Modern Cryptography: Lecture 11

Public Key Encryption I/II

Daniel Slamanig

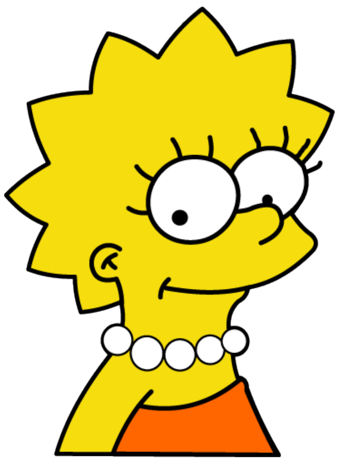
AUSTRIAN INSTITUTE OF TECHNOLOGY

Organizational

- Where to find the slides and homework?
  - https://danielslamanig.info/ModernCrypto19
- How to contact me?
  - daniel.slamanig@ait.ac.at
- Tutors: Guillermo Perez, Karen Klein
  - guillermo.pascualperez@ist.ac.at; karen.klein@ist.ac.at
- Official page at TU, Location etc.
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=3463&dsrid=417&courseNr=192062&semester=2019W
- Tutorial, TU site
  - https://tiss.tuwien.ac.at/course/courseDetails.xhtml?dswid=3593&dsrid=246&courseNr=192063
- Exam for the second part: Thursday 30.01.2020 15:00-17:00 (Tutorial slot)

Overview Public Key Encryption

$m := Dec_{skA}(c)$



secret key $\boldsymbol{sk_A}$
public key $\boldsymbol{pk_A}$

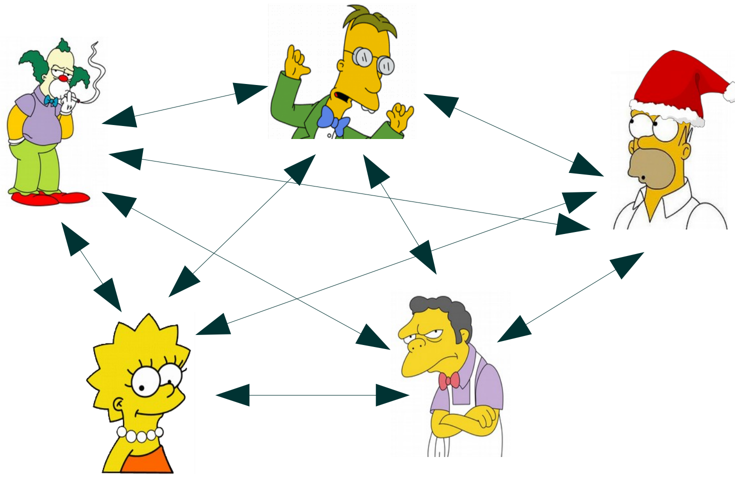
$c := Enc_{pkA}(m)$

Insecure channel

public key $\boldsymbol{pk_A}$

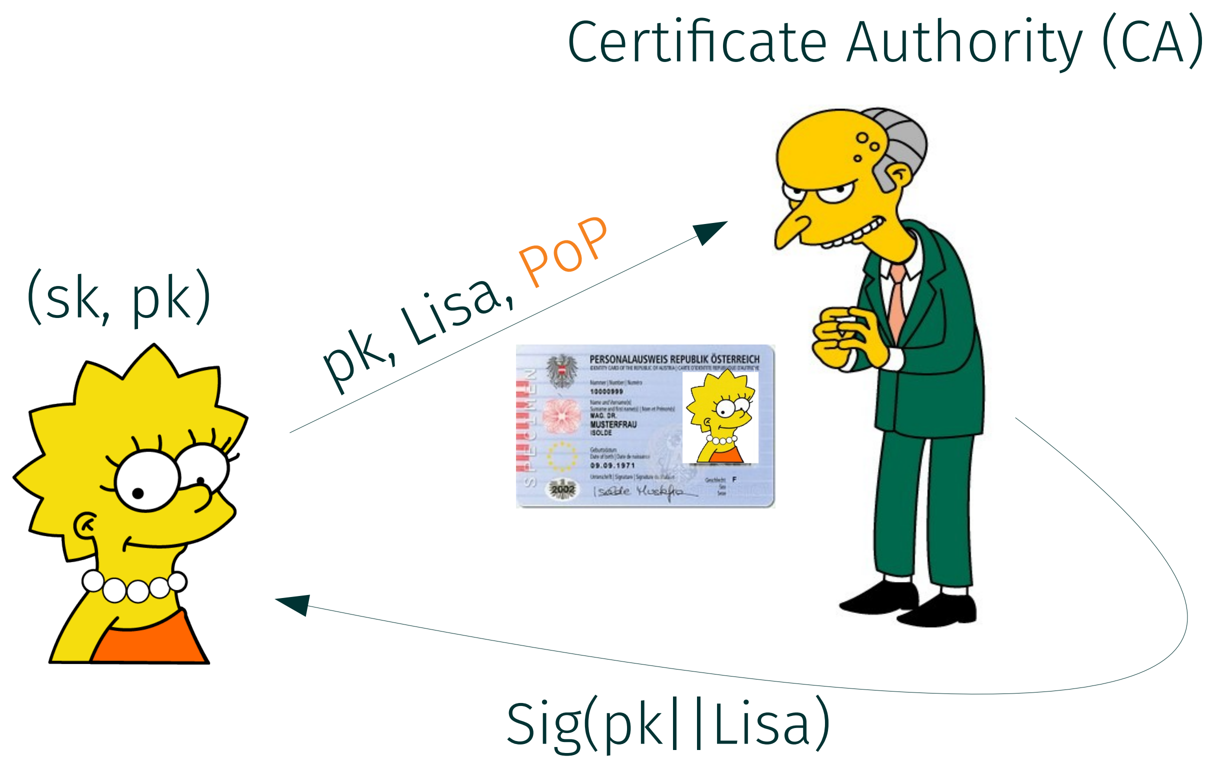
A: $\boldsymbol{pk_A}$

Overview Public Key Encryption



- Now every user has a secret key **sk** and a public key **pk** (secret key **sk** cannot be efficiently computed from **pk**)
- Reduced effort for key management; no shared secret!
- Authentic copy of **pk** can be made public
- How to guarantee that public keys are authentic in practice?
  - Public keys look "random" and no relation to identiy of the holder exists - so binding must be done explicitly
  - Let some trusted entity (CA) explicitly "certify" the connection between **ID** and **pk**
  - Later in the course we will then see an alternative approach
    - public key = identity (identity-based encryption)
    - But setting is different

