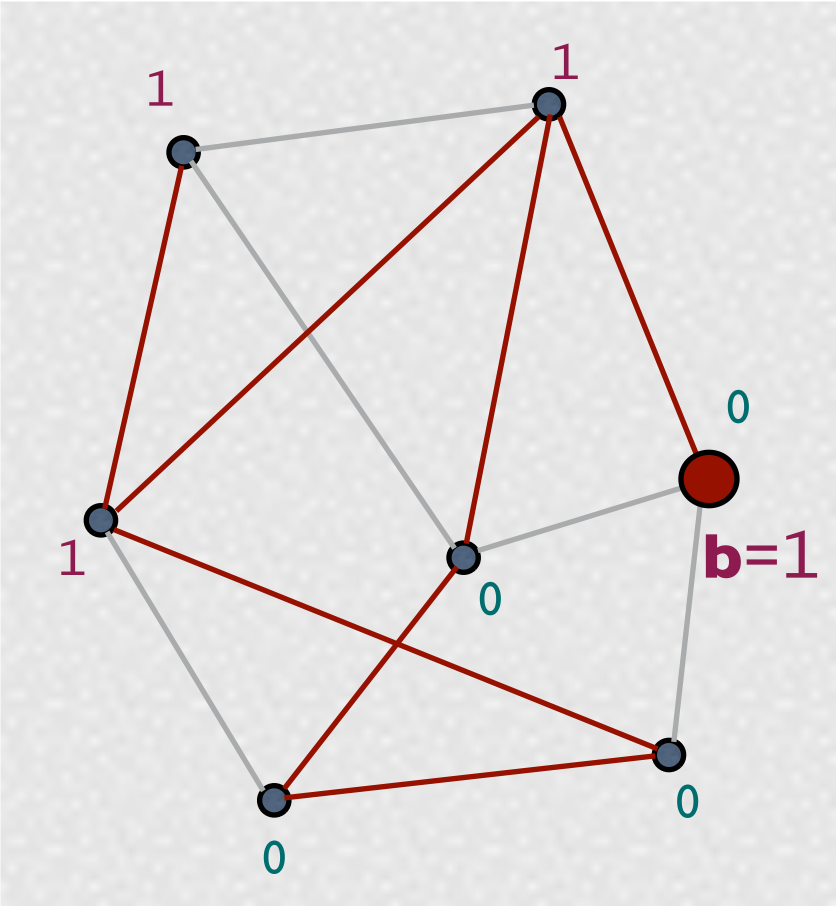
Question: Does the protocol protects the anonymity of the source?

Strong anonymity (Chaum)

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

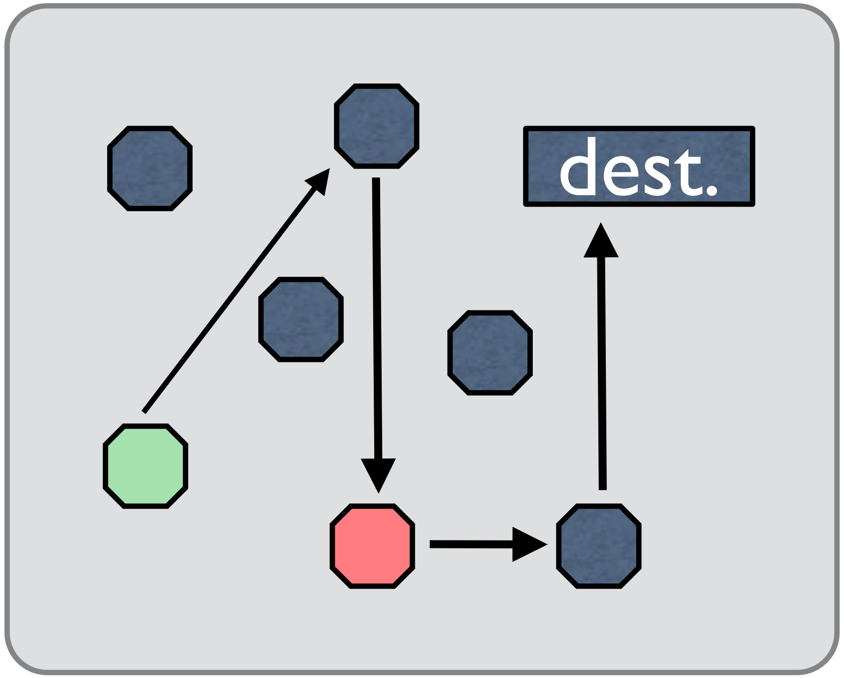
> the *a posteriori* probability that a certain node is the source is equal to its *a priori* probability

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



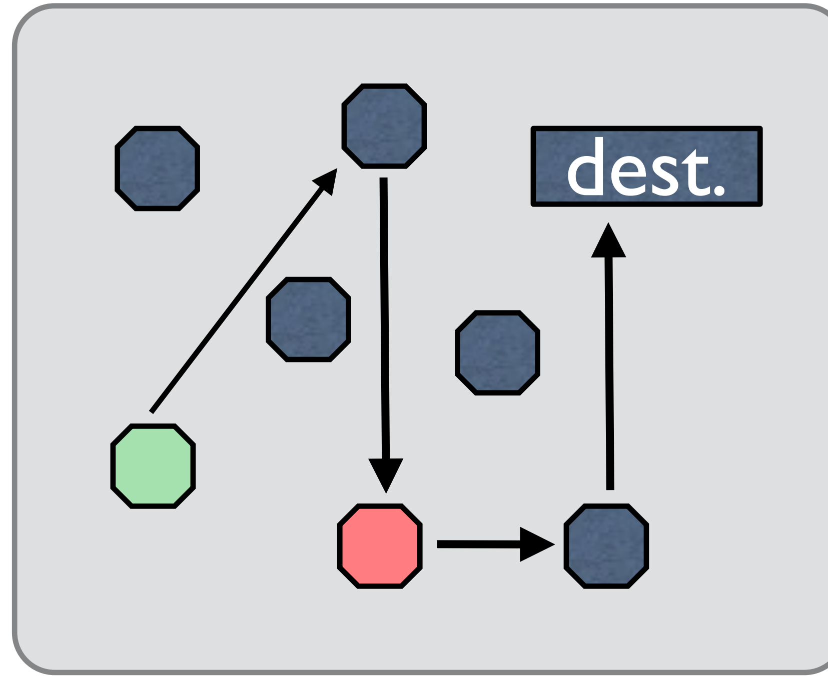
Example 3: Crowds [Rubin and Reiter'98]

- 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



Example 3: 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
  - Password checker: The password
  - DC: the identity of the source
  - Crowds: the identity of the initiator
- Public information (Observables)
  - Password checker: The result (OK / Fail) and the execution time
  - DC: the declarations of the nodes
  - Crowds: the identity of the agent forwarding to 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 course we assume:

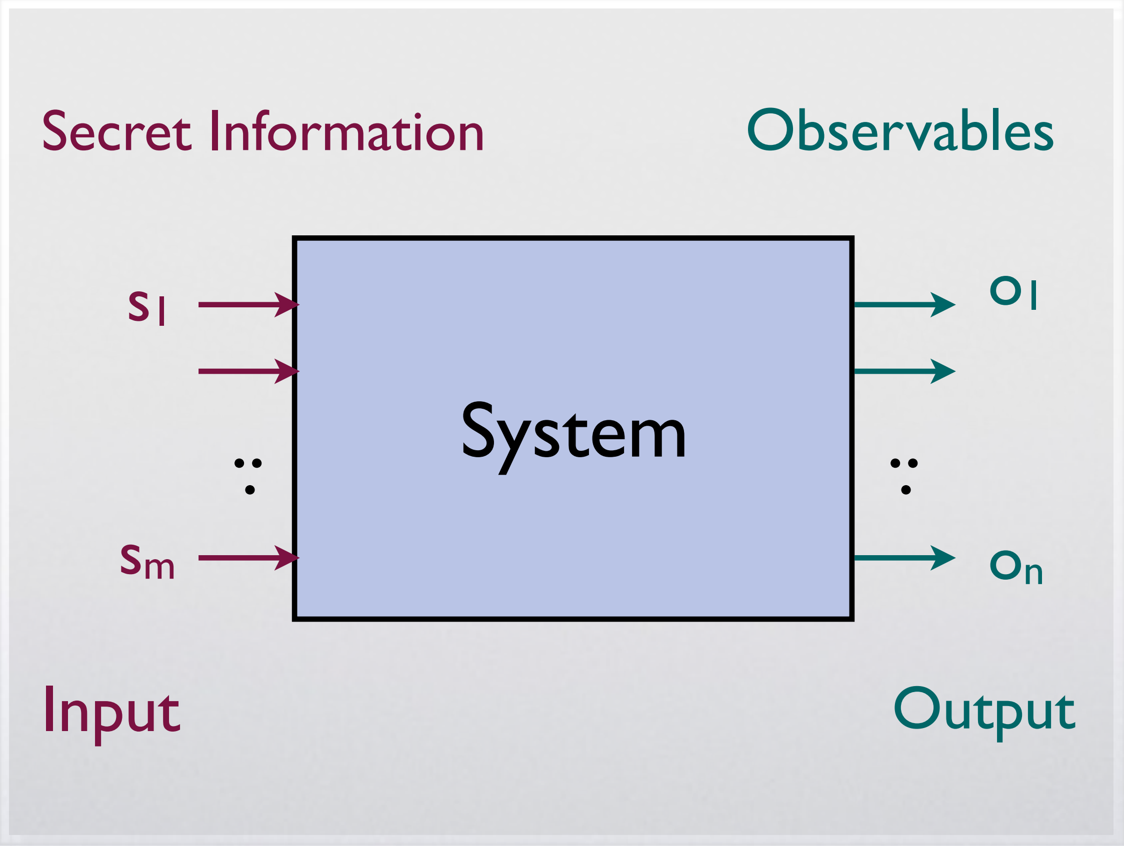
- Secrets: elements of a random variable S
- Observables: elements of a random variable O
- For each secret s, the probability that we obtain an observable o is given by p(o | s)

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

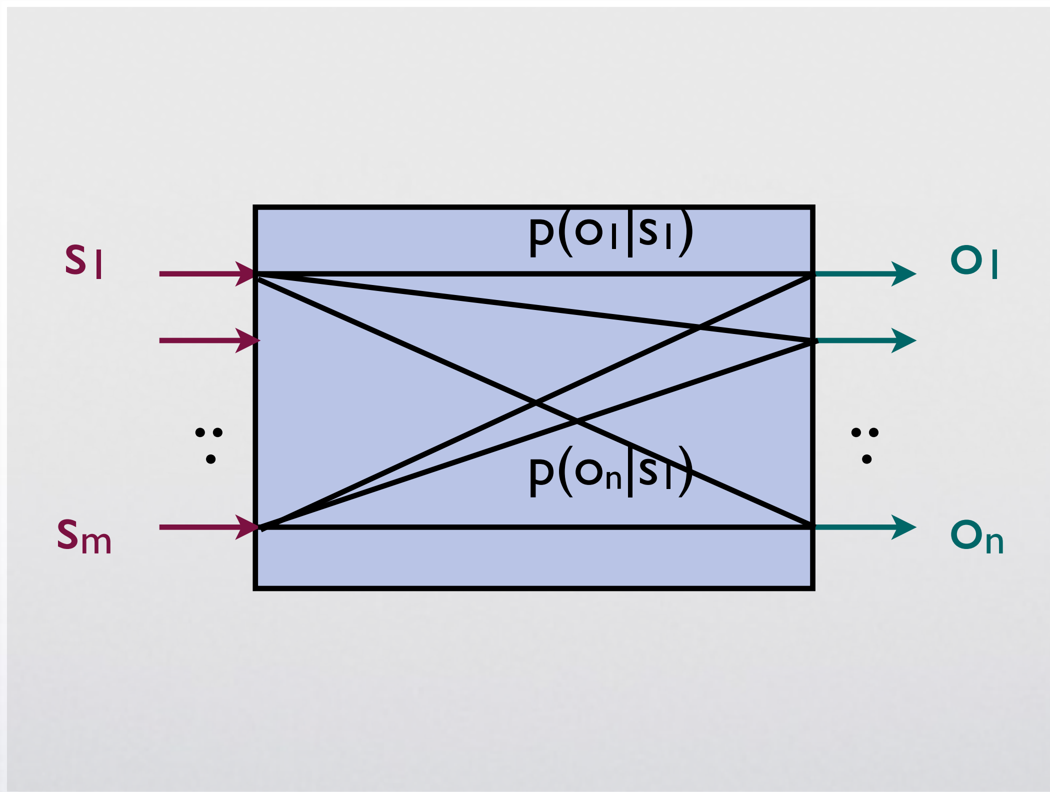
- No nondeterminism: everything is (either deterministic or) probabilistic, although we may not know the distribution

The basic model:

Systems = 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

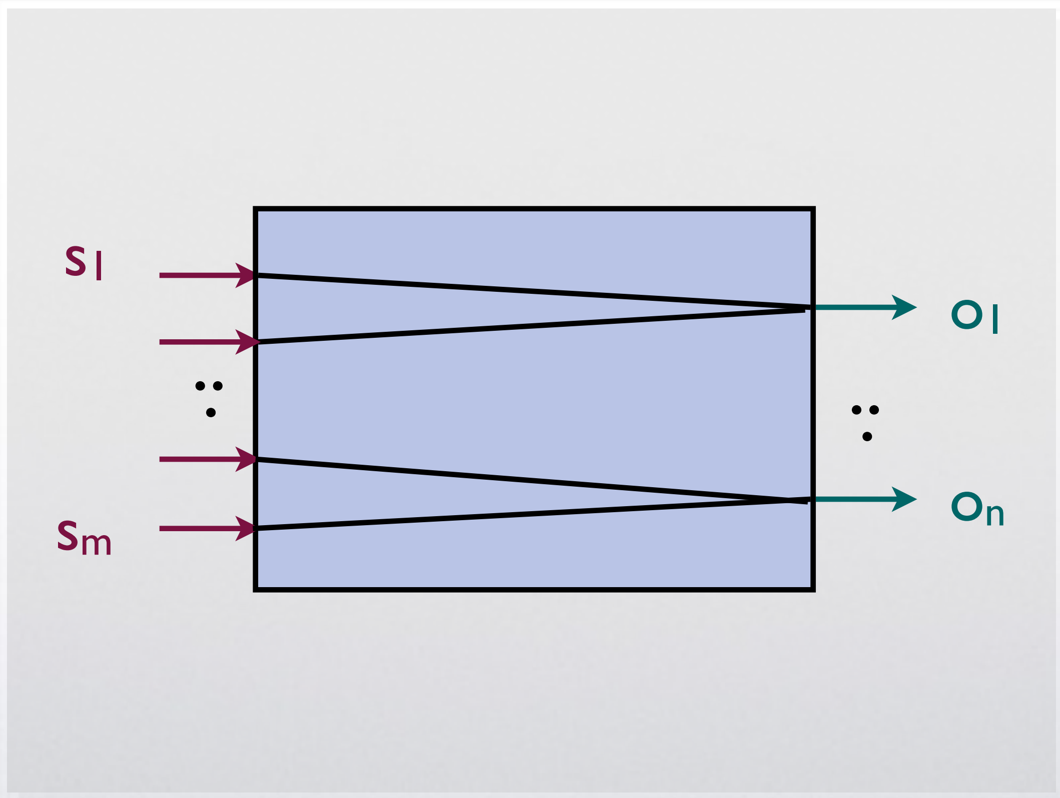
In a information-theoretic channel these conditional probabilities are independent from the input distribution

