UNIVERSITÄT PASSAU

Fakultät für Informatik und Mathematik

6090: Security of Computer and Embedded Systems

Week 7: Cryptographic Foundations Part 2

Elif Bilge Kavun

elif.kavun@uni-passau.de

This Week’s Outline

- Hash Functions
- Asymmetric (Public-key) Encryption

Cryptographic Hashes: Requirements

- <u>“Hash” Motivation:</u> Create a data “fingerprint”

Cryptographic Hashes: Requirements

- "Hash" Motivation: Create a data "fingerprint"

- A hash function $h(x)$ (in the general sense) has the properties
  - Compression
    - h maps an input x of an arbitrary bit length to an output $h(x)$ of fixed bit length n
  - Polynomial time computable
    - There exists a Turing machine *M* and a polynomial *p(n)*, such that *M* computes the function, and such that *M* runs in time ≤ *p(n)* for all inputs of length *n*

Cryptographic Hashes: Requirements

- Example (longitudinal redundancy check):
  - Given m blocks of n-bit input $b_1, \ldots, b_m$, form the n -bit checksum c from the bitwise XOR of every block, i.e., (for $1 \leq i \leq n$)

$$c_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

Cryptographic Hashes: Requirements

- Example (longitudinal redundancy check):

  - Given m blocks of n-bit input $b_1, \ldots, b_m$, form the n -bit checksum c from the bitwise XOR of every block, i.e., (for $1 \leq i \leq n$)

$$c_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

- Cryptographic techniques can be seen as a refinement of checksum techniques **to handle an active forger**

Cryptographic Hashes: Requirements

- $h(x)$ is a cryptographic hash function if it is additionally
  - One-way (or pre-image resistant)
    - Given y, it is hard to compute an x where $h(x) = y$

Cryptographic Hashes: Requirements

- $h(x)$ is a cryptographic hash function if it is additionally
  - One-way (or pre-image resistant)
    - Given y, it is hard to compute an x where $h(x) = y$
  - And usually either
    - 2nd-preimage resistance (weak collision resistance)
      - It is computationally infeasible to find a second input that has the same output as any specified input, i.e., given x to find an $x' \neq x$ such that $h(x') \neq h(x)$

Cryptographic Hashes: Requirements

- $h(x)$ is a cryptographic hash function if it is additionally
  - One-way (or pre-image resistant)
    - Given y, it is hard to compute an x where $h(x) = y$
  - And usually either
    - 2nd-preimage resistance (weak collision resistance)
      - It is computationally infeasible to find a second input that has the same output as any specified input, i.e., given x to find an $x' \neq x$ such that $h(x') \neq h(x)$
    - Collision resistance (implies 2nd-preimage resistance – strong collision resistance)
      - It is difficult to find two distinct inputs x, x' where $h(x') \neq h(x)$
      - If there is any, such a pair is called cryptographic hash collision
      - It requires a hash value at least twice as long as that required for pre-image resistance; otherwise collisions may be found by a birthday attack (we'll see that later in "Attacks" lecture)
      - Collision resistance implies second pre-image resistance but does not imply pre-image resistance
      - A hash-function which is only second pre-image resistant is considered insecure and is therefore not recommended for real applications

Cryptographic Hashes: Summary

- A “cryptographic” hash function must be deterministic
  - Same message always results in the same hash

