Introduction

CSE 5351: Introduction to cryptography

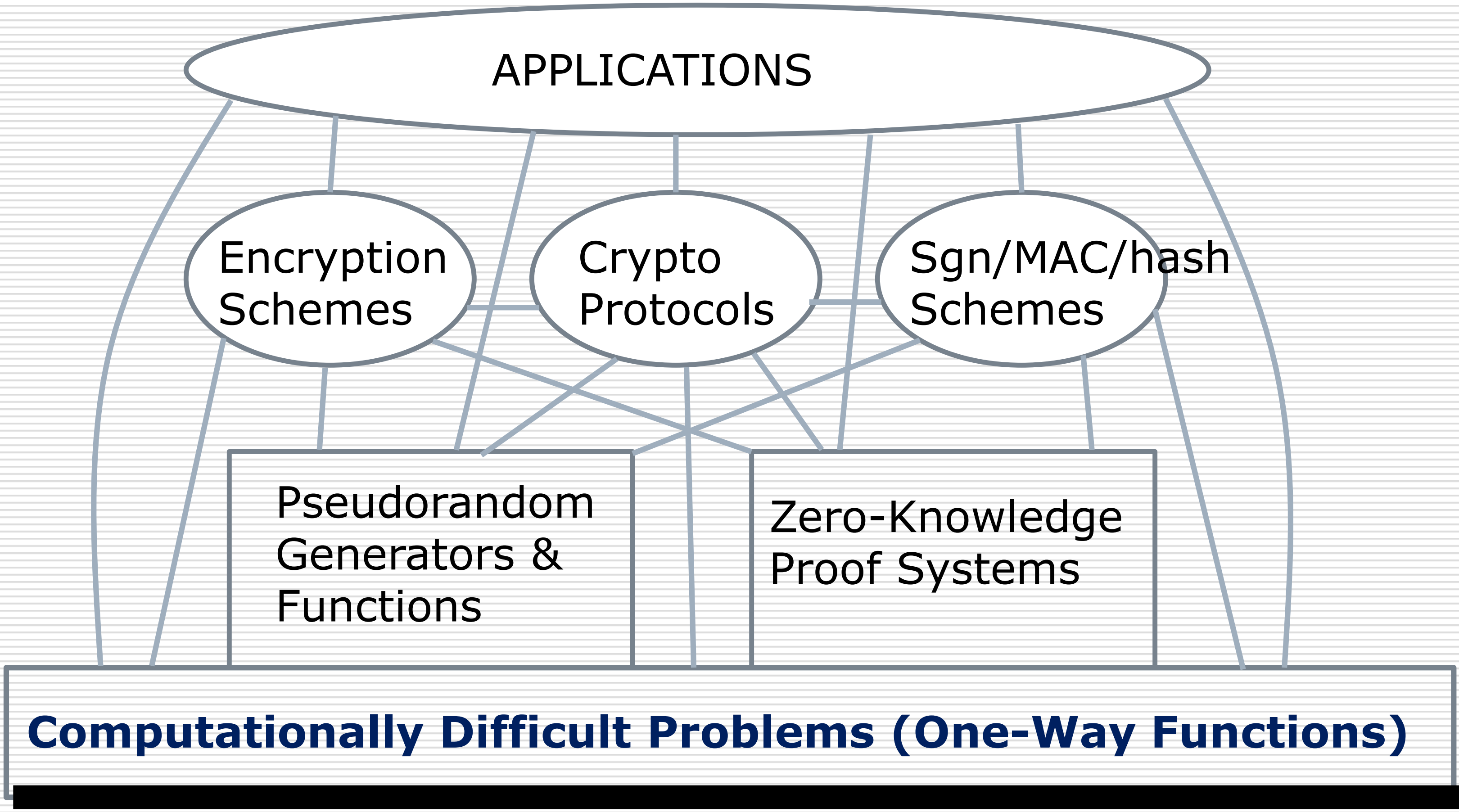
Reading assignment:

Chapter 1 of Katz & Lindell

Cryptography

- Merriam-Webster Online Dictionary:
  1. secret writing
  2. the enciphering and deciphering of messages in secret code or cipher.
- Modern cryptography is more than secret writing.