This means that we can model systems abstracting from the input distribution

Particular case: **Deterministic systems**

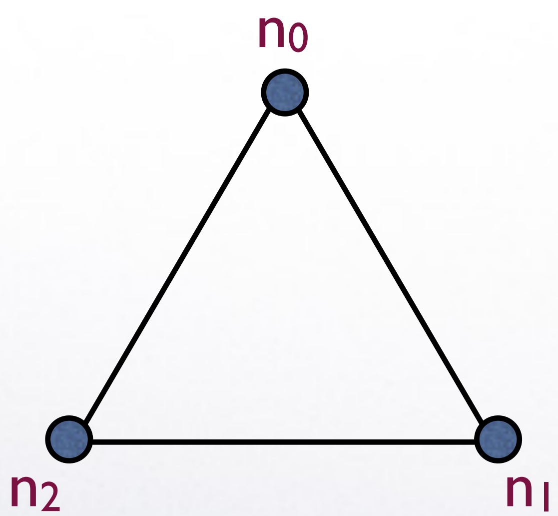
In these systems an input generates only one output

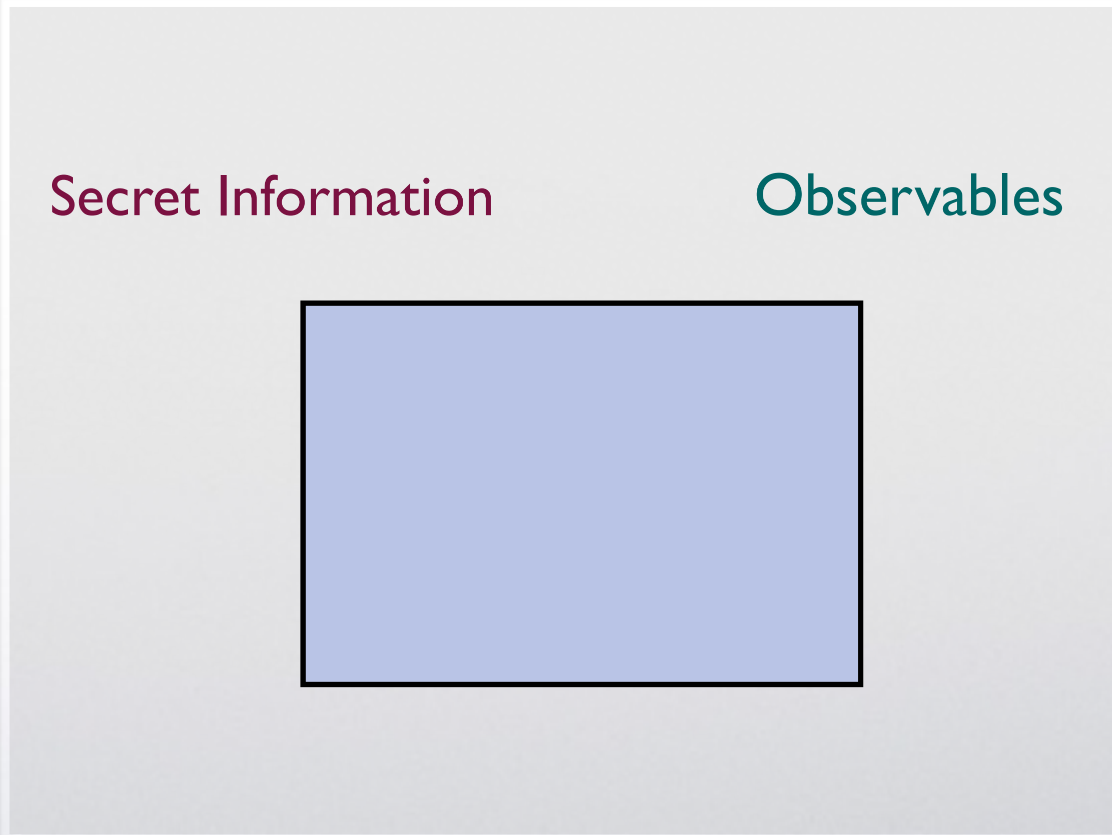
Still interesting: the problem is how to retrieve the input from the output



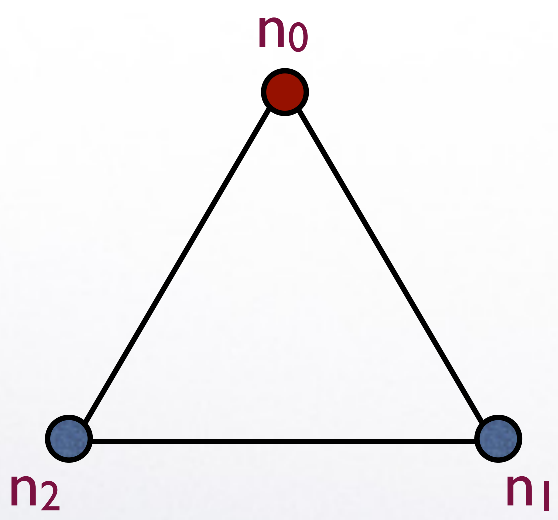
The entries of the channel matrix can be only 0 or 1

