Markovian Modeling for Information Leakage Quantification

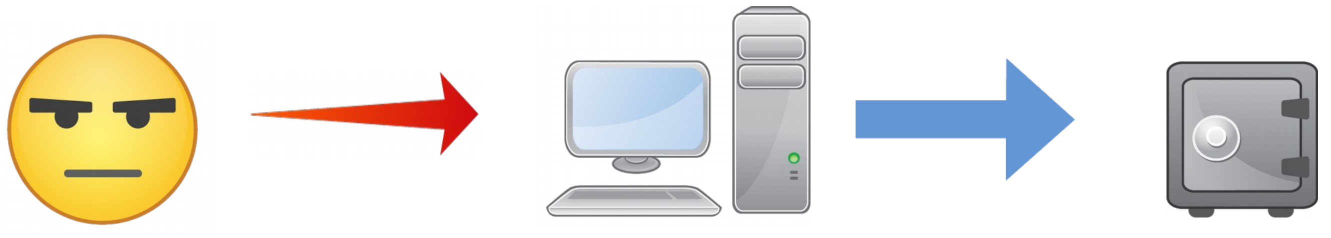
Fabrizio Biondi

EQUIPE PROJET
TAMIS
CENTRE EMETTEUR
RENNES

29/03/2016

Quantitative System Security

SCENARIO



ATTACKER observes SYSTEM depending on SECRET

QUESTION

How much is the <u>difference</u> between the attacker's knowledge about the secret before and after the observation?

Measuring leakage

EFFECTIVENESS of the attack

=

ATTACKER'S IGNORANCE before the attack

-

ATTACKER'S IGNORANCE after the attack

Different measures exist
(Shannon entropy, min-entropy, guessing entropy...)

A quick example

The secret is a 64-bit password (18,446,744,073,709,551,616 possible values)

The attacker does not know anything about it, so his *prior entropy* H(s) is **64** bits

Observing the system, he learns that the secret is divisible by 8, i.e. ends in 000 (2,305,843,009,213,693,952 possible values, corresponding to 61 bits)

His *posterior entropy* H(s|o) is **61** bits

He learnt **64 – 61 = 3** bits of information

A more complex example

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o  := r XOR s
```

Channel matrix model →

s \ o	0	1
0	0.25	0.75
1	0.75	0.25

Prior $H(s) = H(1/2,1/2) = 1$

$-$

Posterior $H(s|o) = P(o=0)H(s|o=0) + P(o=1)H(s|o=1) =\sim 0.8112..$

$=$

Leakage $\boxed{\sim 0.1887..}$

Non-terminating systems

We quantify entropy before and after
the observed system's termination...

..so what if it doesn't terminate?
(e.g. webservices, reactive systems,...)

New questions:

- How much leakage in a given time?
- How much time for a given leakage?
- How much leakage at the limit? Is it finite?
- What is the leakage rate per time unit?

Non-terminating systems

We quantify entropy before and after
the observed system's termination...

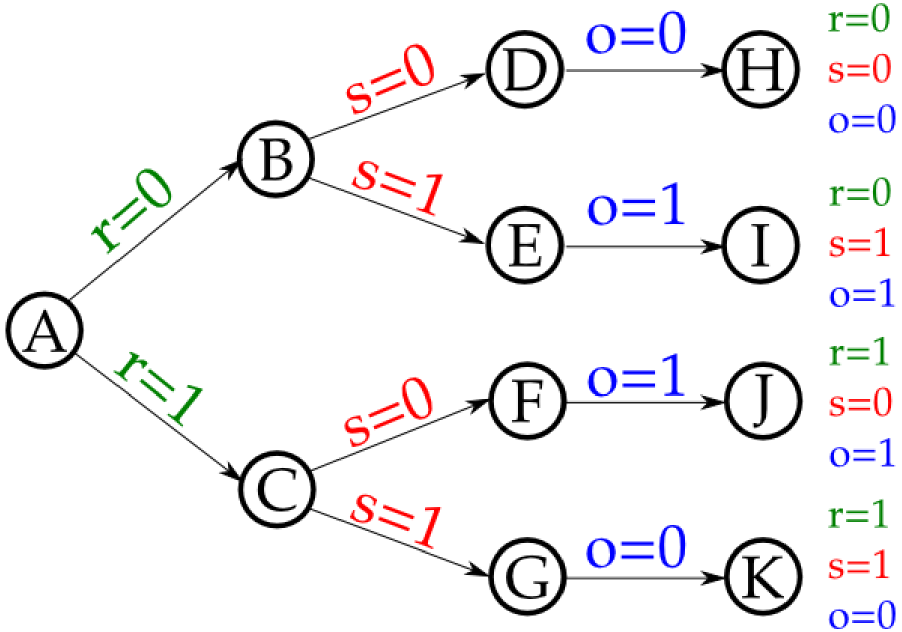
..so what if it doesn't terminate?
(e.g. webservices, reactive systems,...)

Different approach:

- Explicit time (discrete time steps)
- Some variables are observables
- Attacker always knows values of observable variables

Bit XOR as a Markov chain

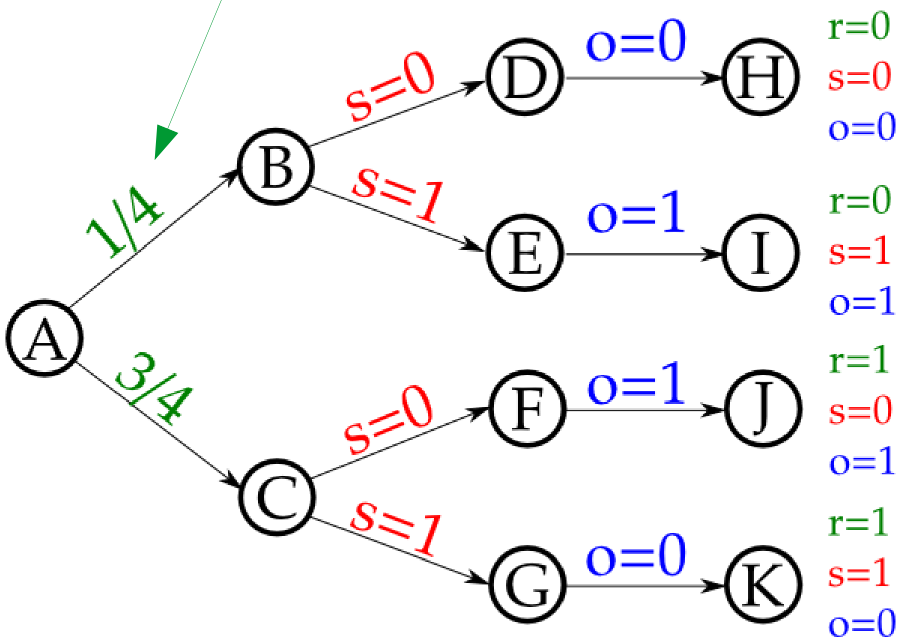
```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```



