Cryptographic Primitives in Blockchain

Dr. Keyur Parmar

Indian Institute of Information Technology, Vadodara

Outline

Introduction

What is a fingerprint? Is it useful? How?

Which attributes of a fingerprint make it useful?

Which attributes of a fingerprint make it useful?

- Uniqueness (i.e., unique to each person)

Which attributes of a fingerprint make it useful?

- Uniqueness (i.e., unique to each person)
- Persistence (i.e., does not change over time)

Which attributes of a fingerprint make it useful?

- Uniqueness (i.e., unique to each person)
- Persistence (i.e., does not change over time)
- Size (i.e., very small as compared to the person)

Goal: To generate fingerprints for digital data, e.g., a fingerprint for a text message or a video.

Goal: To generate fingerprints for digital data, e.g., a fingerprint for a text message or a video.

Is it difficult to generate a fingerprint for digital data?

Cryptographic Hash Functions

What is a cryptographic hash function?

Cryptographic Hash Function

- Input:

Cryptographic Hash Function

- Input:
  - A variable-length data block M

Cryptographic Hash Function

- Input:
    - A variable-length data block M
- Output:

Cryptographic Hash Function

- Input:
  - A variable-length data block M
- Output:
  - A fixed-size hash code $h = H(M)$

Cryptographic Hash Function - SHA256 - Example-1

- Input (Text):

Cryptographic Hash Function - SHA256 - Example-1

- Input (Text):
  - Keyur

Cryptographic Hash Function - SHA256 - Example-1

- Input (Text):
  - Keyur
- Output (Hexadecimal):

Cryptographic Hash Function - SHA256 - Example-1

- Input (Text):
  - Keyur
- Output (Hexadecimal):
  - ABDCF0E8 41704B32 BFA2B901 F44CD8B4 A99DD384 D63A044F 552BF2AC EA0C46D3

Cryptographic Hash Function - SHA256 - Example-1

- Input (Text):
  - Keyur
- Output (Hexadecimal):
  - ABDCF0E8 41704B32 BFA2B901 F44CD8B4 A99DD384 D63A044F 552BF2AC EA0C46D3
- Output (Binary):

Cryptographic Hash Function - SHA256 - Example-1

- Input (Text):
  - Keyur
- Output (Hexadecimal):
  - ABDCF0E8 41704B32 BFA2B901 F44CD8B4 A99DD384 D63A044F 552BF2AC EA0C46D3
- Output (Binary):
  - 10101011 11011100 11110000 11101000 01000001 01110000 01001011 00110010 10111111 10100010 10111001 00000001 11110100 01001100 11011000 10110100 10101001 10011101 11010011 10000100 11010110 00111010 00000100 01001111 01010101 00101011 11110010 10101100 11101010 00001100 01000110 11010011

Cryptographic Hash Function - SHA256 - Example-2

- Input:

Cryptographic Hash Function - SHA256 - Example-2

- Input:
  - Ubuntu DVD (1.9 GB): ubuntu-18.04.2-desktop-amd64.iso

Cryptographic Hash Function - SHA256 - Example-2

- Input:
  - Ubuntu DVD (1.9 GB): ubuntu-18.04.2-desktop-amd64.iso
- Output (Hexadecimal):

Cryptographic Hash Function - SHA256 - Example-2

- Input:
  - Ubuntu DVD (1.9 GB): ubuntu-18.04.2-desktop-amd64.iso
- Output (Hexadecimal):
  - `22580B9F 3B186CC6 6818E60F 44C46F79 5D708A1A D86B9225 C458413B 638459C4`

Cryptographic Hash Function - SHA256 - Example-2

- Input:
  - Ubuntu DVD (1.9 GB): ubuntu-18.04.2-desktop-amd64.iso
- Output (Hexadecimal):
  - `22580B9F 3B186CC6 6818E60F 44C46F79 5D708A1A D86B9225 C458413B 638459C4`
- Output (Binary):