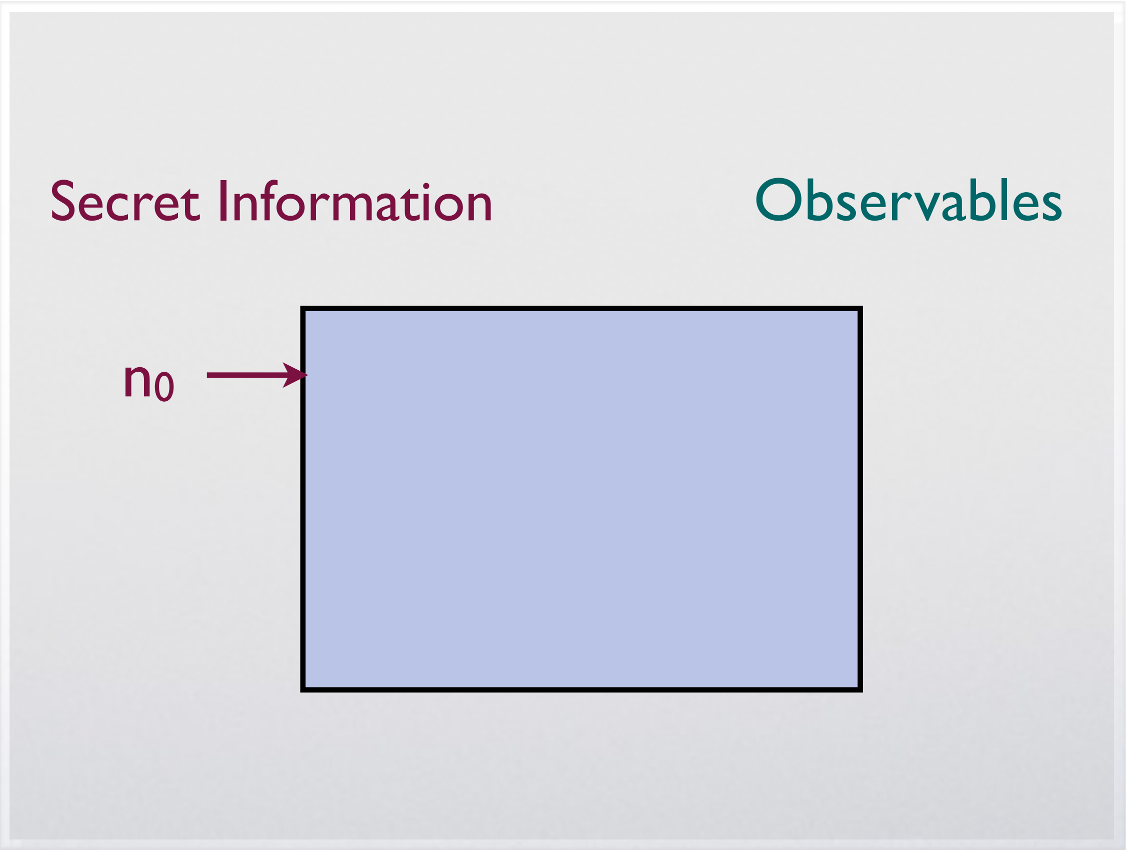
Example: DC nets (ring of 3 nodes, b=1)



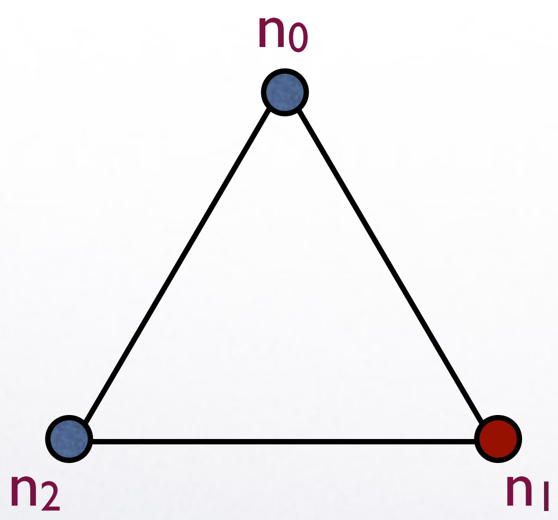


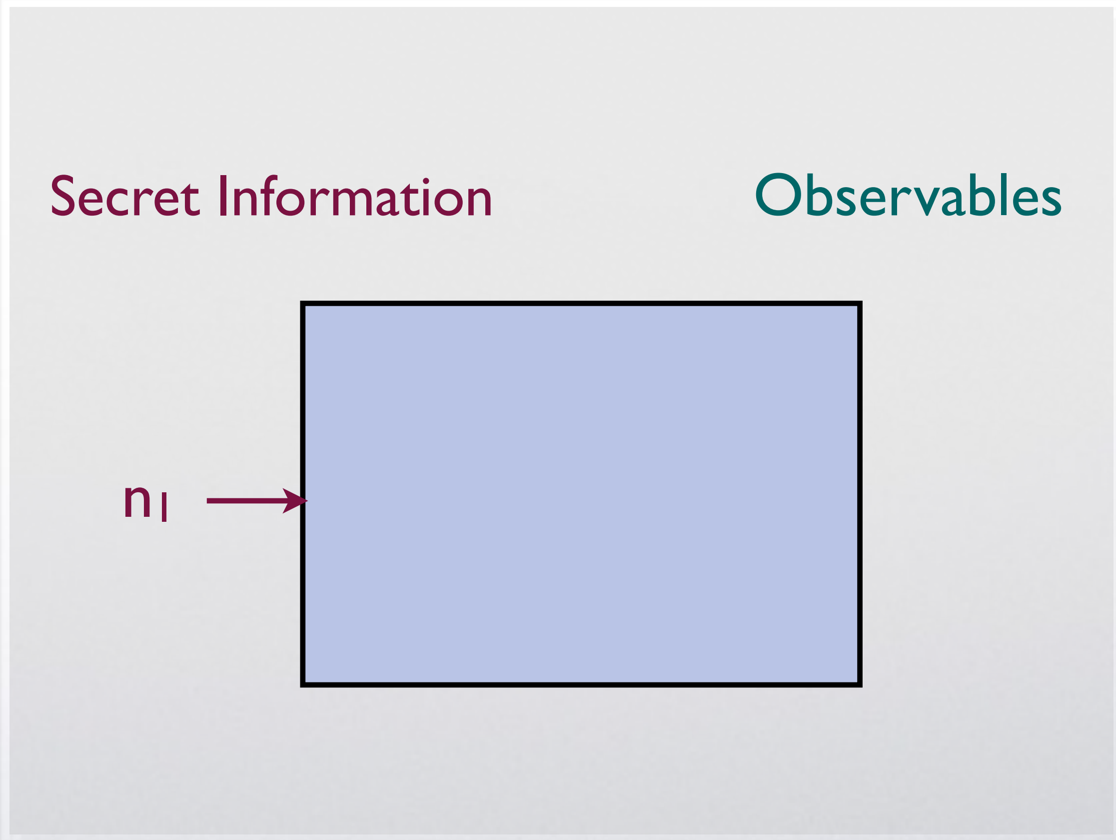
Example: DC nets (ring of 3 nodes, b=1)



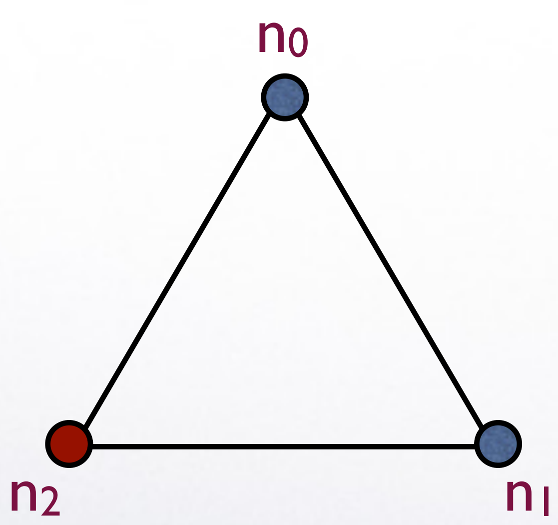


Example: DC nets (ring of 3 nodes, b=1)



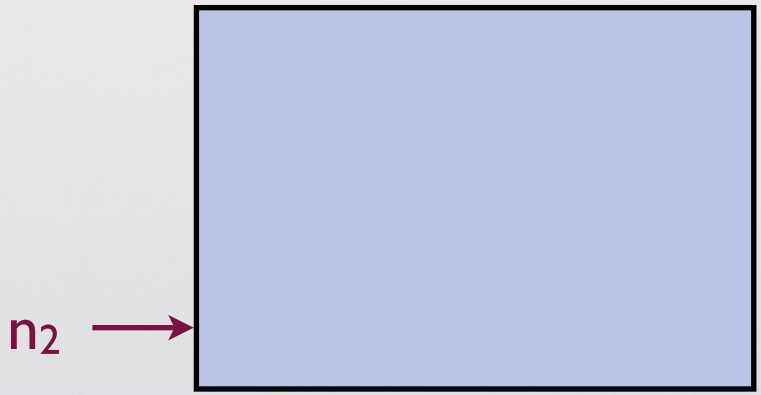


Example: DC nets (ring of 3 nodes, b=1)