Bit XOR as a Markov chain

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```

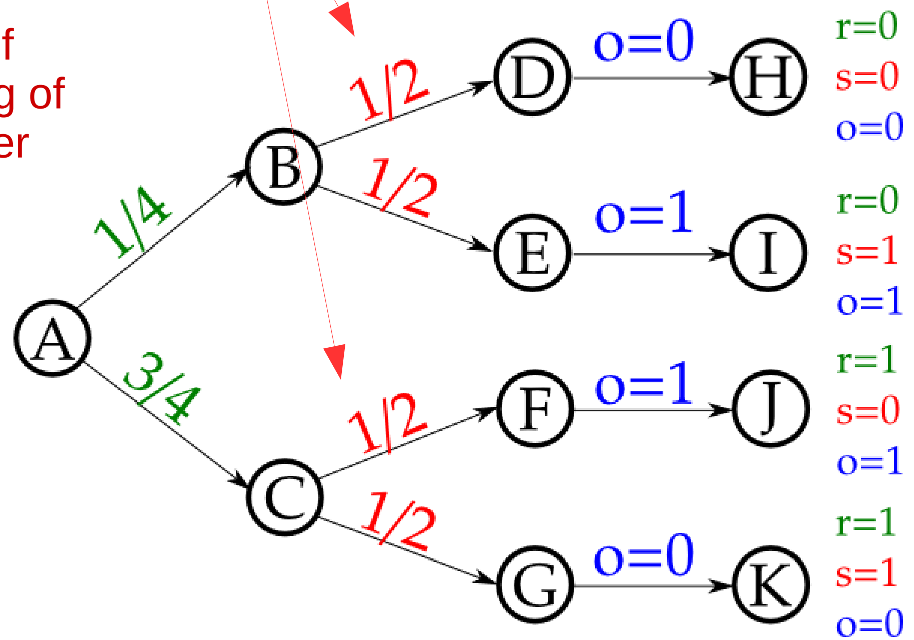
from
code



Bit XOR as a Markov chain

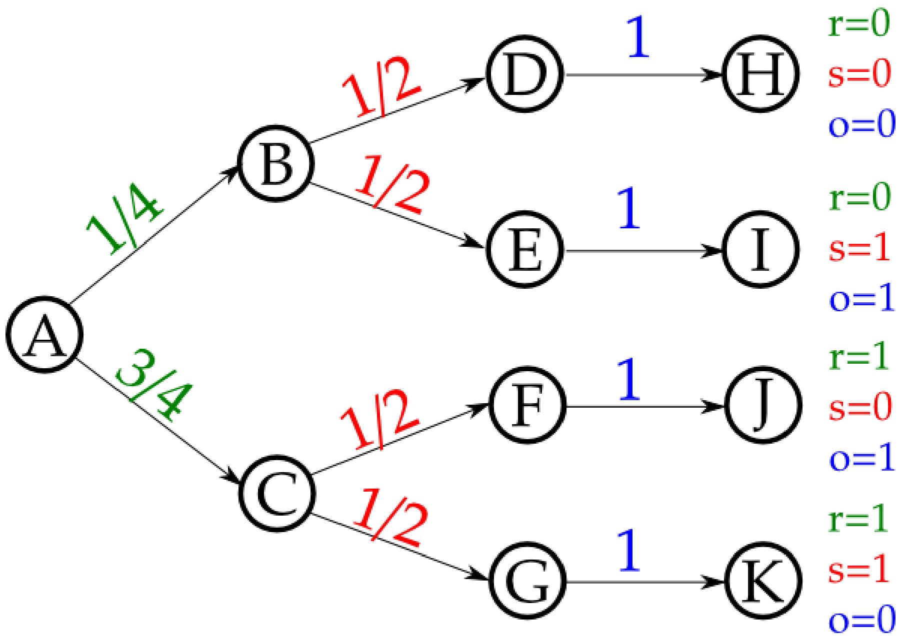
```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```

part of
encoding of
attacker



Bit XOR as a Markov chain

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```

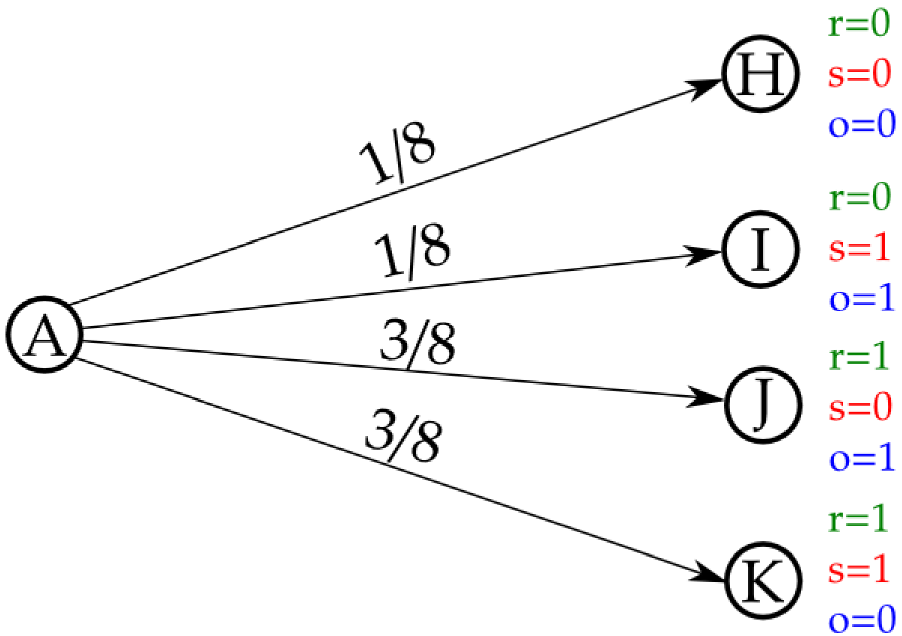


Now it's a Markov chain