A Structural View of Cryptography

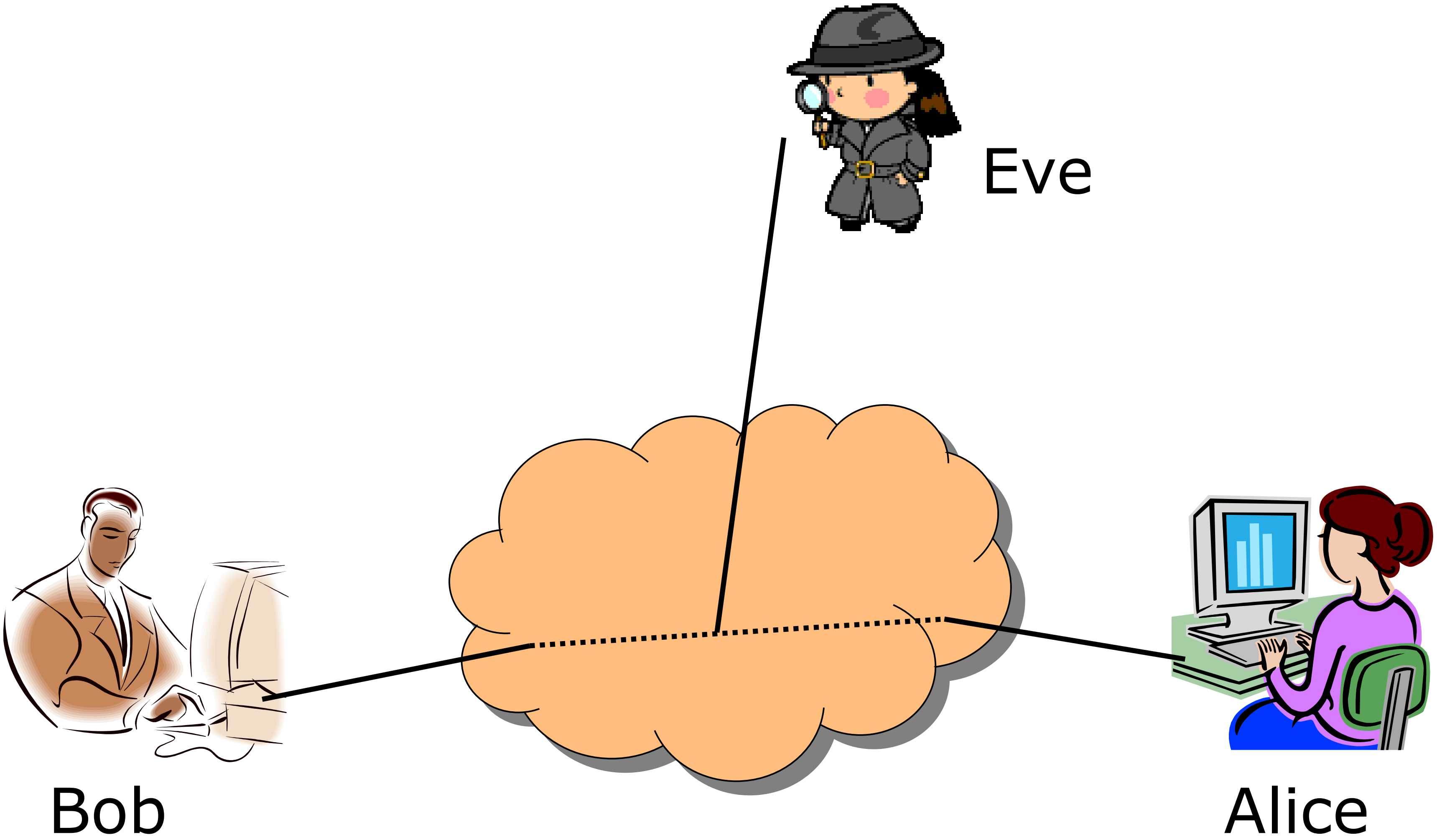


Basic objectives of cryptography

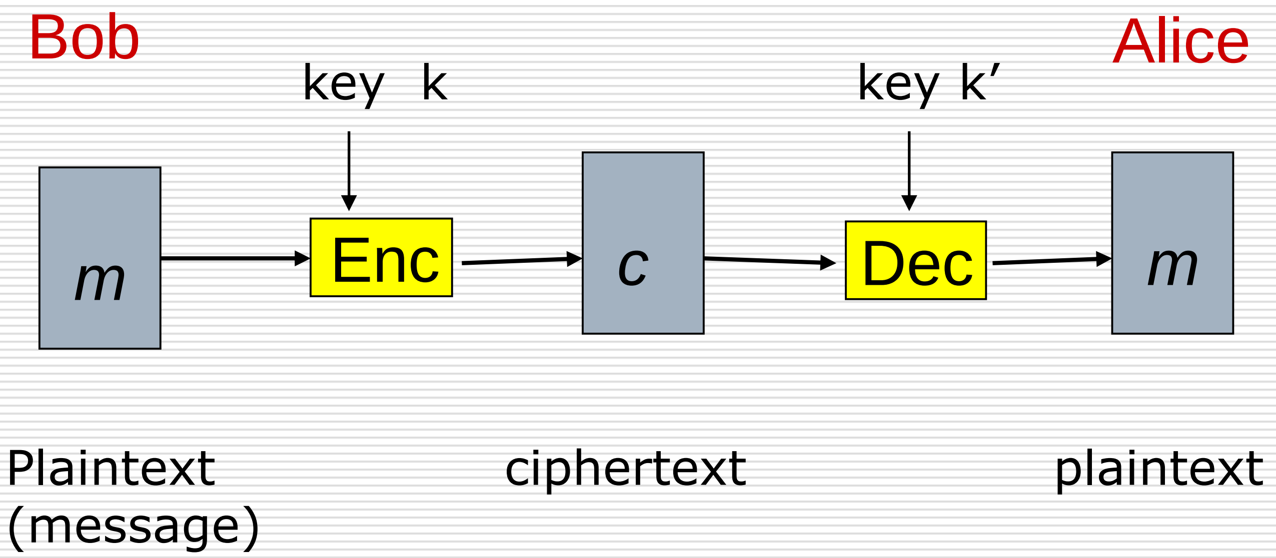
- Protecting data privacy (secret writing)
- Authentication:
  - Message authentication: allowing the recipient to check if a received message has been modified.
  - Data origin authentication: allowing the recipient to verify the origin of a received message.
  - Entity authentication: allowing the entities of a (connection-oriented) communication to authenticate each other.
- Non-repudiation: to prevent the sender from later denying that he/she sent the message.

Main characters of cryptography

- Alice
- Bob
- Eve (eavesdropper, adversary)


Eve
Bob
Alice

Encryption and secrecy



Encryption and secrecy

- Encryption protects secrecy of transmitted messages
  - Encryption Enc_k : plaintext $m \rightarrow$ ciphertext c
  - Decryption $\mathrm{Dec}_{k'}$: ciphertext $c \rightarrow$ plaintext m
  - Encryption key: k
  - Decryption key: k'

$$\left.\begin{array}{l} k \\ k' \end{array}\right\} \text{same or different}$$

Private-key encryption

- Also called symmetric-key encryption
- Encryption key k = decryption key k'
- Dec(k, Enc(k,m)) = m
- Or, $\mathrm{Dec}_k(\mathrm{Enc}_k(m)) = m$