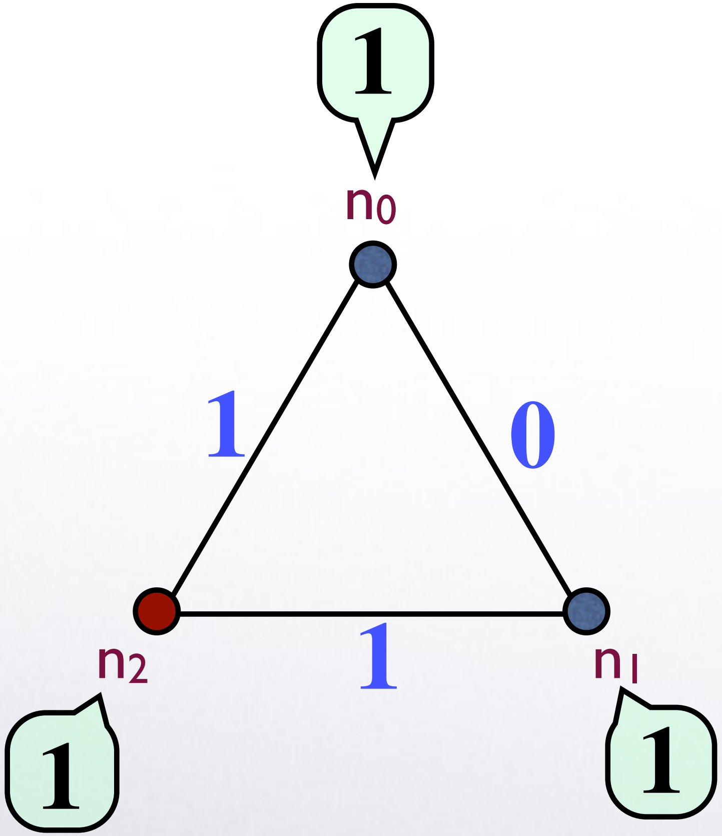


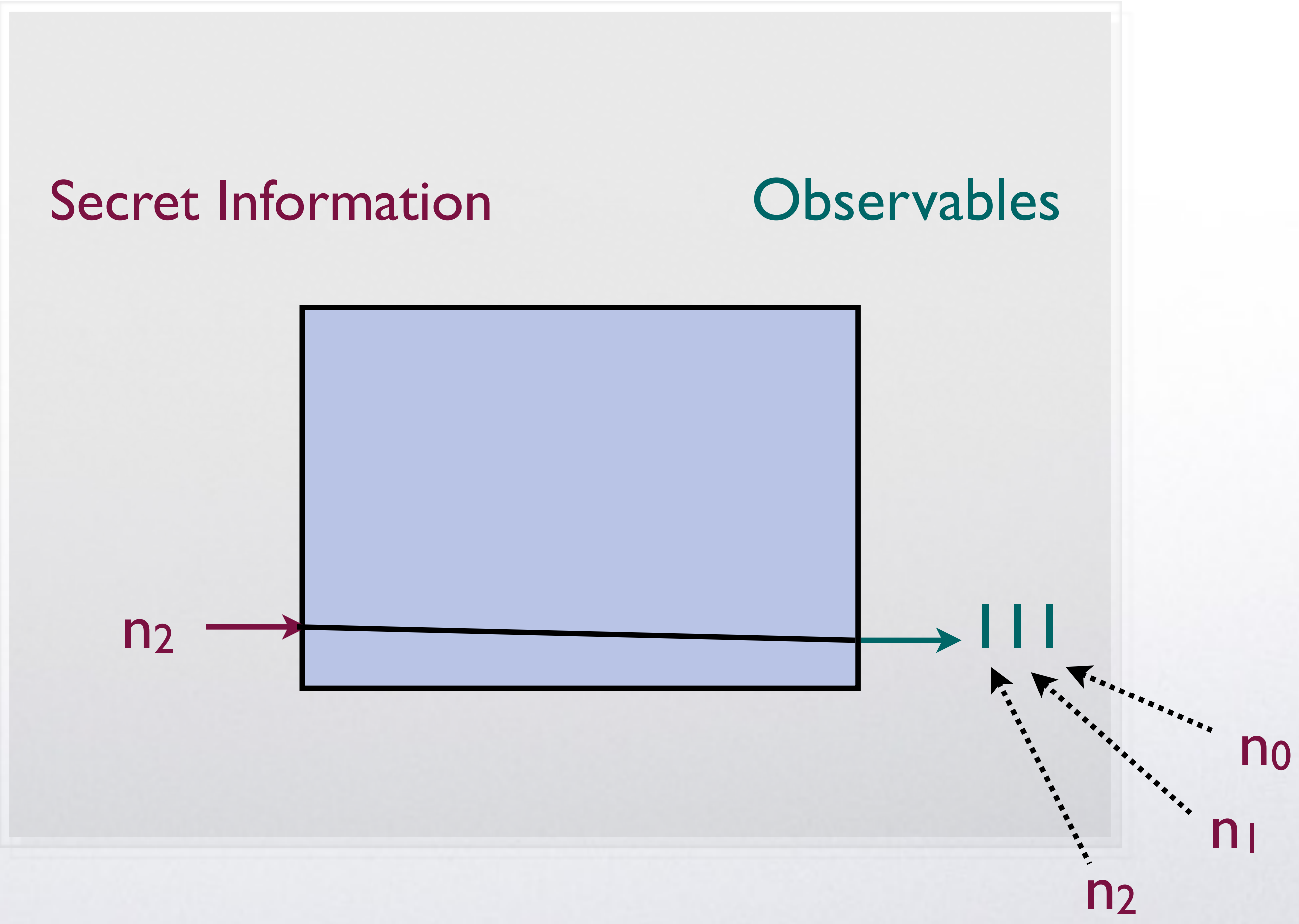
Secret Information

Observables

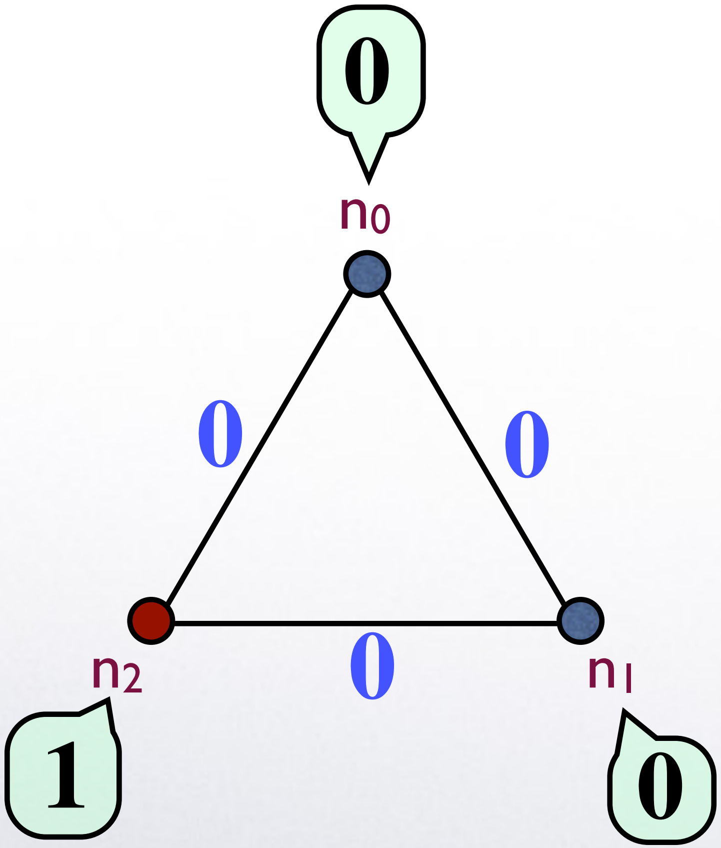


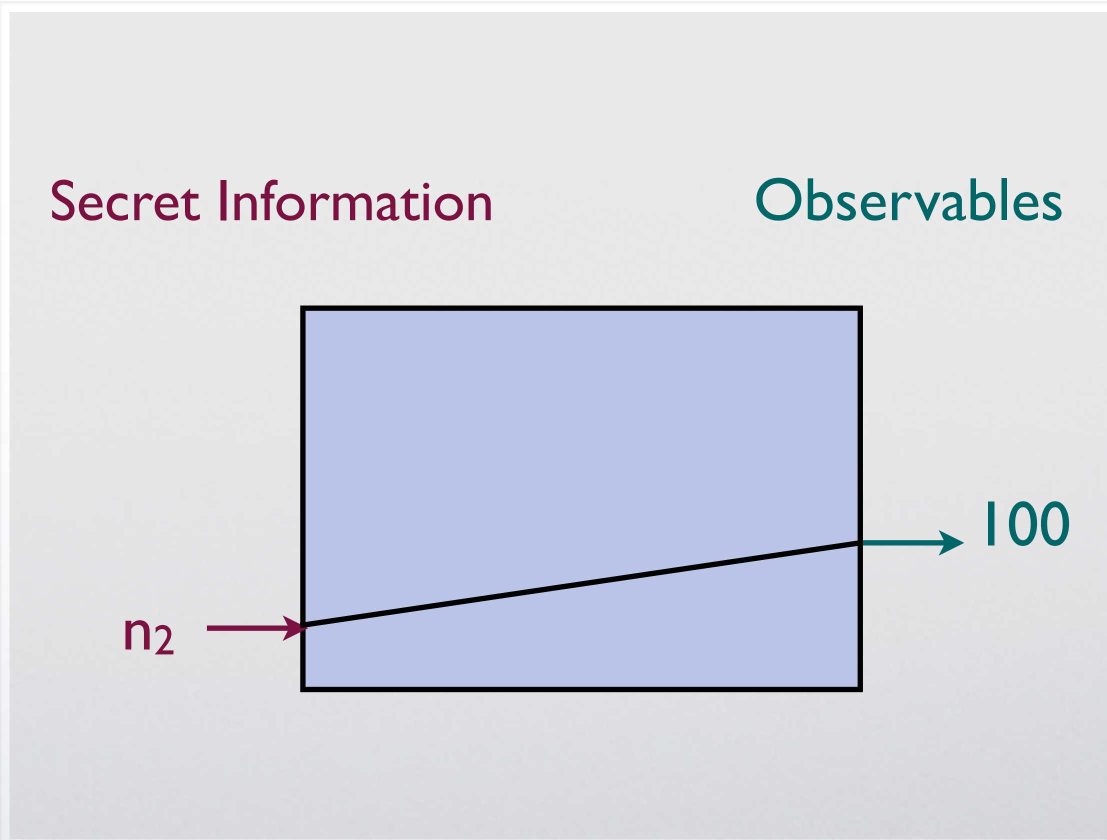