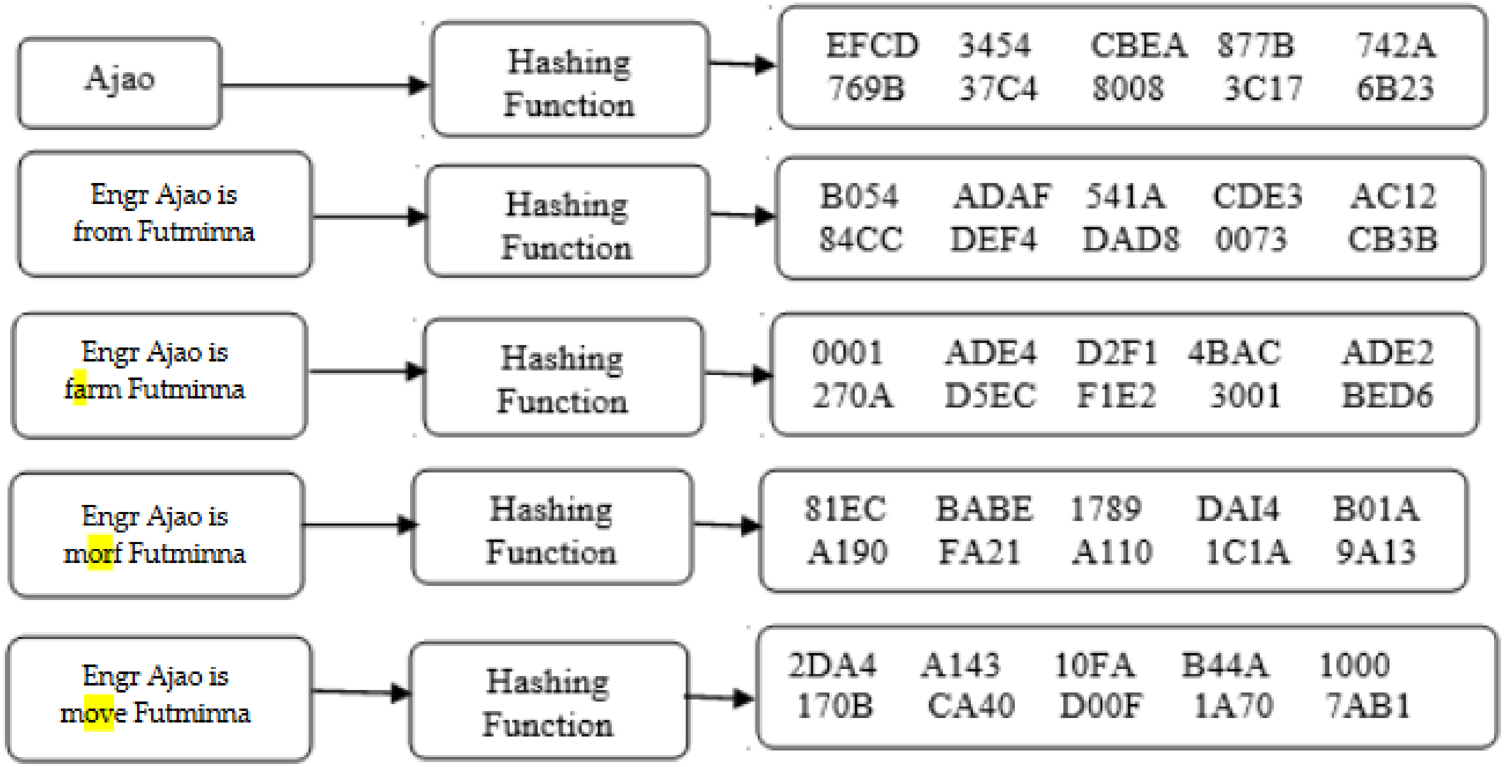
- Quick to compute the hash value for any given message

- Infeasible to generate a message yielding a given hash value
  - Reversing the hash value generation process

- Infeasible to find two different messages with the same hash value

- A small change to a message should change the hash value extensively so that a new hash value appears uncorrelated with the old hash value (avalanche effect)

Cryptographic Hashes: Avalanche Effect



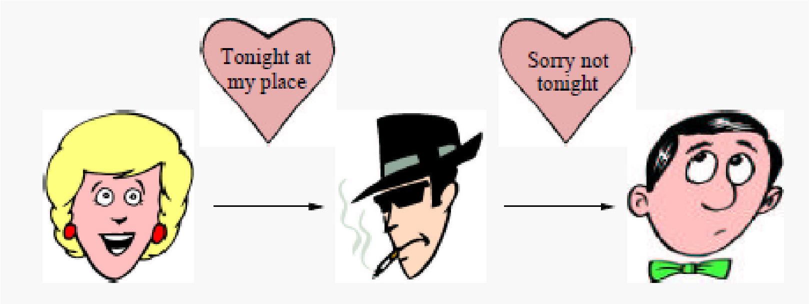
Picture: Lukman et al., Crypto Hash Algorithm-Based Blockchain Technology for Managing Decentralized Ledger Database in Oil and Gas Industry, Multidisciplinary Scientific Journal, vol. 2, 300–325, MDPI, 2019.

