# Cryptography, Number Theory, and RSA

## Introduction to Computer Science

Ruben Niederhagen

November 2021

Department of Mathematics and Computer Science (IMADA)

University of Southern Denmark

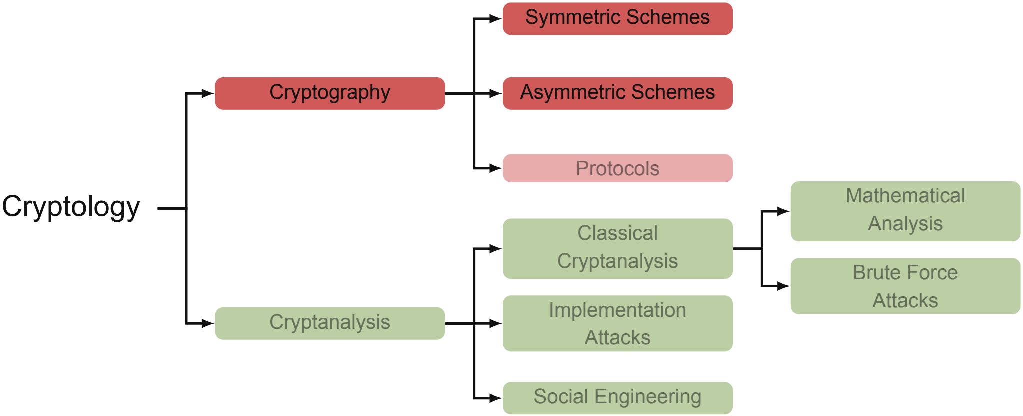
# Outline



- Symmetric key cryptography
- Public key cryptography
- Introduction to number theory
- RSA
  - Correctness of RSA
  - Modular exponentiation
  - Greatest common divisor
  - Primality testing
- Digital signatures with RSA
- Combining symmetric and public key systems

# Cryptography vs. Cryptanalysis





# Why is Cryptography helpful?

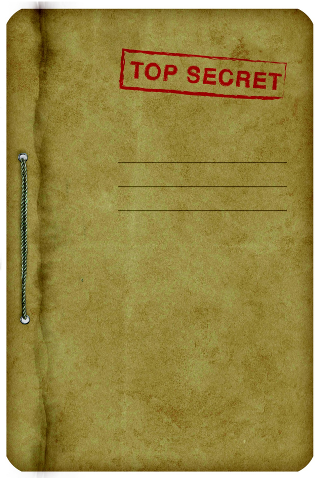


- A system has formal or informal security requirements.
- The requirements include security goals for protecting assets.
  - “The data must not be accessible by a third party.”
  - “It must be ensured, that the data comes from device X.”
  - …
- These security goals can be reached by the use of *cryptographic primitives* and *protocols* (security mechanisms).
- Cryptography is *necessary* for security, but *not sufficient*.

# Confidentiality

- Information must not be accessible by third parties.

$\Rightarrow$ Encryption



- The correctness of information (data is not modified).

$\rightarrow$ Detection of manipulation.

$\Rightarrow$ Cryptographic Hash Function

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 6b | 94 | 1a | 73 | 8d | a5 | c6 | 6a | d7 | 91 | 93 | 24 | 83 | 62 | 30 | 83 |
| 05 | ac | 60 | 83 | 18 | 50 | 21 | 01 | 12 | 21 | 09 | 5c | 2c | 59 | 4e | cf |
| ce | 1c | 91 | f9 | 60 | 66 | 55 | e6 | b2 | cc | 35 | 99 | 91 | cc | 8e | cc |
| 78 | a4 | 97 | b9 | d1 | 8d | 1e | dd | 3e | 9e | e8 | 4d | c0 | e0 | ff | fe |
| 4d | 20 | 9b | 0e | a0 | fa | 75 | 2b | 18 | 07 | cf | 4c | dc | 39 | 6d | e5 |
| ca | e9 | eb | 47 | 1c | 79 | e5 | ea | 87 | d3 | de | 7d | ac | ec | 68 | 69 |
| f5 | aa | 19 | bb | 42 | bb | 9e | ff | dd | 2f | cb | f6 | b0 | 11 | bb | 33 |
| 32 | 0a | 0b | 43 | e3 | 7c | b6 | be | 9b | 56 | 6e | de | eb | f7 | 1f | cc |
| cb | 2b | ba | 67 | 42 | 41 | 20 | 26 | 75 | 43 | 75 | 43 | 93 | 9b | 90 | 1b |
| 2b | 0b | 44 | ff | 6a | b2 | 10 | 26 | 4d | c4 | dc | 4d | 54 | 94 | d8 | 48 |
| 15 | e9 | a1 | 93 | 69 | 29 | 7d | 82 | fe | 88 | 3e | 27 | 1d | 95 | 3e | f6 |
| a6 | 79 | 73 | bc | 43 | bd | 4d | be | 94 | 9e | 1e | fd | 7b | 86 | 44 | e8 |
| 24 | 5a | 82 | ed | 4b | 7b | db | 63 | b1 | 3d | ff | bb | f6 | 7f | 1e | 28 |
| ce | f1 | 31 | 7d | 81 | 6e | a6 | 0d | 18 | 23 | bd | f1 | 28 | c6 | e3 | f4 |
| f8 | bf | e8 | 27 | f5 | e6 | 3c | aa | b3 | e1 | 5f | ce | 41 | 90 | cf | fe |
| 3e | 98 | fe | 9b | 3e | 37 | 03 | c3 | 73 | f7 | fb | 41 | 8e | 3e | ad | 24 |
| 06 | b1 | b1 | f4 | ca | 34 | 4c | b6 | 68 | 49 | f9 | 1f | 8e | fa | ff | 2c |
| d0 | 3c | d2 | 42 | 4e | 62 | 3c | 4c | 1a | c9 | 66 | ba | 03 | 6b | 65 | 28 |
| 9e | 8b | 92 | 88 | f4 | 16 | 59 | 41 | e6 | a3 | e4 | 5d | 7a | 12 | 3f | a9 |
| b2 | 50 | b6 | 83 | 74 | 90 | 33 | f8 | 66 | 2d | 39 | 09 | 8d | 8c | d0 | f1 |
| 24 | 17 | a5 | 84 | 9c | e7 | 12 | e9 | a4 | 85 | 64 | 0f | 8e | 91 | 8f | 27 |
| 42 | 3e | de | 5a | 08 | 9b | c8 | f6 | b0 | 49 | ac | 85 | 5d | 62 | a7 | c9 |
| 60 | 56 | c1 | 4e | b3 | 12 | 56 | 41 | 73 | e1 | 65 | 3e | 85 | ef | c0 | 94 |
| 0f | ef | 49 | 76 | bc | 43 | 7a | 48 | 0b | bd | 40 | 2a | c8 | 3e | f8 | 1c |

# Authenticity

- Provable validity and credibility of data and subjects.