Leakage computation in P-time

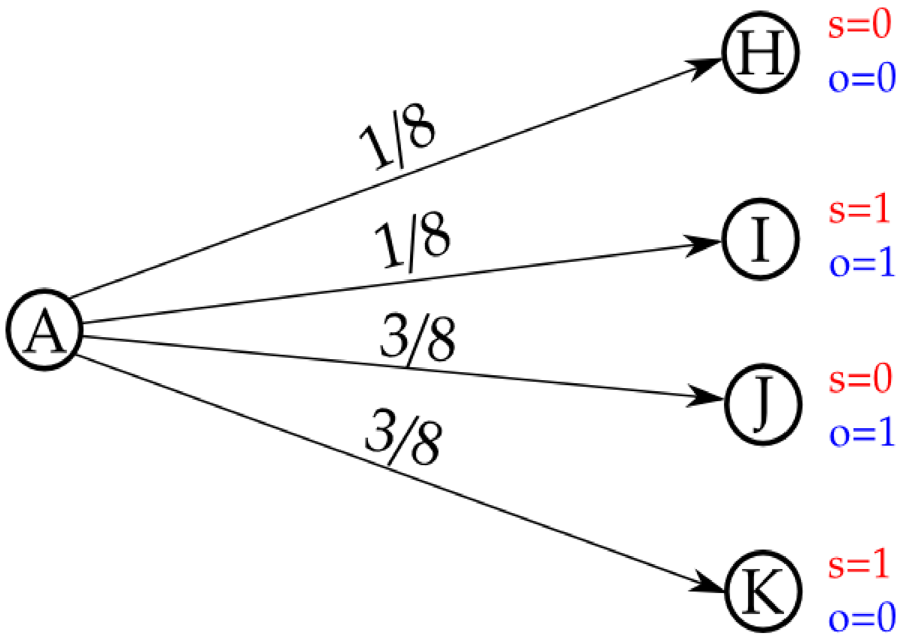
A more complex example

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```



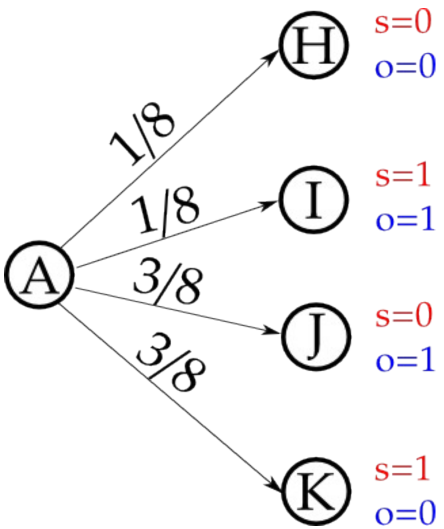
A more complex example

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```



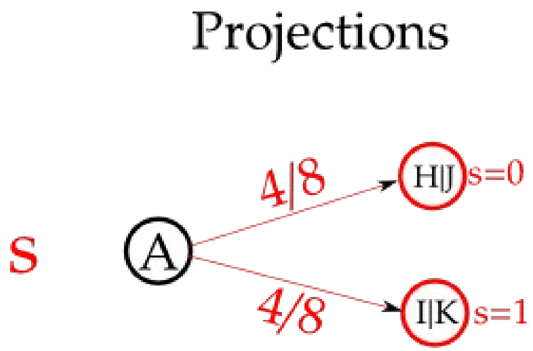
A more complex example

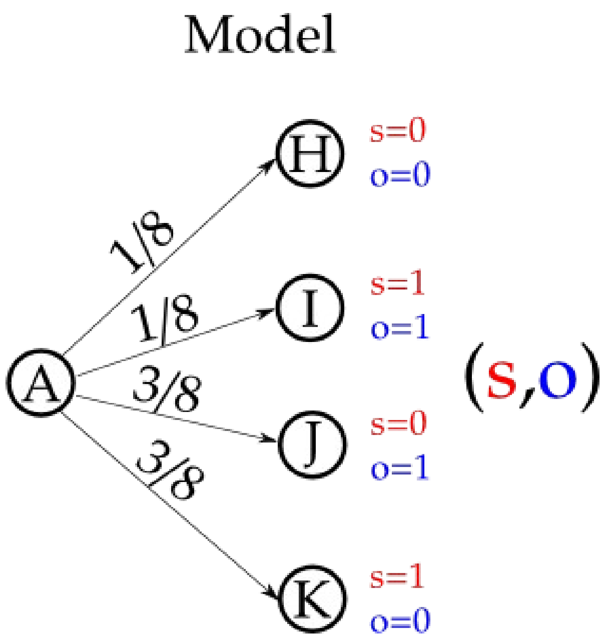
```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```