Example: DC nets (ring of 3 nodes, b=1)



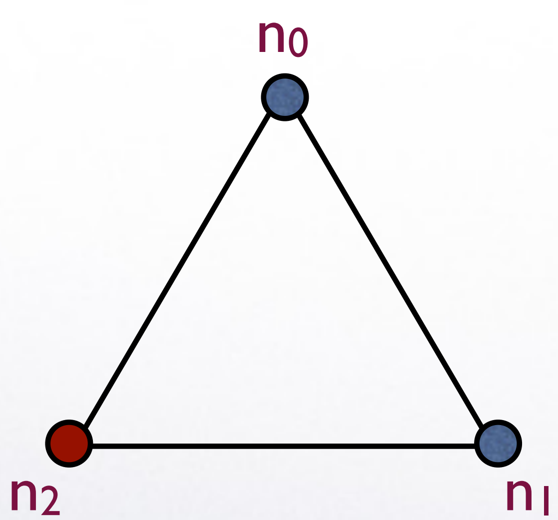


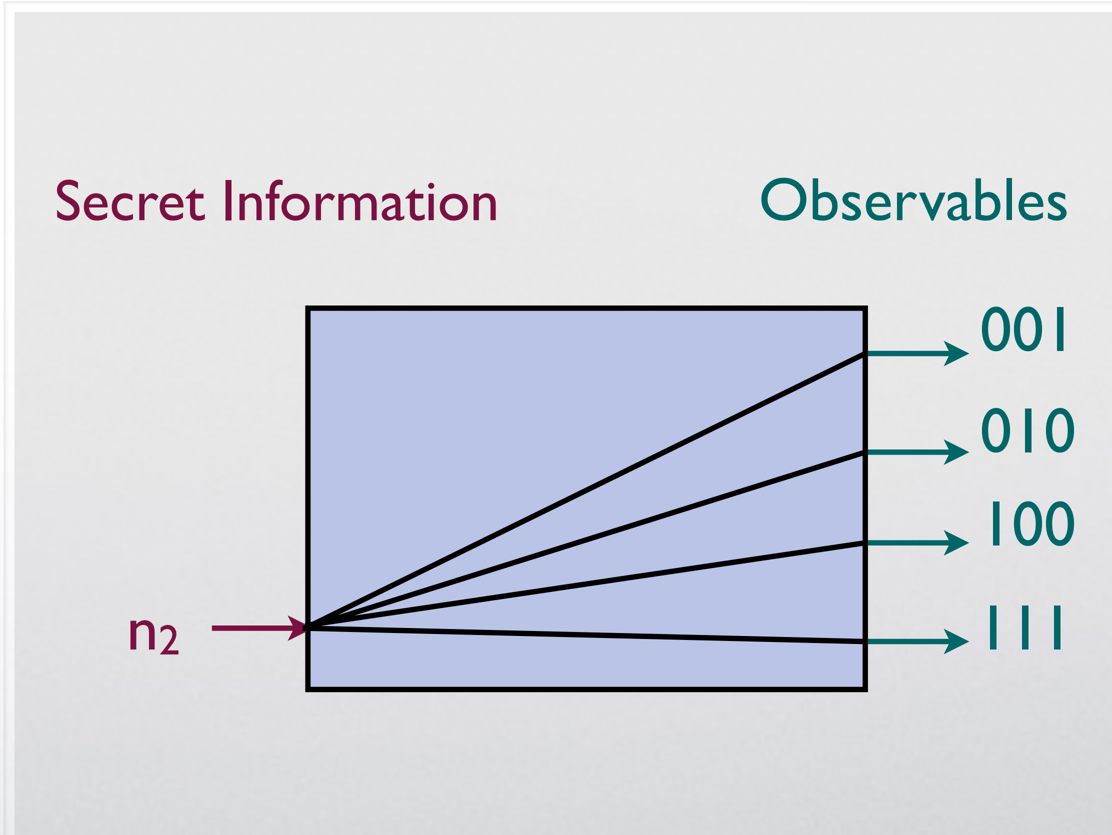
Example: DC nets (ring of 3 nodes, b=1)