⇒ Data:

- Message Authentication Code
- Digital Signature

→ Often depends on integrity.

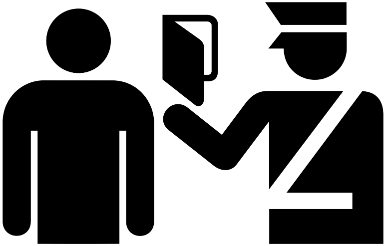
⇒ Subjects:

- Authentication
- Authorization

→ *Verifier* authenticates *proofer*.


```
-----BEGIN PGP SIGNATURE-----
Comment: This signature is for the .tar version of the archive
Comment: git archive --format tar --prefix=linux-4.18.16/ v4.18.16
Comment: git version 2.19.1

iQIzBAABCAAdFiEEZH8oZUiU471FcZm+ONu9yGCSaT4FAlvK3c8ACgkQONu9yGCS
aT5iCg/9GI/DfM4YtqUZYttppVxjgB1ryJng3+H3avf2EaIMnS0v+SNhQRvbn2vl
aVmjHxyHG0Zsqz23MJ3X2j2UMfMUg5tl7IcdQ6rYw4kCb/Mp76m+fZytteKpDltN
WGdyOsJ7bRhhbhAb4uU5MDKqksS93HPUHntJzbiN57VlTKHlc20D/8DAF/fuyjSu
P9EvP2a0gqEK0Ga0FY7Jiyp8ezo7oJCvA86PXo0686tHcY912rUzVXOoUhjHULcA
xE4gbaRAMKW2aW+itq/f5YWFovdUT5ay6mpBhse4pxqYHSFpVMWwCAd9BiPNBWN8
9ppsLX0aGsR2+MGQCZTAsl/0smBXFI3YZFkvjw2mQfOHptCyWC8qyaIzivFjh+OV
VKvIsvpTGQ0oP7hxcjOkKJdIdOgqiwgTV1HuxECdTuhxBMesiMgL1ZcKF2T/P5pP
RhyEnRoJfzjiPkkz/DaPOMtP59wcb7PYzwQVzeQP2MXQD8+KyJwF6EYDhhlDWJVt
fBGpoxGYDGySxgA1ux/Uf/nhvEri0lpSzQvOWdk/RDXFlZYFI2sxWptWnCvNLhwj
vaYy4cnugrF3rjL20vav/1i0ZQ9043nIVj+8uimahN8ZI+w9GCipfVoMuBwaJtxD
QU8CGOJhJqOXTJCoogiS+9BOXz/n7YM3S1XOPt75tUj5nUXZNgM=
=/5LG
-----END PGP SIGNATURE-----
```

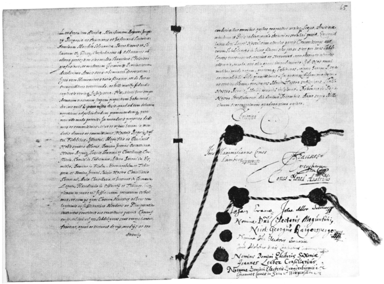



# Non-Repudiation



- Guarantee of reliability of actions.
- A subject cannot deny an applied action afterwards.

$\Rightarrow$ Digital Signature



| **Security Goals** | | **Security Mechanisms** |
|---|---|---|
| Confidentiality | $\rightarrow$ | Encryption |
| Integrity | $\rightarrow$ | Cryptographic Hash Function |
| Authenticity | $\rightarrow$ | Message Authentication Code, Digital Signature |
| Non-Repudiation | $\rightarrow$ | Digital Signature |
| ... | $\rightarrow$ | ... |

# What is encryption?



- Every key $k \in \mathcal{K}$ of a key space $\mathcal{K}$ uniquely defines an encryption function $E_k : \mathcal{M} \to \mathcal{C}$ is a bijection from a message space $\mathcal{M}$ to a ciphertext space $\mathcal{C}$, called the encryption function.
- For each key $d \in \mathcal{K}$, $D_k : \mathcal{C} \to \mathcal{M}$ denotes a bijection from $\mathcal{C}$ to $\mathcal{M}$, called the decryption function.
- Decryption is the inverse of encryption, i.e. $D_k(E_k(m)) = m$.

# Caesar Cipher (with key $= 3$)



| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| P | Q | R | S | T | U | V | W | X | Y | Z | Æ | Ø | Å |
| 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 |
| S | T | U | V | W | X | Y | Z | Æ | Ø | Å | A | B | C |
| 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 0 | 1 | 2 |

$$c_i = m_i + 3 \pmod{29}$$

Suppose the following was encrypted using a Caesar cipher and the Danish alphabet. The key is unknown.

*ZQOØQOØ, RI.*

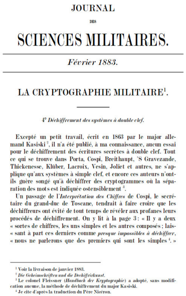
What does it say?

What does this say about how many keys should be possible?

# Kerckhoffs's Principle



- Requirement for modern crypto systems.
- *The security of a scheme must not depend on the secrecy of the scheme but on the secret key.*
  - $\Rightarrow$ Large key space.
  - $\Rightarrow$ Effort to break a scheme must be higher than the resources of an adversary.
  - $\Rightarrow$ The scheme and implementation should be publicly available and auditable.



JOURNAL

DES

SCIENCES MILITAIRES.

*Février 1883.*

LA CRYPTOGRAPHIE MILITAIRE[1].

