

Lecture 9. RSA public-key encryption and signatures

Bruno Grenet



<https://membres-ljk.imag.fr/Bruno.Grenet/IntroCrypto.html>

Introduction to cryptology
Université Grenoble Alpes – IM²AG
M1 INFO, MOSIG & AM

A Method for Obtaining Digital Signatures and Public- Key Cryptosystems

R. L. Rivest, A. Shamir, and L. Adleman
MIT Laboratory for Computer Science
and Department of Mathematics

An encryption method is presented with the novel property that publicly revealing an encryption key does not thereby reveal the corresponding decryption key. This has two important consequences:

- (1) Couriers or other secure means are not needed to transmit keys, since a message can be enciphered using an encryption key publicly revealed by the intended recipient. Only he can decipher the message, since only he knows the corresponding decryption key.**
- (2) A message can be "signed" using a privately held decryption key. Anyone can verify this signature using the corresponding publicly revealed encryption key. Signatures cannot be forged, and a signer cannot later deny the validity of his signature. This has obvious**

I. Introduction

The era of "electronic mail" [10] may soon be upon us; we must ensure that two important properties of the current "paper mail" system are preserved: (a) messages are *private*, and (b) messages can be *signed*. We demonstrate in this paper how to build these capabilities into an electronic mail system.

At the heart of our proposal is a new encryption method. This method provides an implementation of a "public-key cryptosystem", an elegant concept invented by Diffie and Hellman [1]. Their article motivated our research, since they presented the concept but not any practical implementation of such a system. Readers familiar with [1] may wish to skip directly to Section V for a description of our method.

II. Public-Key Cryptosystems

In a "public-key cryptosystem" each user places in a public file an encryption procedure E. That is, the public file is a directory giving the encryption procedure of each user. The user keeps secret the details of his corresponding decryption procedure D. These procedures have the following four properties:

- (a) Deciphering the enciphered form of a message M

A Method for Obtaining Digital Signatures and Public-Key Cryptosystems

R. Rivest, A. Shamir & L. Adleman (1978)

- ▶ Basics of RSA encryption scheme
- ▶ Signature using the encryption scheme in *reverse mode*

Pros

- ▶ First proposal of a public-key encryption scheme
- ▶ Use of computational difficulty as security

Cons

- ▶ As presented, the encryption scheme is completely unsafe!
- ▶ The signature is not a good idea!

Remark

- ▶ Already known to GCHQ (UK) in 1973, declassified only in 1997 Clifford Cocks

Contents of this lecture

1. The maths of RSA: the trapdoor permutation \cong content of the original paper

- ▶ $\mathbb{Z}/N\mathbb{Z}$ where $N = p \times q$
- ▶ Designing a *trapdoor permutation*

2. RSA encryption scheme

- ▶ What should be added to obtain a proper encryption scheme?

3. RSA signatures

- ▶ How to obtain a proper signature scheme?

Contents

1. The maths of RSA: the trapdoor permutation

2. RSA encryption scheme

3. RSA signatures

Representation and ring operations

General context

Computations *modulo* $N = p \times q$ where p, q are prime numbers

Representation and modular operations

- ▶ $\mathbb{Z}/N\mathbb{Z} = \{0, 1, \dots, N - 1\}$ with *modular* addition, subtraction and multiplication:
 1. Perform the operation in the integers
 2. Reduce the result *modulo* N
- ▶ Complexity: $O(\log^2 N)$ or $O(\log N \log \log N)$

Example: $\mathbb{Z}/35\mathbb{Z}$

$$21 + 17 = 38 = 3$$

$$5 \times 10 = 50 = 15$$

$$-12 = 23$$

Detour by a fundamental algorithm

The extended Euclidean Algorithm (xGCD)

Input: $a, b \in \mathbb{Z}, a > b \geq 0$

Output: $g, u, v \in \mathbb{Z}$ s.t. $g = au + bv$
and $g = \gcd(a, b)$

1. $(r_0, u_0, v_0) \leftarrow (a, 1, 0)$
2. $(r_1, u_1, v_1) \leftarrow (b, 0, 1)$
3. $i \leftarrow 2$
4. while $r_{i-1} \neq 0$:
5. $(q_i, r_i) \leftarrow \text{QUOREM}(r_{i-2}, r_{i-1})$
6. $(u_i, v_i) \leftarrow (u_{i-2} - q_i u_{i-1}, v_{i-2} - q_i v_{i-1})$
7. $i \leftarrow i + 1$
8. return $(r_{i-2}, u_{i-2}, v_{i-2})$

i	r_i	u_i	v_i	q_i
0	21	1	0	—
1	15	0	1	—
2	6	1	-1	1
3	3	-2	3	2
4	0	5	-7	2
				...