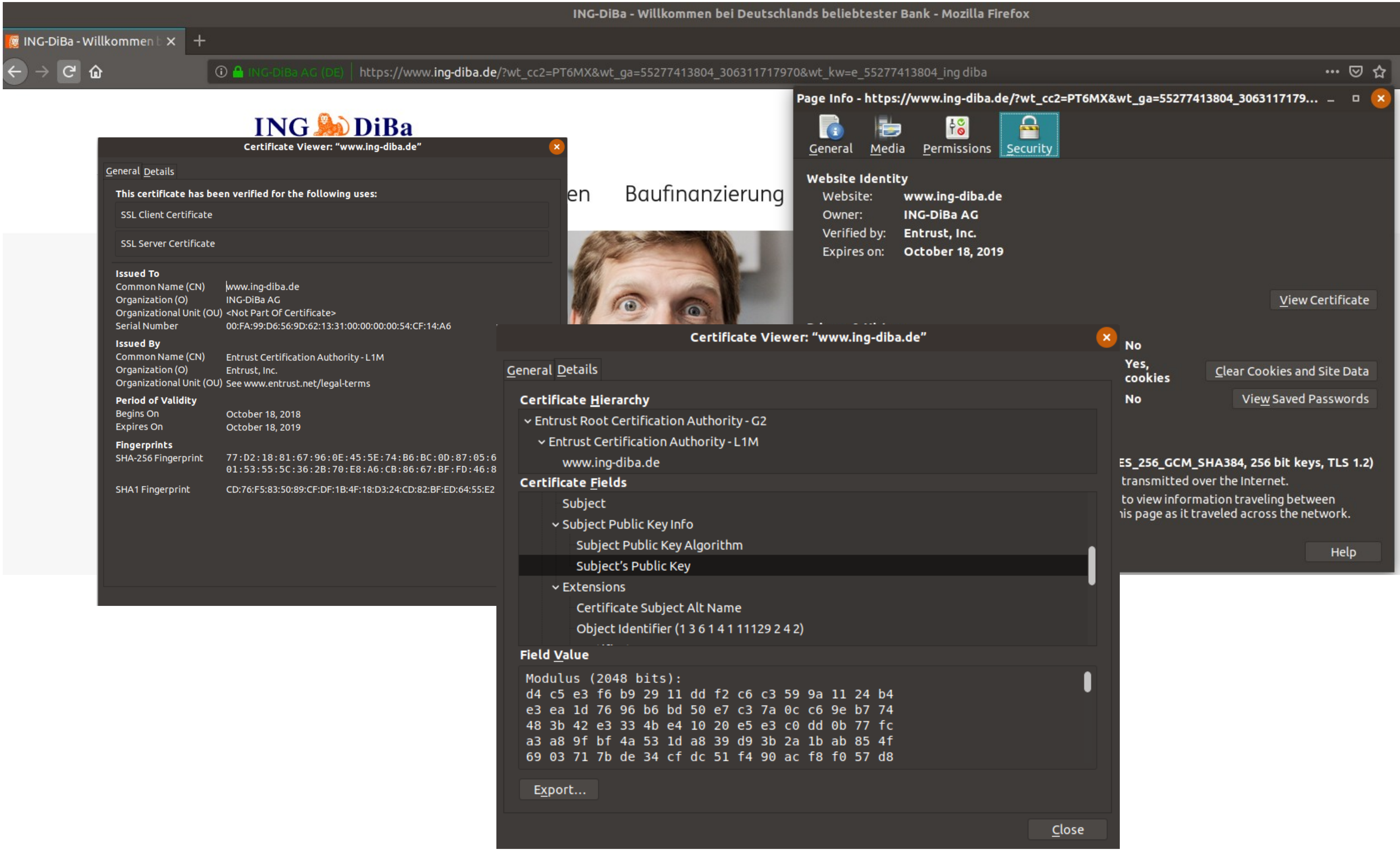
Certifying Public Keys



- Demonstrate that you hold sk for pk
  - Proof of Possession (PoP)
- CA certifies pk||ID
  - ID: mail, domain, etc.
- CA is trusted to operate properly (PKI model)
  - CA is “self-certified”

- Alternative models
  - Web of trust (e.g., PGP)
  - Decentralized PKI (DPKI)
    - “Self Sovereign Identity” (e.g., Sovrin)

Overview Public Key Encryption



Public Key Encryption: Definition

DEFINITION 11.1 A public-key encryption scheme is a triple of PPT algorithms (Gen, Enc, Dec) such that:

1. The **key-generation** algorithm **Gen** takes as input the security parameter 1^n and outputs a pair of keys (pk, sk) (the message space **M** is implicit in the public key).
2. The **encryption** algorithm **Enc** takes as input a public key pk and a message m from some message space. It outputs a ciphertext c, and we write this as $c \leftarrow \mathrm{Enc}_{pk}(m)$. (We often also write $c \leftarrow \mathrm{Enc}(m, pk)$)
3. The deterministic **decryption** algorithm **Dec** takes as input a private key sk and a ciphertext c, and outputs a message m or a special symbol $\perp$ denoting failure. We write this as $m := \mathrm{Dec}_{sk}(c)$. (We often also write $m := \mathrm{Dec}(c, sk)$).

It is required that, except possibly with negligible probability over $(pk, sk) \leftarrow \mathrm{Gen}(1^n)$, we have

$$\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(m)) = m$$

for any message $m \in \mathbf{M}$.

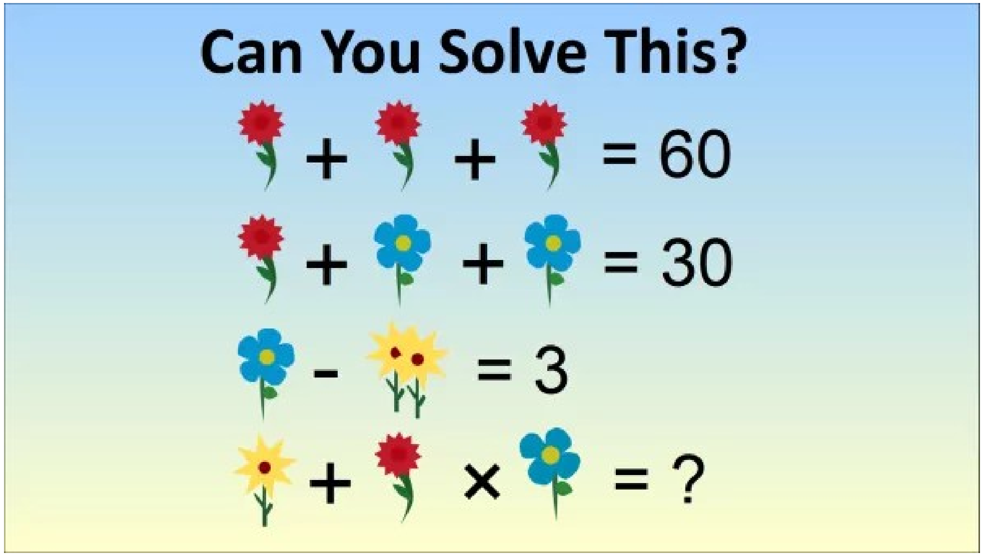
Some Remarks on the Definition

- The encryption algorithm may be deterministic or probabilistic
- The decryption algorithm may be perfectly correct (never fails) or may fail with negligible probability

- Every instance has an associated message space M (which we assume to be implicitly defined when seeing the public key)
  - In the simplest case we encrypt bits
    - it is easy to extend such a scheme to bitstrings $\{0,1\}^k$
  - Usually M represents some algebraic structure which does not contain all bitstrings of some fixed size
    - typically we have efficient ways to injectively encode messages from $\{0,1\}^k$ into elements from M

Constructing Public Key Encryption

- Need some hard problems to rely on!