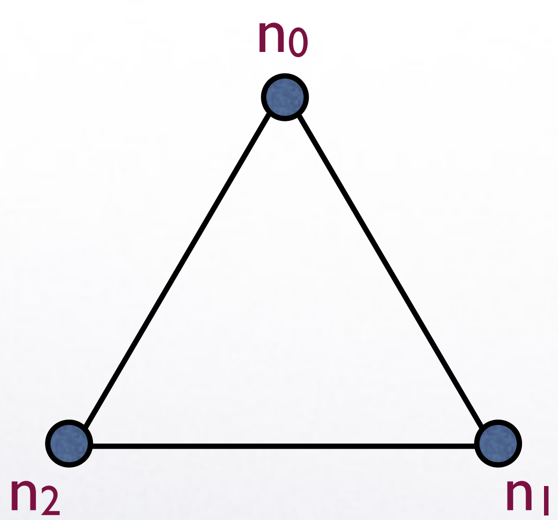


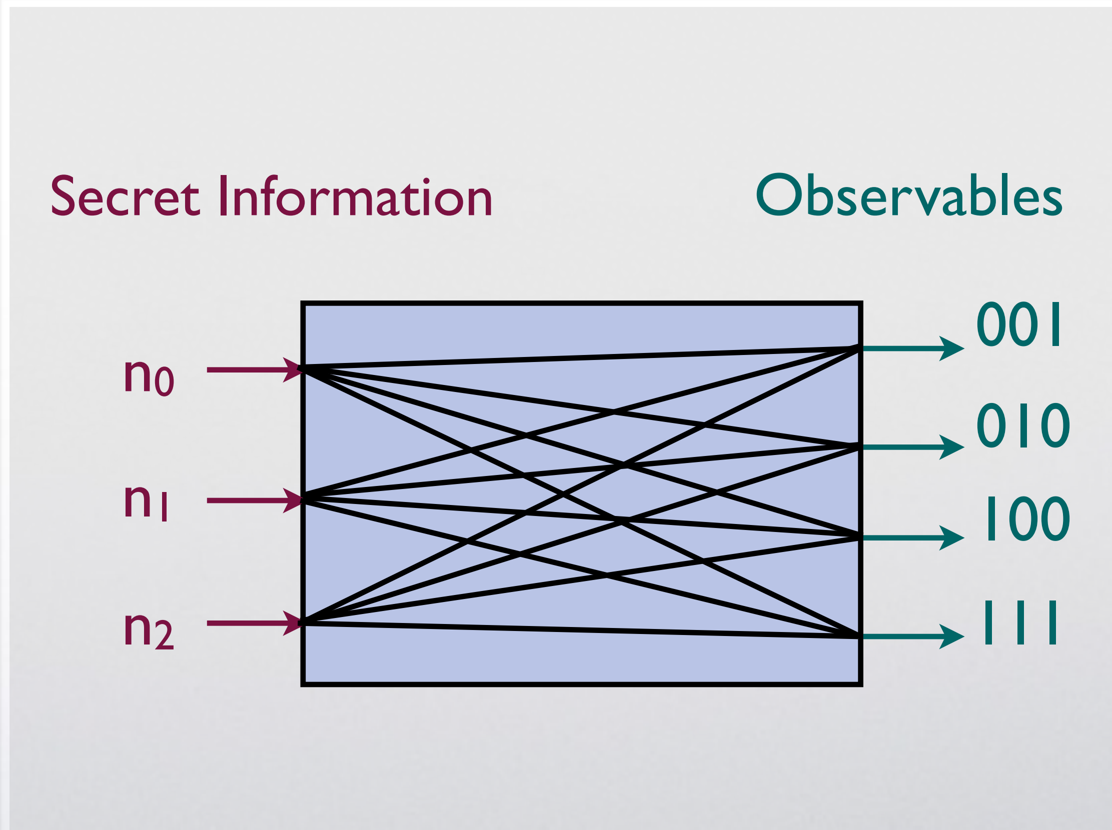
Example: DC nets (ring of 3 nodes, b=1)





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





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

	001	010	100	111
n_0	1/4	1/4	1/4	1/4
n_1	1/4	1/4	1/4	1/4
n_2	1/4	1/4	1/4	1/4

	001	010	100	111
n_0	1/3	2/9	2/9	2/9
n_1	2/9	1/3	2/9	2/9
n_2	2/9	2/9	1/3	2/9

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

Quantitative Information Flow

- Intuitively, the **leakage** is the (probabilistic) information that the adversary **gains** about the **secret** through the **observables**

- Each observable **changes** the **prior** probability distribution on the secret values into a **posterior** probability distribution according to the **Bayes** theorem

- In the average, the posterior probability distribution gives a **better hint** about the actual secret value

Observables: prior ⇒ posterior

Observables: prior ⇒ posterior

p(n)		001	010	100	111
½	n_0	⅓	$^2/_9$	$^2/_9$	$^2/_9$
¼	n_1	$^2/_9$	⅓	$^2/_9$	$^2/_9$
¼	n_2	$^2/_9$	$^2/_9$	⅓	$^2/_9$

prior secret prob

p(o|n)
conditional prob

Observables: prior ⇒ posterior

p(n)
½
¼
¼
prior secret prob

	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

p(o|n)
conditional prob

	001	010	100	111
n_0	⅙	1/9	1/9	1/9
n_1	1/18	1/12	1/18	1/18
n_2	1/18	1/18	1/12	1/18

p(n,o)
joint prob

Observables: prior ⇒ posterior

p(n)		001	010	100	111
½	n_0	⅓	2⁄9	2⁄9	2⁄9
¼	n_1	2⁄9	⅓	2⁄9	2⁄9
¼	n_2	2⁄9	2⁄9	⅓	2⁄9

prior secret prob

p(o|n)
conditional prob

	001	010	100	111	
p(o)	5⁄18	¼	¼	2⁄9	obs prob
n_0	⅙	1⁄9	1⁄9	1⁄9	
n_1	1⁄18	1⁄12	1⁄18	1⁄18	
n_2	1⁄18	1⁄18	1⁄12	1⁄18	

p(n,o)
joint prob

$$p(n|o) = \frac{p(n,o)}{p(o)}$$

Bayes theorem

obs prob

p(o)	5/18	1/4	1/4	2/9
	001	010	100	111

p(n,o)
joint prob

	001	010	100	111
n_0	1/6	1/9	1/9	1/9
n_1	1/18	1/12	1/18	1/18
n_2	1/18	1/18	1/12	1/18

