Aalto University
School of Science

CS-E4530 Computational Complexity Theory

Lecture 13: Cryptography

Aalto University
School of Science
Department of Computer Science

Spring 2017

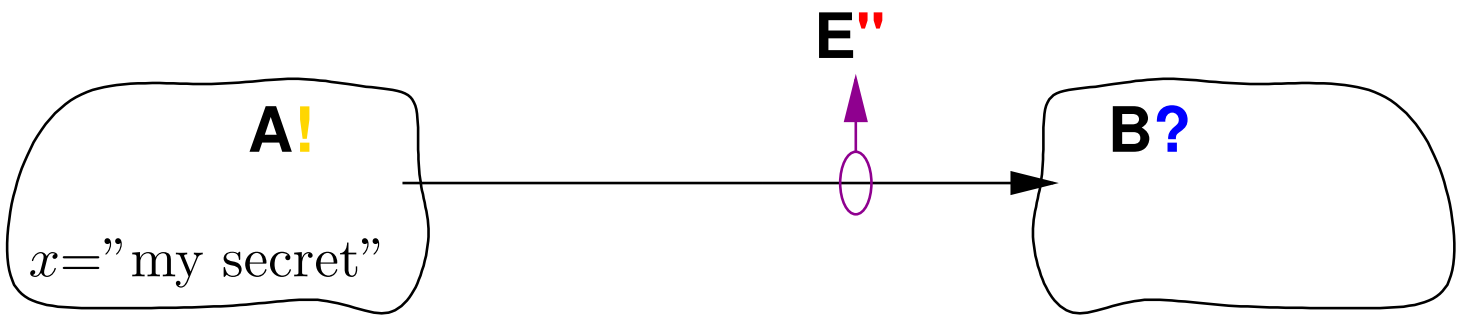
Agenda

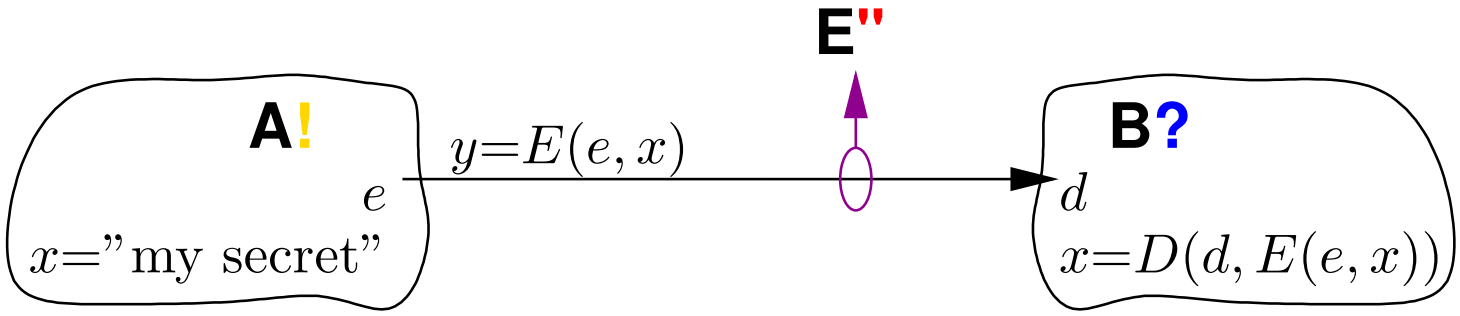
- Cryptosystems
- One-way functions
- The RSA public-key cryptosystem
- Cryptography and complexity
- Randomised cryptography
- Cryptographic protocols
- Signatures
- Mental poker
- Interactive proofs
- Zero knowledge

(C. Papadimitriou: *Computational Complexity*, Chapter 12)

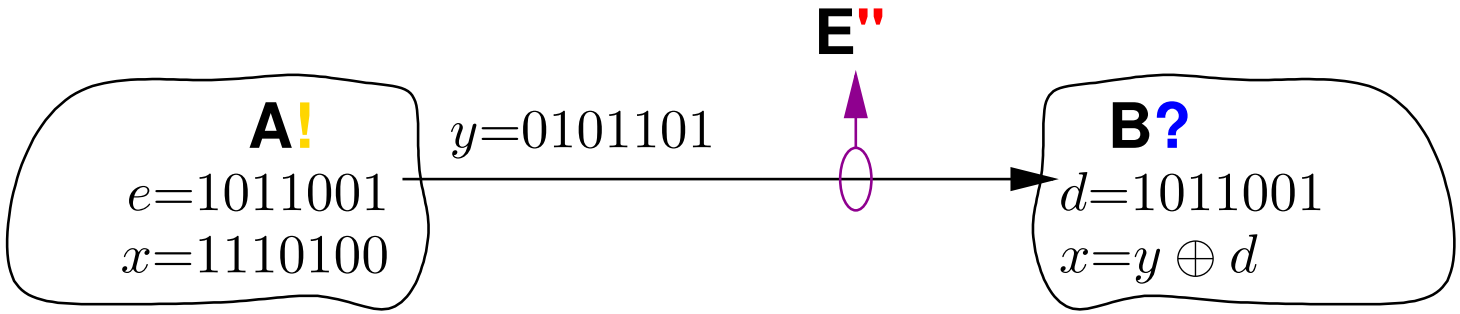
Cryptosystems

- Two parties, Alice and Bob, wish to communicate in the presence of a malevolent eavesdropper Eve.