- Will look into constructions from factoring-related problems
  - RSA in particular
- Will look at constructions from DL-related problems (next lecture)
  - We already have discussed DDH and CDH

Factoring

- Every integer $N>1$ can be uniquely (up to ordering) written as $N=\prod_i p_i^{e_i}$
  - p_i are distinct primes and $e_i \geq 1$ for all i
- Given a factorization it is easy to compute the composite N

- Computing the factorization is hard for certain forms of composites
  - Hardest if numbers to factor have only large prime factors

- A trivial algorithm to find the factors of any given N is trival division
  - Inefficient as it represents an exponential-time algorithm

Factoring

- Two types of algorithms
  - Generic ones: apply to arbitrary N
  - Specific ones: tailored to work for N of some specific form

- **Specific algorithm**
  - Pollard's p– 1 method: Factor N=pq when p-1 has small prime factors
    - Choosing uniform n-bit primes p,q, small prime factors of p-1 and q-1 are very unlikely
- **General purpose algorithms**
  - Pollard's rho method: $O(N^{1/4} \cdot \text{polylog}(q))$ runtime (still exponential)
  - Fastest general purpose factoring algorithm is the general number field sieve
    - Subexponential with runtime $2^{O((\log N)^{1/3} \cdot (\log \log N)^{2/3})}$

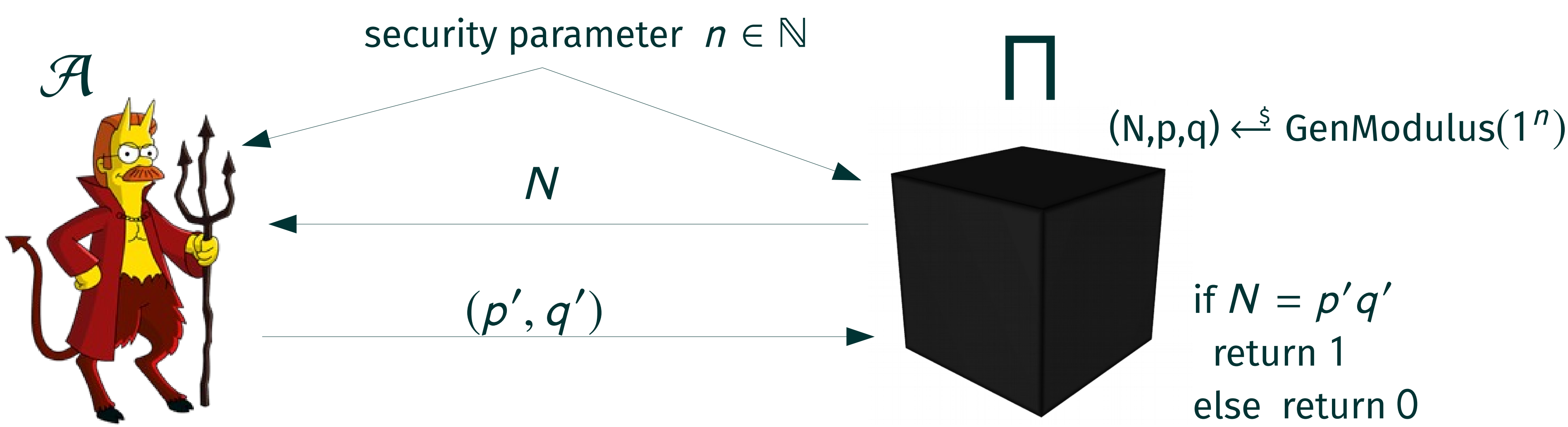
Factoring

- Let **GenModulus** be a polynomial-time algorithm that on input 1^n outputs (N,p,q) where N=pq and p,q are n-bit primes.

DEFINITION 8.45: Factoring is hard relative to GenModulus if for all PPT algorithms A there exists a negligible function such that

$$\Pr[\text{Factoring}_{A,\text{GenModulus}}(n)=1] \leq \text{negl}(n) .$$

$\text{Factor}_{A,\text{GenModulus}}(n)$

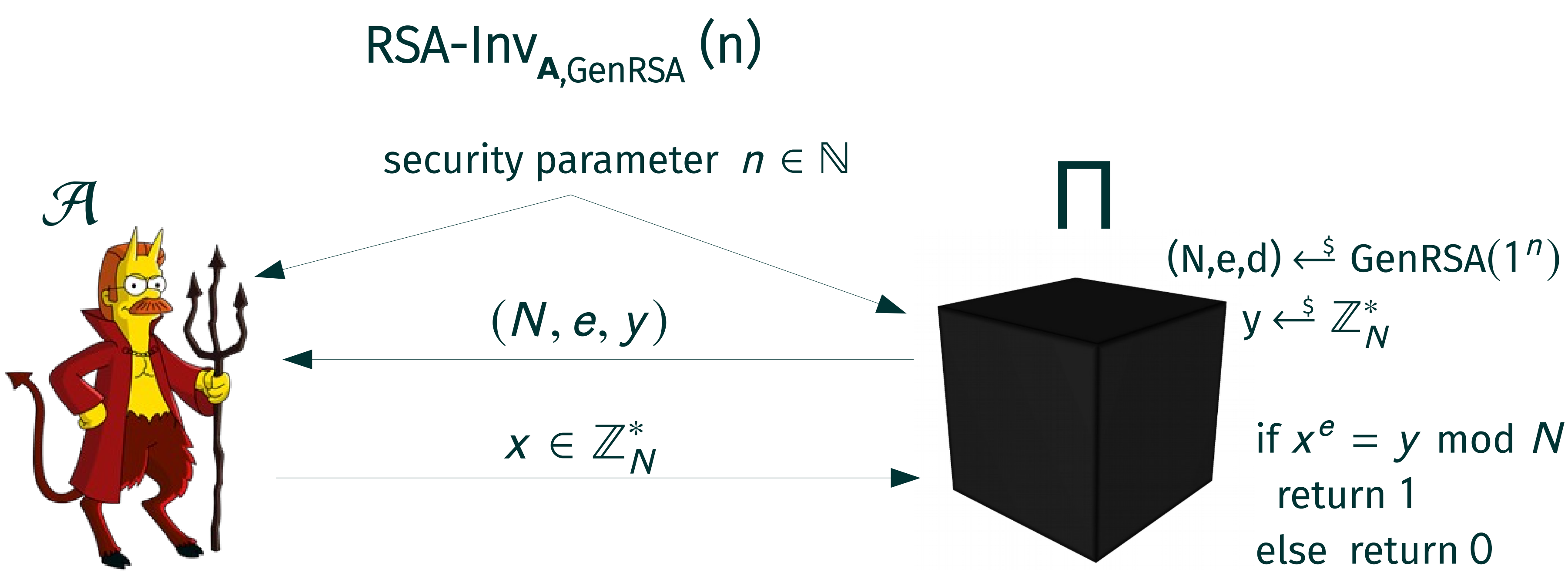


RSA Assumption

- Let **GenRSA** be a polynomial-time algorithm that on input 1^n outputs (N,e,d) where N=pq and p,q are n-bit primes and e,d>0 are integers s.t. gcd(e,φ(N))=1 and ed = 1 mod φ(N).