Cryptographic Hashes: Summary

- A “cryptographic” hash function must be deterministic
  - Same message always results in the same hash
- Quick to compute the hash value for any given message
- Infeasible to generate a message yielding a given hash value
  - Reversing the hash value generation process
- Infeasible to find two different messages with the same hash value
- A small change to a message should change the hash value extensively so that a new hash value appears uncorrelated with the old hash value (avalanche effect)
- Uses: Information-security applications
  - Digital signatures
  - Message authentication codes (MACs)
  - Other forms of authentication
- Hash value also called message digest or Modification Detection Code (MDC)

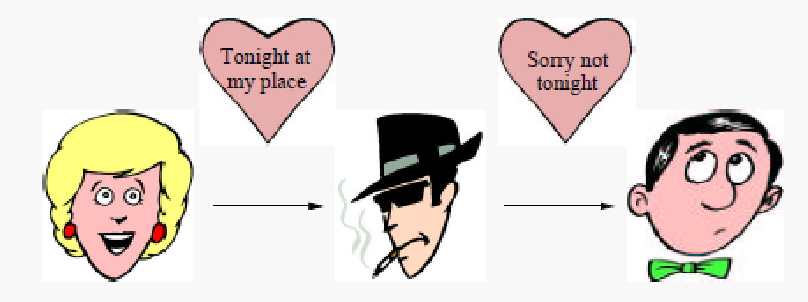
Application: Message Integrity

- *Message* or *data integrity* is the property that data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source

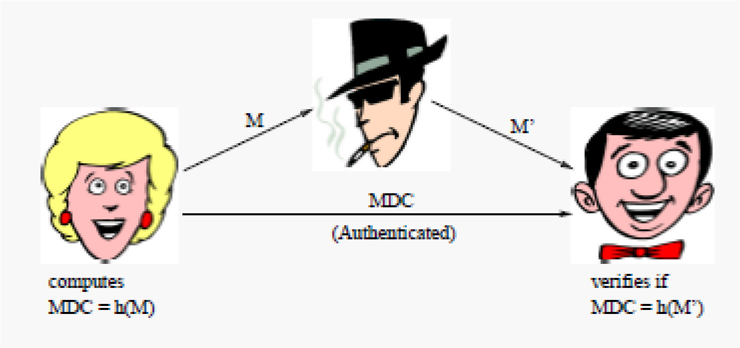


Application: Message Integrity

- *Message* or *data integrity* is the property that data has not been altered in an unauthorized manner since the time it was created, transmitted, or stored by an authorized source



- Message integrity: MDC provides checkable fingerprint
  - Requires 2nd-preimage resistance and authenticated MDC
  - Typical application: Signed hashes



Application: Password Files

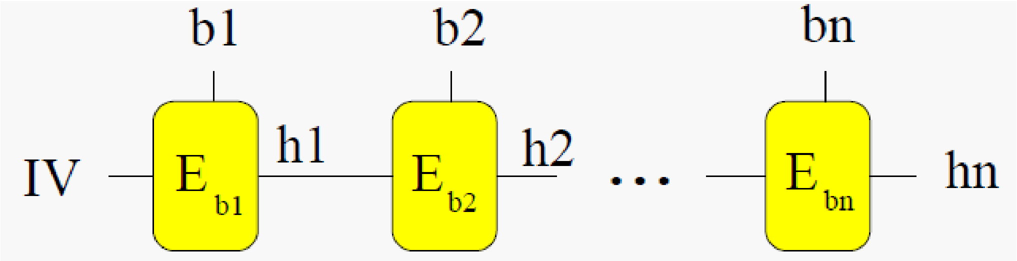
- For password p, store $h(p)$ in password file
- Requires mainly only pre-image resistance. Why?
- Often combined with *salt* s, i.e., store pair $(s, h(s, p))$

Constructing a Cryptographic Hash Function

- Block chaining techniques can be used (Rabin 1978)
  - Divide message M into fixed size blocks $b_1, \ldots, b_n$
  - Use symmetric encryption algorithm, e.g., DES

$$h_0 = IV\,(\text{initial value})$$
$$h_1 = E_{bi}\ (h_{i-1})$$

  - Similar to Cipher Block Chaining (CBC), but no secret key



- Modern algorithms (e.g., SHA-1/2/3, MD4, MD5, ...) are much more complex and use specially designed functions
  - A number of collision results has shaken confidence in their properties
  - Modern applications based on hashes still "appear" safe, e.g., no preimage attacks yet (except SHA-1)

Asymmetric (Public-key) Encryption

- What is it?
  - What is the basis?

Asymmetric (Public-key) Encryption

- What is it?
  - What is the basis?