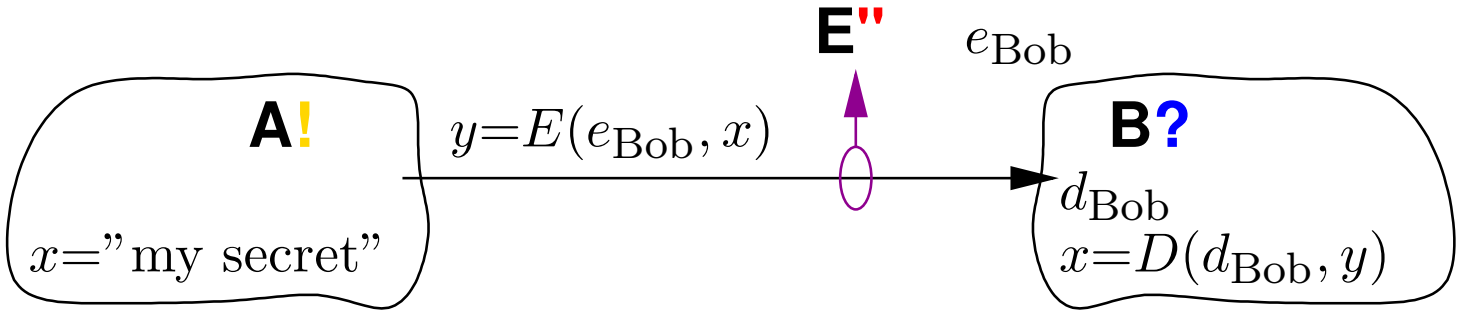


- Alice and Bob agree on two algorithms E (encoding) and D (decoding) which are assumed to be known to general public.
- Confidentiality is ensured by two strings $e, d \in \Sigma^*$ (here $\Sigma = \{0,1\}$), the *encoding* and *decoding key*, respectively.
- If Alice wants to send a message $x \in \Sigma^*$ to Bob, she computes the encrypted message $y = E(e,x)$ and transmits this to Bob over the unreliable channel.
- Bob receives y and computes $D(d,y) = x$. (e and d have been carefully selected to make D a left inverse of E.)
- E and D should be *polynomial-time algorithms*.
- There should be *no way for the eavesdropper* to compute x or d from y and e.



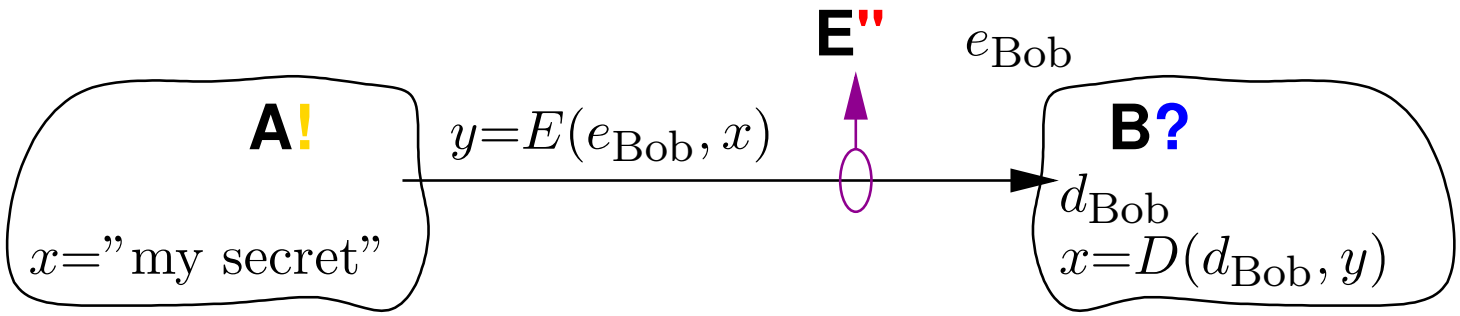
- A simple solution: *one-time pad*
  - Choose d and e to be the same arbitrary string e of length $|x|$.
  - Let both $E(e,x) = e \oplus x$ and $D(e,y) = e \oplus y$ (the exclusive or of the corresponding strings, i.e. the ith bit of $D(e,y)$ is one iff exactly one of e_i or y_i is.)
  - Now $e \oplus (e \oplus x) = x$ and hence $D(e,E(e,x)) = x$
  - Furthermore, if the eavesdropper could derive x from y, then she would also know $e = x \oplus y$. (Equivalently, if the eavesdropper does not to know e, then she cannot possibly derive x from y.)
- One-time pads have limited usability:
  1. How to protect the communication agreeing on the key e?
  2. Long keys are needed (as long as the messages).