<u>DEFINITION 8.46:</u> The RSA problem is hard relative to GenRSA if for all PPT algorithms A there exists a negligible function such that

$$\Pr[\text{RSA-Inv}_{A,\text{GenRSA}}(n)=1] \leq \text{negl}(n) .$$

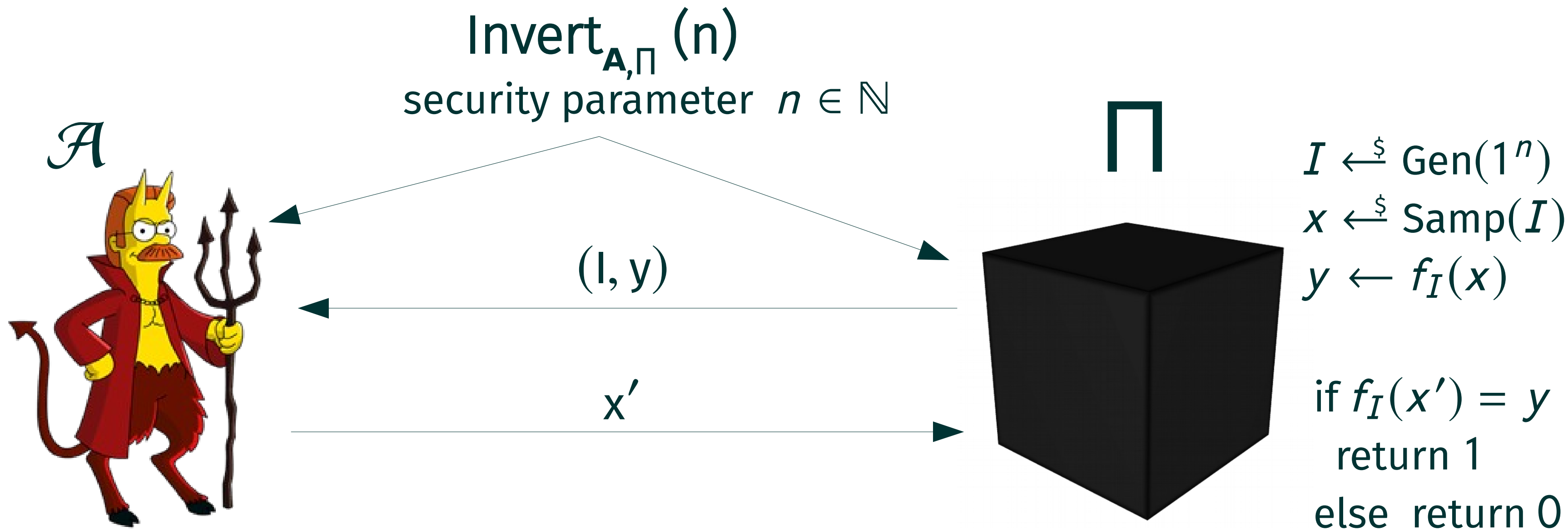


One-Way Permutation (OWP)

- DEFINITION 8.75: A triple Π = (Gen, Samp, f) of PPT algorithms is a family of permutations if the following hold:
  - The **parameter-generation algorithm Gen**, on input 1^n , outputs parameters I with |I| ≥ n. Each value of I defines a set D_I that constitutes the domain and range of a permutation (i.e., bijection) $f_I : D_I \to D_I$.

DEFINITION 8.76: The family of permutations Π = (Gen, Samp, f) is one-way if for all PPT algorithms A there exists a negligible function negl such that

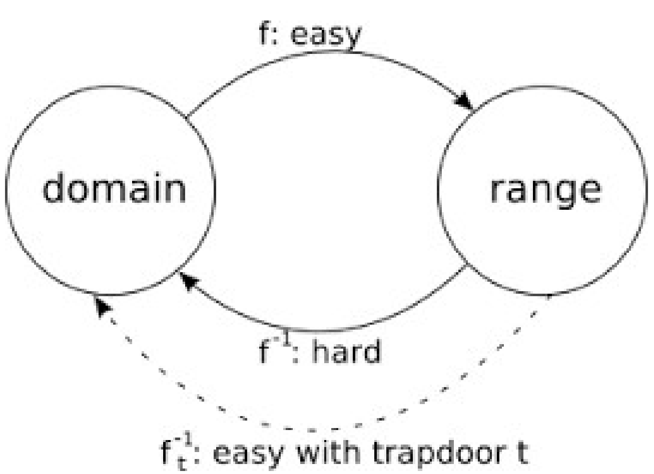
$$\Pr[\mathsf{Invert}_{A,\Pi}(n)=1] \leq \mathsf{negl}(n) .$$



Trapdoor One-Way Permutation

- DEFINITION 13.1: A triple Π = (Gen, Samp, f, Inv) of PPT algorithms is a family of trapdoor permutations if the following hold:
  - The **parameter-generation algorithm Gen**, on input 1^n, outputs parameters (I, td) with $|I| \geq n$. Each value of I defines a set D_I that constitutes the domain and range of a permutation (i.e., bijection) $f_I : D_I \rightarrow D_I$.
  - Let Gen' be Gen that only outputs I. **Then, (Gen',Samp,f) is a family of OWPs**.
  - Let (I, td) be the output of Gen(1^n). The deterministic **inverting algorithm Inv**, on input td and $y \in D_I$, outputs an element $x \in D_I$. We write this as $x := Inv_{td}(y)$. We require that with all but negl. probability over (I, td) output by Gen(1^n) and uniform choice of $x \in D_I$, we have

$$Inv_{td}(f_I(x))=x.$$



One-Way Permutation – Candidates

- RSA Assumption
  - Is it a OWP? Yes, we assume.
- Best currently known way to break RSA assumption is to factor N and then compute e'th roots mod p and q and use CRT to recover the final result
  - RSA Assumption implies Factoring
- Do we need to factor?
  - Computing e'th roots modulo N yields a facotring algorithm? Unknown for $e \geq 3$.
  - Not known to be equivalent to factoring
- Equivalence known for square roots!
  - Not a special case of RSA (2 not coprime to $\varphi(N)$)
  - Rabin cryptosystem (not popular in practice)