Detour by a fundamental algorithm

The extended Euclidean Algorithm (xGCD)

Input: $a, b \in \mathbb{Z}, a > b \geq 0$

Output: $g, u, v \in \mathbb{Z}$ s.t. $g = au + bv$
and $g = \gcd(a, b)$

1. $(r_0, u_0, v_0) \leftarrow (a, 1, 0)$
2. $(r_1, u_1, v_1) \leftarrow (b, 0, 1)$
3. $i \leftarrow 2$
4. while $r_{i-1} \neq 0$:
5. $(q_i, r_i) \leftarrow \text{QUOREM}(r_{i-2}, r_{i-1})$
6. $(u_i, v_i) \leftarrow (u_{i-2} - q_i u_{i-1}, v_{i-2} - q_i v_{i-1})$
7. $i \leftarrow i + 1$
8. return $(r_{i-2}, u_{i-2}, v_{i-2})$

Correction

- ▶ For all i , $\gcd(a, b) = \gcd(r_i, r_{i+1})$
- ▶ For all i , $r_i = a \cdot u_i + b \cdot v_i$

Consequence

$\gcd(a, b) = 1 \iff$

there exists $u, v \in \mathbb{Z}$ s.t. $1 = a \cdot u + b \cdot v$

Complexity

The bit complexity of the extended Euclidean Algorithm is $O(\log(a) \log(b))$

Inversion and division in $\mathbb{Z}/N\mathbb{Z}$

Definition

$a \in \mathbb{Z}/N\mathbb{Z}$ is **invertible** if there exists $b \in \mathbb{Z}/N\mathbb{Z}$ s.t. $a \times b = 1$ *modular \times*

► one can *divide by a* in $\mathbb{Z}/N\mathbb{Z}$

Theorem

$a \in \mathbb{Z}/N\mathbb{Z}$ is invertible modulo N \iff $\gcd(a, N) = 1$ *previous slide* $\iff \exists u, v$ s.t. $1 = au + Nv$

Proof

$\exists u, v$ s.t. $1 = au + Nv \iff \exists u, au \pmod N = 1 \iff a$ is invertible

Algorithms

Inverse: Use the extended Euclidean Algorithm

Running time: $O(\log^2 N)$

or $O(\log N \log^2 \log N)$

Division: Use multiplication and inverse

Same running time

Invertible elements of $\mathbb{Z}/N\mathbb{Z}$

Definition

- ▶ The **multiplicative group** $\mathbb{Z}/N\mathbb{Z}^\times$ is the set of invertible elements of $\mathbb{Z}/N\mathbb{Z}$
- ▶ Its number of elements is denoted $\varphi(N)$

