Lecture 2

Randomness as a resource

Plan of the Lecture

Unpredictability is not always a nuisance: it can be a resource

1 Rapid gallery of tasks

2 Computational complexity

3 Randomized algorithms

4-5 Cryptography

6 Zero-knowledge proofs

GALLERY OF TASKS

Role of randomness (usually)

Only "lack of structure"

- Sampling
  - Polls
- "Monte Carlo" computations
  - Optimization
  - Software testing
  - Randomized algorithms (see next)

Secrecy / ignorance

- Games
  - Gambling, strategies

Monte Carlo method (1)

Optimization: find the maximum of $f(x_1, x_2, \dots, x_n)$

Without randomness

Sample in some pre-established points, or in some pre-established directions, and keep the maximum value.

With randomness

- Start from a point
- Find the change of f in several directions
- Choose in which direction to move at random (with suitable probabilities)
- Go to the next point and repeat.

At each step, it is actually possible to follow a direction in which f decreases
$\Rightarrow$ Randomness helps not to get stuck on local maxima

Monte Carlo method (2)

Testing a complex software (e.g. airplane control)

Without randomness
Ask a team of engineers to list all the emergency situations they can come up with, then test that the software copes with those.

With randomness
Create situations at random and see if the software copes with them

$\Rightarrow$ Randomness helps not to rely on human perception of "what is likely to happen".

Gambling

- Various activities go under “gambling”:
  - Games: cards, dice, roulette, pachinko, mahjong…
  - Betting
- **Randomness = ignorance ⇒ fairness**
  - Neither the client, nor the house have privileged knowledge (⇒ the house wins on average)
- Famous non-criminal “break the house” cases:
  - Joseph Jagger in Monaco, 1873
  - The MIT Blackjack team (1979-2000)

Strategies: game theory

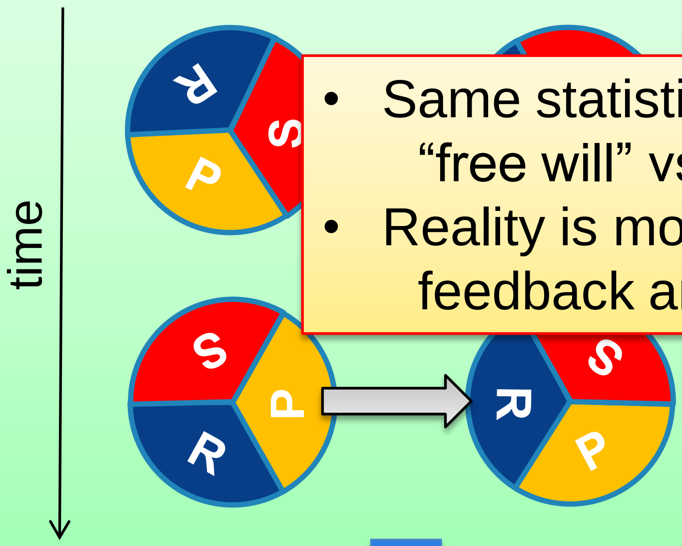
Rock-Paper-Scissor

- The best is for each player to draw their choice with probability 1/3.
- If one player predictably deviates from this, he will lose.
- An example of “Nash equilibrium” realized by a “mixed” strategy, that is a strategy that uses randomness.

This kind of games is widely used to model human and animal behavior

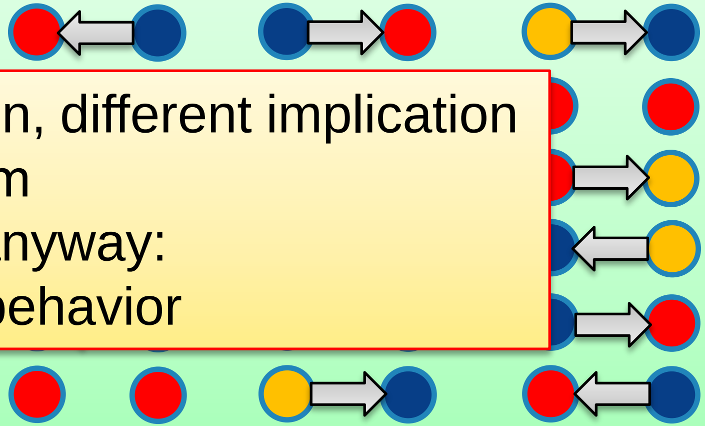
Where is the randomness?

Individuals playing a mixed strategy



Each individual uses randomness

Populations playing a mixed strategy



Randomness from "sampling" (the population has varying traits, which determine each individual's behavior)

- Same statistical conclusion, different implication
  "free will" vs. determinism
- Reality is more complex anyway:
  feedback and adaptive behavior

Suggested Readings

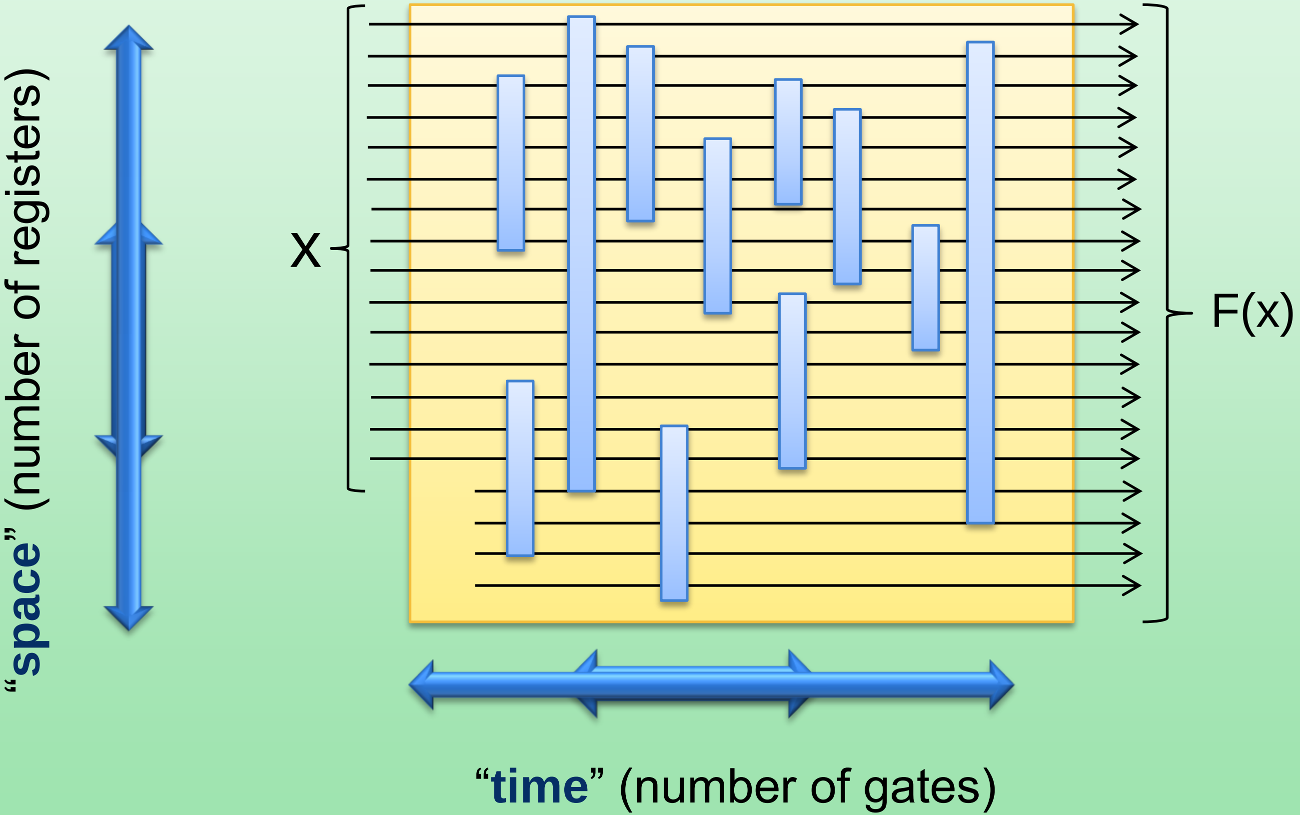
Wikipedia pages:

- http://en.wikipedia.org/wiki/Game_of_chicken
- http://en.wikipedia.org/wiki/Matching_pennies
- http://en.wikipedia.org/wiki/Rock,_Paper,_Scissors
- http://en.wikipedia.org/wiki/Unscrupulous_diner%27s_dilemma
- http://en.wikipedia.org/wiki/Evolutionary_game_theory
- http://en.wikipedia.org/wiki/Monte_Carlo_method
- http://en.wikipedia.org/wiki/Roulette
- http://en.wikipedia.org/wiki/Joseph_Jagger
- http://en.wikipedia.org/wiki/MIT_Blackjack_Team

TOOL: COMPUTATIONAL COMPLEXITY

Elementary notions

Notion of Scaling

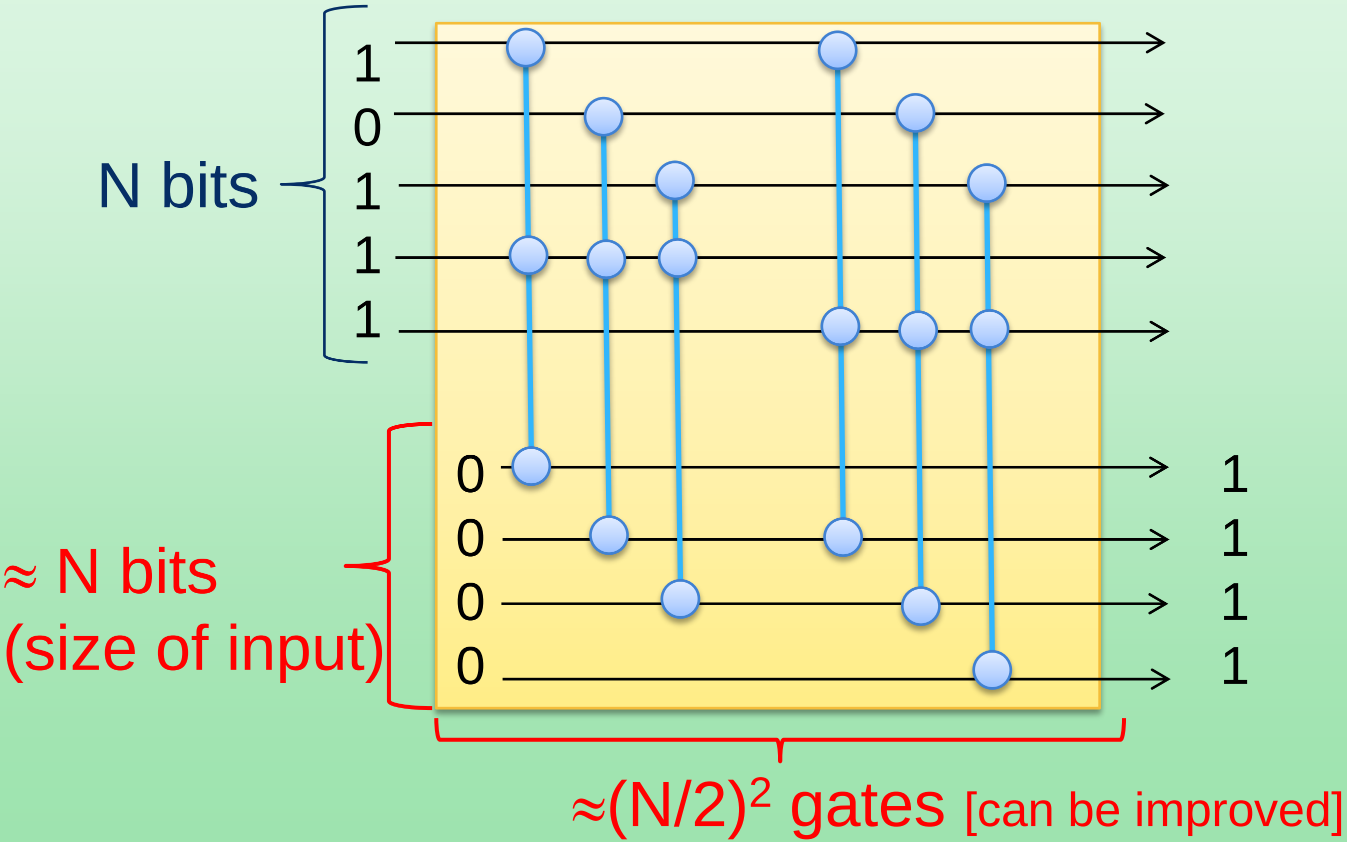


How do time and space change with size(X)=N?