Textbook RSA Encryption

- KeyGen(1^n): Pick two random n-bit primes p,q, set N = pq, pick e s.t. gcd(e ,φ(N)) = 1, compute d := e^{-1} mod φ(N) output (sk , pk) := ((d , N), (e ,N))
- Enc (m, pk): On input m ∈ $\mathbb{Z}_N$ and pk = (e , N) , compute and output

$$c := m^e \bmod N$$

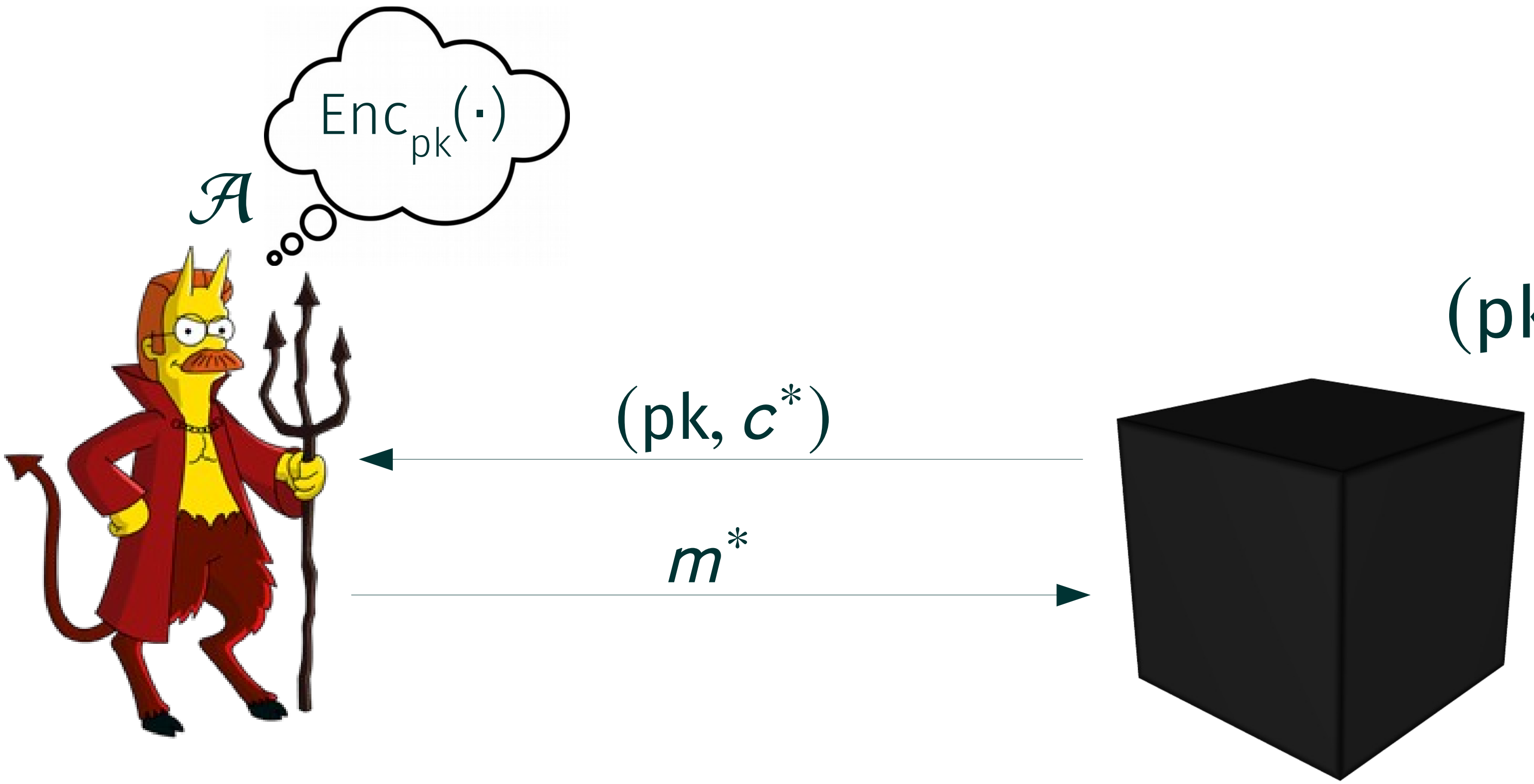
- Dec (c, sk): On input c an d sk = (d , N) , compute and output

$$m := c^d \bmod N$$

We have for all m ∈ $\mathbb{Z}_N$ that $m = (m^e)^d \bmod N$

Proof of correctness of RSA will be done as a HW.

OW-CPA Security



$(pk, sk) \xleftarrow{\$} \mathrm{Gen}(1^n)$
$m \xleftarrow{\$} \mathcal{M}$
$c^* \xleftarrow{\$} \mathrm{Enc}_{pk}(m)$
if $m^* = m$
  return 1
else return 0

A public-key encryption scheme Π = (Gen, Enc, Dec) has one-way encryptions in the presence of an eavesdropper if for all PPT adversaries A there is a negligible function negl s.t.

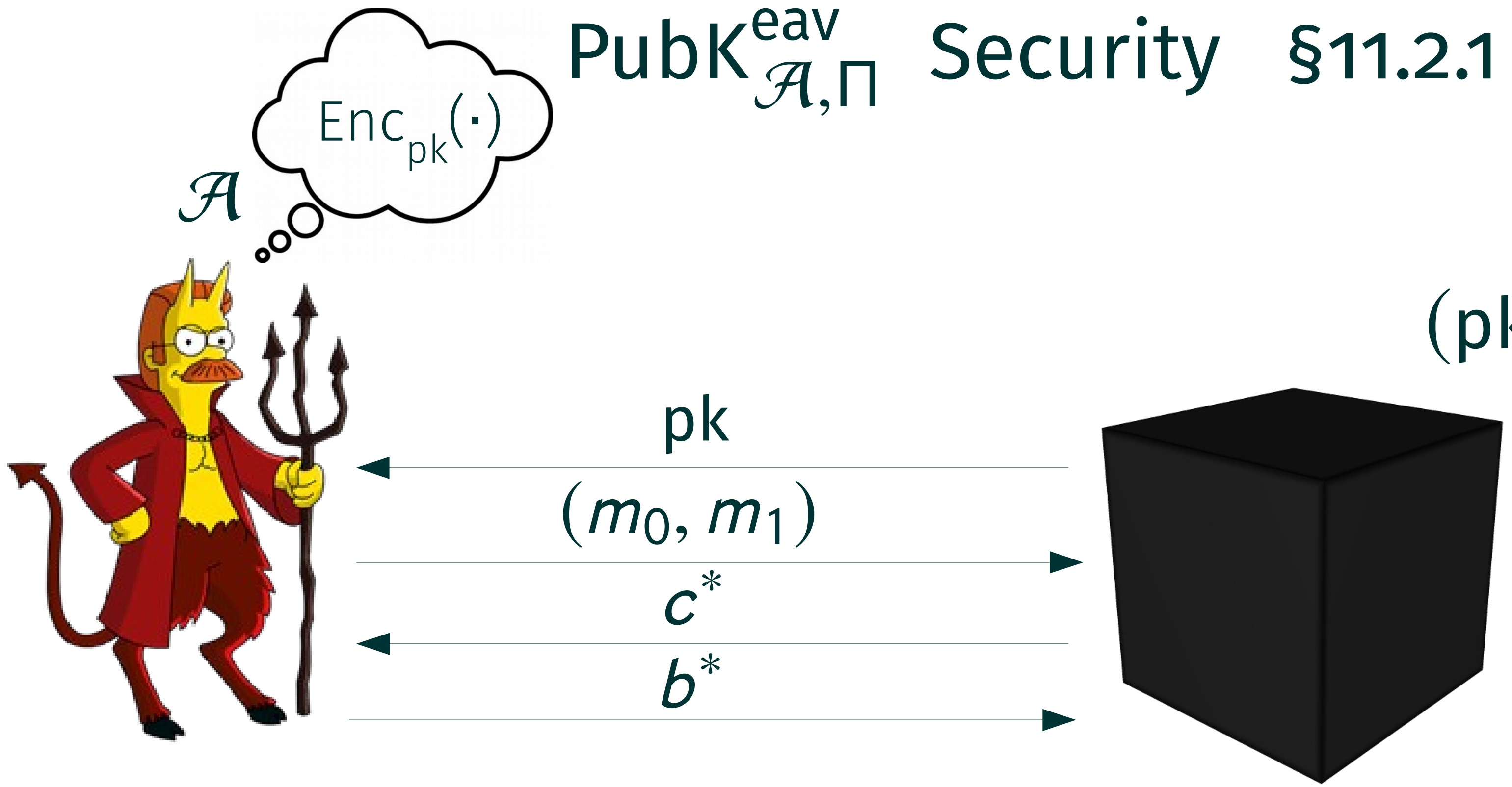
$$\Pr[\mathrm{PubK}^{\text{ow-cpa}}_{A,\Pi}(n)=1] \leq \mathrm{negl}(n)\,.$$