A more complex example

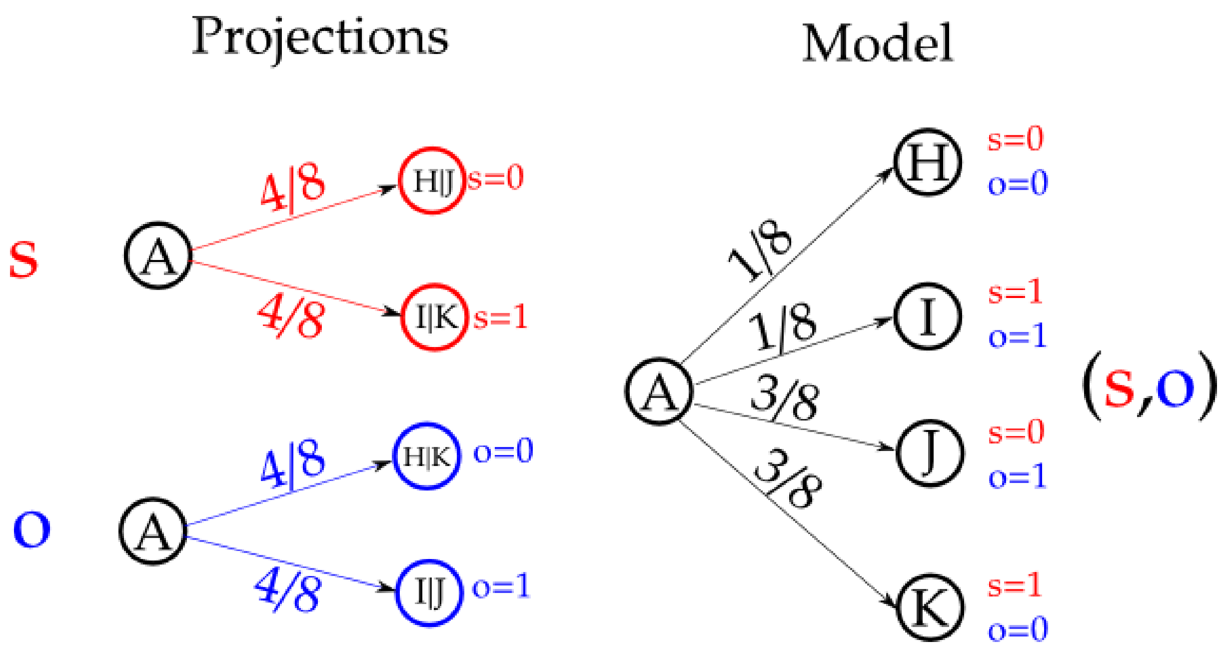
```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```





A more complex example

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```



A more complex example

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```

Projections		Model
H(s)		H(s,o)
+	=	+
H(o)		$\boxed{\text{Leakage}}$

A more complex example

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```

Projections		Model
1		1.8112...
+	=	+
1		$\boxed{\text{Leakage}}$

A more complex example

```
random bit r := {0 → 1/4 , 1 → 3/4 };
secret bit s := {0 → 1/2 , 1 → 1/2 };
output bit o := r XOR s
```

$$\boxed{\text{Leakage} \;=\; 0.1887\ldots\text{ bits}}$$

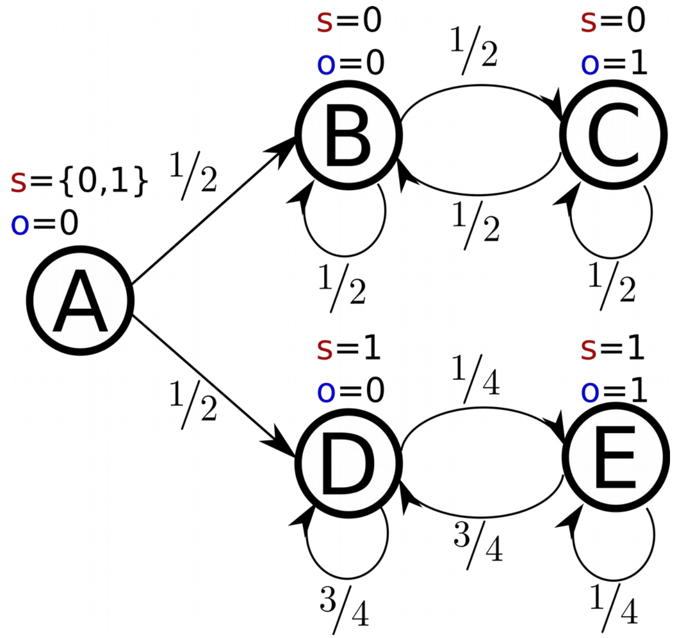
(18.87% of the secret)

Non-terminating leaking program