4° *Déchiffrement des systèmes à double clef.*

Excepté un petit travail, écrit en 1863 par le major allemand Kasiski[2], il n'a été publié, à ma connaissance, aucun essai pour le déchiffrement des écritures secrètes à double clef. Tout ce qui se trouve dans Porta, Cospi, Breithaupt, 'S Gravezande, Thickenesse, Klüber, Lacroix, Vesin, Joliet et autres, ne s'applique qu'aux systèmes à simple clef, et encore ces auteurs n'ont-ils guère songé qu'à déchiffrer des cryptogrammes où la séparation des mots est indiquée ostensiblement[3].

Un passage de l'*Interprétation des Chiffres* de Cospi, le secrétaire du grand-duc de Toscane, tendrait à faire croire que les déchiffreurs ont évité de tout temps de révéler aux profanes leurs procédés de déchiffrement. On y lit à la page 3 : « Il y a deux « sortes de chiffres, les uns simples et les autres composés ; lais- « sant à part ces derniers comme *presque impossibles à déchiffrer*, « nous ne parlerons que des premiers qui sont les simples[4]. »

[1] Voir la livraison de janvier 1883.
[2] *Die Geheimschriften und die Dechiffrirkunst.*
[3] Le colonel Fleissner (*Handbuch der Kryptographie*) a adopté, sans modification aucune, la méthode de déchiffrement du major Kasiski.
[4] Je cite d'après la traduction du Père Nicéron.

# Vigenère Cipher

- Vigenère cipher shifts the input by a rotating key of some length $m$:

$$c_i = m_i + k_{i \bmod m} \pmod{n}$$

- Example for key $\{3, 1, 4\}$:

| | | | | | | |
|---|---|---|---|---|---|---|
| Input | H | e | l | l | o | … |
| Key | 3 | 1 | 4 | 3 | 1 | … |
| Output | K | f | p | o | p | … |

Department of Mathematics
and Computer Science (IMADA)

- The One-Time-Pad (OTP) uses an infinitely long key:

$$c_i = m_i + k_{++} \pmod n$$

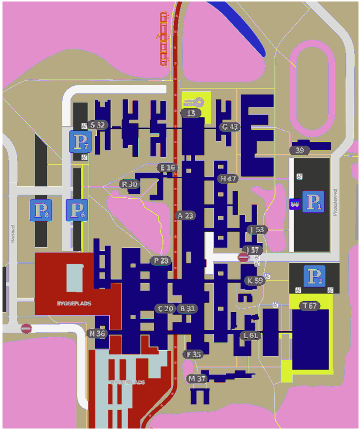
- The same key index must never be used again.
- All ciphertexts are independent from all plain texts and from each other.
- If the key of the OTP is uniform and perfectly random, the cipher is unbreakable.

→ Note: If a Vigenère key has the same length as the message and each key is used only once, it is the same as the OTP.

# A more Visual Approach

**Original**


**Caesar**



# A more Visual Approach

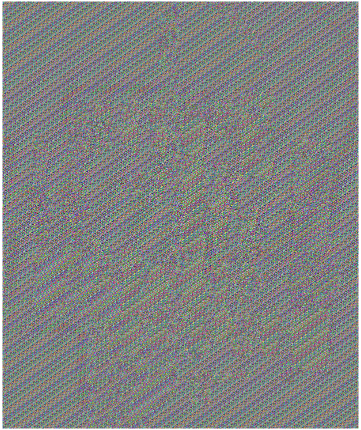
**Original**


**Vigenère**

Department of Mathematics
and Computer Science (IMADA)

**Original**

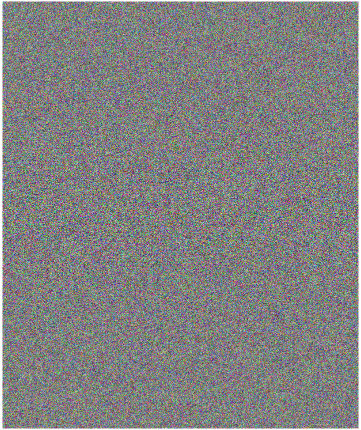

**Vigenère (longer key)**



# A more Visual Approach

## Original


## One-Time-Pad



# A more Visual Approach

**Original**


**One-Time-Pad (imperfect key)**

# Symmetric Key Systems



- ~~Caesar Cipher~~
- ~~Vigenère Cipher~~
- One-Time Pad
- ~~Enigma~~
- ~~DES~~
- ~~Triple DES~~
- (IDEA)
- Blowfish
- **<u>AES</u>**

Department of Mathematics
and Computer Science (IMADA)

- ~~Caesar Cipher~~
- ~~Vigenère Cipher~~
- One-Time Pad
- ~~Enigma~~
- ~~DES~~
- ~~Triple DES~~
- (IDEA)
- Blowfish
- **AES**

**Problem:**

Both sides need to know a shared secret key.

How do we securely communicate the key?

**Solution:**

Use public key cryptography:

- Use a public key for encryption.
- Use a private (secret) key for decryption.

Bob — 2 keys -$PK_B$,$SK_B$

$PK_B$ — Bob's public key
$SK_B$ — Bob's private (secret) key

For Alice to send $m$ to Bob,
to encrypt $m$, Alice computes:
$c = E(m, PK_B)$.

To decrypt $c$, Bob computes:
$r = D(c, SK_B)$.
$r = m$

It must be "hard" to compute $m$ from $(c, PK_B)$.

It must be "hard" to compute $SK_B$ from $PK_B$.

Department of Mathematics
and Computer Science (IMADA)

**Definition**

Suppose $a, b \in \mathbb{Z}$, $a > 0$.

Suppose $\exists c \in \mathbb{Z}$ s.t. $b = ac$.

Then *a divides b*: $a \mid b$.

$a$ is a *factor* of $b$.

$b$ is a *multiple* of $a$.

$e \nmid f$ means $e$ does not divide $f$.