A public-key cryptosystem



- Bob generates a pair of keys $(d_{\text{Bob}}, e_{\text{Bob}})$ and announces e_{Bob} *openly*.
  (Key d_{Bob} is private but e_{Bob} is well-known to Alice and the general public.)
- Alice can send a message x to Bob by transmitting $E(e_{\text{Bob}}, x)$.
  (Key e_{Bob} is publicly known.)
- Bob can decode the message by $D(d_{\text{Bob}}, E(e_{\text{Bob}}, x)) = x$.
- It should be computationally infeasible to deduce
  - d_{Bob} from e_{Bob}, and
  - x from $y = E(e_{\text{Bob}}, x)$ without knowing d_{Bob}.

A public-key cryptosystem



- The difficulty of compromising a public-key cryptosystem rests with the difficulty of guessing x from y (and e).
- Once we have a correct guess x, it can be checked simply by testing $E(e_{\text{Bob}}, x) = y$.
- Since x cannot be more than polynomially longer than y, **compromising a public-key cryptosystem** (given y, find an x such that $E(e_{\text{Bob}}, x) = y$) **is a problem in FNP**.[1]

[1] **FNP**, or "function-**NP**" is the class of polynomially length-bounded partial functions f such that for any given x and y, the condition "$f(x) = y$" can *checked* in polynomial time. The family of functions *computable* in polynomial time is denoted **FP**.

One-Way Functions

- Secure public-key cryptosystems can exist only if $\mathbf{P} \neq \mathbf{NP}$.
- Even if we assume $\mathbf{P} \neq \mathbf{NP}$, the existence of a secure public-key cryptosystem is not immediate but what is needed is a *one-way function* (for which the inverse is in $\mathbf{FNP} \setminus \mathbf{FP}$).

Definition (One-way functions)

Let f be a function from strings to strings. We say that f is a one-way function if the following holds:
(i) f is one-to-one and for all $x \in \Sigma^*$, $|x|^{\frac{1}{k}} \leq |f(x)| \leq |x|^k$ (for some k).
(ii) f is in $\mathbf{FP}$.
(iii) The inverse f^{-1} of f is not in $\mathbf{FP}$.

- Notice that if (i) and (ii) hold, then f^{-1} is in $\mathbf{FNP}$.
  ($x = f^{-1}(y)$ if $f(x) = y$ which can be checked in polynomial time).

Candidates for one-way functions

- *Integer multiplication:*

$$f_{\text{MULT}}(p, C(p), q, C(q)) = pq,$$

where $p < q$ are prime numbers and $C(p), C(q)$ are their primality certificates.
(i) f_{MULT} is one-to-one, (ii) polynomial-time computable and (iii) we know of no polynomial-time algorithm which inverts f_{MULT}, i.e., factors products of large primes.

- *Exponentiation modulo a prime:*

$$f_{\text{EXP}}(p, C(p), r, x) = (p, C(p), r^x \bmod p),$$

where p is a prime number, $C(p)$ is its primality certificate, r is a primitive root modulo p ($r^{p-1} = 1 \bmod p$) and integer $x < p$.
No polynomial-time algorithm inverting f_{EXP} is known (the *discrete logarithm problem*).

- *The RSA function:*

$$f_{\text{RSA}}(x, e, p, C(p), q, C(q)) = (x^e \bmod pq, pq, e)$$

  where $p < q$ are prime numbers, $C(p), C(q)$ are their primality certificates, e is a relative prime to $\phi(pq) = (p-1)(q-1)$ (Euler's totient function) and integer $x < pq$.

- f_{RSA} is one-to-one.
- f_{RSA} is polynomial-time computable.
- No polynomial-time algorithm for inverting f_{RSA} has been announced.

The RSA Public-key Cryptosystem

- The RSA key is (p, q, d, e) where $d = e^{-1} \bmod \phi(pq)$, i.e. $ed = 1 + k\phi(pq)$ for some k. Bob's public key is (pq, e) and Bob's private key is (p, q, d). Given p, q and e one can compute d by extended Euclid's algorithm.
- Alice encrypts a message x by

$$y = x^e \bmod pq$$

- Bob decrypts a message y by

$$y^d = x^{ed} = x^{1+k\phi(pq)} = x(x^{\phi(pq)})^k = x \bmod pq,$$

  where $x^{\phi(pq)} = 1 \bmod pq$ by an extension of Fermat's theorem.
- If one can invert f_{MULT} then one can invert f_{RSA}. In particular, any algorithm that factors integers can be used to invert f_{RSA}: if one knows p and q, then one can compute $\phi(pq) = (p-1)(q-1)$ and from it and e then recover d by Euclid's algorithm.

Cryptography and Complexity

- Linking the existence of one-way functions and **NP**-completeness?
- **UP** is a class closer to one-way functions:

Definition (The class UP)

A nondeterministic Turing machine is called *unambiguous* if for any input x there is at most one accepting computation. **UP** is the class of languages accepted by unambiguous polynomial-time nondeterministic Turing machines.

- $\mathbf{P} \subseteq \mathbf{UP} \subseteq \mathbf{NP}$.