Cryptographic Hash Function - SHA256 - Example-2

- Input:
  - Ubuntu DVD (1.9 GB): ubuntu-18.04.2-desktop-amd64.iso
- Output (Hexadecimal):
  - 22580B9F 3B186CC6 6818E60F 44C46F79 5D708A1A D86B9225 C458413B 638459C4
- Output (Binary):
  - 00100010 01011000 00001011 10011111 00111011 00011000 01101100 11000110 01101000 00011000 11100110 00001111 01000100 11000100 01101111 01111001 01011101 01110000 10001010 00011010 11011000 01101011 10010010 00100101 11000100 01011000 01000001 00111011 01100011 10000100 01011001 11000100

Applications of Cryptographic Hash Functions

Applications of Cryptographic Hash Functions

- Message authentication

Applications of Cryptographic Hash Functions

- Message authentication
- Digital signature

Applications of Cryptographic Hash Functions

- Message authentication
- Digital signature
- One-way password file (e.g., `/etc/shadow`)

Applications of Cryptographic Hash Functions

- Message authentication
- Digital signature
- One-way password file (e.g., `/etc/shadow`)
- Pseudo-random number generator (PRNG)

Applications of Cryptographic Hash Functions

- Message authentication
- Digital signature
- One-way password file (e.g., `/etc/shadow`)
- Pseudo-random number generator (PRNG)
- Blockchain

How to design a cryptographically secure hash function?

Requirements for a Cryptographic Hash Functions

Requirements for a Cryptographic Hash Functions

1. Variable input size

Requirements for a Cryptographic Hash Functions

1. Variable input size
2. Fixed output size

Requirements for a Cryptographic Hash Functions

1. Variable input size
2. Fixed output size
3. Efficiency

Requirements for a Cryptographic Hash Functions

1. Variable input size
2. Fixed output size
3. Efficiency
4. One-way property (Preimage resistant)

Requirements for a Cryptographic Hash Functions

1. Variable input size
2. Fixed output size
3. Efficiency
4. One-way property (Preimage resistant)
5. Second preimage resistant (Weak collision resistant)

Requirements for a Cryptographic Hash Functions

1. Variable input size
2. Fixed output size
3. Efficiency
4. One-way property (Preimage resistant)
5. Second preimage resistant (Weak collision resistant)
6. Collision resistant (Strong collision resistant)

Requirements for a Cryptographic Hash Functions

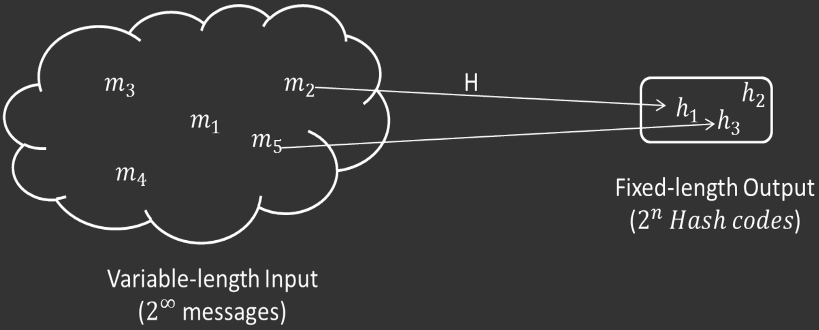
- Preimage - For a hash code $h = H(m)$, m is the preimage of h.

Requirements for a Cryptographic Hash Functions

- Preimage - For a hash code $h = H(m)$, m is the preimage of h.
- As there is a many-to-one mapping, for any given hash code h, there will be multiple preimages.

Requirements for a Cryptographic Hash Functions

- Preimage - For a hash code $h = H(m)$, m is the preimage of h.
- As there is a many-to-one mapping, for any given hash code h, there will be multiple preimages.



Requirements for a Cryptographic Hash Functions

- Let's assume that the length of the hash code is n bits, and the hash function H takes as input messages of length b bits with $b > n$.

Requirements for a Cryptographic Hash Functions

- Let's assume that the length of the hash code is n bits, and the hash function H takes as input messages of length b bits with $b > n$.
- The total number of possible hash codes is 2^n, the total number of possible messages is 2^b.