Example: Caesar's shift cipher

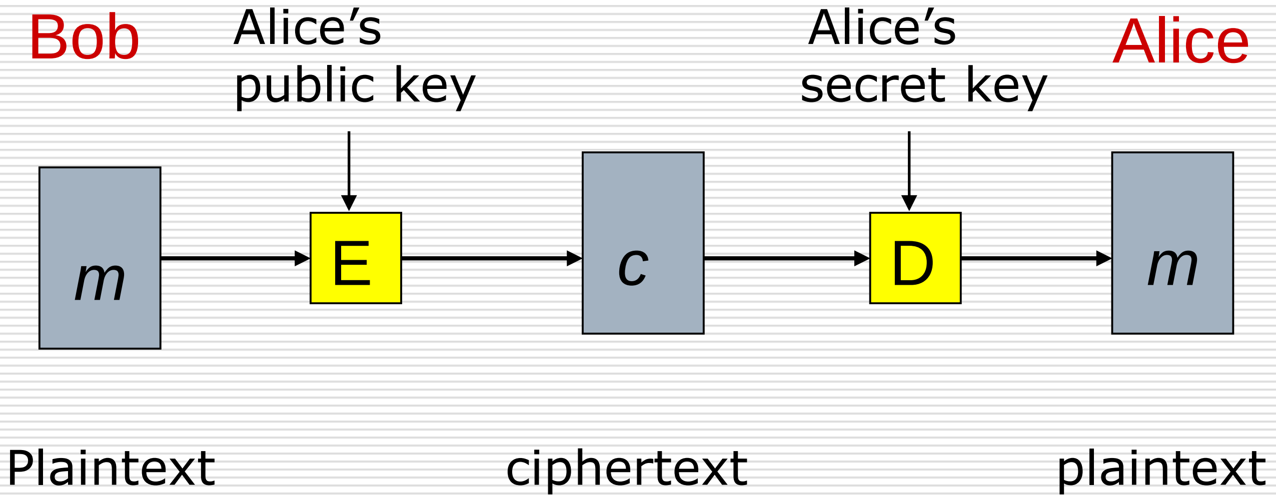
- Plaintext: a sequence of English characters

  $m = m_1 m_2 \ldots m_t$
- Each character represented as an integer in 0-25
- Key k: an integer in 0-25
- $\mathrm{Enc}_k(m) = c = c_1 c_2 \ldots c_t$ *where* $c_i = [(m_i + k) \bmod 26]$
- $\mathrm{Dec}_k(c) = m = m_1 m_2 \ldots m_t$ where $m_i = [(c_i - k) \bmod 26]$
- Example: Enc_3(ohio) = rklr Dec_3(rklr) = ohio

Public-key encryption

- Also called asymmetric encryption
- Using a pair of keys (*pk*, *sk*)
  - *pk* is public, known to everyone (who wishes to know)
  - *sk* is secret, known only to the key's owner (say Alice)
- From *pk*, it is hard to derive *sk*.
- $\mathrm{Dec}_{sk}(\mathrm{Enc}_{pk}(m)) = m$.

Public-key Encryption



Example: RSA

- Public key $pk = (N, e)$
- Secret key $sk = (N, d)$
- Encryption: $\mathrm{Enc}_{pk}(m) = \left[m^e \bmod N \right]$
- Decryption: $\mathrm{Dec}_{sk}(c) = \left[c^d \bmod N \right]$