Theorem (Ko 1985, Grollman & Selman 1988)

$\mathbf{UP} = \mathbf{P}$ *iff there are no one-way functions.*

- Most likely $\mathbf{P} \neq \mathbf{UP}$ and $\mathbf{UP} \neq \mathbf{NP}$.
  ☞ $\mathbf{NP}$-completeness is not useful in identifying one-way functions.
- $\mathbf{UP}$-completeness does not seem to be useful either:
  $\mathbf{UP}$ is a semantical class with no known complete problems.
- Moreover, complexity theory does not seem to be the right tool for analysing the *security* of cryptosystems because it is based on worst-case performance estimates:

  Even if we could show that compromising a cryptosystem is a hard computational problem in the worst case, this is not enough for the security of a cryptosystem.

Security considerations

- For security it is not enough that decoding is hard in the worst case – it is unacceptable if an eavesdropper can easily decode half (or any reasonable fraction) of the possible messages easily.
- For the definition of a one-way function one would need to replace condition "(iii) f^{-1} not in $\mathbf{FP}$" by a stronger requirement:

  (iii') There is no polynomial-time algorithm for inverting f on a polynomial fraction of the inputs of length n.
- Actually even this condition is not strong enough, because it assumes that the inverting method is deterministic. Also randomised algorithms should be considered, and even non-uniform families of circuits. (In practice, an attack on a cryptosystem could focus only on the currently used key size and invest massive amounts of computation for constructing a circuit that works specifically for this key size.)

Trapdoor functions

- Moreover, not all one-way functions are usable for cryptographic purposes.
- In addition to properties (i–iii) in the definition of a one-way function a couple of further requirements need to be satisfied:

  (iv) One can sample the domain of the function f efficiently (find efficiently arguments for which the function is "defined").

  (v) There is a polynomially computable function d of the input of f that makes the inversion problem computationally easy.

- One-way functions satisfying the two additional requirements are called *trapdoor functions*.
- f_{RSA} is a trapdoor function – supposing that indeed condition (iii) holds and f_{RSA}^{-1} is not in $\mathbf{FP}$.

Randomised Cryptography

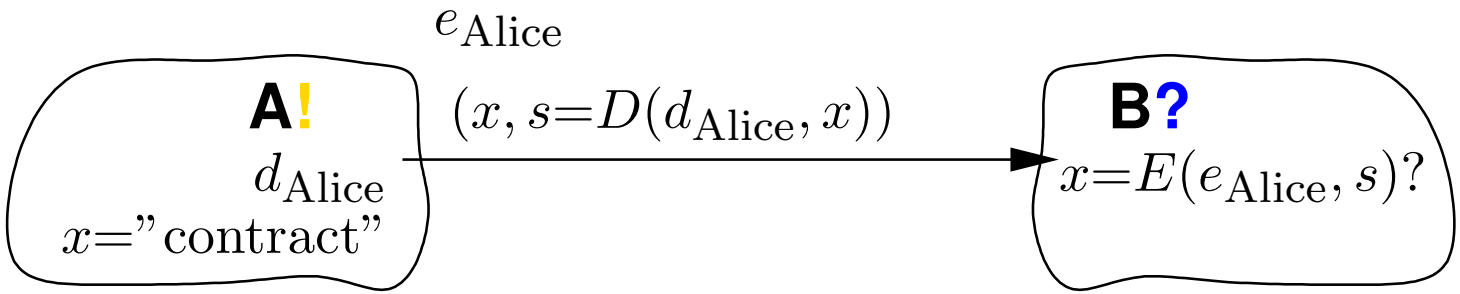
- Doubts on security remain even if very strong one-way functions are used.
- For example, even if f can be inverted only on a few strings, this could be a serious threat if the strings are important ones ("ATTACK NOW", "SELL ALL STOCKS").
- An important case is to be able to send a single confidential bit $b \in \{0,1\}$.
  E.g., in RSA the encoding of a bit b is $b^e = b$, and the encrypted message would be the same as the original one!
- A remedy: *randomised public-key cryptosystem*
  - For sending a bit b, generate a random integer $x \leq \frac{pq}{2}$ and then transmit $y = (2x+b)^e \bmod pq$.
  - After receiving, decode the original message $(2x+b)$; b can then be read as the least significant bit of the decrypted integer.
  - A longer message can be broken into a sequence of bits which can be sent individually as above.

Cryptographic Protocols

- Protocol: a set of interacting computations sharing input and output, aimed at achieving some common goal or consensus state.
- Following the protocol desired to be easy; breaking it desired to be hard.
- Protocols and computational complexity are closely related.

Signatures

- *Problem:* Alice wants to send Bob a signed document x, i.e., a signed message $\mathrm{S}_{\mathrm{Alice}}(x)$ that contains x and identifies unmistakably the sender.
- *A solution:* Use a public-key cryptosystem.
  Alice has public and private keys: $e_{\mathrm{Alice}}, d_{\mathrm{Alice}}$.
  1. Alice sends: $\mathrm{S}_{\mathrm{Alice}}(x) = (x, D(d_{\mathrm{Alice}}, x))$
  2. Bob takes the second part and "decodes" it:
     $E(e_{\mathrm{Alice}}, D(d_{\mathrm{Alice}}, x)) = D(d_{\mathrm{Alice}}, E(e_{\mathrm{Alice}}, x)) = x$
     (this works for commutative systems like RSA).
  3. If decoding equals to the first part (x), Bob accepts x.