Requirements for a Cryptographic Hash Functions

- Let's assume that the length of the hash code is n bits, and the hash function H takes as input messages of length b bits with $b > n$.
- The total number of possible hash codes is 2^n, the total number of possible messages is 2^b.
- On average, each hash code corresponds to 2^{b-n} preimages.

Requirements for a Cryptographic Hash Functions

- Let's assume that the length of the hash code is n bits, and the hash function H takes as input messages of length b bits with $b > n$.
- The total number of possible hash codes is 2^n, the total number of possible messages is 2^b.
- On average, each hash code corresponds to 2^{b-n} preimages.
- If the input is arbitrarily large, then the number of preimages per hash code is arbitrarily large. Hence, there will be collisions.

Requirements for a Cryptographic Hash Functions

- Let's assume that the length of the hash code is n bits, and the hash function H takes as input messages of length b bits with $b > n$.
- The total number of possible hash codes is 2^n, the total number of possible messages is 2^b.
- On average, each hash code corresponds to 2^{b-n} preimages.
- If the input is arbitrarily large, then the number of preimages per hash code is arbitrarily large. Hence, there will be collisions.
- Is it possible to design a collision resistant hash function?

One-way Property (Preimage Resistant)

For any given hash code h, it is computationally infeasible to find the input message m such that $H(m) = h$.

Second Preimage Resistant (Weak Collision Resistant)

For any given message m_1, it is computationally infeasible to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

Collision Resistant (Strong Collision Resistant)

It is computationally infeasible to find a pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

Security Attacks

- Two categories of attacks on hash functions.
  - Brute-force attacks - depend only on the bit length. (e.g., bit length of the hash code)
  - Cryptanalysis - depends on design flaw(s) of a particular hash function.

- Given the following entry of the "/etc/shadow" file (of Ubuntu 16.04), find the password of the user, "Alice" which contains [A-Za-z1-9] (Uppercase and/or lowercase letters and/or numbers).

```
Alice:$6$YIjqPaC8$ckYvhWkRkymmv/twBcANwa/L
WNjLsAdCHRToK3G9GIImPUWERnmFW2bUoOmLH
zUJ2tGr433QaOnHLdjDjc4Bs/:17648:0:99999:7:::
```

- Which property of the hash function you have to attack to find the password of Alice? Why?

Preimage Attack - A Brute-force Approach

For any given hash code h, it is computationally infeasible to find the input message m such that $H(m) = h$.

- Choose values of m' at random and compute $H(m')$. Continue until a collision occurs, i.e., $H(m') = h$.

Preimage Attack - A Brute-force Approach

For any given hash code h, it is computationally infeasible to find the input message m such that $H(m) = h$.

- Choose values of m' at random and compute $H(m')$. Continue until a collision occurs, i.e., $H(m') = h$.
- What is the level of effort required to perform the preimage attack, i.e., to find the message m such that $H(m) = h$?

Preimage Attack - A Brute-force Approach

For any given hash code h, it is computationally infeasible to find the input message m such that $H(m) = h$.

- Choose values of m' at random and compute $H(m')$. Continue until a collision occurs, i.e., $H(m') = h$.
- What is the level of effort required to perform the preimage attack, i.e., to find the message m such that $H(m) = h$?
- For an n-bit hash code, the level of effort is proportional to 2^n.

Second Preimage Attack - A Brute-force Approach

For any given message m_1, it is computationally infeasible to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

- Choose values of m_2 at random $(m_2 \neq m_1)$ and compute $H(m_2)$. Continue until a collision occurs, i.e., $H(m_2) = H(m_1)$.

Second Preimage Attack - A Brute-force Approach

For any given message m_1, it is computationally infeasible to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

- Choose values of m_2 at random $(m_2 \neq m_1)$ and compute $H(m_2)$. Continue until a collision occurs, i.e., $H(m_2) = H(m_1)$.
- What is the level of effort required to perform the second preimage attack, i.e., to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$?