**Theorem**

*$a, b, c \in \mathbb{Z}$. Then*

1. *if $a|b$ and $a|c$, then $a|(b+c)$,*
2. *if $a|b$, then $a|bc$ $\forall c \in \mathbb{Z}$, and*
3. *if $a|b$ and $b|c$, then $a|c$.*

$$p \in \mathbb{Z}, \; p > 1$$

**Definition (prime)**
*p* is *prime* if 1 and *p* are the only positive integers which divide *p*.

**Definition (composite)**
*p* is *composite* if it is not prime.

**Example (primes)**
$2, 3, 5, 7, 11, 13, 17, \ldots$

**Example (composites)**
$4, 6, 8, 9, 10, 12, 14, 15, 16, \ldots$

**Theorem**

$a \in \mathbb{Z}$, $d \in N$

$\exists$ *unique* $q, r$, $0 \leq r < d$ *s.t.*

$$a = dq + r$$

*d – divisor*

*a – dividend*

*q – quotient*

*r – remainder = a mod d*

**Definition (relatively prime)**

$gcd(a, b)$ = greatest common divisor of $a$ and $b$
= largest $d \in \mathbb{Z}$ s.t. $d|a$ and $d|b$

If $gcd(a, b) = 1$, then $a$ and $b$ are *relatively prime* or *coprime*.

Department of Mathematics
and Computer Science (IMADA)

**Definition (congruence)**

$a \equiv b \pmod{m}$ — $a$ is congruent to $b$ modulo $m$ if $m \mid (a - b)$.

$m \mid (a - b) \;\Rightarrow\; \exists k \in \mathbb{Z}$ s.t. $a = b + km$.

**Theorem**

$a \equiv b \pmod{m} \qquad c \equiv d \pmod{m}$

*Then* $a + c \equiv b + d \pmod{m}$ *and* $ac \equiv bd \pmod{m}$.

**Proof.**

$\exists k_1, k_2$ s.t.

$a = b + k_1 m \qquad c = d + k_2 m$

$$\begin{aligned} a + c &= b + k_1 m + d + k_2 m \\ &= b + d + (k_1 + k_2)m \end{aligned}$$

$\square$

Department of Mathematics
and Computer Science (IMADA)

**Definition (congruence)**

$a \equiv b \pmod{m}$ — $a$ is congruent to $b$ modulo $m$ if $m \mid (a - b)$.

$m \mid (a - b) \;\Rightarrow\; \exists k \in \mathbb{Z}$ s.t. $a = b + km$.

**Example**

1. $15 \equiv 22 \pmod{7}$?
2. $15 \equiv 1 \pmod{7}$?
3. $15 \equiv 37 \pmod{7}$?
4. $58 \equiv 22 \pmod{9}$?

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}$.

$\Rightarrow \; r = m$.

**Example**

$p = 5$, $q = 11$, $e = 3$, $d = 27$, $m = 8$.

Then $N = p \cdot q = 5 \cdot 11 = 55$,
$(p - 1)(q - 1) = 4 \cdot 10 = 40$,
$gcd(e, (p - 1)(q - 1)) = gcd(3, 40) = 1$,
$e \cdot d = 81$. So $e \cdot d \equiv 1 \pmod{40}$.

To encrypt $m$: $c = 8^3 \pmod{55}$
$= \; 512 \pmod{55} = \; 17$.

To decrypt $c$: $r = 17^{27} \pmod{55}$
$= \; 1667711322168688287513535727415473$
$\pmod{55} = \; 8$.

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}$.

$\boxed{\Rightarrow\ r = m.}$

**Why does RSA work?**
Chinese Remainder Theorem and
Fermat's Little Theorem.

**Theorem (Fermat's Little Theorem)**

*$p$ is a prime, $p \nmid a$.*

*Then* $a^{p-1} \equiv 1 \pmod{p}$
*and* $a^p \equiv a \pmod{p}$.

**Example**

$3^7 \equiv 3 \pmod{7}$

$3^6 \equiv 1 \pmod{7}$

$23^{40} \equiv 1 \pmod{41}$

Department of Mathematics
and Computer Science (IMADA)

**Theorem (The Chinese Remainder Theorem)**

*Let $n_1, n_2, ..., n_k$ be pairwise relatively prime. For any integers $x_1, x_2, ..., x_k$, there exists $x \in \mathbb{Z}$ s.t. $x \equiv x_i \pmod{n_i}$ for $1 \leq i \leq k$, and this integer is uniquely determined modulo the product $N = n_1 n_2 ... n_k$.*

$$
\begin{aligned}
x &\equiv x_1 \pmod{n_1} \\
x &\equiv x_2 \pmod{n_2} \\
&\cdots \\
x &\equiv x_k \pmod{n_k}
\end{aligned}
$$

$$\rightarrow x \equiv x_1 \equiv x_2 \equiv \cdots \equiv x_k \pmod{N = n_1 n_2 ... n_k}$$

Department of Mathematics
and Computer Science (IMADA)

**Correctness of RSA**

Consider $r = D(E(m, PK_A), SK_A)$.
Note $\exists k$ s.t. $e_A d_A = 1 + k(p_A - 1)(q_A - 1)$.

$$r \equiv (m^{e_A} \pmod{N_A})^{d_A} \pmod{N_A} \equiv m^{e_A{}^{d_A}} \equiv m^{e_A d_A} \equiv m^{1+k(p_A-1)(q_A-1)} \pmod{N_A}.$$

Consider $r \pmod{p_A}$. Use Fermat's little theorem.
$r \equiv m^{1+k(p_A-1)(q_A-1)} \equiv m \cdot (m^{(p_A-1)})^{k(q_A-1)} \equiv m \cdot 1^{k(q_A-1)} \equiv m \pmod{p_A}$.

Consider $r \pmod{q_A}$. Use Fermat's little theorem.
$r \equiv m^{1+k(p_A-1)(q_A-1)} \equiv m \cdot (m^{(q_A-1)})^{k(p_A-1)} \equiv m \cdot 1^{k(p_A-1)} \equiv m \pmod{q_A}$.

Apply the Chinese Remainder Theorem:
$gcd(p_A, q_A) = 1, \Rightarrow r \equiv m \pmod{N_A}$.