- Not very efficient or secure. Altenatively, one can apply a cryptographic hash function and sign the hashed value.

Mental Poker

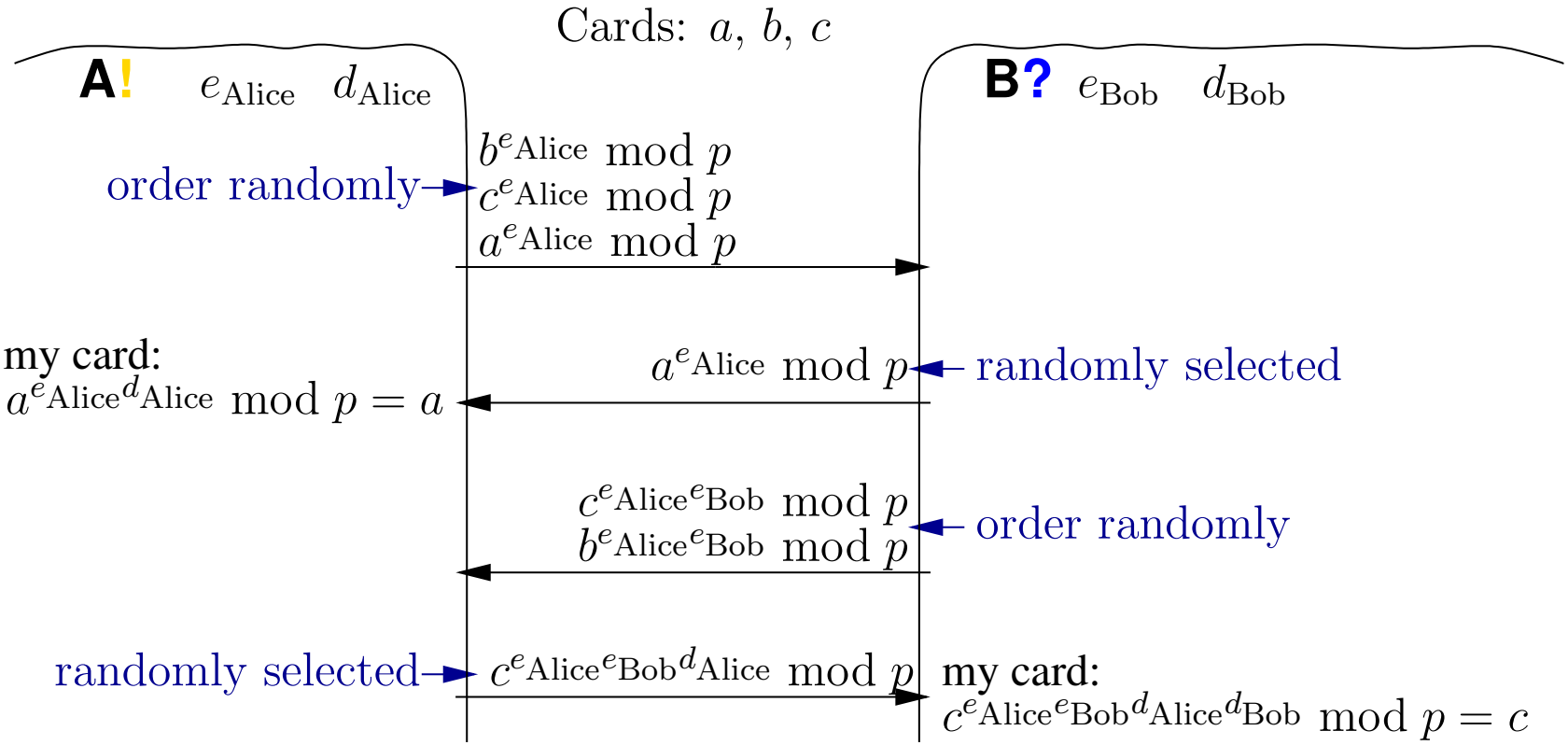
- *Problem:* Alice and Bob have agreed upon three n-bit numbers $a < b < c$ (cards). They need to randomly choose one card each such that:
  1. Their cards are different.
  2. All six pairs of distinct cards are equiprobable as outcomes.
  3. Alice's card is known to Alice but not to Bob and similarly for Bob.
  4. The outcome is indisputable.
- *A solution:* The players agree on a single large prime p.
  Each player has two *secret* keys: $e_{\text{Alice}}, d_{\text{Alice}}, e_{\text{Bob}}, d_{\text{Bob}}$ such that $e_{\text{Alice}} d_{\text{Alice}} = e_{\text{Bob}} d_{\text{Bob}} = 1 \bmod (p-1)$.
  Now, e.g., $e_{\text{Alice}} d_{\text{Alice}} = k(p-1)+1$ and hence,
  $x^{e_{\text{Alice}} d_{\text{Alice}}} = x(x^{p-1})^k = x \bmod p$ (by Fermat's theorem)

Mental Poker: the protocol

- Alice encrypts the three cards and sends to Bob the encrypted messages $a^{e_{\text{Alice}}} \bmod p, b^{e_{\text{Alice}}} \bmod p, c^{e_{\text{Alice}}} \bmod p$ in some random order.
- Bob picks one, say $a^{e_{\text{Alice}}} \bmod p$, and sends it to Alice who decrypts it with d_{Alice} and keeps as her card.
- Bob encrypts the two remaining cards $b^{e_{\text{Alice}} e_{\text{Bob}}} \bmod p, c^{e_{\text{Alice}} e_{\text{Bob}}} \bmod p$ and sends them to Alice in some random order.
- Alice picks one, say $b^{e_{\text{Alice}} e_{\text{Bob}}} \bmod p$, decodes it with d_{Alice} and sends $b^{e_{\text{Alice}} e_{\text{Bob}} d_{\text{Alice}}} \bmod p$ to Bob.
- Bob decrypts this with d_{Bob} and takes as his card.