- Before this…

Background: One-way Functions

- A function $f: X \to Y$ is a *one-way function*, if
  - f is "easy" to compute for all $x \in X$
  - f^{-1} is "difficult" to compute
- Example
  - Problem of *modular cube roots* (check the references, internet, etc. to understand this)
    - Select primes $p = 48611$ and $q = 53993$
    - Let $n = pq = 2624653723$ and $X = \{1, 2, \dots, n-1\}$
    - Define $f(x) = x^3 \bmod n$ where $f: X \to N$
    - Compute $f(2489991) = 2489991^3 \bmod 2624653723 = 1981394214$ → (somehow) **Easy!**
    - Invert f (which is f^{-1}): Means finding "x", which is cubed and went through a modulo operation (and we only have the remainder of this operation) → **Difficult!**

Background: Trapdoor One-way Function

- A *trapdoor one-way function* is a one-way function
  - Given extra information (the *trapdoor* information), it is feasible to find an $x \in X$ where $f(x) = y$
- Example
  - Computing modular cube roots (introduced in the previous slide) is easy when p and q are known
    - Basic number theory

Asymmetric (Public-key) Encryption

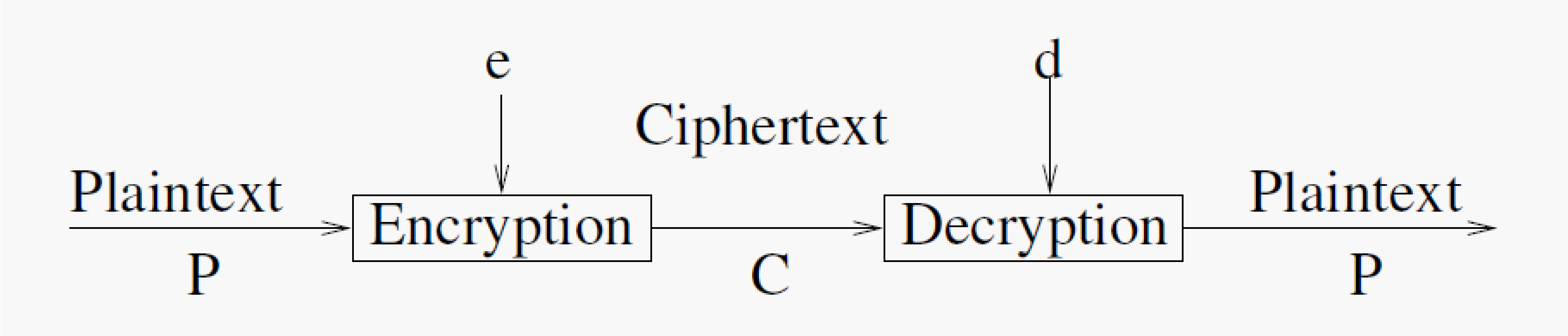
- Some example algorithms
  - RSA (Rivest-Shamir-Adleman)
    - Very famous, widely-deployed
    - Based on difficulty of factoring large numbers
  - Elliptic Curve Cryptography
    - Famous alternative, “lightweight”
    - Based on the algebraic structure of elliptic curves over finite fields (finding shortest vector)
  - NTRUEncrypt
    - Based on the difficulty of factoring certain polynomials in a truncated polynomial ring into a quotient of two polynomials having very small coefficients
  - Post-quantum Public Key Algorithms
    - Based on lattice distance problem
    - Code-based
    - Isogeny-based