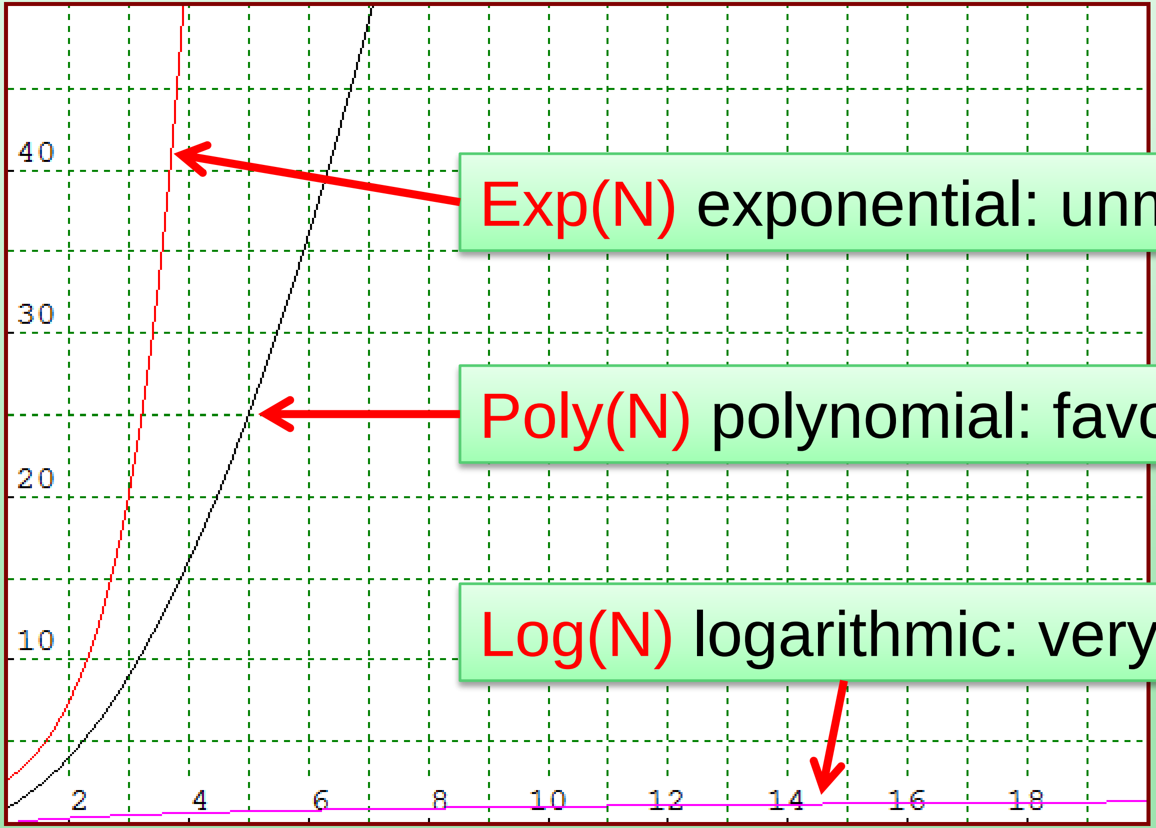
Pedestrian example

5x3 in binary:

```
5   101
3    11
   ----
    101
   101
   ----
   1111 = 15
```



Usual scaling



Exp(N) exponential: unmanageable

Poly(N) polynomial: favorable

Log(N) logarithmic: very favorable

Frequent notation

$F(N) = O\big(f(N)\big)$:

$F(N) \leq Mf(N)$ for large N and a constant $M < \infty$.

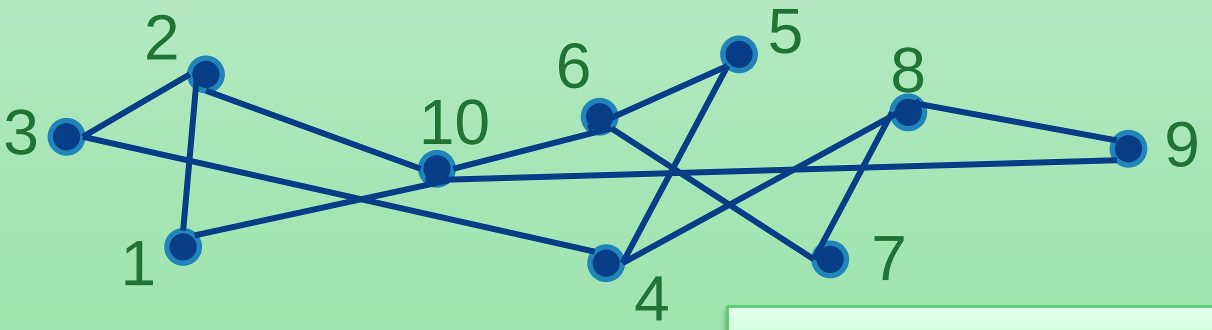
Examples:

- $1000000 \log N = O(\log N)$
- $N + \log N = O(N)$
- $\mathrm{e}^{\mathrm{N}} = 2^{\mathrm{N}\log_2 \mathrm{e}} = 2^{O(N)} \neq O(2^{\mathrm{N}})$

This is what is meant by Log, Poly, Exp scaling

The famous P&NP

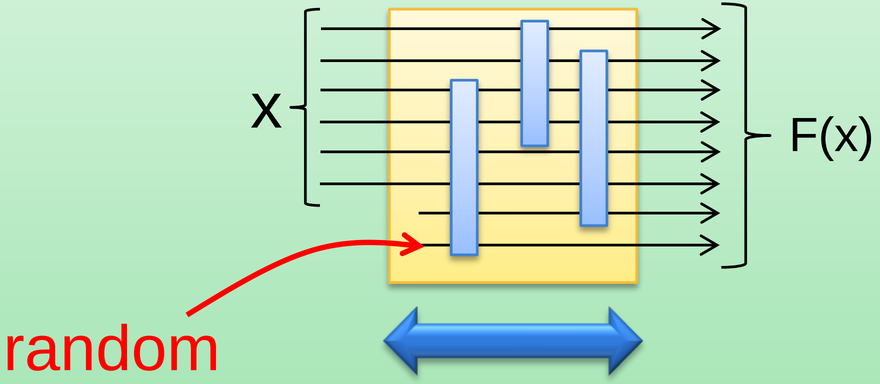
- P = "polynomial time" $\Rightarrow$ *efficiently solvable*
- NP = "nondeterministic polynomial time": *efficiently verifiable* if a solution is given
  - Example: traveling salesman with length L



Still, it may be that P=NP…

When randomness matters: BPP

"Bounded error probabilistic polynomial time"



~~More than Poly(N)~~

Poly(N) with some error

Suggested Readings

Wikipedia pages:

- http://en.wikipedia.org/wiki/Complexity_class
- http://en.wikipedia.org/wiki/Computational_complexity_theory

However, for the experts, there are some 500 classes! Check them here: https://complexityzoo.uwaterloo.ca/Complexity_Zoo

IS BPP=P?

The curious story of primality tests

The task

Primality test PRIMES(m):
Given an integer number m, find out if it is prime.

Number of bits required to encode m: $N(m) = \lfloor \log_2 m \rfloor$ the smallest integer larger than $\log_2(m)$

$\Rightarrow$ The input $m \approx 2^N$ is Exp in N

Remark: the *number of digits in base 10* is $N_{10}(m) = \lfloor \log_{10} m \rfloor \approx (\log_{10} 2) N(m) \approx 0.3 N(m)$; so one can also say that the input is Exp in the number of digits

Obvious deterministic test

Divide m by *all the odd integers* up to $\sqrt{m}$:

- If you find a divider, m is certainly not prime
- If you don't find a divider, m is certainly prime

Advantage: deterministic, unconditionally correct

Problem: $Time(m) \approx \sqrt{m} \approx 2^{N/2}$.