Department of Mathematics
and Computer Science (IMADA)

$$x \equiv x_1 \pmod{n_1} \qquad\qquad r \equiv m \pmod{p}$$
$$x \equiv x_2 \pmod{n_2} \qquad\qquad r \equiv m \pmod{q}$$
$$\rightarrow$$
$$\rightarrow x \equiv x_1 \equiv x_2 \pmod{N = n_1 n_2} \qquad\qquad \rightarrow r \equiv m \equiv m \pmod{N = pq}$$

We consider the special case where $n_1 = p$ and $n_2 = q$ are two primes (hence $N = pq$), and where $x_1 = x_2 = m$.

Clearly, $m \equiv m \pmod{p}$ and $m \equiv m \pmod{q}$ for any $m$.
So since $r$ fulfills $r \equiv m \pmod{p}$ and $r \equiv m \pmod{q}$, then $r \equiv m \pmod{N}$.

In particular, $0 \leq r, m \leq N - 1$, so we must have $r = m$.

Department of Mathematics
and Computer Science (IMADA)

**Correctness of RSA**

Consider $r = D(E(m, PK_A), SK_A)$.

Note $\exists k$ s.t. $e_A d_A = 1 + k(p_A - 1)(q_A - 1)$.

$$r \equiv (m^{e_A} \pmod{N_A})^{d_A} \pmod{N_A} \equiv m^{e_A{}^{d_A}} \equiv m^{e_A d_A} \equiv m^{1+k(p_A-1)(q_A-1)} \pmod{N_A}.$$

Consider $r \pmod{p_A}$. Use Fermat's little theorem.

$$r \equiv m^{1+k(p_A-1)(q_A-1)} \equiv m \cdot (m^{(p_A-1)})^{k(q_A-1)} \equiv m \cdot 1^{k(q_A-1)} \equiv m \pmod{p_A}.$$

Consider $r \pmod{q_A}$. Use Fermat's little theorem.

$$r \equiv m^{1+k(p_A-1)(q_A-1)} \equiv m \cdot (m^{(q_A-1)})^{k(p_A-1)} \equiv m \cdot 1^{k(p_A-1)} \equiv m \pmod{q_A}.$$

Apply the Chinese Remainder Theorem:

$gcd(p_A, q_A) = 1, \Rightarrow r \equiv m \pmod{N_A}$.

$\Rightarrow$ So $D(E(m, PK_A), SK_A) = m$.

# Security of RSA

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}$.

$\Rightarrow \; r = m$.

**RSA problem:**
Given $c$ and $PK_A = (N_A, e_A)$, find $m$ such that:

$$c = m^{e_A} \pmod{N_A}.$$

This is believed to be hard to solve for large values.

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}$.

$\Rightarrow \ r = m$.

The primes $p_A$ and $q_A$ are kept secret with $d_A$.

What happens if Eve can factor $N_A$?

Then she can find $p_A$ and $q_A$.
From them and $e_A$, she finds $d_A$.

Then she can decrypt just like Alice.

Factoring must be hard!

Department of Mathematics
and Computer Science (IMADA)

**Theorem**

*N composite $\Rightarrow$ N has a prime divisor $\leq \sqrt{N}$.*

```
procedure FACTOR(N)
    for i = 2 to √N do
        if i divides N then
            return (i, N/i)        ▷ divisor found
        end if
    end for
    return -1                      ▷ divisor not found
end procedure
```

**Corollary**

*There is an algorithm for factoring N that does $O(\sqrt{N})$ tests of divisibility.*

Check all possible divisors between 2 and $\sqrt{N}$.
Not finished in your grandchildren's life time for $N$ with 3072 bits.

**Problem:**

The length of the input is $n = \lceil \log_2(N+1) \rceil$.

So the running time is $O(2^{n/2})$ — exponential.

**Open Problem:**

Does there exist a polynomial time factoring algorithm?

Use primes which are at least 2048 (or 3072) bits long.
So $2^{2047} \leq p_A, q_A < 2^{2048}$ — so $p_A \approx 10^{616}$.

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = \underline{m^{e_A}} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = \underline{c^{d_A}} \pmod{N_A}$.

$\Rightarrow\ r = m$.

**How do we implement RSA?**

We need to encrypt and decrypt:
compute $a^k \pmod{n}$.

**Example**

$p = 5$, $q = 11$, $e = 3$, $d = 27$, $m = 8$.

Then $N = 55$.

$e \cdot d = 81$.

So $e \cdot d \equiv 1 \pmod{4 \cdot 10}$.

To encrypt $m$: $c = 8^3 \pmod{55} = 17$.

To decrypt $c$: $r = 17^{27} \pmod{55} = 8$.

SDU

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}$.

$\Rightarrow\ r = m$.

**Example**

$p = 5$, $q = 11$, $e = 3$, $d = 27$, $m = 8$.

Then $N = p \cdot q = 5 \cdot 11 = 55$,
$(p - 1)(q - 1) = 4 \cdot 10 = 40$,
$gcd(e, (p - 1)(q - 1)) = gcd(3, 40) = 1$,
$e \cdot d = 81$. So $e \cdot d \equiv 1 \pmod{40}$.

To encrypt $m$: $c = 8^3 \pmod{55}$
$= 512 \pmod{55} = 17$.

To decrypt $c$: $r = 17^{27} \pmod{55}$
$= 1667711322168688287513535727415473$
$\pmod{55} = 8$.

**Theorem**

*For all nonnegative integers,* $b, c, m$*,* $b \cdot c \pmod{m} = (b \pmod{m}) \cdot (c \pmod{m}) \pmod{m}$.

**Example**

$a^3 \pmod{n} = a \cdot a^2 \pmod{n} = (a \pmod{n})(a^2 \pmod{n}) \pmod{n}$.

$$\begin{aligned} 8^3 \pmod{55} &= 8 \cdot 8^2 \pmod{55} \\ &= 8 \cdot 64 \pmod{55} \\ &= 8 \cdot 9 \pmod{55} \\ &= 72 \pmod{55} \\ &= 17 \end{aligned}$$

$\Rightarrow$ Computing modulo often keeps the numbers (relatively) small!

We need to encrypt and decrypt: compute $a^k \pmod{n}$.

$a^2 \pmod{n} \equiv a \cdot a \pmod{n}$ — 1 modular multiplication
$a^3 \pmod{n} \equiv a \cdot (a \cdot a \pmod{n}) \pmod{n}$ — 2 mod mults
Guess: $k - 1$ modular multiplications.