p(o|n)
conditional prob

	001	010	100	111
n_0	1/3	2/9	2/9	2/9
n_1	2/9	1/3	2/9	2/9
n_2	2/9	2/9	1/3	2/9

post secret prob

	p(n|001)
n_0	3/5
n_1	1/5
n_2	1/5

Password-checker 1

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

   end if
end for
```

Let us construct the channel matrix

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

Password-checker 1

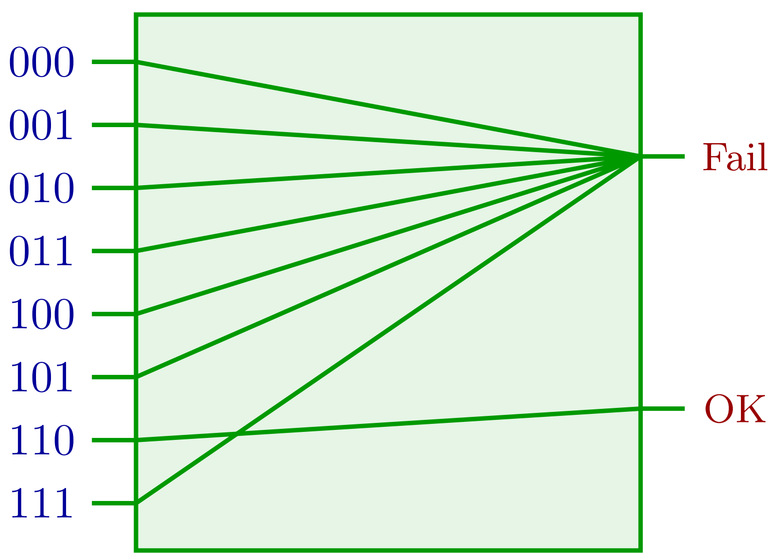
```
out := OK
for i = 1, ..., N do
    if x_i ≠ K_i then
        out := FAIL

    end if
end for
```

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: $out \in \{\mathsf{OK}, \mathsf{FAIL}\}$



	Fail	OK
000	1	0
001	1	0
010	1	0
011	1	0
100	1	0
101	1	0
110	0	1
111	1	0

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)

Password-checker 2

```
out := OK
for i = 1, ..., N do
   if x_i ≠ K_i then
      { out := FAIL
        exit() }
   end if
end for
```

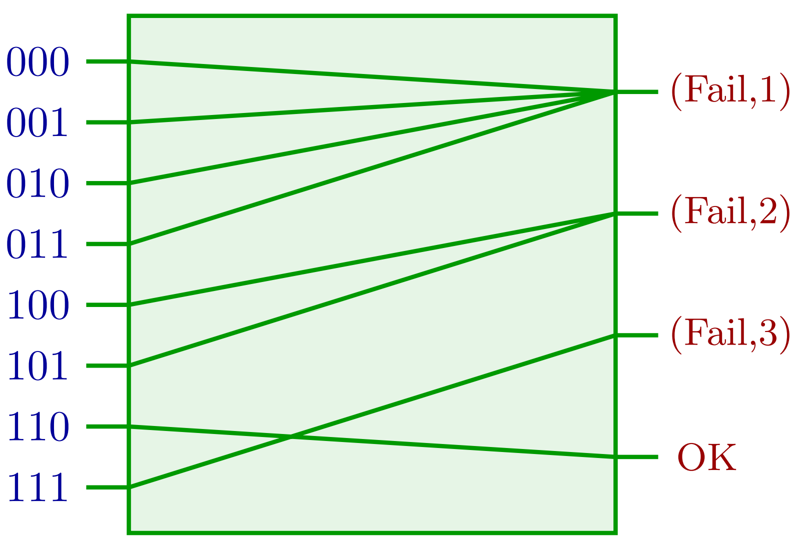
Assume the user string is $x_1x_2x_3 = 110$

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

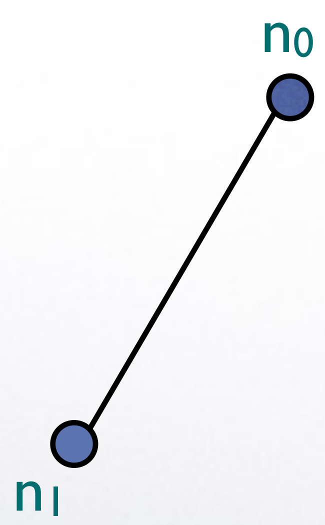


	(Fail, 1)	(Fail, 2)	(Fail, 3)	OK
000	1	0	0	0
001	1	0	0	0
010	1	0	0	0
011	1	0	0	0
100	0	1	0	0
101	0	1	0	0
110	0	0	0	1
111	0	0	1	0

Exercise 1

- Assuming that the possible passwords have uniform prior distribution, compute the matrix of the joint probabilities, and the posterior probabilities, for the two password-checker programs

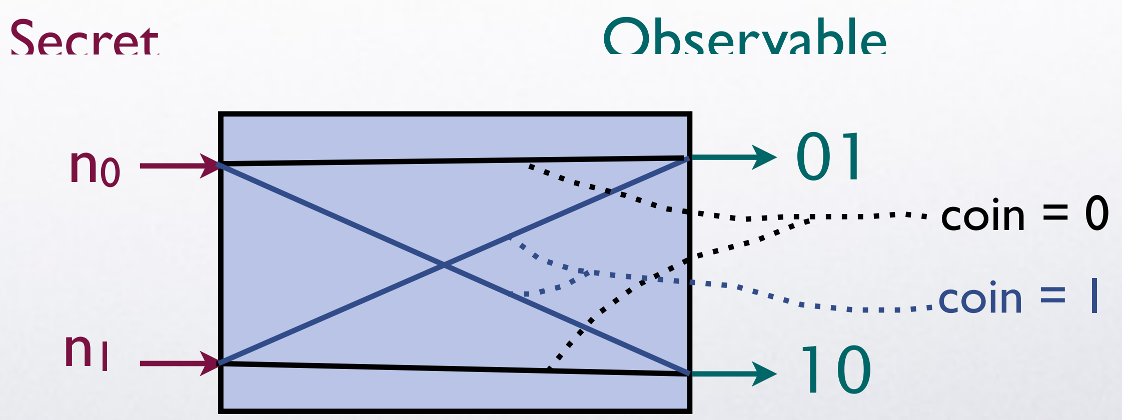
Example: DC nets. Ring of 2 nodes, and assume b = 1



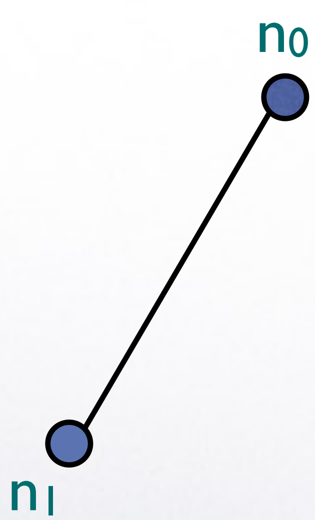
Let us construct the channel matrix

Input: n_0 , n_1

Output: the declarations of n_1 and n_0: $d_1 d_0 \in \{01, 10\}$



Example: DC nets. Ring of 2 nodes, and assume b = 1



Let us construct the channel matrix

Input: n_0 , n_1

Output: the declarations of n_1 and n_0: $d_1 d_0 \in \{01, 10\}$

Fair coin: p(0) = p(1) = ½

	01	10
n_0	½	½
n_1	½	½

Biased coin: p(0) = ⅔ p(1) = ⅓

	01	10
n_0	⅔	⅓
n_1	⅓	⅔

Exercise 2

- DC nets: Assuming that n_0 and n_1 have uniform prior distribution, compute the matrix of the joint probabilities, and the posterior probabilities, in the two cases of fair coins, and of biased coins

- Same exercise, but now assume that the prior distribution is 2/3 for n_0 and 1/3 for n_1