Security of Textbook RSA

- One-way security (OW-CPA) under RSA Assumption
  - Adversary gets public key and encryption of a random message
  - Adversary needs to output the message
- Very weak security guarantees
  - Guarantees only for uniformly random messages
  - Adversary has to reconstruct entire message

- Interesting property: homomorphic PKE
  - Given two ciphertexts c_1 and c_2 under same public key, we can operate on the underlying plaintexts without prior decryption
    - $c_1 = m_1^e \bmod N$, $c_2 = m_2^e \bmod N$: $c_1 c_2 = (m_1 m_2)^e \bmod N$
  - Problem (no CCA secuirty – see next lecture), but also interesting feature (if at least IND-CPA secure)

$\mathrm{PubK}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ Security §11.2.1



$(pk, sk) \xleftarrow{\$} \mathrm{Gen}(1^n)$
$b \xleftarrow{\$} \{0, 1\}$
$c^* \xleftarrow{\$} \mathrm{Enc}_{pk}(m_b)$
if $b^* = b$
  return 1
else return 0

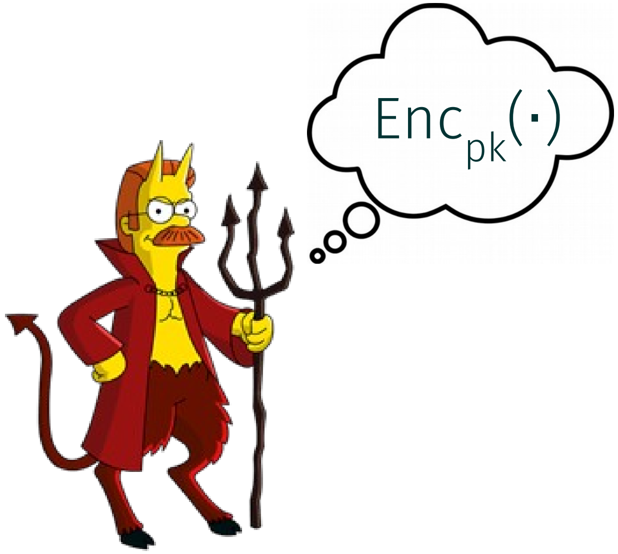
A public-key encryption scheme Π = (Gen, Enc, Dec) has indistinguishable encryptions in the presence of an eavesdropper if for all probabilistic polynomial-time adversaries A there is a negligible function negl s.t.

$$\Pr[\mathrm{PubK}^{\mathrm{eav}}_{A,\Pi}(n)=1] \le ½ + \mathrm{negl}(n) .$$

Some Observations

PROPOSITION 11.3 If a public-key encryption scheme has indistinguishable encryptions in the presence of an eavesdropper, it is IND-CPA-secure.

Analogous for one-wayness

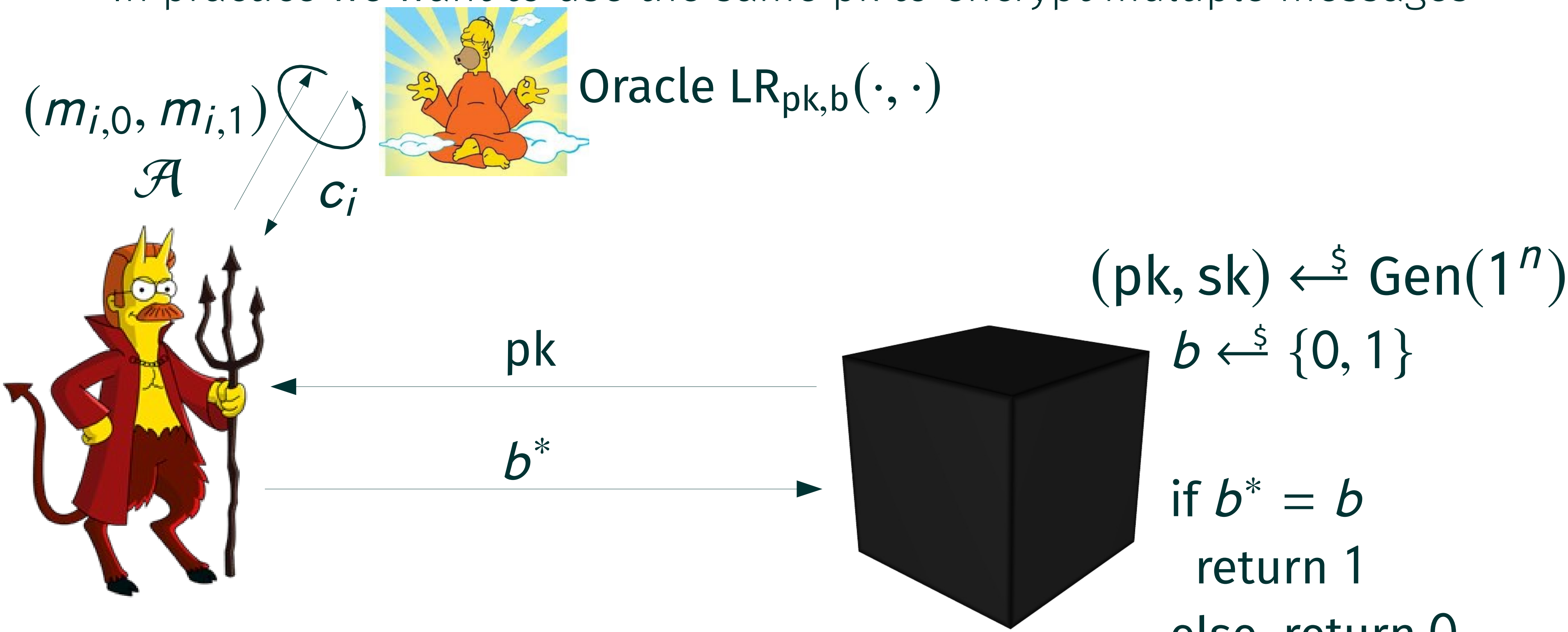


THEOREM: No public-key encryption scheme can be perfectly secret.