```
secret bit s := {0 → 1/2 , 1 → 1/2 };
observable bit o = 0;
while (true) do
   if (s==0)
       o := {0 → 1/2 , 1 → 1/2 };
   else
       o := {0 → 3/4 , 1 → 1/4 };
   fi
od
```

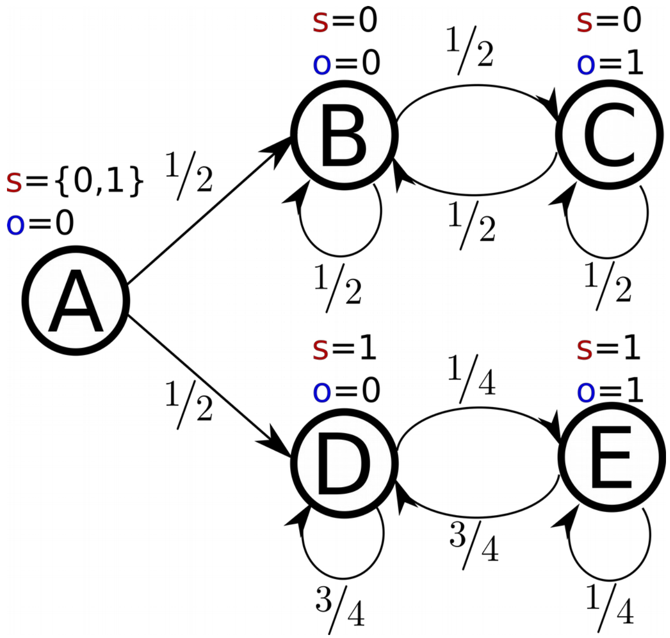
Non-terminating leaking program

```
secret bit s := {0 → 1/2 , 1 → 1/2 };
observable bit o = 0;
while (true) do
   if (s==0)
       o := {0 → 1/2 , 1 → 1/2 };
   else
       o := {0 → 3/4 , 1 → 1/4 };
   fi
od
```



Non-terminating leaking program