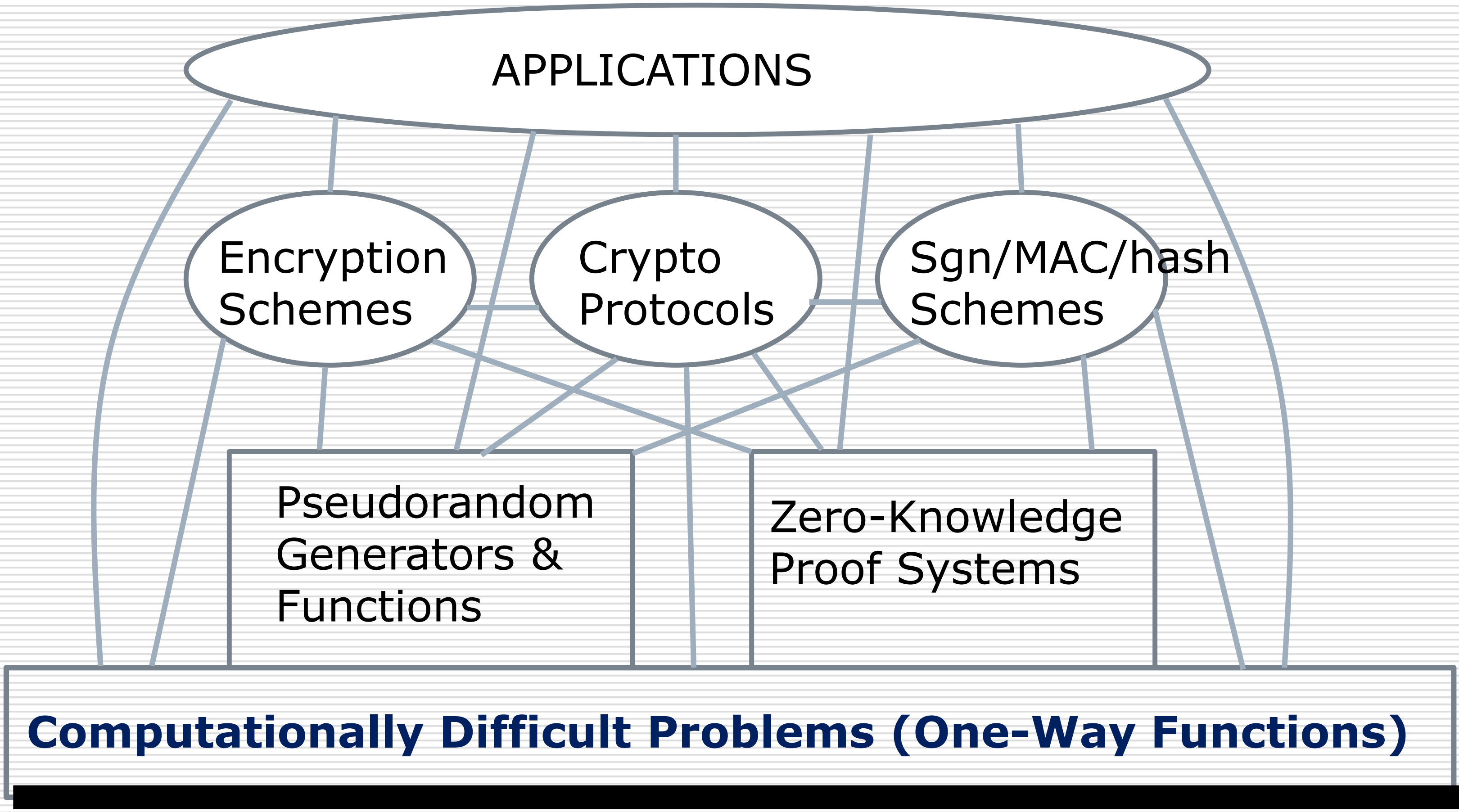
Message authentication codes

- Ensuring data integrity using private keys.
- Alice and Bob share a private key k.
- Alice sends to Bob the augmented message (m, x), where $x = \mathrm{MAC}_k(m)$.
- Bob on receiving (m', x'), checks if $x' = \mathrm{MAC}_k(m')$. If so, accepts m' as authentic.

Digital signatures

- Ensuring data integrity and non-repudiation using public-key methods
- $s = \mathrm{Sign}_{sk}(m)$
- $\mathrm{Verify}_{pk}(m', s') =$ true or false.
- Hash-then-sign: $s = \mathrm{Sign}_{sk}(h(m))$, where h is a cryptographic hash function.

A Structural View of Cryptography



Pseudorandom generators (1)

- Randomness and security of cryptosystems are closely related.
- Vernam's one-time pad encryption scheme:
  - To encrypt a message *m* (a string of bits)
  - Randomly generate a bit string *k*
  - Encrypt m as $c = m \oplus k$ bit by bit
  - *c* looks random to anyone not knowing the key *k*.

Pseudorandom generators (2)

- Expensive to generate truly random bits.
- Psuedorandom generators are algorithms that, on input a short random bit string, generate a longer, random-like bit string.

Cryptographic primitives

- These are often regarded as basic cryptographic primitives:
  - Pseudorandom generators/functions
  - Encryption schemes
  - Cryptographic hash functions
  - MACs, digital signatures
- They are often used as building blocks to build cryptographic protocols.