Nothing much better until the 1970s.

Randomized algorithm (1)

For a an integer number, 1<a<m, consider the equation

$$\left(\frac{a}{m}\right) = a^{(m-1)/2} \bmod m$$

"Jacobi symbol"

If you are not familiar with primes, don't worry: all you need to know is that all these numbers are efficiently computable.

- <u>If m is prime</u>, the equation is satisfied for all values of a.
  Proof: follows from "Euler's theorem"
- <u>If m is composite</u>, the equation is satisfied by approximately half the values of a.

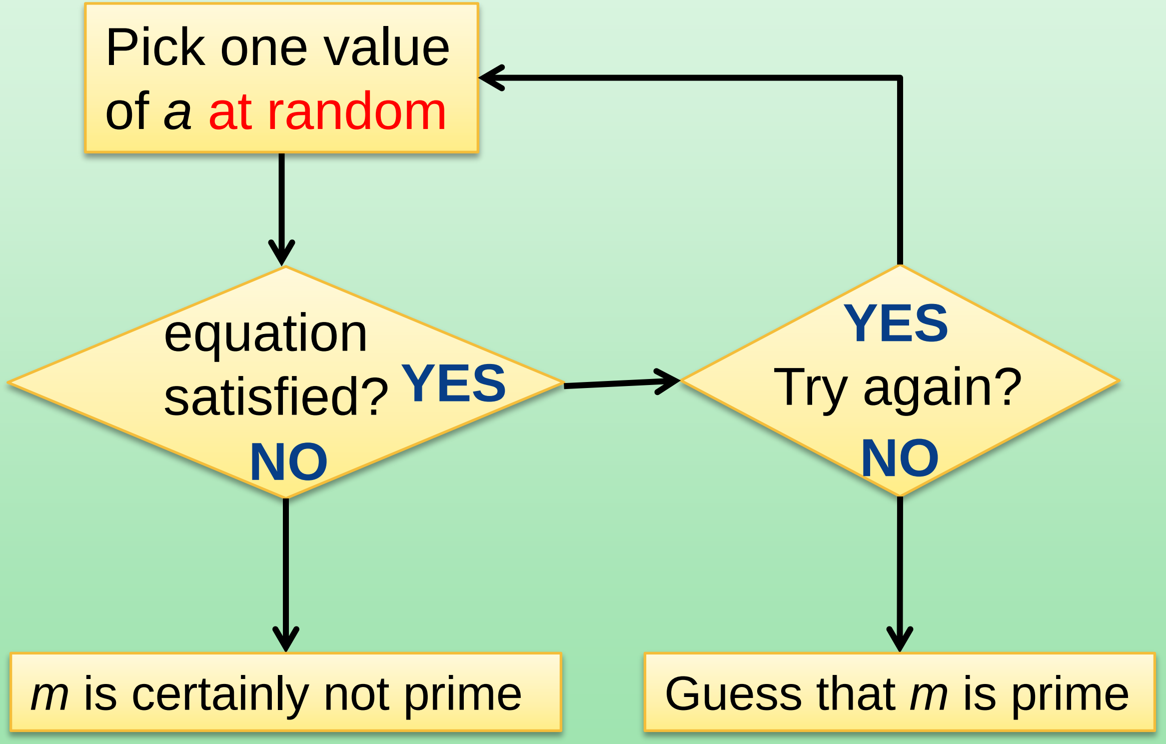
Idea: pick one value of a and try your luck!

Are you familiar with mod? If yes, skip this. If not:

$\boldsymbol{c = a \bmod b}$, where a,b,c are positive integers, means that the remainder of the division of a by b is c, that is $a = nb + c$ where n is a positive integer. Clearly one has $b > c$.

Example: $a\ mod\ 2 = 0$ for even numbers, $a\ mod\ 2 = 1$ for odd numbers

Randomized algorithm (2)



Can be wrong, but the probability of wrong guess decreases exponentially with the number of calls

Randomness is useful…

- Solovay-Strassen algorithm 1973
  - Scales as $O(\log^3 m) = O(N^3)$
- Miller-Rabin 1974, then others
- Randomized algorithms later found for many other problems

$\Rightarrow$ BPP is probably larger than P

… or maybe not!

- 2002: Agrawal, Kayal and Saxena (AKS) find a deterministic algorithm that scales Log: PRIMES is in P.
  - An improved version has $O(\log^6 m) = O(N^6)$
- De-randomization is one of the big topics in computer science.
- Maybe BPP = P after all.

Suggested Readings

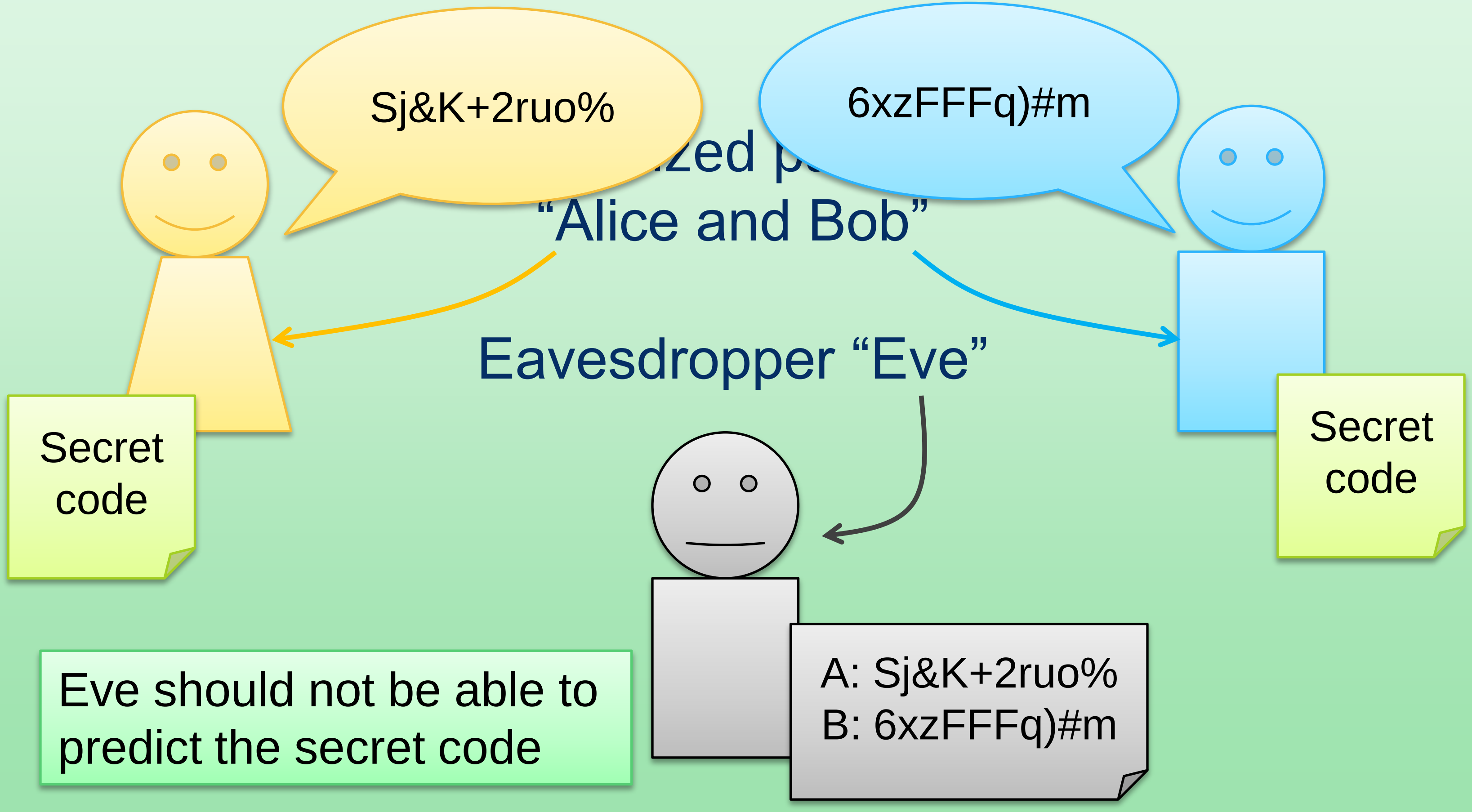
Wikipedia pages:

- http://en.wikipedia.org/wiki/Primality_test

CRYPTOGRAPHY (1)

Secret keys

Scenario


Sj&K+2ruo%
6xzFFFq)#m
“Alice and Bob”
Eavesdropper “Eve”
Secret code
Secret code
A: Sj&K+2ruo%
B: 6xzFFFq)#m
Eve should not be able to predict the secret code

Historical notes

- Until the 20th century: someone invents a code, someone else finds how to break it.
- A bad idea: trust that *the protocol* won't be discovered ("security by obscurity")
  - It may work at times (e.g. Navajos)…
  - … but big failures (e.g. Enigma machine)
- Kerckhoff's principle (1883): let the protocol be known $\Rightarrow$ need to guarantee only the security of the "key"

One-time pad (OTP; Vernam 1917)

Message

$\underbrace{01001000}_{\text{H}}\underbrace{01101001}_{\text{i}}$

ASCII

- Has a meaning

Key

0110111101000000

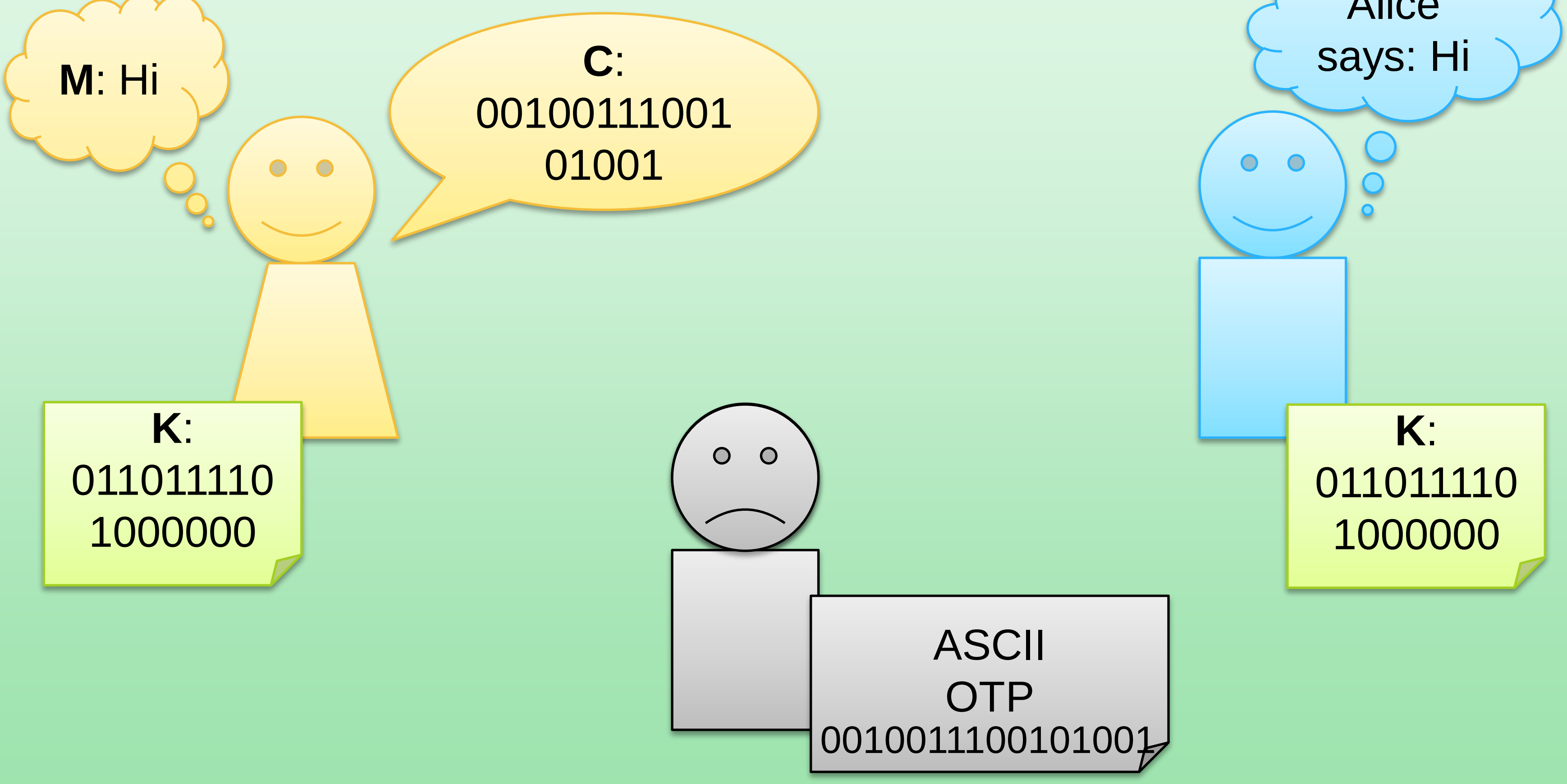
- Drawn at random (no meaning)
- As long as the message

Cypher
(encrypted text)

```
M 0100100001101001
K 0110111101000000
------------------
C 0010011100101001
```

- Binary sum bit by bit

OTP in action



Security of OTP

If K is random for Eve, C carries no information about M
$\Rightarrow$ Eve cannot learn anything by hearing C

Proof (more formal: Shannon 1940s)
Suppose c_j=0: if k_j=0, then m_j=0; if k_j=1, then m_j=1.
$\Rightarrow$ If Eve knows *nothing* about k_j, she can just as well guess m_j by tossing a coin. Similar if c_j=1.