```
secret bit s := {0 → 1/2 , 1 → 1/2 };
observable bit o = 0;
while (true) do
   if (s==0)
       o := {0 → 1/2 , 1 → 1/2 };
   else
       o := {0 → 3/4 , 1 → 1/4 };
   fi
od
```



Attacker sees infinite string of bits

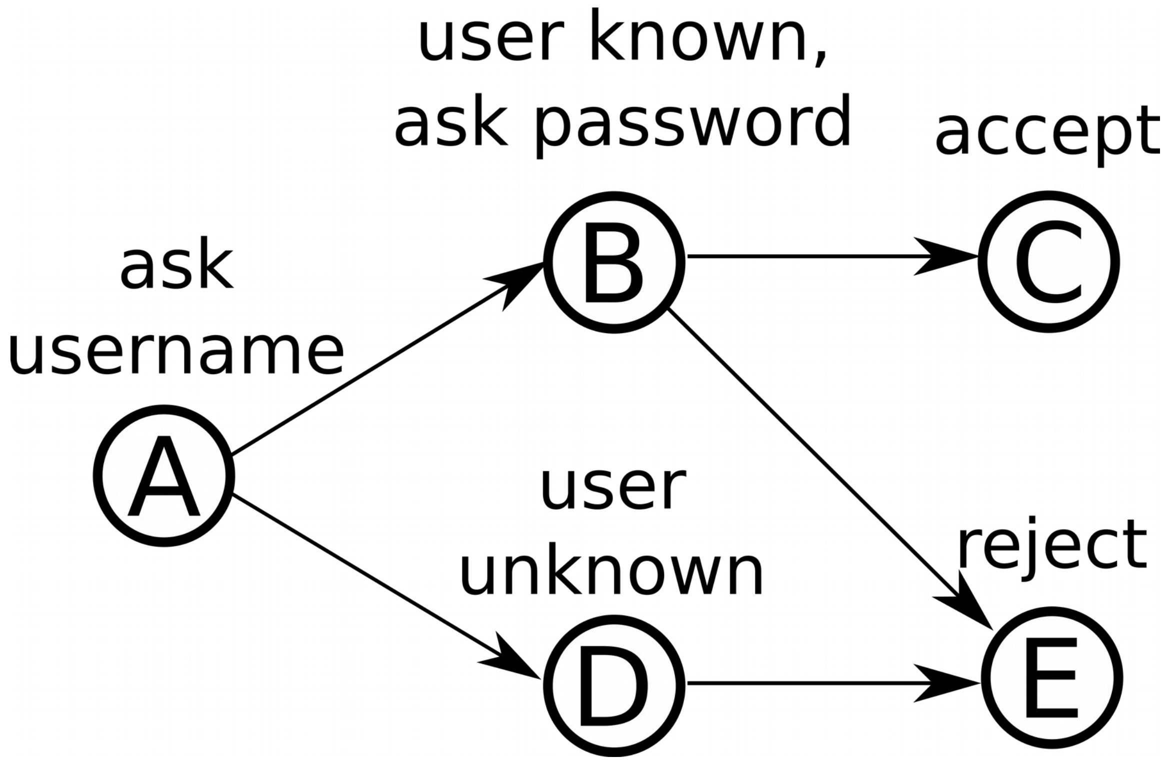
Leakage rate is 0

Leakage is 1 bit, but only at the limit

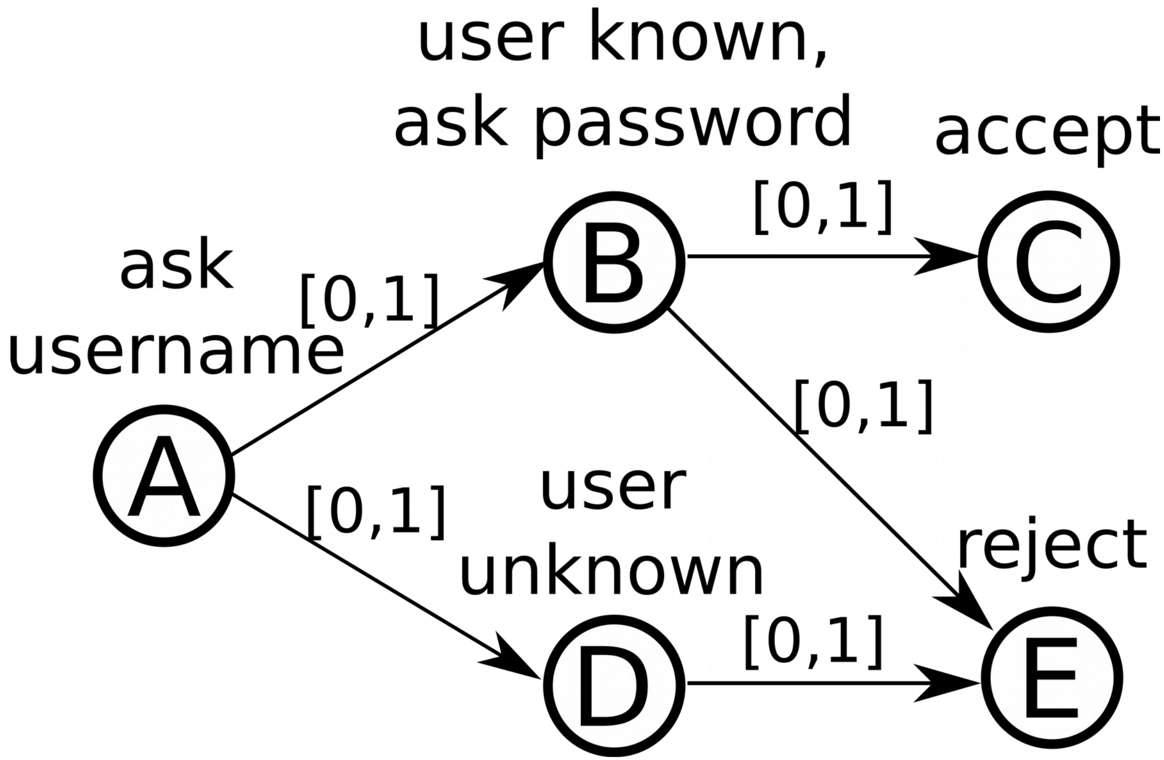
Contributions

- Finiteness of leakage of non-terminating systems can be characterized
- Limit computation of leakage and leakage rate
- Algorithm for computing limiting distribution of non-ergodic Markov chains in $\sim O(|C|^{3.3})$
- **Bounded time:** Pseudopolynomial algorithm to compute leakage from time t_1 to time t_2 in $O(t_2|C|^2)$)
- **Bounded leakage**: Finding time at which a given leakage *L* is reached is Skolem-equivalent (decidability unknown)

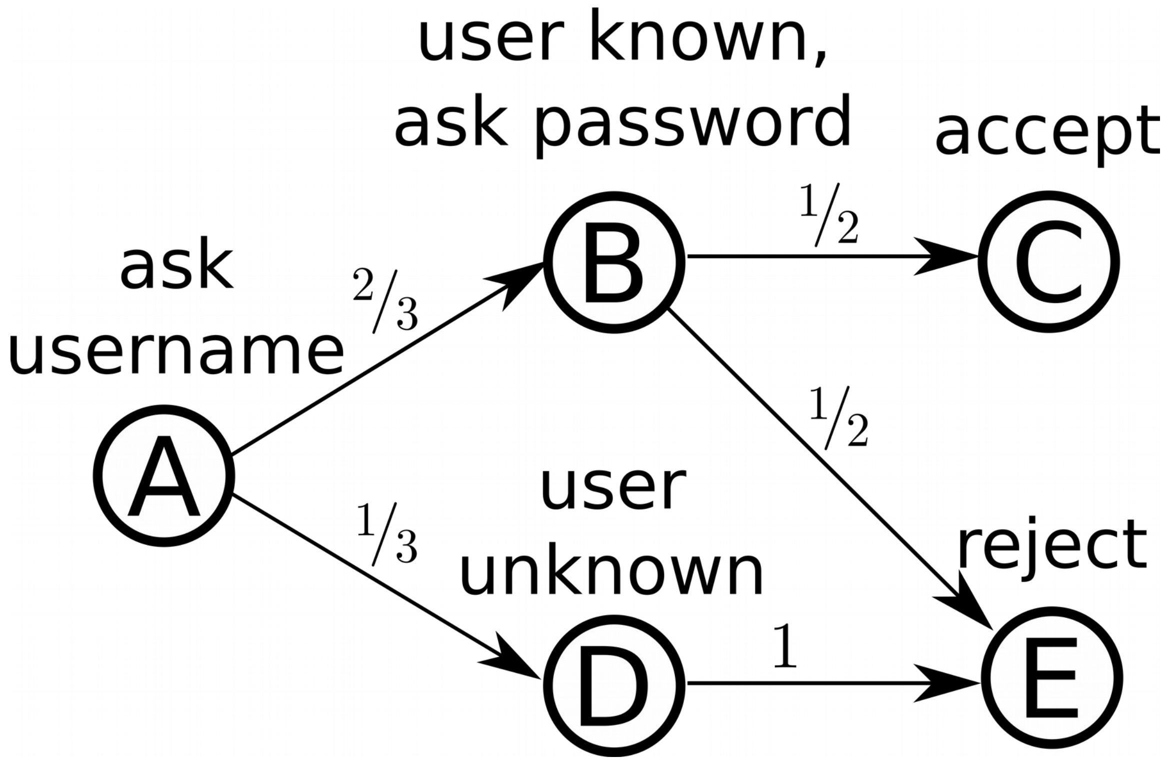
Specification



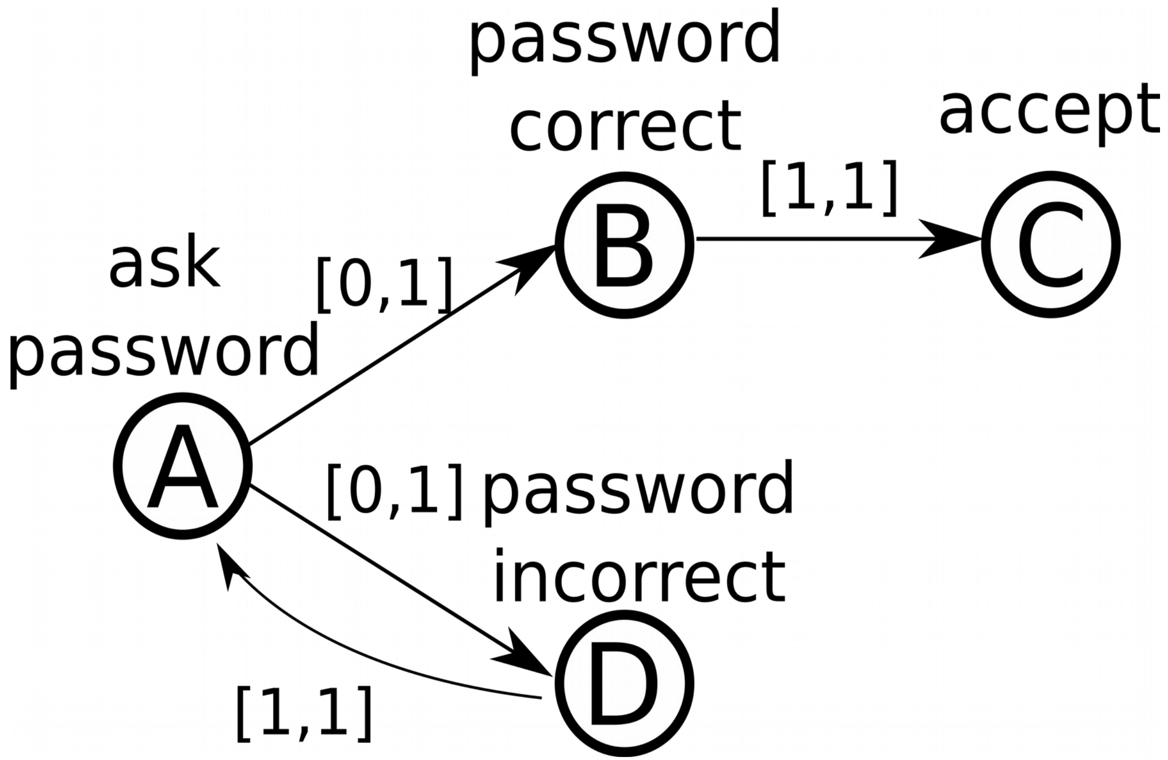
Interval Markov Chain



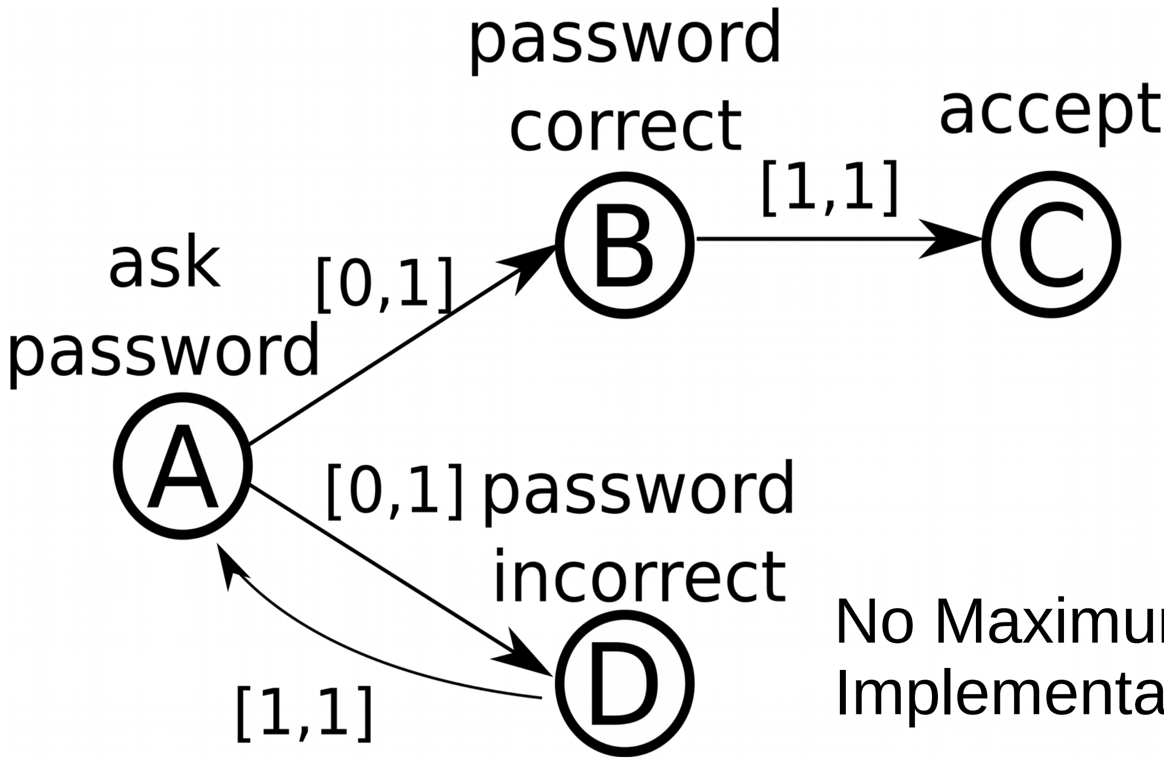