☞ Conditions (i–iv) satisfied.

Mental Poker: the protocol



Interactive Proofs

- A nondeterministic algorithm can be seen as a simple protocol: Alice has exponential computing power (to find a good certificate) and Bob has polynomial (to check the certificate).
- What languages can be accepted if Bob can use randomisation?

Definition

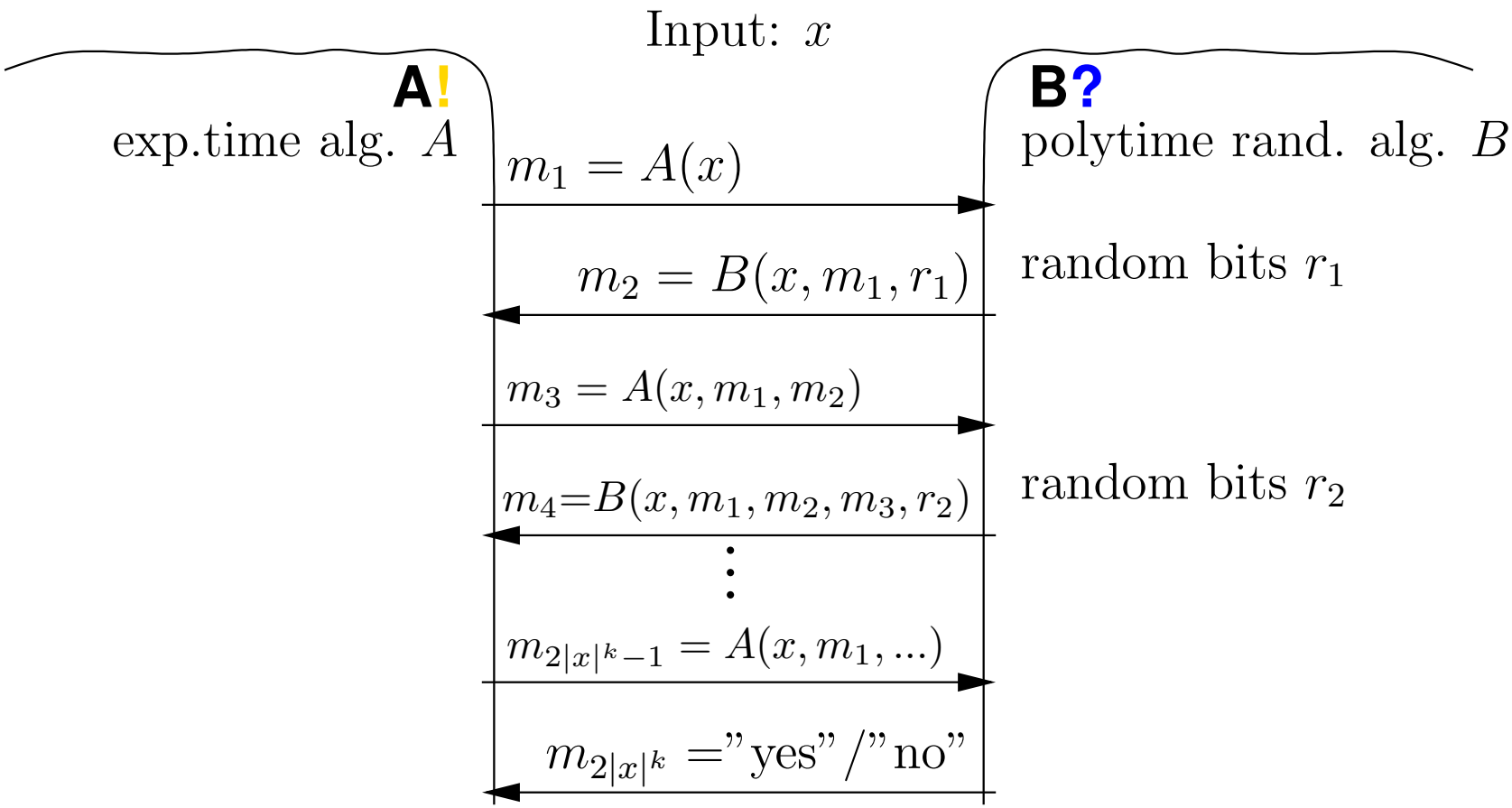
An *interactive proof system* (A, B) is a protocol between Alice and Bob. Alice runs an exponential time algorithm A while Bob has a polynomial-time randomised algorithm B.
The input x is known to both algorithms.
The two exchange a sequence of messages $m_1, m_2, \ldots, m_{2|x|^k}$ where Alice sends the odd-numbered ones and Bob even-numbered and $|m_i| \leq |x|^k$ for all i.