THEOREM 11.4 No deterministic public-key encryption scheme is IND-CPA-secure.

Why?

- In practice we want to use the same pk to encrypt multiple messages



THEOREM 11.6 If a public-key encryption scheme Π is IND-CPA-secure, then it also has indistinguishable multiple encryptions.

Proof Idea

- Let us fix a polynomial bound t=poly(n) on the queries to LoR
- We now define a sequence of "intermediate experiments"
  - Let us start in an experiment where LoR has bit b=0
    - Adversary submits $((m_{1,0},m_{1,1}), \ldots, (m_{t,0},m_{t,1}))$ and LoR always return encryptions of $m_{i,0}$
    - Adversary sees $(E_{pk}(m_{1,0}), \ldots, E_{pk}(m_{t,0}))$
  - Let the i'th experiment change the first i positions in the responses to $(E_{pk}(m_{1,1}), \ldots, E_{pk}(m_{i,1}))$
  - After t steps we end up with LoR replying $(E_{pk}(m_{1,1}), \ldots, E_{pk}(m_{t,1}))$ and thus are in the experiment where LoR has bit b=1
- If the probability of distuinguishing the first and the last experiment is neglible, we have proven our claim
- Formally, we use a hybrid argument

$$\underbrace{(E_{pk}(m_{1,0}), \ldots, E_{pk}(m_{t,0})) \approx (E_{pk}(m_{1,1})}_{\text{Reduction to IND-CPA}}, \ldots, E_{pk}(m_{t,0})) \approx \cdots \approx (E_{pk}(m_{1,1}), \ldots, \underbrace{E_{pk}(m_{t,0})) \approx (E_{pk}(m_{1,1}), \ldots, E_{pk}(m_{t,1}))}_{\text{Reduction to IND-CPA}}$$

...

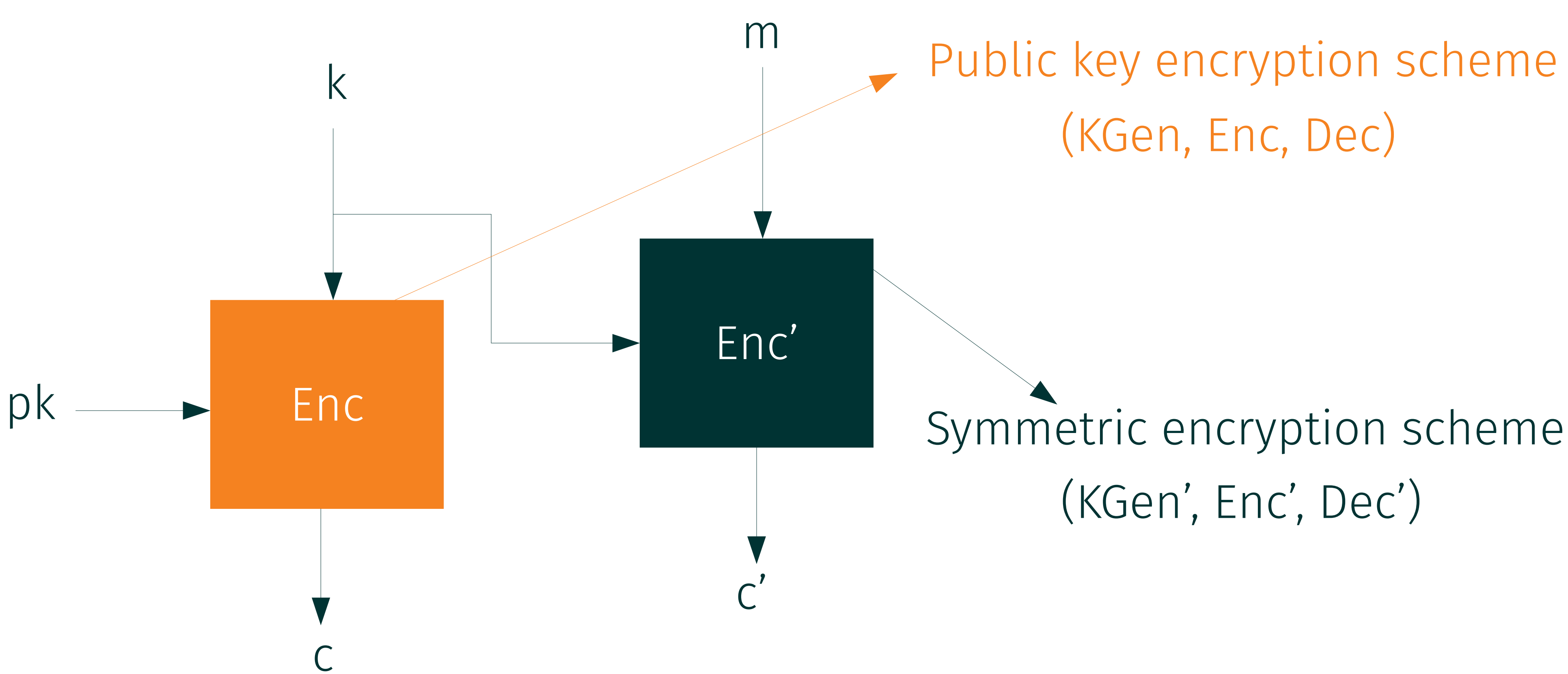
Arbitrary Long Messages

- We can use this fact to construct from any PKE $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ another PKE $\Pi' = (\text{Gen}, \text{Enc}', \text{Dec}')$.
- Assume that Π encrypts messages from $\{0,1\}^m$, then we can construct a scheme for messages of length $\{0,1\}^{m \cdot k}$ for any $k \in \mathbb{N}$
- Encryption simply looks as follows and decryption works the obvious way:
  - $\text{Enc}'_{pk}(m) := \text{Enc}_{pk}(m_1), \ldots, \text{Enc}_{pk}(m_k)$

CLAIM 11.7 Let Π and Π' be as above. If Π is IND-CPA-secure, then so is Π'.