Level of security: **"information-theoretical" or "unconditional" security**:

- As opposed to "security based on computational assumptions".
- Notice that there are "conditions" for unconditional security: K must be random for Eve, as long as the message, kept secret…

A balance of OTP

Advantages

- Unbreakable in principle, provided the Key:
  - is as long as the message
  - is used only once
  - is unpredictable for Eve
  - does not leak out

Practical problems

- Key distribution
  - Once a key has been used, Alice and Bob have to find a way of sharing a new one
- Key storage
  - Large and *safe* memory (who keeps its key?)

e-banking uses OTP. But not **e-commerce**: your computer cannot have an OTP key stored for any possible site you may want to buy from…

Suggested Readings

Wikipedia pages:

- Simon Singh, The code book
  http://en.wikipedia.org/wiki/The_Code_Book
- http://en.wikipedia.org/wiki/One-time_pad

And here is the webpage for ASCII code: http://www.ascii-code.com/

CRYPTOGRAPHY (2)

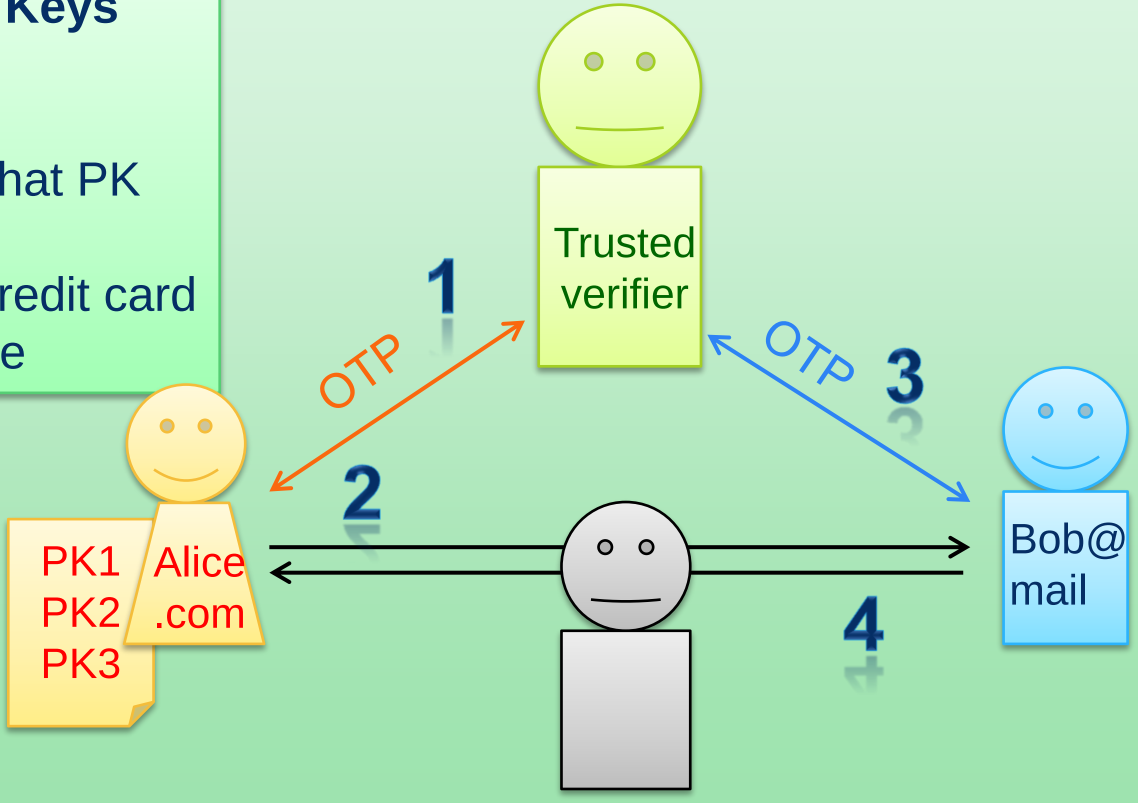
Public keys

Online transactions

1. Verifier learns the **Public Keys** (PKs) of Alice
2. Alice sends a PK to Bob
3. Bob checks with Verifier that PK belongs to Alice
4. If Yes, Bob encodes his credit card number and sends to Alice

- Eve learns PK during 2 (knows as much as Bob)
- Bob's text 4 contain info about the message (credit card number)

⇒ how can this be secure?



RSA (1): preparing the key

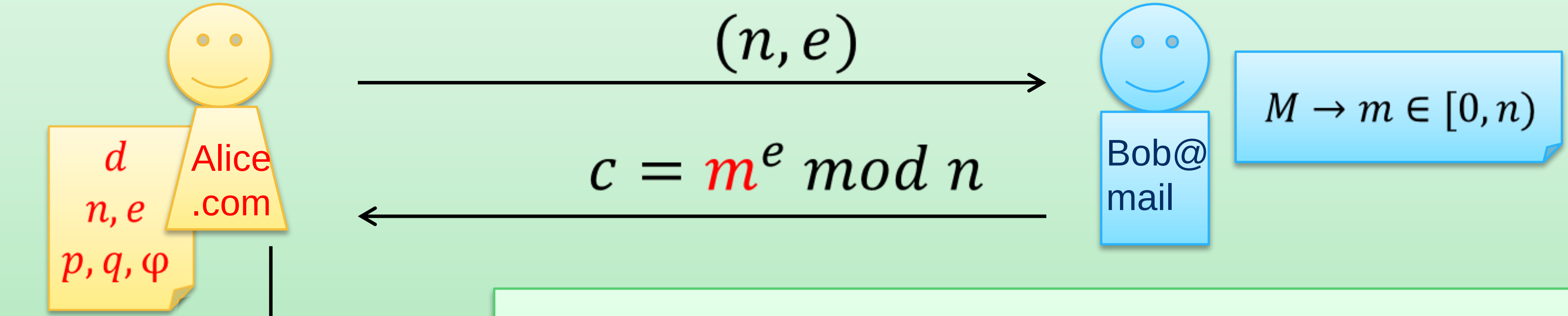
Rivest-Shamir-Adleman 1977

1. Choose **at random** two primes p, q. (of approximately same size)
2. Compute $n = pq$ and $\varphi(n) = (p-1)(q-1)$.
3. Choose e such that $1 \leq e < \varphi(n)$ and co-prime with $\varphi(n)$.
4. Find d such that $de = 1 \; mod \; \varphi(n)$ i.e. $de = \varphi(n)k + 1$

The idea behind: if $c = m^e \; mod \; n$, then $m = c^d \; mod \; n$.