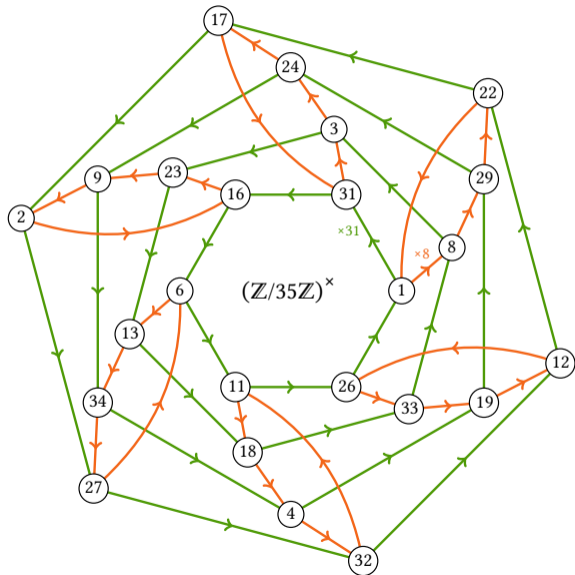
Proposition

If $N = p \times q$ with primes $p \neq q$, $\varphi(N) = (p-1)(q-1)$

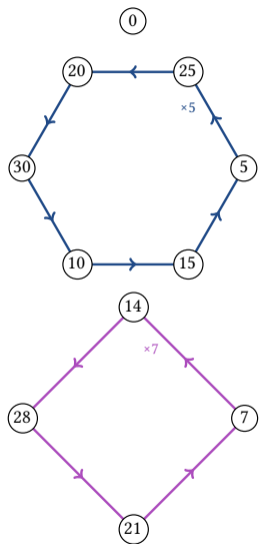
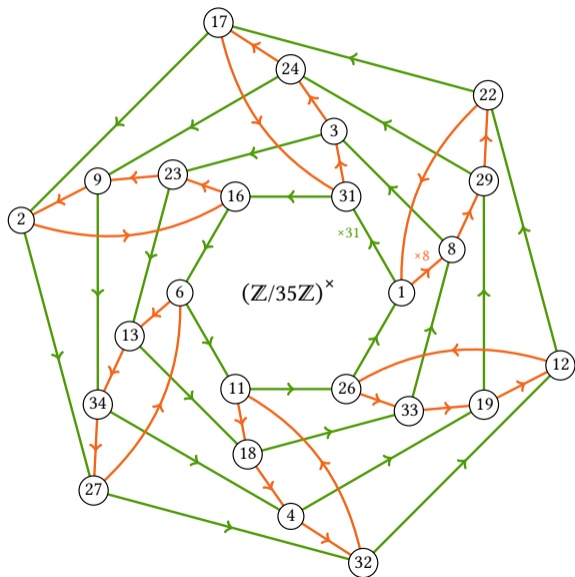
Proof

- a is not invertible in $\mathbb{Z}/N\mathbb{Z} \Leftrightarrow \gcd(a, N) \neq 1$
 $\Leftrightarrow a$ is divisible by p or q .
- Multiples of p : $0, p, 2p, 3p, \dots, (q-1)p$
— Multiples of q : $0, q, 2q, 3q, \dots, (p-1)q$
} # non-invertible elements
= $1 + (p-1) + (q-1) = p+q-1$
- $\varphi(N) = pq - (p+q-1) = (p-1)(q-1)$

The multiplicative group is **not** cyclic!



The multiplicative group is **not** cyclic!



Non-invertible elements

The "RSA theorem"

Theorem

Let $N = p \times q$ with primes $p \neq q$. Then for all $a \in \mathbb{Z}/N\mathbb{Z}$, $a^{1+\varphi(N)} = a$.

Proof

(1) Fermat little thm: $\forall a \in \{1, \dots, p-1\}$, $a^{p-1} \bmod p = 1$ (for prime p)

$$\left\{ ax \bmod p : 1 \leq x \leq p-1 \right\} = \left\{ y : 1 \leq y \leq p-1 \right\}$$
$$a^{p-1} \prod_{x=1}^{p-1} x = \prod_{x=1}^{p-1} ax = \prod_{y=1}^{p-1} y \Rightarrow a^{p-1} = 1$$

$$(2) a^{1+\varphi(N)} = a^{1+(p-1)(q-1)} = a \times (a^{p-1})^{q-1} = a \times (a^{q-1})^{p-1}$$

$$\hookrightarrow a^{1+\varphi(N)} \bmod p = a^{1+\varphi(N)} \bmod q = a$$

therefore $a^{1+\varphi(N)} - a$ is divisible by p and q , thus by $N = pq$

$$\text{Therefore } a^{1+\varphi(N)} - a \bmod N = 0$$



The RSA trapdoor permutation

The original (unsafe!) RSA encryption scheme

Definition as an encryption scheme

Encryption key: (N, e) where $N = p \times q$ with primes $p \neq q$ and $\gcd(e, \varphi(N)) = 1$

Decryption key: (N, d) where $d \times e \bmod \varphi(N) = 1$

Encryption: Given $m \in \mathbb{Z}/N\mathbb{Z}$, compute $c = m^e \bmod N$

Decryption: Given $c \in \mathbb{Z}/N\mathbb{Z}$, compute $m = c^d \bmod N$

Correction

$$c^d = (m^e)^d = m^{e \cdot d} = m^{1 + k\varphi(N)} = m^{1 + \varphi(N) + (k-1)\varphi(N)} = m^{1 + (k-1)\varphi(N)} = m$$

The algorithms and complexities

Key generation

1. Generate two random primes $p \neq q$
 - ▶ Sample random (odd) integers
 - ▶ Test their primality
2. Compute $N = p \times q$ and $\varphi(N) = (p - 1) \times (q - 1)$
3. Generate e, d such that $e \times d \bmod \varphi(N) = 1$
 - ▶ Sample random integers e
 - ▶ Apply xGCD($e, \varphi(N)$) to test invertibility and get d