Cryptographic protocols

- A cryptographic protocol:
  - Involves two or more parties
  - Often combines different primitives
  - Accomplishes a more sophisticated task, e.g., tossing a coin over the phone

Example cryptographic protocol

- Protocol for user identification
  - using a digital signature scheme
  - Alice has a key pair (pk, sk)
- Alice $\rightarrow$ Bob: “I’m Alice”
- Alice $\leftarrow$ Bob: a random challenge c
- Alice $\rightarrow$ Bob: a response $s = \mathrm{Sign}_{sk}(c)$
- Bob checks if $\mathrm{Verify}_{pk}(c,s) = \mathrm{true}$

Is this protocol secure?

- Suppose Bob has a key pair (pk, sk)

- Alice → Bob: “I’m Alice”
- Alice ← Bob: “What’s your password?”
- Alice → Bob: a response c = $\text{Enc}_{pk}(m)$, where m is Alice’s password
- Bob checks if $\text{Dec}_{sk}(c)$ is correct.

One-way functions

- Modern cryptosystems are based on (trapdoor) one-way functions and difficult computational problems.
- A function *f* is one-way if it is easy to compute, but hard to invert.
  - Easy to compute: $x \xrightarrow{f} f(x)$
  - Hard to compute: $x \xleftarrow{f^{-1}} f(x)$
- Trapdoor: some additional information that makes f^{-1} easy to compute.

“Assumed” one-way functions

- No function has been proved one-way.
- Some functions are believed to be one-way.
- For example:
  - Integer multiplication
  - Discrete exponentiation
  - Modular powers

"Assumed" one-way functions

- Integer multiplication:

$$f(x,y) = x \cdot y \qquad \left(x,\ y:\ \text{large primes}\right)$$

- Discrete exponentiation:

$$f(x) = b^x \bmod n \qquad \left(x:\ \text{integers},\ 1 < x < n\right)$$

- Modular powers:

$$f(x) = x^b \bmod n \qquad \left(x:\ \text{integers},\ 1 < x < n\right)$$

Cryptanalysis

- Science of studying attacks against cryptographic schemes.
- Kerkhoff’s principle: the adversary knows all details about a cryptosystem except the secret key.
- Cryptography + Cryptanalysis = Cryptology

Attacks on encryption schemes

- Attacks are different in
  - Objectives: e.g. to obtain partial information about a plaintext, to fully decipher it, or to obtain the secret key
  - Levels of computing power
  - Amount of information available
- When studying the security of an encryption scheme, we need to specify the type of attacks.

Different types of attacks

- Different types of attacks (classified by the amount of information that may be obtained by the attacker):
  - Ciphertext-only attack
  - Known-plaintext attack
  - Chosen-plaintext attack (possibly adaptively)
  - Chosen-ciphertext attack (possibly adaptively)
  - Chosen plaintext & ciphertext attack (possibly adaptively)

Ciphertext-only attacks

- Given: a ciphertext *c*
- Q: what is its plaintext of *c*?
- An encryption scheme at least must be able to resist this type of attacks.

Known-plaintext attacks

- Given: (m_1,c_1), (m_2,c_2), …, (m_k,c_k) and a new ciphertext c.
- Q: what is the plaintext of c?

Chosen-plaintext attacks

- Given: (m_1,c_1), (m_2,c_2), …, (m_k,c_k), where $m_1, m_2, \ldots, m_k$ are chosen by the adversary, and a new ciphertext c.
- Q: what is the plaintext of c?
- Adaptively-chosen-plaintext attack: $m_1, m_2, \ldots, m_k$ are chosen adaptively.

Chosen-ciphertext attacks

- Given: (m_1,c_1), (m_2,c_2), …, (m_k,c_k), where c_1, c_2, …, c_k are chosen b y the adversary; and a new ciphertext c.
- Q: what is the plaintext of c?
- Adaptively-chosen-ciphertext attack: c_1, c_2, …, c_k are chosen adaptively.

Different types of adversaries …

- Classified by the amount of computing resources available by the adversary:
  - The attacker has unbounded computing power
  - The attacker only has a polynomial amount of computing power (polynomial in some security parameter, typically the key length).