The protocol

- $m_1 = A(x)$.
- For all $i \leq 2|x|^k$, $m_{2i} = B(x; m_1; \ldots; m_{2i-1}, r_i)$ and $m_{2i+1} = A(x; m_1; \ldots; m_{2i})$ where r_i is the polynomially long random string used by Bob at the ith stage (r_i is not known to A).
- For the last message $m_{2|x|^k} \in \{\text{“yes”}, \text{“no”}\}$ signalling accept/reject.
- (A, B) *decides* a language L iff for all strings x,
  - if $x \in L$, then the probability that x is accepted by (A, B) is at least $1 - \frac{1}{2^{|x|}}$.
  - if $x \notin L$, then the probability that x is accepted by (A', B) is at most $\frac{1}{2^{|x|}}$ for any exponential algorithm A' replacing A.

IP is the class of languages decided by interactive proof systems.

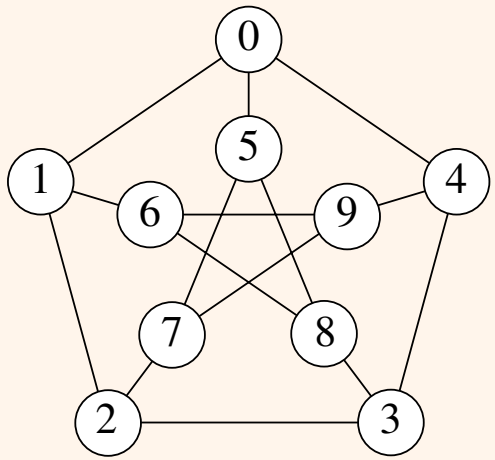
The protocol illustrated

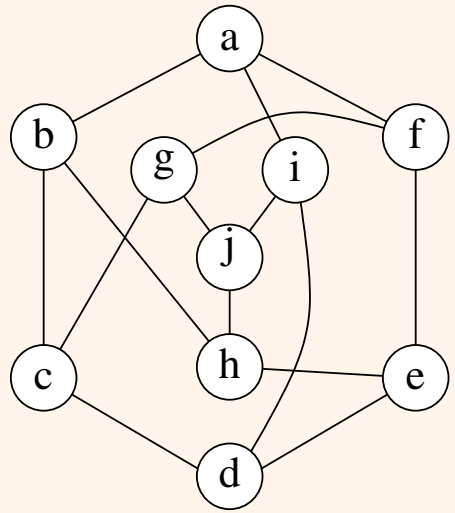


- $\mathbf{NP} \subseteq \mathbf{IP}$
- $\mathbf{BPP} \subseteq \mathbf{IP}$
- GRAPH ISOMORPHISM $\in \mathbf{NP}$ (not known to be $\mathbf{NP}$-complete or in $\mathbf{P}$)
- GRAPH NONISOMORPHISM $\in \mathbf{IP}$ (not known to be in $\mathbf{NP}/\mathbf{BPP}$)

Example

Are these two graphs isomorphic?