This is too many!

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

$p_A$ and $q_A$ have $\geq 2048$ bits each.

So at least one of $e_A$ and $d_A$ has $\geq 2048$ bits.

To either encrypt or decrypt would need $\geq 2^{2047} \approx 10^{616}$ operations
(age of the universe: $4.3 \cdot 10^{17}$ seconds).

Department of Mathematics
and Computer Science (IMADA)

We need to encrypt and decrypt: compute $a^k \pmod{n}$.

$a^2 \pmod{n} \equiv a \cdot a \pmod{n}$ — 1 modular multiplication
$a^3 \pmod{n} \equiv a \cdot (a \cdot a \pmod{n}) \pmod{n}$ — 2 mod mults

How do you calculate $a^4 \pmod{n}$ with less than 3 mod mults?
$a^4 \pmod{n} \equiv (a^2 \pmod{n})^2 \pmod{n}$ — 2 mod mults

In general: $a^{2s} \pmod{n}$?
$a^{2s} \pmod{n} \equiv (a^s \pmod{n})^2 \pmod{n}$

In general: $a^{2s+1} \pmod{n}$?
$a^{2s+1} \pmod{n} \equiv a \cdot ((a^s \pmod{n})^2 \pmod{n}) \pmod{n}$

# Modular Exponentiation

**procedure** EXP(a,k,n) ▷ Compute $a^k \pmod{n}$
  **if** $k < 0$ **then return** -1 ▷ Error
  **if** $k = 0$ **then return** 1
  **if** $k = 1$ **then return** $a \pmod{n}$
  **if** $k$ is odd **then**
    $c_1 \leftarrow \text{EXP}(a, k-1, n)$
    **return** $a \cdot c_1 \pmod{n}$
  **end if**
  **if** $k$ is even **then**
    $c_2 \leftarrow \text{EXP}(a, k/2, n)$
    **return** $c_2 \cdot c_2 \pmod{n}$
  **end if**
**end procedure**

To compute $3^6 \pmod{7}$: $\text{Exp}(3, 6, 7)$

$c_2 \leftarrow \text{Exp}(3, 3, 7)$

$c_1 \leftarrow \text{Exp}(3, 2, 7)$

$c_2 \leftarrow \text{Exp}(3, 1, 7)$

$3 \pmod{7} = 3$

$c_2 \cdot c_2 \pmod{n} = 3 \cdot 3 \pmod{7} = 2$

$a \cdot c_1 \pmod{n} = 3 \cdot 2 \pmod{7} = 6$

$c_2 \cdot c_2 \pmod{n} = (6 \cdot 6) \pmod{7} = 1$

```
procedure EXP(a,k,n)            ▷ Compute a^k (mod n)
    if k < 0 then return -1                     ▷ Error
    if k = 0 then return 1
    if k = 1 then return a (mod n)
    if k is odd then
        c1 ← EXP(a, k − 1, n)
        return a · c1 (mod n)
    end if
    if k is even then
        c2 ← EXP(a, k/2, n)
        return c2 · c2 (mod n)
    end if
end procedure
```

How many modular multiplications?

Divide exponent by 2 every other time.

How many times can we do that?

$\lfloor \log_2(k) \rfloor$ — So at most $2\lfloor \log_2(k) \rfloor$ modular multiplications.

This is quite cheap!

$\Rightarrow$ We can compute modular exponentiation efficiently using *square-and-multiply* and frequent modulo operations.

Department of Mathematics
and Computer Science (IMADA)

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}$.

$\Rightarrow\ r = m$.

Use $N_A = 35$ $e_A = 11$ to create keys.

What are $p_A$ and $q_A$?

What is $d_A$? Try $d_A = 11$ and check it.

Encrypt 4. Decrypt the result.

Department of Mathematics
and Computer Science (IMADA)

$N_A = p_A \cdot q_A$, where $p_A, q_A$ prime.

$\boxed{\begin{array}{l} gcd(e_A, (p_A - 1)(q_A - 1)) = 1. \\ e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}. \end{array}}$

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}$.

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}$.

$\Rightarrow\ r = m$.

**How do we implement RSA?**

We need to find: $e_A, d_A$.
$gcd(e_A, (p_A - 1)(q_A - 1)) = 1$.
$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}$.

Choose random $e_A$.
Check that:

$$gcd(e_A, (p_A - 1)(q_A - 1)) = 1.$$

Find $d_A$ such that:

$$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}.$$

Department of Mathematics
and Computer Science (IMADA)

**Theorem**

$a, b \in N$. $\exists\, s, t \in \mathbb{Z}$ s.t. $sa + tb = gcd(a, b)$.

**Proof.**

Let $d$ be the smallest positive integer in $D = \{xa + yb \mid x, y \in \mathbb{Z}\}$.

$d \in D \;\Rightarrow\; d = x'a + y'b$ for some $x', y' \in \mathbb{Z}$.

$gcd(a, b)|a$ and $gcd(a, b)|b$, so $gcd(a, b)|x'a$, $gcd(a, b)|y'b$, and $gcd(a, b)|\,(x'a + y'b = d)$.

We will show that $d|gcd(a, b)$, so $d = gcd(a, b)$.

Suppose $a = dq + r$ with $0 \le r < d$ and some $q$.

$$\begin{aligned} r &= a - dq \\ &= a - q(x'a + y'b) \\ &= (1 - qx')a - (qy')b \end{aligned}$$

$\Rightarrow\; r \in D$

$r < d$, $d$ is smallest positive integer in $D \Rightarrow r = 0 \;\Rightarrow d|a$. Similarly, one can show that $d|b$.

Therefore, $d|gcd(a, b)$. □

SDU

How do you find $d$, $s$ and $t$?

Let $d = gcd(a, b)$. Write $b$ as $b = aq + r$ with $0 \leq r < a$.
Then, $d|b \; \Rightarrow \; d|(aq + r)$.
Also, $d|a \; \Rightarrow \; d|(aq) \; \Rightarrow \; d|((aq + r) - aq) \; \Rightarrow \; d|r \Rightarrow \; d|a, \; d|b, \; d|(a \text{ mod } b)$.