Asymmetric (Public-key) Encryption

- What is it?
  - It is based on difficulty of certain (mathematical) problems

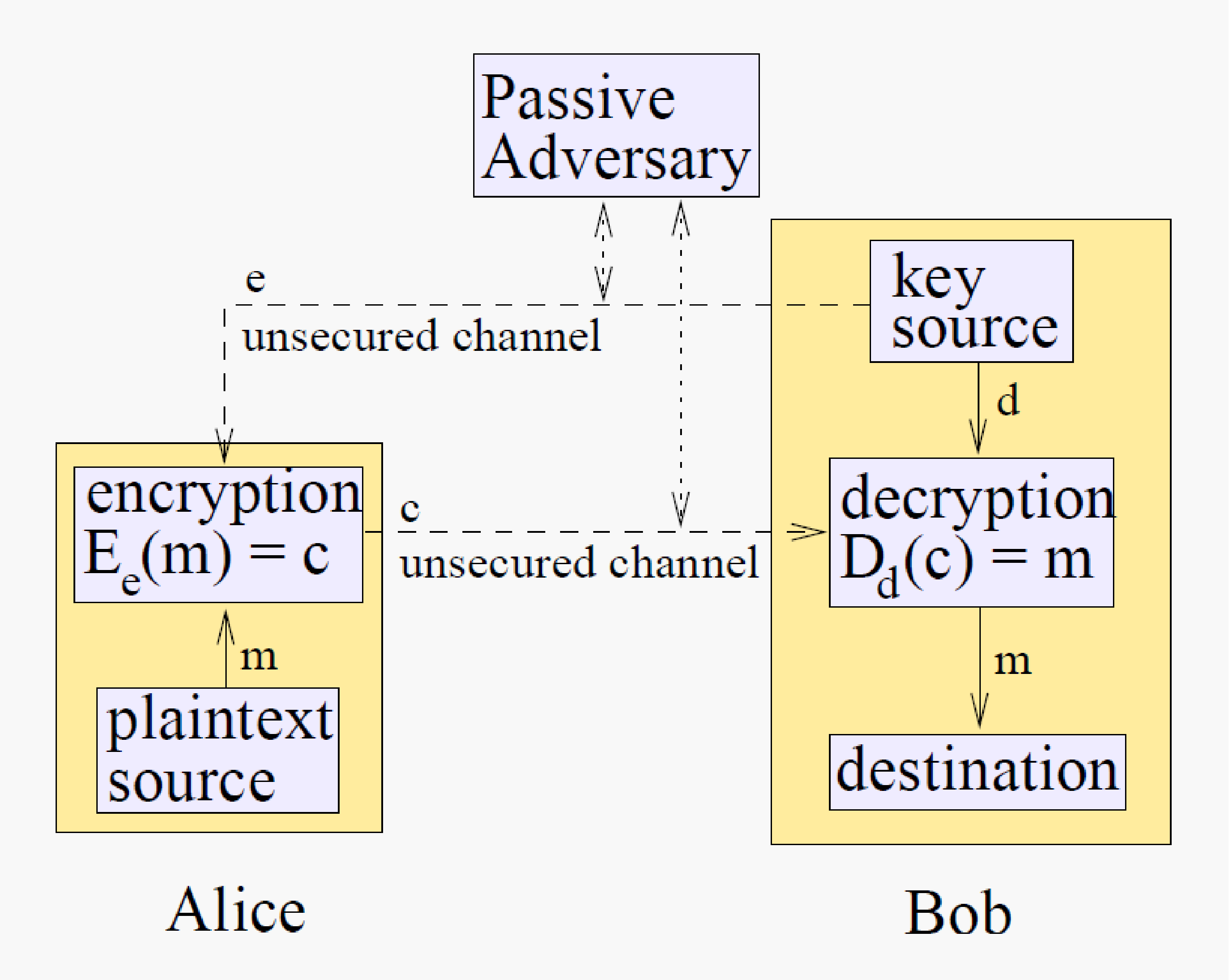
Asymmetric (Public-key) Encryption

- Public-key cryptography is based on two keys: e and d
  - Scheme is designed so that, given a pair (E_e, D_d),
    - Knowing E_e, it is infeasible
    - Given $c \in C$ to find an $m \in M$ where $E_e(m) = c$
  - This implies it is infeasible to determine d from e
  - E_e constitutes a trapdoor one-way function with trapdoor d
- Public key e can be public information



Asymmetric (Public-key) Encryption

- When Alice can determine the *message authenticity* of *e*, public-key cryptography provides her a *confidential* channel to Bob



Asymmetric (Public-key) Encryption

- Published after 1976 challenge by Diffie and Hellman
- RSA named after inventors: Rivest, Shamir, Adleman, 1978
- Security comes from difficulty of factoring large numbers
  - Keys are functions of a pairs of large ($\geq$ 100 digits) prime numbers
- Most popular public-key algorithm
- Used in many applications, e.g., PGP, PEM, SSL, ...
- Requires some basic number theory to appreciate

Number Theory: Prime Numbers

- Numbers

$$N = \{0,1,2,\ldots\}$$
$$Z = \{0,1,-1,\ldots\}$$
$$\text{Primes} = \{2,3,5,7,\ldots\}$$

- Every $n \in N$ has a unique set of prime factors

Number Theory: Prime Numbers

- Numbers

$$N = \{0,1,2,\ldots\}$$
$$Z = \{0,1,-1,\ldots\}$$
$$\text{Primes} = \{2,3,5,7,\ldots\}$$