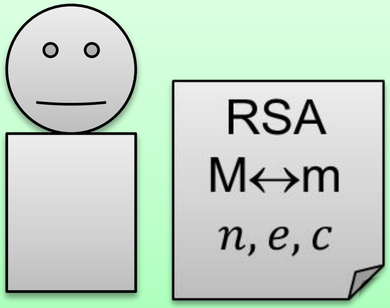
Alice's public key is (n, e)
Alice keeps secret the private key d and all that allows to compute it directly: $p, q, \varphi(n)$

RSA(2): encoding & decoding



$$m = c^d \bmod n$$

$m \to M$



- d is contained in n, if only Eve could factor $n \to (p, q)$
- "One-way" function: multiply is easy, factoring is not…
- … but it's possible: security based on limited computational power of Eve.

Maths of RSA

Lemma: if $c = a \, mod \, b$, then $a^n \, mod \, b = c^n \, mod \, b$.
Proof: $c = a \, mod \, b$ means $a = bk + c$ for some integer k. The result follows by expanding a^n using Netwon's binomial.

RSA encoding and decoding: if $c = m^e \, mod \, n$, then $m = c^d \, mod \, n$.
Proof: using the previous lemma, $c^d \, mod \, n = m^{ed} \, mod \, n$, with $ed = k\varphi(n) + 1$ for some integer k. Euler's totient theorem states that $a^{\varphi(n)} = 1 mod \, n$ if a is co-prime with n.
Therefore, if m is co-prime with n, it holds:

$$m^{ed} = \left(m^k\right)^{\varphi(n)} m = (1 + k'n)m$$

whence $m^{ed} \, mod \, n = m$.
The case where m is co-prime with n happens with exponentially low probability. In this case, it is easy for factor n, so Bob can check it

A balance of RSA

Advantages

- Key generated only by the receiver
  - The choice of (p,q) must be random for Eve.
  - Key distribution is not a problem
  - Key storage: only one location

Potential issues

- Security based on an assumption of limited computational power
  - 768-bit (231-digit) key cracked in 2010
  - 1024-bit (308-digit) keys being used as secure (need 150-digit primes).

Suggested Readings

Wikipedia pages:

- http://en.wikipedia.org/wiki/RSA_(algorithm)

And here are some articles related to RSA algorithm:

- http://arstechnica.com/security/2010/01/768-bit-rsa-cracked-1024-bit-safe-for-now/
- http://arstechnica.com/security/2013/10/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography/

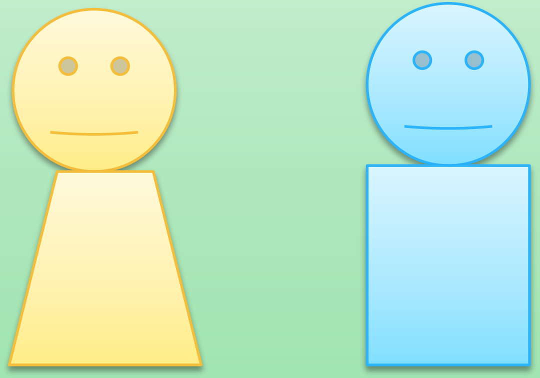
ZERO-KNOWLEDGE PROOFS

The goal

Alice wants to convince Bob that she knows the proof of a theorem, without showing the actual proof to Bob.

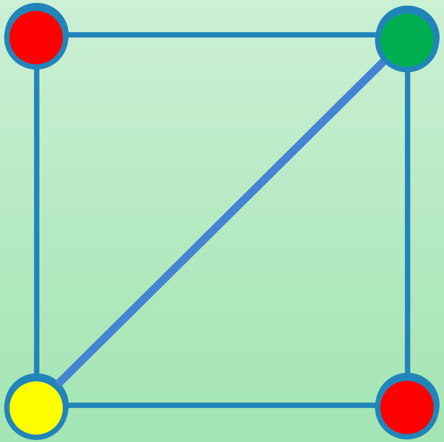
Different scenario:

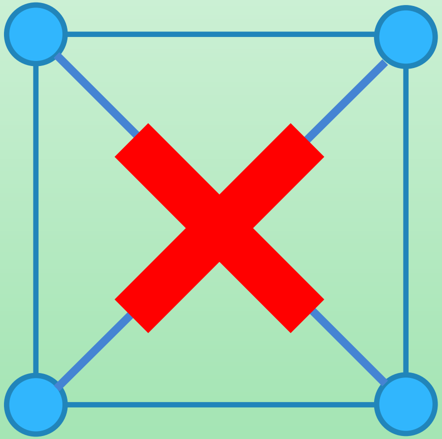
- Alice and Bob don't trust each other
- There is no Eve



Three-coloring

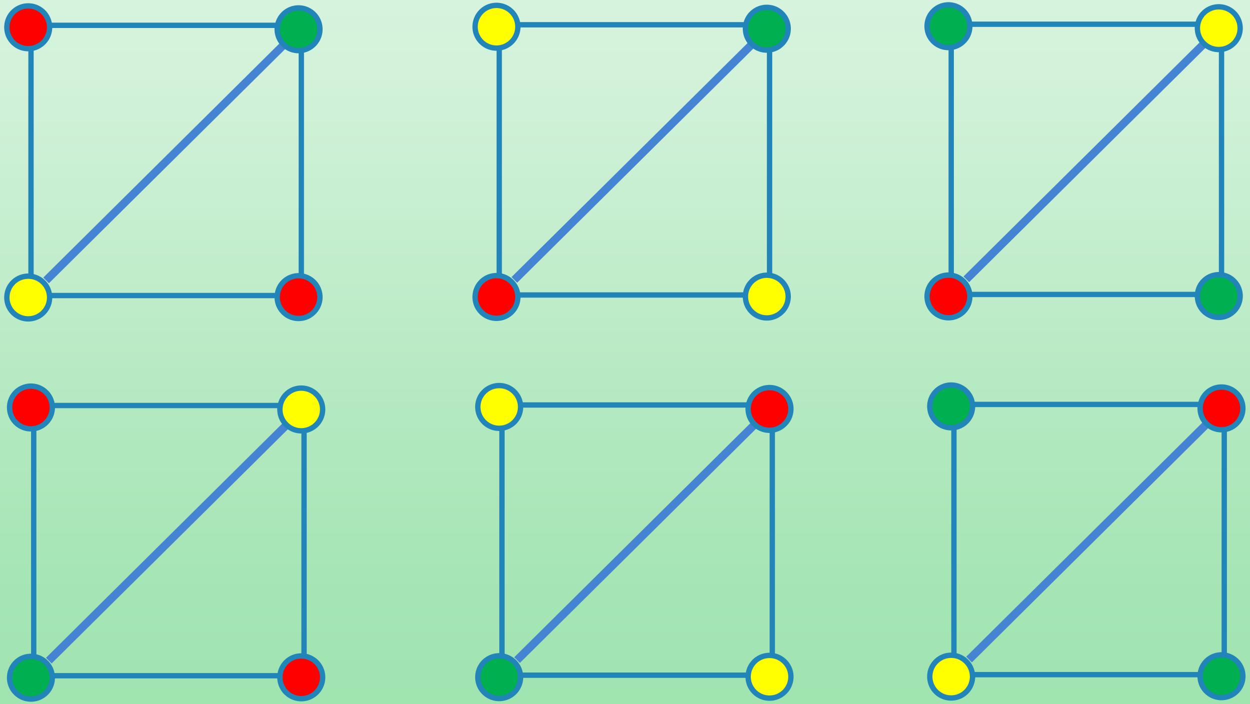
Can a planar graph be "colored" with only three colors, in such a way that two connected nodes have different colors?