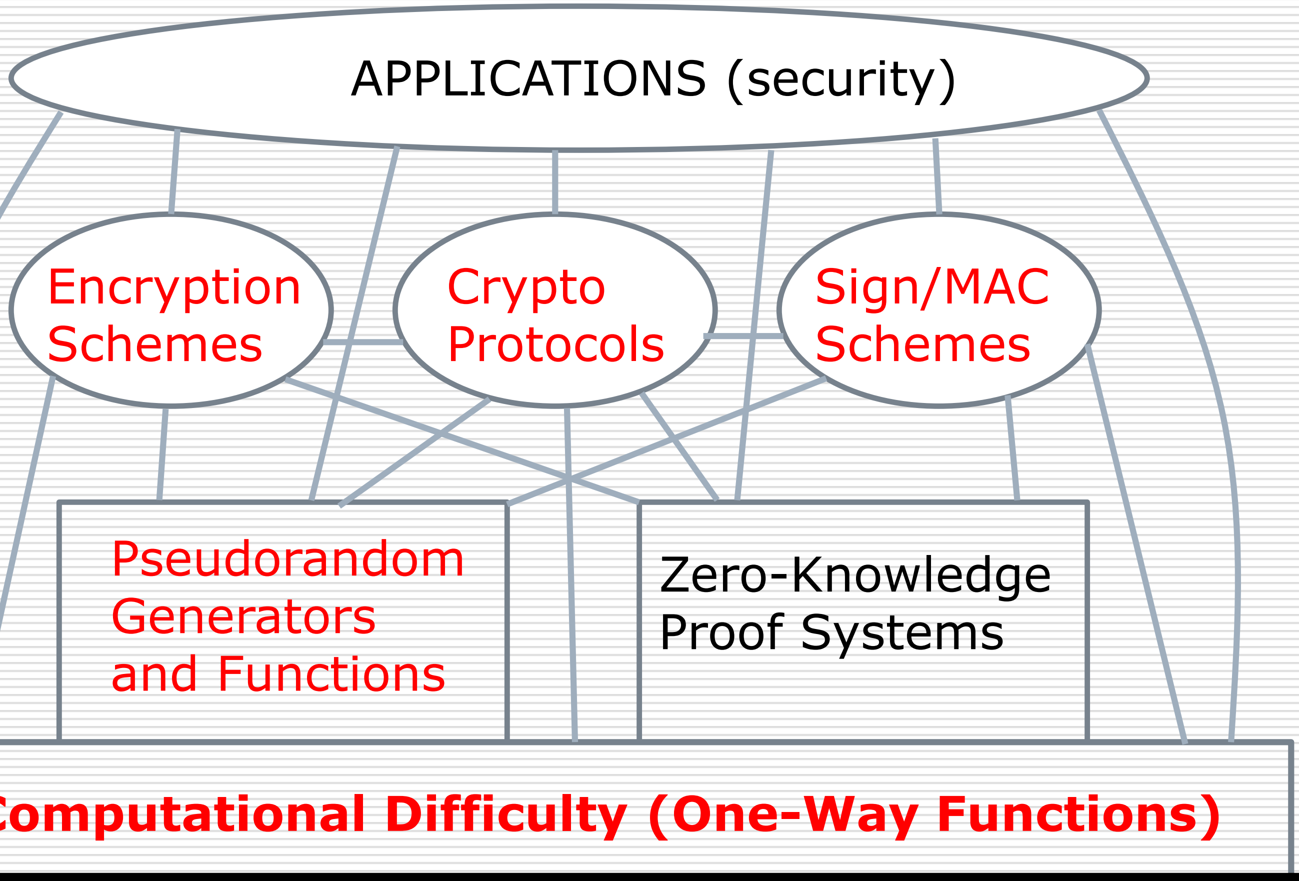
Unconditional security

- Secure even if the adversary has infinite computational resources (CPU time and memory storage).
- For example, Vernam's one-time pad is unconditionally secure against ciphertext-only attack.

Computational security

- Secure if the attacker has only polynomial amount of computational resources.
- For example, RSA is considered computationally secure; it may take thousands years to decipher a ciphertext.
  - Why is RSA not unconditionally secure?

This course:


APPLICATIONS (security)
Encryption Schemes
Crypto Protocols
Sign/MAC Schemes
Pseudorandom Generators and Functions
Zero-Knowledge Proof Systems
Computational Difficulty (One-Way Functions)