Arbitrary Long Messages in Practice

- The previous method is rather inefficient
- In practice so called "hybrid encryption" is used
  - Formal discussion after the holidays via the KEM/DEM paradigm

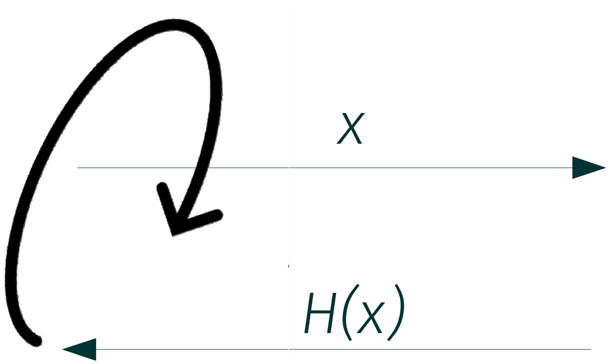


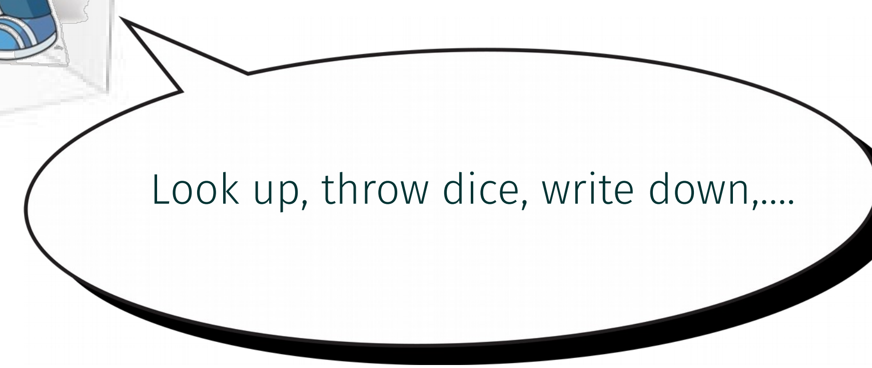
Random Oracle Model (ROM)

Mihir Bellare, Phillip Rogaway

- Function H that can be accessed in a black-box way
  - Answers consistently for values x already seen
  - For new values x, choose random n bit string as answer

Truly random function H: $\{0,1\}^* \rightarrow \{0,1\}^n$





- Do they exist?
  - NO! But let us assume cryptographic hash functions behave “approximately” like ROs

Random Oracle Model (ROM)

Mihir Bellare, Phillip Rogaway

- Why ROM?
  - Allows efficient constructions of cryptographic primitives with “provable security” guarantees
  - The security proofs are then in the ROM
  - Efficient signature and encryption schemes (RSA-OAEP, RSA-PSS, etc.)
- How are they used in security proofs?
  - Sample a random H at the beginning of an experiment
  - Output of ROM fully hidden unless queried, i.e., H(m||r) for r a large random string
  - Typically we assume that the reduction can “program” the random oracle, i.e., can choose the answers to the oracle calls
    - This is easily possible as all the answers are independent
    - Can embed information usable to the reduction in oracle answers (we will see examples)

Criticism of the ROM

- Often considered as a “heuristic” argument for security instead of a real proof, as ROM is a very strong idealization
- There are schemes that can be shown secure in the ROM, but insecure when ROM is replaced with **any** real hash function
  - Though, this example is very artificial
  - No realistic example of this type known
- Proofs in the ROM for practical constructions appear to be very robust!

RSA Encryption in the ROM (A hybrid encryption scheme)

- Let H: $\mathbb{Z}_N \rightarrow \{0,1\}^k$ be a hash function modeled as a random oracle
- Let RSA encryption and decryption be as follows:
  - $Enc(m , pk) := (H(x) \oplus m , x^e \bmod N)$ for $m \in \{0,1\}^k$ and $x \leftarrow^{\$} \mathbb{Z}_N^*$
  - $Dec((c_1,c_2), sk) := H(c_2^d \bmod N) \oplus c_1$

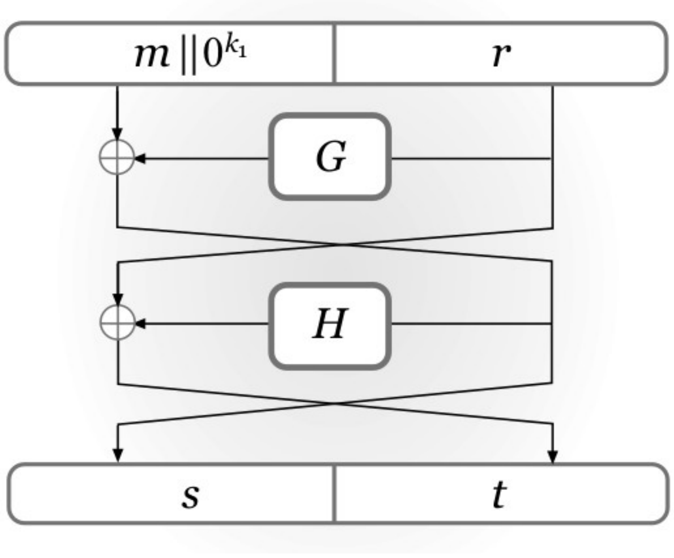
CLAIM: The above construction is CPA-secure under the RSA assumption in the ROM.

Proof idea:

- To obtain information about m from (c_1,c_2) one has to learn information about H(x)
- If the adversary does not query H(x), then challenge ciphertext is independent from m_b
- To learn information about H(x), adversary has to query it. We can embed RSA challenge y as $c^* = (r, y)$ with r uniformly random
- Challenge ciphertext is hidden information theoretically unless random oracle queried on x s.t. $y = x^e \bmod N$
- If this happens, we have an adversary against the RSA assumption (thus we can rule out, that the adversary queries x to H).

Standardized Padded Variants of RSA

- Use of textbook RSA on preprocessed messages
- ~~RSA-PKCS# 1 v1.5 (should not be used!!)*~~
  - "Padded RSA": Basically, encrypt m':=m||r with random r
    - PKCS(m, r) = 0x00||0x02||r||0x00||m
  - No proof of security for assumed CPA secure version known
  - Definitely no CCA security (see next lecture)
- RSA-OAEP (Optimal Asymmetric Encryption Padding)
  - More complex preprocessing
  - Two-round Feistel network with G and H as round functions
    - Invertible!
  - Proof of IND-CCA security in the ROM; thus also IND-CPA secure



*Matthew Green: "PKCS#1v1.5 is awesome — if you're teaching a class on how to attack cryptographic protocols. In all other circumstances it sucks."

RSA Implementation (Pitfalls)

- Small public exponents, i.e., e=3
  - Efficient encryption (only two multiplications)
  - Various attack scenarios known (to reconstruct the message)
    - If the same message is encrypted under at least 3 different public keys
    - If short messsages are encrypted (and no modular reduction required)
- Reasonable choice of public exponent: e=65537
  - Avoids low-exponent attacks and reasonable fast: $65537 = 2^{16}+1$
- Private exponents must not be too small
  - Brute force attacks
  - Even if $d \approx N^{1/4}$ (Wiener, improved by Boneh & Durfee) attacks are known