$O(\log^4 N)$
 $O(\log N)$ samples
 $O(\log^3 N)$
 $O(\log^2 N)$
 $O(\log^2 N)$
 $1 + O(1/\sqrt{N})$ samples
 $O(\log^2 N)$

Encryption and decryption

- ▶ Modular exponentiation in $\mathbb{Z}/N\mathbb{Z}$
 - ▶ *Binary powering*, using $a^n = \begin{cases} a^{\lfloor n/2 \rfloor} \cdot a^{\lfloor n/2 \rfloor} & \text{for even } n \\ a \cdot a^{\lfloor n/2 \rfloor} \cdot a^{\lfloor n/2 \rfloor} & \text{for odd } n \end{cases}$

$O(\log^3 N)$

Remark

- ▶ All complexities can be slightly improved

Attacks on the trapdoor

Possible goals

Key recovery: Given (N, e) , compute d s.t. $d \times e \bmod \varphi(N) = 1$

Plaintext recovery: Given (N, e) and c , compute m s.t. $m^e \bmod N = c$

Computational problems

Modular e -th root: Given N, c, e , compute m s.t. $m^e \bmod N = c$

Computation of φ : Given $N = p \times q$ (for unknown p, q), compute $\varphi(N) = (p-1)(q-1)$

Factorization: Given $N = p \times q$, compute p and q

Reductions between problems

▶ Plaintext recovery \iff modular e -th root

▶ Computation of $\varphi \implies$ Key recovery \implies plaintext recovery

▶ Computation of $\varphi \iff$ Factorization of N : \iff If you can compute p and q you can compute $\varphi(N) = (p-1)(q-1)$

\iff Consider $(x-p)(x-q) = x^2 - (p+q)x + pq$
 $= x^2 - (N - \varphi(N) + 1)x + N \rightarrow$ Compute p, q from $N, \varphi(N)$ using root finding

Integer factorization

Complexity of integer factorization

- ▶ Brute force algorithm: $O(\sqrt{N}) = O(2^{\frac{\log N}{2}})$
- ▶ ...
- ▶ General Number Field Sieve: $2^{O(\log^{\frac{1}{3}} N \log^{\frac{2}{3}} \log N)}$ *Lenstra, Lenstra (1993) and others...*
- ▶ Quantum algorithms:
 - ▶ $O(\log^3 N) = O(2^{3 \log \log N})$ *Shor (1994)*
 - ▶ $O(\log^{2.5} N) = O(2^{2.5 \log \log N})$ *Regev (2025)*

(Remark: no known NP-hardness result \rightarrow could be polynomial in $\log N$)

Current record: 829-bit (250-digit) integer factorization

- ▶ Boudot, Gaudry, Guillevic, Heninger, Thomé, Zimmermann (Feb. 2020)
- ▶ Software: [CADO-NFS](#)
- ▶ Hardware: (mainly) academic clusters
- ▶ Approx. 2,700 core-years in a few months

Contents

1. The maths of RSA: the trapdoor permutation

2. RSA encryption scheme

3. RSA signatures

The original RSA scheme is unsafe!

Deterministic encryption

- ▶ Two ciphertexts are equal iff the corresponding messages are equal
- ▶ The scheme cannot be IND-CPA/CCA secure

Examples of other difficulties

Small exponent: If e and m are small: $m^e \bmod N = m^e$ in $\mathbb{Z} \rightarrow m = \sqrt[e]{c}$ in \mathbb{Z}

Related messages: Given the ciphertexts of m and $m + \delta$ with small $\delta \rightarrow m$

Multiple receivers: Given the ciphertexts of m with several distinct keys $\rightarrow m$

The original RSA encryption scheme is severely flawed and **should never be used!**

- ▶ Solution: use (randomized) padding

The padded RSA encryption scheme: overview

Construction

Parameters: n : number of bits of N ; ℓ : length of the messages

- $\text{Gen}_n()$:
1. $p, q \leftarrow$ two random primes s.t. $p \times q$ has bit-length n
 2. $N \leftarrow p \times q, \quad \varphi_N \leftarrow (p - 1) \times (q - 1)$
 3. $e \leftarrow \mathbb{Z}/\varphi_N\mathbb{Z}^\times, \quad d \leftarrow e^{-1} \bmod \varphi_N$
 4. return $\text{ek} = (N, e), \text{dk} = (N, d)$