Maximum Leakage Implementation



Interval Markov Chain



Interval Markov Chain



No Maximum Leakage Implementation exists

(however...)

Contributions

- P-time algorithm to verify if Maximum Leakage Implementation exists
- If MLI exists, it can be synthesized in P-time[1]
- If MLI does not exist, we can determine in P-time if infinite leakage is possible (meaning specification allows for non-terminating implementations)
- If infinite leakage not possible, intervals can be modified to improve specification

[1] Taolue Chen and Tingting Han, *On the Complexity of Computing Maximum Entropy Markovian Models*, FSTTCS 2014

The QUAIL analyzer

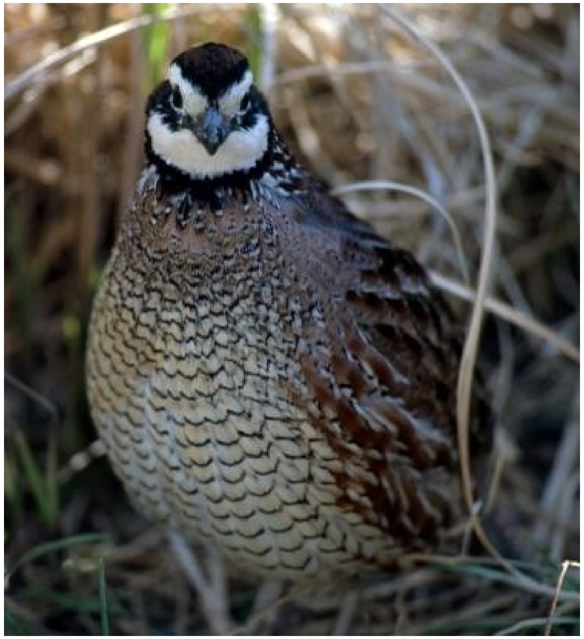
Our Quantitative Analyzer for Information Leakage implements the analysis of terminating systems.

Powerful imperative WHILE language with loops, arrays, for... translated to Markovian semantics

Available at
`https://project.inria.fr/quail`

Examples:

- Voting protocols
- Dining cryptographers