Second Preimage Attack - A Brute-force Approach

For any given message m_1, it is computationally infeasible to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

- Choose values of m_2 at random $(m_2 \neq m_1)$ and compute $H(m_2)$. Continue until a collision occurs, i.e., $H(m_2) = H(m_1)$.
- What is the level of effort required to perform the second preimage attack, i.e., to find another message m_2 such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$?
- For an n-bit hash code, the level of effort is proportional to 2^n.

What should be the length of the hash code h to prevent the preimage/second preimage attack?

Collision Resistant Attack - A Brute-force Approach

It is computationally infeasible to find a pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

- What is the level of effort required to perform the collision resistant attack, i.e., to find the pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$?

Collision Resistant Attack - A Brute-force Approach

It is computationally infeasible to find a pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

- What is the level of effort required to perform the collision resistant attack, i.e., to find the pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$?
- The level of effort required is significantly less than the effort required for a preimage/second preimage attack. Why?

Collision Resistant Attack - A Brute-force Approach

It is computationally infeasible to find a pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

- What is the level of effort required to perform the collision resistant attack, i.e., to find the pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$?
- The level of effort required is significantly less than the effort required for a preimage/second preimage attack. Why?
    - Birthday Paradox!

Collision Resistant Attack - A Brute-force Approach

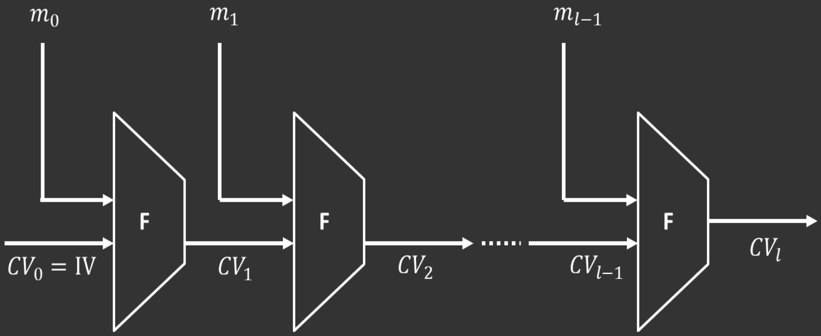
It is computationally infeasible to find a pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$.

- What is the level of effort required to perform the collision resistant attack, i.e., to find the pair of messages (m_1, m_2) such that $m_1 \neq m_2$ and $H(m_1) = H(m_2)$?
- The level of effort required is significantly less than the effort required for a preimage/second preimage attack. Why?
  - Birthday Paradox!
- For an n-bit hash code, the level of effort is **roughly** proportional to $2^{n/2}$.

What should be the length of the hash code h to prevent the preimage/second preimage/collision-resistant attack?

Merkle-Damgård Construction

- Most hash functions in use today follow the Merkle-Damgård structure.



- IV - Initial value
- m_i - i-th input block
- CV_i - Chaining variable
- CV_l - Hash code
- l - Number of input blocks
- F - Compression function

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
  - SHA-224

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
  - SHA-224
  - SHA-256

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
  - SHA-224
  - SHA-256
  - SHA-384

Cryptographic Hash Functions - Examples

- Secure Hash Algorithm (SHA)-2 (Year: 2002) and SHA-3 (Year: 2015)
    - SHA-224
    - SHA-256
    - SHA-384
    - SHA-512

Public-key Cryptography

What are the applications of public-key cryptography?

Is it difficult to distribute a key/secret between communicating parties?

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography
- Modern cryptography

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography
- Modern cryptography
    - Symmetric-key algorithms (AES, DES, etc.)

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography
- Modern cryptography
  - Symmetric-key algorithms (AES, DES, etc.)
    - Permutation and substitution

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography
- Modern cryptography
  - Symmetric-key algorithms (AES, DES, etc.)
    - Permutation and substitution
    - A secret key

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography
- Modern cryptography
  - Symmetric-key algorithms (AES, DES, etc.)
    - Permutation and substitution
    - A secret key
  - Public-key algorithms (RSA, ElGamal, etc.)