- $\text{Enc}_{\text{ek}}(m)$:
1. $r \leftarrow \{0, 1\}^{n-\ell}$ $m \in \{0, 1\}^\ell$
 2. while $\hat{m} = r \| m \notin \mathbb{Z}/N\mathbb{Z}$: $r \leftarrow \{0, 1\}^{n-\ell}$
 3. return $c = \hat{m}^e \bmod N$ $\text{ek} = (N, e)$

- $\text{Dec}_{\text{dk}}(c)$:
1. $\hat{m} \leftarrow c^d \bmod N$ $\text{dk} = (N, d)$
 2. Return $m = \hat{m}_{[n-\ell..n]}$

Correction

- ▶ As for the original RSA

Security of padded RSA

The security depends on $n - \ell$

number of padding bits

Small values of $n - \ell$

- ▶ $2^{n-\ell}$ possible paddings
- ▶ Sufficient to break $2^{n-\ell}$ original RSA instances

→ Not secure!

Very large value of $n - \ell$: $\ell = 1$

- ▶ If computing e -th root in $\mathbb{Z}/N\mathbb{Z}$ is hard, IND-CPA secure encryption scheme
- ▶ Very inefficient secure encryption scheme, one bit at a time
- ▶ Slightly better if used as a KEM

still useless!

Medium values of $n - \ell$

- ▶ Open problem!

Padded RSA in practice

RSA PKCS1

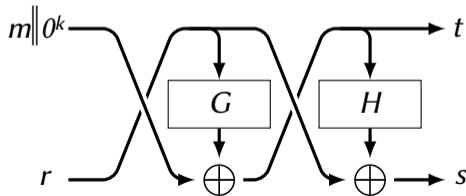
- ▶ Standardized by RSA laboratories
- ▶ Padding: $m \rightarrow 0x00\|0x02\|r\|0x00\|m$ where r is random
- ▶ Attack using failure of the unpadding procedure
 - ▶ Used against SSL 3.0
 - ▶ Workaround: in case of failure, return a random value
 - ▶ Prevents IND-CCA security

Bleichenbacher (1998)

RSA Optimal Asymmetric Encryption Padding (OAEP)

Bellare, Rogaway (1994)

- ▶ Padding: $m \rightarrow s\|t$ where
 - ▶ G, H : hash functions
 - ▶ r : random bits
- ▶ Standardized as PKCS1 v2
- ▶ IND-CCA secure under two assumptions
 - ▶ RSA trapdoor is *one-way*
 - ▶ G and H are random oracles



Contents

1. The maths of RSA: the trapdoor permutation

2. RSA encryption scheme

3. RSA signatures

Original (unsafe!) version

Signatures are not “encryption with the private key”

Construction

- $\text{Gen}_n()$:
1. $p, q \leftarrow$ two random primes s.t. $p \times q$ has bit-length n
 2. $N \leftarrow p \times q, \quad \varphi_N \leftarrow (p-1) \times (q-1)$
 3. $e \leftarrow \mathbb{Z}/\varphi_N\mathbb{Z}^\times, \quad d \leftarrow e^{-1} \bmod \varphi(N)$
 4. return $\text{vk} = (N, e), \text{sk} = (N, d)$

$\text{Sign}_{\text{sk}}(m)$: 1. return $m^d \bmod N$

$m \in \mathbb{Z}/N\mathbb{Z}$

$\text{Vrfy}_{\text{vk}}(m, \sigma)$: 1. test whether $m = \sigma^e \bmod N$

Correction

- ▶ As for the original RSA encryption scheme

Attacks

existential forgeries

1. The adversary chooses σ and computes $m = \sigma^e \bmod N$
2. The adversary sees (m_1, σ_1) and (m_2, σ_2) and computes $m = m_1 \cdot m_2$ and $\sigma = \sigma_1 \cdot \sigma_2$

RSA FDH (*Full Domain Hash*)

Construction

- $\text{Gen}_n()$: 1. Compute $\text{vk} = (N, e)$, $\text{sk} = (N, d)$ as previously
2. Choose a hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}/N\mathbb{Z}$