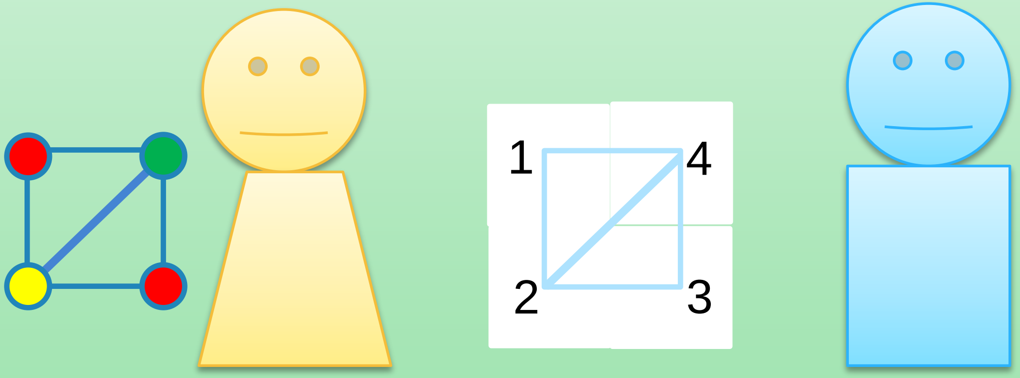
An obvious symmetry

Each 3-coloring comes in 6 versions (permutations)

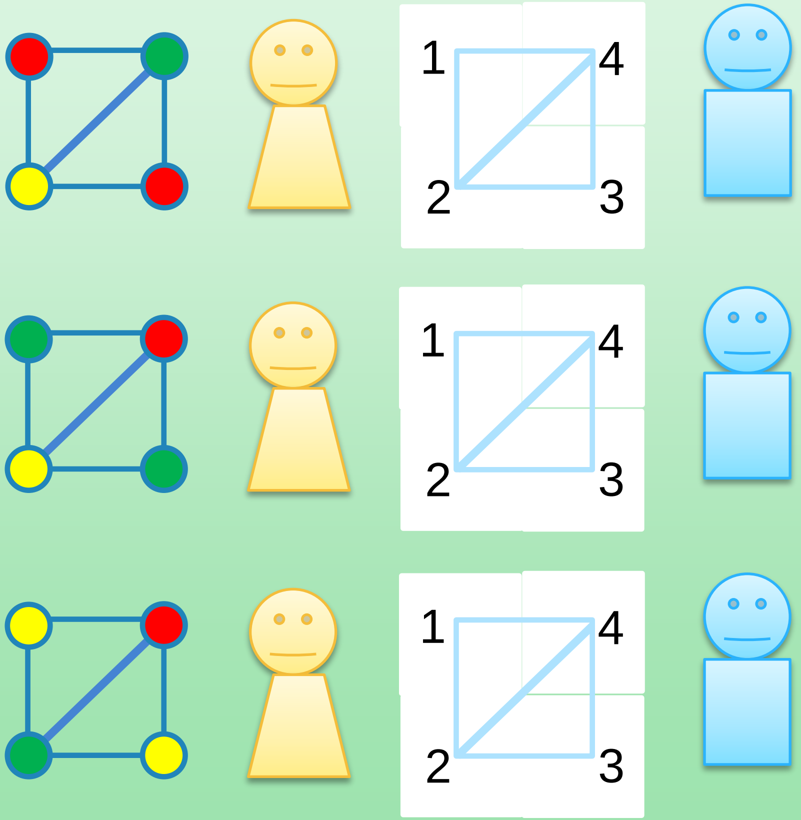


The goal, again

Alice has found a 3-coloring and wants to convince Bob that she has indeed, but without revealing the solution to Bob



Repeat the game



- If Alice is caught at fault, she did not have the proof.
- If Alice is never caught at fault:
  - she must have the 3-coloring indeed, provided she cannot *predict* which nodes Bob will look to;
  - Bob will not learn how to color the graph, unless he can *predict* which runs correspond to which permutation.

From coloring to infinity

But what if Alice's proof is not about 3-coloring?

The statement "I have a proof of [theorem] of length n" can be mapped to a graph, and the statement is true if and only if the graph is 3-colorable.

Hint of the argument:

1. "Having a proof of [theorem] of length n" is NP
   - This can be formalized, but intuitively: it is hard to find a proof, but easy to check if someone gives you a proof.
2. By the Cook-Levin theorem, any NP problem can be solved if one can solve one of the "NP-complete" problems, and 3-coloring is such.

Cryptographic remark on ZKP

This kind of ZKP relies on:

1. Alice must choose a permutation before learning about Bob’s choices (commitment)
2. Bob must not learn which permutation is being chosen in each run.

This is OK if you do it “in person”. But as an algorithm between communicating stations, it requires a primitive called “*bit commitment*”. This can be implemented under computational assumptions, but not unconditionally.

This presentation was inspired by a lecture that Avi Wigderson (http://www.math.ias.edu/avi/) gave in NUS in December 2012

Suggested Readings

[Sanjeev Arora and Boaz Barak: Computational complexity]

Wikipedia page:

- http://en.wikipedia.org/wiki/Cook%E2%80%93Levin_theorem

Summary of Lecture 2

Unpredictability is not always a nuisance: it can be a resource

- Tasks
  - Lack of structure: polls, Monte Carlo optimization
  - Games (objective or subjective randomness?)
- Randomized algorithms
  - Can we de-randomize (i.e. BPP=P)?
- Cryptography: untrusted parties
  - Secret communication: OTP, RSA
  - Other scenarios (ZKP)