An interactive proof system deciding GRAPH NONISOMORPHISM

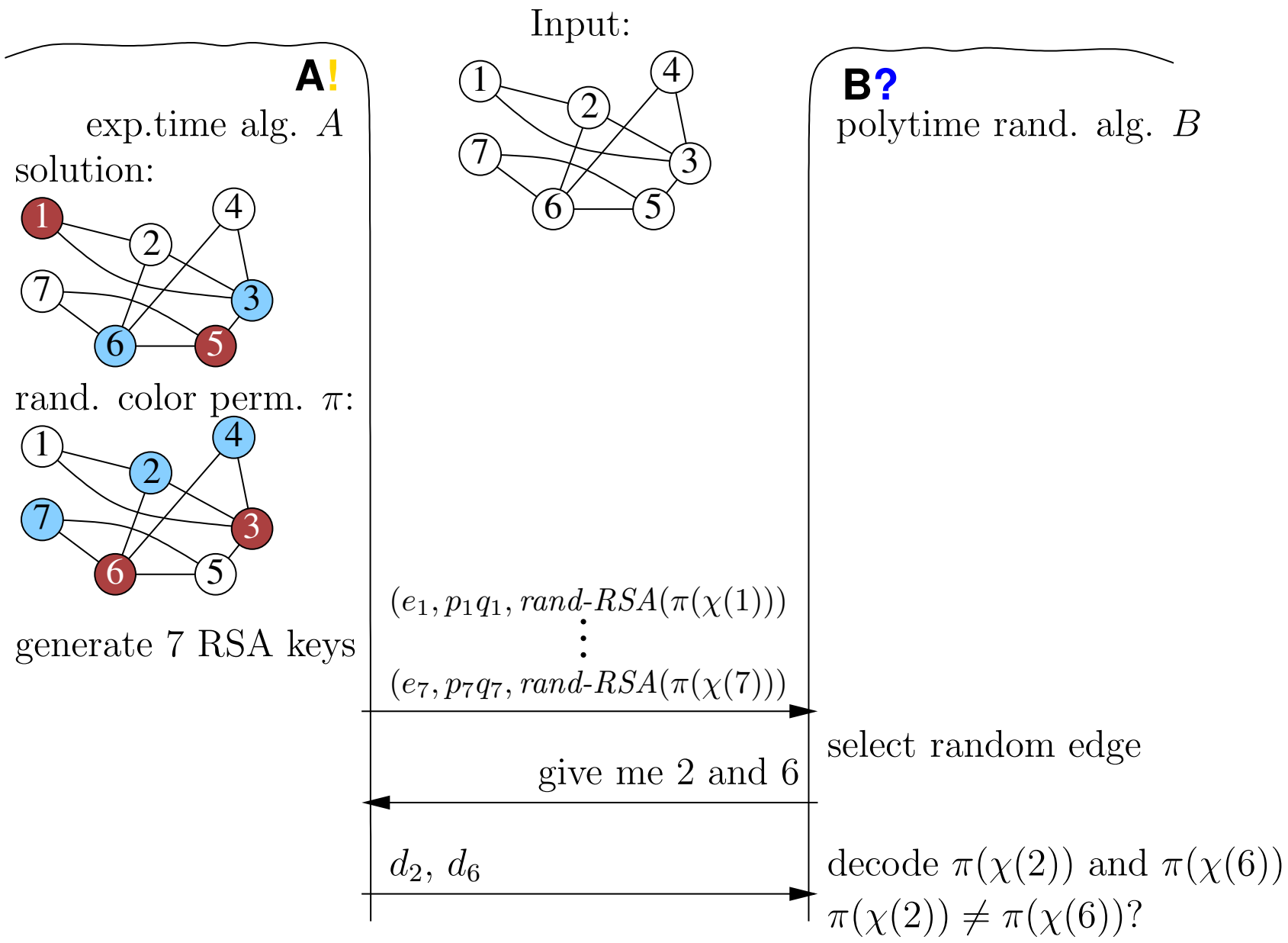
- Bob: on input $x = (G, G')$ repeats for $i = 1, \ldots, |x|$ rounds: Chooses random bit b_i and if $b_i = 1$ then $G_i = G$ else $G_i = G'$; generates a random permutation π_i and sends $m_{2i-1} = (G, \pi_i(G_i))$.
- Alice: checks whether the two graphs received are isomorphic. If they are then sends $m_{2i} = 1$ else $m_{2i} = 0$.
- After $|x|$ rounds Bob accepts if random bits $b_1, \ldots, b_{|x|}$ and Alice's replies $m_2, \ldots, m_{2|x|}$ are identical.
- If G and G' are not isomorphic, then x is accepted since $b_i = 1$ iff m_{2i-1} contains two isomorphic graphs iff $m_{2i} = 1$.
- If G and G' are isomorphic, then Alice always only sees a graph and its permuted copy, and hence needs to guess a random bit correctly $|x|$ times: perfect success in this has probability $\frac{1}{2^{|x|}}$.

Zero Knowledge

- *Challenge:* Design an interactive protocol such that given a graph (V,E), in the end Bob is convinced that with very high probability Alice has a legal 3-colouring of the graph but Bob has no clue about the actual 3-colouring. Such a protocol is called a *zero-knowledge proof*.
- *Protocol:* Suppose Alice's colouring is $\chi : V \mapsto \{00,11,01\}$. Then in each round:
  1. Alice generates a random permutation π of the colours; generates for each vertex i an RSA public-private key (p_i,q_i,e_i,d_i) and computes (y_i,y_i'), a randomised RSA coding of $\pi(\chi(i))$; and then reveals (e_i,p_iq_i,y_i,y_i') for each vertex $i \in V$ to Bob.
  2. Bob picks at random an edge $\{i,j\} \in E$ and Alice reveals d_i,d_j.
  3. Bob decodes y_i,y_i' and y_j,y_j' to obtain colours $\pi(\chi(i))$ and $\pi(\chi(j))$ and checks that they are different.

- If Alice has no legal colouring, Bob has at least $\frac{1}{|E|}$ probability of finding an edge $\{i,j\} \in E$ such that $\chi(i) = \chi(j)$.
- If this is repeated $k|E|$ times, the probability for Bob to find out that Alice has no legal colouring is at least $1 - e^{-k}$.
- Note: Bob has not learned anything about the actual colouring. Bob sees random public keys, encryptions of colours, and colours $\pi(\chi(i))$ and $\pi(\chi(j))$ but these are randomly chosen pairs of different colours. (The permutation π changes at each round.)
- Zero knowledge: The interactions in the protocol form a random string drawn from a distribution that was available to Bob in the beginning.
- Assuming one-way functions exist, *all* problems in $\mathbf{NP}$ have zero-knowledge proofs (Goldreich, Micali & Wigderson 1991).

One round illustrated



Learning Objectives

- The concepts of public-key cryptosystems, one-way functions and their relationships.
- The role of complexity theory in analysing the security of cryptosystems.
- Examples of cryptographic protocols: signatures, mental poker, interactive proofs, zero knowledge.