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography
- Modern cryptography
  - Symmetric-key algorithms (AES, DES, etc.)
    - Permutation and substitution
    - A secret key
  - Public-key algorithms (RSA, ElGamal, etc.)
    - Mathematical functions

Public-key (Asymmetric-keys) Cryptography

- Public-key cryptography - true revolution in cryptography
- Modern cryptography
  - Symmetric-key algorithms (AES, DES, etc.)
    - Permutation and substitution
    - A secret key
  - Public-key algorithms (RSA, ElGamal, etc.)
    - Mathematical functions
    - Two keys - one is public, the other is private/secret

Symmetric-key Cryptosystem - Encryption/Decryption

- m - Plaintext (Message)
- c - Ciphertext
- k - Secret-key shared between Alice and Bob

Public-key Cryptography

- The need for public-key cryptosystems...

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems
  - Digital signatures

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems
  - Digital signatures
- History

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems
  - Digital signatures
- History
  - Diffie and Hellman - 1976

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems
  - Digital signatures
- History
  - Diffie and Hellman - 1976
  - R. Merkle - 1975 (did not publish until 1978)

Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

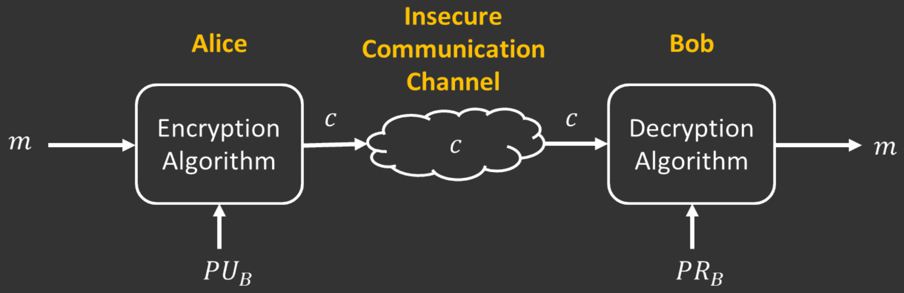
Public-key Cryptography

- The need for public-key cryptosystems...
  - Key distribution problems of symmetric-key cryptosystems
  - Digital signatures
- History
  - Diffie and Hellman - 1976
  - R. Merkle - 1975 (did not publish until 1978)
  - Bobby Inman (Director of the NSA) - Discovered at NSA in the mid-1960s

---

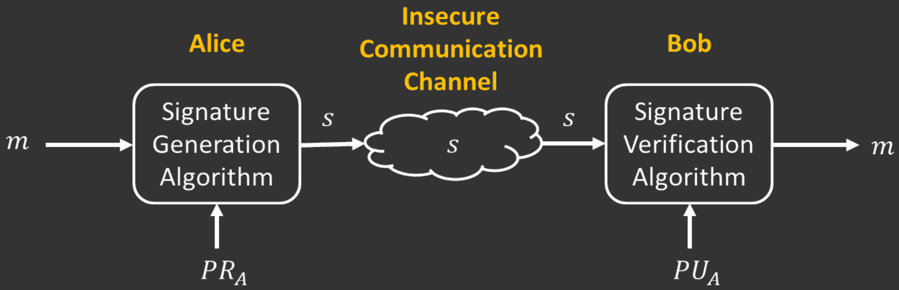
Diffie and Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 6, (1976).

Public-key Cryptosystem - Encryption/Decryption



- m - Plaintext (Message)
- c - Ciphertext
- PU_B - Bob's public-key
- PR_B - Bob's private-key

Public-key Cryptosystem - Digital Signature



- m - Plaintext (Message)
- c - Ciphertext
- PU_A - Alice's public-key
- PR_A - Alice's private-key

Applications for Public-key Cryptosystems

Applications for Public-key Cryptosystems

- Classification of public-key cryptosystems

Applications for Public-key Cryptosystems

- Classification of public-key cryptosystems
    - Encryption/decryption - Uses recipient's public key to encrypt the message.