- Authentication protocols
- Smart grids

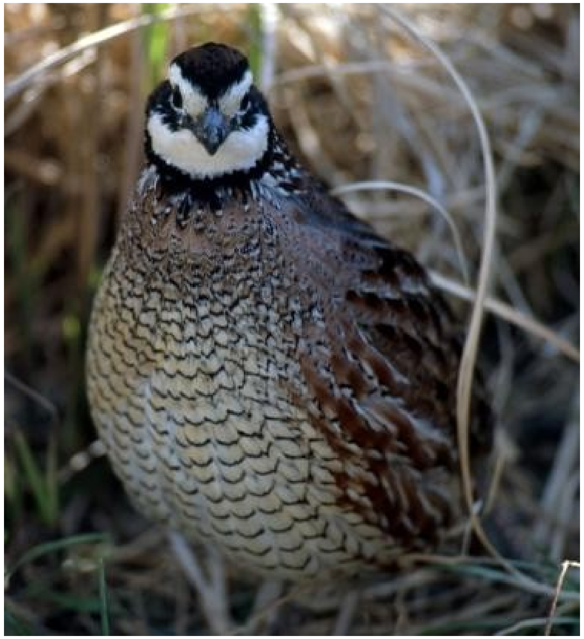


The QUAIL analyzer

Our Quantitative Analyzer for Information Leakage implements the analysis of terminating systems.

Powerful imperative WHILE language with loops, arrays, for... translated to Markovian semantics

Available at
`https://project.inria.fr/quail`



Analysis strategy:
- Precise trace analysis
- Statistical analysis (in progress)
- Compositional hybrid analysis (in progress)

Conclusions

- Markov chains can be used to model programs and compute their leakage
- Time-dependent leakage and leakage rate can be computed for non-terminating processes
- Specifications can be modeled with Interval Markov Chain, and their Maximum Leakage Implementation synthesized if it exists
- The QUAIL analyzer performs precise analysis of programs written in a WHILE language: **`https://project.inria.fr/quail`**
- All papers available at **`https://people.rennes.inria.fr/Fabrizio.Biondi/`**

Thank you for your attention!