Let $d' = gcd(a, r) = gcd(a, b - aq)$.
Then, $d'|a \; \Rightarrow \; d'|(aq)$
Also, $d'|(b - aq) \; \Rightarrow \; d'|((b - aq) + aq) \; \Rightarrow \; d'|b \Rightarrow \; d'|a, \; d'|(a \text{ mod } b), \; d'|b$.

Thus, $gcd(a, b) = gcd(a, b \pmod a) = gcd(b \pmod a, a)$.
We can reduce to a "smaller" problem $\Rightarrow$. *Extended Euclidean Algorithm*

**Example**

Compute $s$ and $t$ such that $s \cdot 6 + t \cdot 9 = gcd(6, 9)$:

$$9 = 0 \cdot 6 + 1 \cdot 9$$
$$6 = 1 \cdot 6 + 0 \cdot 9$$

$gcd(6, 9) = gcd(9 \bmod 6, 6)$

$$9 - 1 \cdot 6 = (0 \cdot 6 + 1 \cdot 9) - 1(1 \cdot 6 + 0 \cdot 9)$$
$$3 = -1 \cdot 6 + 1 \cdot 9 = gcd(6, 9)$$

# The Extended Euclidean Algorithm

$d_0 \leftarrow b \qquad s_0 \leftarrow 0 \qquad t_0 \leftarrow 1$

$d_1 \leftarrow a \qquad s_1 \leftarrow 1 \qquad t_1 \leftarrow 0$

$n \leftarrow 1$

**while** $d_n > 0$ **do**

**begin**

$n \leftarrow n + 1$

$q_n \leftarrow \lfloor d_{n-2}/d_{n-1} \rfloor$

$d_n \leftarrow d_{n-2} - q_n d_{n-1}$

$s_n \leftarrow s_{n-2} - q_n s_{n-1}$

$t_n \leftarrow t_{n-2} - q_n t_{n-1}$

**end**

**return** $s \leftarrow s_{n-1}, \quad t \leftarrow t_{n-1}, \quad gcd(a, b) \leftarrow d_{n-1}$

Finding **multiplicative inverses** modulo $m$:

Given $a$ and $m$, find $x$ s.t. $a \cdot x \equiv 1 \pmod{m}$.

Should also find a $k$, s.t. $ax = 1 + km$.
So solve for an $s$ in an equation $sa + tm = 1$.

This can be done if $gcd(a, m) = 1$.
Just use the *Extended Euclidean Algorithm*.

If the result, $s$, is negative, add $m$ to $s$.
Now, for $s' = s + m$, we have
$(s' - m)a + tm \equiv 1 \pmod{m}$.

**Examples:**
Calculate the following:

1. $gcd(6, 9)$
2. $s$ and $t$ such that $s \cdot 6 + t \cdot 9 = gcd(6, 9)$
3. $gcd(15, 23)$
4. $s$ and $t$ such that $s \cdot 15 + t \cdot 23 = gcd(15, 23)$

$\boxed{N_A = p_A \cdot q_A, \text{ where } p_A, q_A \text{ prime.}}$

$gcd(e_A, (p_A - 1)(q_A - 1)) = 1.$

$e_A \cdot d_A \equiv 1 \pmod{(p_A - 1)(q_A - 1)}.$

- $PK_A = (N_A, e_A)$
- $SK_A = (N_A, d_A)$

To encrypt:
$c = E(m, PK_A) = m^{e_A} \pmod{N_A}.$

To decrypt:
$r = D(c, SK_A) = c^{d_A} \pmod{N_A}.$

$\Rightarrow \ r = m.$

**How do we implement RSA?**
We need to find: $p_A, q_A$ — large primes.

Choose numbers at random and check if they are prime?

1. How many random integers of length 1024 are prime?

**Theorem (Prime Number Theorem)**

*About $\frac{x}{\ln x}$ numbers $< x$ are prime.*

So, about $\frac{2^{1024}}{709}$ integers of length 1024 are prime.

$\Rightarrow$ We expect to test about 709 numbers before finding a prime with 1024 bits.

(This holds because the expected number of tries until a "success", when the probability of "success" is $p$, is $1/p$.)

2. How fast can we test if a number is prime?

**Sieve of Eratosthenes:**

Use lists to track multiples of primes:

| | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | ... |
| | 3 | | 5 | | 7 | | 9 | | 11 | | 13 | | 15 | | 17 | | 19 | | 21 | ... |
| | | | 5 | | 7 | | | | 11 | | 13 | | | | 17 | | 19 | | | ... |
| | | | | | 7 | | | | 11 | | 13 | | | | 17 | | 19 | | | ... |

$2^{1024} \approx 10^{308}$ — more than the number of atoms in universe ($10^{78}$ to $10^{82}$).

So we cannot even write out this list!

# Primality Test — Method 2



**procedure** CHECKPRIME($n$)
  **for** $i = 2$ to $\sqrt{n}$ **do**
    **if** $i$ divides $n$ **then**
      **return** 1 ▷ divisor found
    **end if**
  **end for**
  **return** -1 ▷ divisor not found
**end procedure**

The same as factoring.

Check all possible divisors between 2 and $\sqrt{n}$.

Our sun will die before we're done!

Department of Mathematics
and Computer Science (IMADA)

Recall:

**Theorem (Fermat's Little Theorem)**
*Suppose p is a prime.*
*Then for all* $1 \leq a \leq p-1$,

$$a^{p-1} \pmod{p} \equiv 1.$$

Use a randomized primality test:

**Miller–Rabin primality test:**
Starts with Fermat test:

$$2^{14} \pmod{15} \equiv 4 \not\equiv 1.$$

So 2 is a *witness* that 15 is not prime.

**Fermat test:**

**procedure** $\textsc{Prime}(n)$
  **for** $i = 1$ to $r$ **do**
    Choose random $a \in \{1, 2, \ldots, n-1\}$
    **if** $a^{n-1} \pmod{n} \not\equiv 1$ **then**
      **return** composite
    **end if**
  **end for**
  **return** probably prime
**end procedure**

**Problem:**
Does not work well for some numbers!

**Definition (Carmichael Numbers)**

A composite $n$ such that

$$\text{for all } a \in \{1, 2, \ldots, n-1\} \text{ s.t. } \gcd(a, n) = 1, \quad a^{n-1} \pmod{n} \equiv 1$$

is called a *Carmichael number*.