- Every $n \in N$ has a unique set of prime factors
  - $60 = 2^2 \times 3 \times 5$

Number Theory: Prime Numbers

- Numbers

$$N = \{0,1,2,\ldots\}$$
$$Z = \{0,1,-1,\ldots\}$$
$$\text{Primes} = \{2,3,5,7,\ldots\}$$

- Every $n \in N$ has a unique set of prime factors
  - $60 = 2^2 \times 3 \times 5$
- Multiplying numbers is easy, factoring numbers appears hard
  - We cannot factor most numbers with more than 1024 bits

Number Theory: Division/Remainder/Modulo

- *Divisors*: $a \neq 0$ divides b (written $a|b$) if $\exists m.\, ma = b$
    - Examples: $3|6$, $3 \nmid 7$, $3 \nmid 10$
- $\forall a, n. \exists q, \quad r. a = q \times n + r$ where $0 \leq r < n$
    - Here r is the *remainder*, and we write $a \bmod n = r$
    - Examples:

$$
\begin{array}{lll}
6 \ = 2 \times 3 + 0 & \rightarrow & 6 \bmod 3 \ = 0 \\
7 \ = 2 \times 3 + 1 & \rightarrow & 7 \bmod 3 \ = 1 \\
10 = 3 \times 3 + 1 & \rightarrow & 10 \bmod 3 = 1
\end{array}
$$

- $a, b \in Z$ are *congruent modulo n*, if $a \mathrm{mod} n = b \mathrm{mod} n$
    - We write this as $a \equiv b (\mathrm{mod} n)$
    - Example: $7 \equiv 10 (\mathrm{mod} 3)$

Number Theory: GCD

- For $a, b \in N, \gcd(a, b)$ denotes *greatest common divisor*
  - Example: $60 = 2^2 \times 3 \times 5$, $14 = 2 \times 7$, $\gcd(60,14) = 2$
- $a, b \in N$ are *relatively prime* if $\gcd(a, b) = 1$
- GCD can be computed quickly using Euclid's algorithm
  - Examples:

$$\begin{aligned} \gcd(60,14) &: 60 = 4 \times 14 + 4 \\ \gcd(14,4) &: 14 = 3 \times 4 + 2 \\ \gcd(4,2) &: 4 \quad = 2 \times 2 + 0 \end{aligned}$$

- With extended version, one can compute $x, y \in Z$ where

$$\gcd(a, b) = xa + yb$$

Here $2 = 14 - 3 \times 4 = 14 - 3(60 - 4 \times 14) = -3 \times 60 + 13 \times 14$

Number Theory: Inverse

- Suppose that $a, b \in Z$ are relatively prime. There is a $c \in Z$ satisfying $bc \bmod a \ = 1$, i.e., we can compute $b^{-1} \bmod a$
- Proof:
  - From extended Euclidean Algorithm, exists $x, y \in Z$ where
$$1 = ax + by$$
  - Now consider the two sides modulo a. Since $a|ax$, we have $by \bmod a = 1$
  - Assertion follows with $c \coloneqq y$
- Example: $4^{-1} \bmod 7$
  - From Euclidean Algorithm: $1 = 7 \times (-1) + 4 \times 2$
  - Hence solution c is 2
  - Check: $4 \times 2 \bmod 7 = 1$

RSA Algorithms

- Generate a public/private key pair
  - Generate two large distinct primes p and q
  - Compute $n = pq$ and $\Phi = (p-1)(q-1)$
  - Select an e, $1 < e < \Phi$, relatively prime to Φ
  - Compute the unique integer d, $1 < d < \Phi$ where $ed \bmod \Phi = 1$
  - Return public key (n, e) and private key d
- Encryption with key (n, e)
  - Represent the message as an integer $m \in \{0, \dots, n-1\}$
  - Compute $\mathrm{c} = m^e \bmod n$
- Decryption with key d
  - Compute $\mathrm{m} = c^d \bmod n$