$\text{Sign}_{\text{sk}}(m)$: 1. return $H(m)^d \bmod N$ $m \in \{0, 1\}^*$

$\text{Vrfy}_{\text{vk}}(m, \sigma)$: 1. test whether $H(m) = \sigma^e \bmod N$

Properties of H to avoid previous and other problems

1. Forgery: $\sigma \rightarrow h = \sigma^e \rightarrow m$ s.t. $H(m) = h$ *first preimage resistance*
2. $(m_1, \sigma_1), (m_2, \sigma_2) \rightarrow (m, \sigma_1\sigma_2)$ if $H(m) = H(m_1)H(m_2)$ *“multiplicative resistance”*
3. If $H(m_1) = H(m_2)$, $\sigma_1 = \sigma_2$ *collision resistance*
4. The image of H should be (almost) the full $\mathbb{Z}/N\mathbb{Z}$ *full domain*

Bad and good news

- ▶ We *do not know* how to *provably* build a satisfying H *no security proof*
- ▶ Security proof if RSA trapdoor is *one-way* and H is a random oracle

Proof sketch of RSA FDH

(Informal) Theorem

If e -th roots in $\mathbb{Z}/N\mathbb{Z}$ are hard to compute and **H is random**, RSA FDH is secure

Proof sketch

- What is an adversary A_0 against RSA FDH:

- Sends queries m_i , and gets as answer $\sigma_i = H(m_i)^d \pmod N$

- At the end, outputs (m, σ) s.t. $\sigma = H(m)^d \pmod N$ or $\sigma^e = H(m) \pmod N$

- **Assumption**: A_0 always queries $H(m_i)$ and $H(m)$

- From A_0 , we build A_0' that, given c and e , computes h s.t. $h^e = c \pmod N$

Proof sketch of RSA FDH

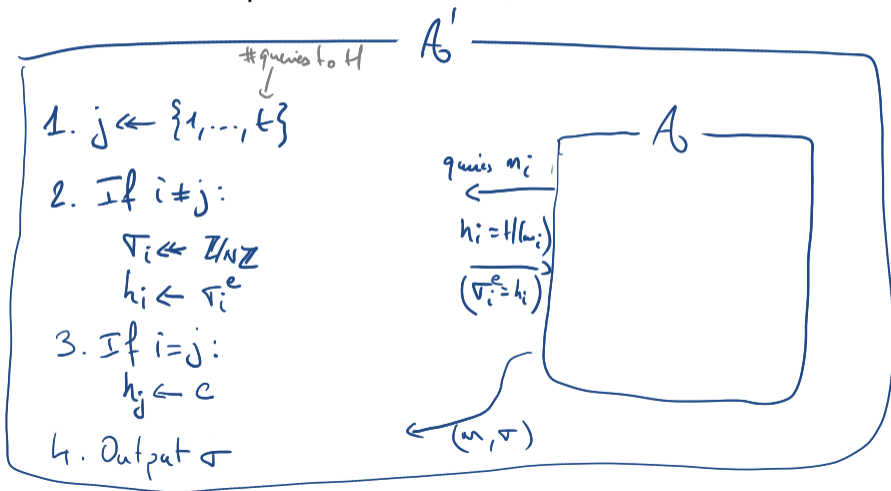
(Informal) Theorem

If e -th roots in $\mathbb{Z}/N\mathbb{Z}$ are hard to compute and H is random, RSA FDH is secure

Proof sketch

$c, e \rightarrow$

σ s.t. $\sigma^e = c$



We want a s.t. $a^e = c = h_j$
We have m, σ s.t. $c = H(m) = \sigma^e$

Conclusion

RSA is a *one-way* trapdoor function

- ▶ One direction is easy to compute: $(m, e) \rightarrow m^e \bmod N$
- ▶ The other direction is (hopefully!) hard to compute: $(c, e) \rightarrow \sqrt[e]{c} \bmod N$
- ▶ But there is a trapdoor: given $d = e^{-1} \bmod \varphi(N)$, easy to compute $m = c^d \bmod N$

Use of RSA trapdoor function

- ▶ No direct use!
- ▶ Public-key encryption scheme \rightarrow RSA OAEP
- ▶ Digital signatures \rightarrow RSA FDH

Security

- ▶ No formal proof that RSA is *one-way* *assumption*
- ▶ Related but not equivalent to the difficulty of integer factorization
- ▶ Typical key sizes: N with ≥ 2048 bits
- ▶ Many other pitfalls: implementation, randomness quality, dependent keys, ...