**Example**

$561 = 3 \cdot 11 \cdot 17$

Only 241 out of 560 numbers are prime-witnesses for 561.

It is likely that Fermat's test does not reveil 561 as prime for several attempts.

For Carmichael numbers with large prime factors, this becomes even more significant.

**Theorem**

*If $p$ is prime,*

$\sqrt{1} \pmod{p} = \{x \mid x^2 \pmod{p} = 1\} = \{1, p-1\}$.

*If $p$ has $> 1$ distinct factors, 1 has at least 4 square roots.*

**Example**

$\sqrt{1} \pmod{15} = \{1, 4, 11, 14\}$

**Example**

Taking square roots of $1 \pmod{561}$:

$$50^{560} \pmod{561} \equiv 1$$
$$50^{280} \pmod{561} \equiv 1$$
$$50^{140} \pmod{561} \equiv 1$$
$$50^{70} \pmod{561} \equiv 1$$
$$50^{35} \pmod{561} \equiv 560$$

$$2^{560} \pmod{561} \equiv 1$$
$$2^{280} \pmod{561} \equiv 1$$
$$2^{140} \pmod{561} \equiv 67$$

2 is a *witness* that 561 is composite.

Department of Mathematics
and Computer Science (IMADA)

**procedure** MILLERRABIN($n, r$)
  Calculate odd $m$ such that $n - 1 = 2^s \cdot m$
  **for** $i = 1$ to $r$ **do**
    Choose random $a \in \{1, 2, \ldots, n-1\}$
    **if** $a^{n-1} \pmod{n} \not\equiv 1$ **then return** composite
    **if** $a^{(n-1)/2} \pmod{n} \equiv n-1$ **then** continue
    **if** $a^{(n-1)/2} \pmod{n} \not\equiv 1$ **then return** composite
    **if** $a^{(n-1)/4} \pmod{n} \equiv n-1$ **then** continue
    **if** $a^{(n-1)/4} \pmod{n} \not\equiv 1$ **then return** composite
    $\ldots$
    **if** $a^{m} \pmod{n} \equiv n-1$ **then** continue
    **if** $a^{m} \pmod{n} \not\equiv 1$ **then return** composite
  **end for**
  **return** probably prime
**end procedure**

**Theorem**

*If $n$ is composite, at most $1/4$ of the $a$'s with $1 \leq a \leq n-1$ will not end in "***return** composite*" during an iteration of the for-loop.*

This means that with $r$ iterations, a composite $n$ will survive to "**return** probably prime" with probability at most $(1/4)^r$. For e.g. $r = 100$, this is less than $(1/4)^{100} = 1/2^{200} < 1/10^{60}$.

A prime $n$ will always survive to "**return** probably prime".

$\Rightarrow$ We can test for primality quite fast!

# Conclusions about Primality Testing



1. Miller–Rabin is a practical primality test.
2. There is a less practical deterministic primality test.
3. Randomized algorithms are useful in practice.
4. Algebra is used in primality testing.
5. Number theory is not useless.

# Combining Symmetric and Public Key Systems



**Problem:**

Public key systems are slow!

**Solution:**

Use symmetric key system for large message.
Encrypt only session key with public key system.

**To encrypt a message $m$ to send to Bob:**

- Choose a random *session key* $k$ for a symmetric key system (e.g., AES).
- Encrypt $k$ with Bob's public key — result $k_e$.
- Encrypt $m$ with $k$ — result $m_e$.
- Send $k_e$ and $m_e$ to Bob.

How does Bob decrypt? Why is this efficient?

# Digital Signatures with RSA



Suppose Alice wants to *sign* a document $m$ such that:

- no one else could *forge* her signature and
- it is easy for others to *verify* her signature.

Note $m$ has arbitrary length.

RSA is used on fixed length messages.

Alice uses a **cryptographically secure hash function** $h$, such that:

- for any message $m'$, $h(m')$ has a fixed length (e.g., 512 bits) and
- it is “hard” for anyone to find two messages $(m_1, m_2)$ such that $h(m_1) = h(m_2)$.

Then Alice “decrypts” $h(m)$ with her secret RSA key $(N_A, d_A)$:

$$s = (h(m))^{d_A} \pmod{N_A}.$$

Bob *verifies* her signature using her public RSA key $(N_A, e_A)$ and $h$:

$$c = s^{e_A} \pmod{N_A}.$$

He accepts if and only if

$$h(m) = c.$$

This works because $s^{e_A} \pmod{N_A} =$

$$((h(m))^{d_A})^{e_A} \pmod{N_A} = ((h(m))^{e_A})^{d_A} \pmod{N_A} = h(m).$$

Department of Mathematics
and Computer Science (IMADA)

**Data in transit:**

- websites,
- emails,
- chat,
- . . .

**Data at rest:**

- disc encryption,
- program or data obfuscation,
- . . .

**Authentication:**

- passports,
- NemID,
- biometry,
- . . .

**Rights management:**

- media access,
- feature activation,
- . . .

**Privacy:**

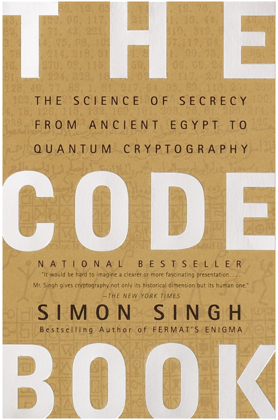
- data mining on anonymized data,
- age verification,
- . . .

**Anonymity:**

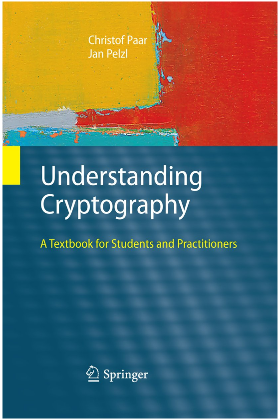
- voting systems,
- bidding systems,
- . . .

# Further Reading (if interested)






"The Code Book"
Simon Singh




"Understanding Cryptography"
Christof Paar, Jan Pelzl

# XKCD — Security