An RSA Example

- Let $p = 47$ and $q = 71$, then $n = pq = 3337$
- Encryption key e must have no factors in common with
$$\Phi = (p-1)(q-1) = 46 * 70 = 3220$$
- Choose $e = 79$ (randomly, $1 < e < \Phi$, relatively prime to Φ)
- Compute the unique integer $d = 79^{-1} \bmod 3220 = 1019$
- Publish e and n, keep private key d secret, discard p and q
- Break message m into small blocks, e.g., $m = 688\ 232\ 687\ 966\ 668$
- Compute $\mathrm{c} = m^e \bmod n$ blockwise. E.g., $c_1 = 688^{79} \bmod 3337 = 1570$
- To decrypt: Compute $m_1 = 1570^{1019} \bmod 3337 = 688$

RSA Security

- Computation of secret key d given (n, e)
  - As difficult as factorization
  - If we can factor $n = pq$, then we can compute $\Phi = (p-1)(q-1)$, hence $d \equiv e^{-1} \bmod \Phi$
- No known polynomial time algorithm
  - But given progress in factoring, n should have at least 1024 bits
- Computation of m, given c, and (n, e)
  - Computation of e^{th} root
  - Unclear (= no proof) whether it is necessary to compute d, i.e., to factorize n
- Progress in number theory could make RSA insecure
  - Or, quantum computers!

RSA Security

- Computation of secret key d given (n, e)
  - As difficult as factorization
  - If we can factor $n = pq$, then we can compute $\Phi = (p-1)(q-1)$, hence $d \equiv e^{-1} \mathrm{mod} \Phi$
- No known polynomial time algorithm
  - But given progress in factoring, n should have at least 1024 bits
- Computation of m, given c, and (n, e)
  - Computation of e^{th} root
  - Unclear (= no proof) whether it is necessary to compute d, i.e., to factorize n
- Progress in number theory could make RSA insecure
  - Or, quantum computers!

- What does this say about our modern IT (security) infrastructures?

Note:
Symmetric and Asymmetric Encryption

- Note on actual implementations
  - Usually symmetric encryption is computational less complex (i.e., faster)
  - Real-world systems often based on
    - Asymmetric keys-pair as long-term key
    - Symmetric session key
    - Long-term key is used for encrypting session key

Recommendations: Symmetric and Asymmetric Encryption

- Recommendations
  - Check: https://www.keylength.com/en/4/
  - Most up-to-date

Date	Security Strength	Symmetric Algorithms	Factoring Modulus	Discrete Logarithm Key	Discrete Logarithm Group	Elliptic Curve	Hash (A)	Hash (B)
Legacy (1)	80	2TDEA	1024	160	1024	160	SHA-1 (2)	
2019 - 2030	112	(3TDEA) (3) AES-128	2048	224	2048	224	SHA-224 SHA-512/224 SHA3-224	
2019 - 2030 & beyond	128	AES-128	3072	256	3072	256	SHA-256 SHA-512/256 SHA3-256	SHA-1 KMAC128
2019 - 2030 & beyond	192	AES-192	7680	384	7680	384	SHA-384 SHA3-384	SHA-224 SHA-512/224 SHA3-224
2019 - 2030 & beyond	256	AES-256	15360	512	15360	512	SHA-512 SHA3-512	SHA-256 SHA-512/256 SHA-384 SHA-512 SHA3-256 SHA3-384 SHA3-512 KMAC256

Recommendations:
Symmetric and Asymmetric Encryption

- Symmetric
  - 56 bits are breakable by brute force, e.g., against DES
  - NIST: Triple DES (with 112) and AES with at least 128 bits considered secure until 2013 (actually even longer)
  - BSI: AES with at least 128 bits considered secure until 2021 (even longer)
  - Usually 256-bit AES is recommended
- Asymmetric
  - 1024-bit RSA keys considered equivalent to 80-bit symmetric keys
  - NIST: 2048 RSA considered secure until 2030
  - BSI: 3072 RSA considered secure until 2021 (even longer)
  - Elliptic-curve cryptography appears secure with shorter keys, e.g., 256-bits
  - Assuming no relevant math or technical breakthroughs

Reading List

- Ross J. Anderson. Security Engineering: A Guide to Building Dependable Distributed Systems. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001.
  - The complete book is available at: http://www.cl.cam.ac.uk/~rja14/book.html
- Alfred J. Menezes, Scott A. Vanstone, and Paul C. Van Oorschot. Handbook of Applied Cryptography. CRC Press, Inc., Boca Raton, FL, USA, 5th edition, 2001.
  - The complete book is available at: http://cacr.uwaterloo.ca/hac/
- Bruce Schneier. Applied Cryptography. John Wiley & Sons, Inc., 2nd edition, 1996.

Thanks for your attention!

- Any questions or remarks?