Applications for Public-key Cryptosystems

- Classification of public-key cryptosystems
  - Encryption/decryption - Uses recipient's public key to encrypt the message.
  - Digital signature - Uses sender's private key to sign the message.

Applications for Public-key Cryptosystems

- Classification of public-key cryptosystems
  - Encryption/decryption - Uses recipient's public key to encrypt the message.
  - Digital signature - Uses sender's private key to sign the message.
  - Key exchange - Uses the private key(s) of sender and/or recipient.

Requirements for Public-key Cryptosystems

Requirements for Public-key Cryptosystems

- Key Generation (public/private keys) - should be feasible.

Requirements for Public-key Cryptosystems

- Key Generation (public/private keys) - should be feasible.
- Encryption - should be computationally feasible given the message M and the public key PU_k.

$$C = E(PU_k, M)$$

Requirements for Public-key Cryptosystems

- Key Generation (public/private keys) - should be feasible.
- Encryption - should be computationally feasible given the message M and the public key PU_k.

$$C = E(PU_k, M)$$

- Decryption - should be computationally feasible given the ciphertext C and the private key PR_k.

$$M = D(PR_k, C) = D[PR_k, E(PU_k, M)]$$

Requirements for Public-key Cryptosystems

- Given the public key PU_k, it must be computationally infeasible to determine the private key, PR_k.

Requirements for Public-key Cryptosystems

- Given the public key PU_k, it must be computationally infeasible to determine the private key, PR_k.
- Given the public key PU_k and a ciphertext C, it must be computationally infeasible to recover the original message M.

Is it difficult to design a public-key cryptosystem? If we have to design a public-key cryptosystem, what do we need?

Trapdoor one-way function

- Trap-door one-way function - Easy to compute in one direction and infeasible to compute in the other direction without trapdoor information.
- Trapdoor one-way function

$$
\begin{aligned}
C &= f_k(M) && \text{easy, if k and M are known} \\
M &= f_k^{-1}(C) && \text{easy, if k and C are known} \\
M &= f_k^{-1}(C) && \text{infeasible, if C is known but k is unknown}
\end{aligned}
$$

Here, "easy" means possible in polynomial time.

Public-key Cryptosystems - Attacks

- Brute-force attack - Use large keys.

Public-key Cryptosystems - Attacks

- Brute-force attack - Use large keys.
    - How much large (keys)?

Public-key Cryptosystems - Attacks

- Brute-force attack - Use large keys.
  - How much large (keys)? Large enough to make brute-force attack impractical and small enough to keep the encryption and decryption feasible.

Public-key Cryptosystems - Attacks

- Brute-force attack - Use large keys.
  - How much large (keys)? Large enough to make brute-force attack impractical and small enough to keep the encryption and decryption feasible.

- Find ways to compute the private key given the public key.

Public-key Cryptosystems - Attacks

- Brute-force attack - Use large keys.
    - How much large (keys)? Large enough to make brute-force attack impractical and small enough to keep the encryption and decryption feasible.
- Find ways to compute the private key given the public key.
- Known/Chosen plaintext attacks to derive the private key.

Digital Signature

What is a signature? Is it useful? How?

Goal: To generate a signature for digital data, e.g., sign a text message or a video.

Goal: To generate a signature for digital data, e.g., sign a text message or a video.

Is it difficult to sign digital data?

Digital Signature

- Masquerade - Insertion of messages into the network from a fraudulent source (e.g., messages, acknowledgments).

Digital Signature

- Masquerade - Insertion of messages into the network from a fraudulent source (e.g., messages, acknowledgments).
- Content modification - Changes to the contents of a message.

Digital Signature

- Masquerade - Insertion of messages into the network from a fraudulent source (e.g., messages, acknowledgments).
- Content modification - Changes to the contents of a message.
- Sequence modification - Modification of a sequence of messages communicated between parties.

Digital Signature

- Masquerade - Insertion of messages into the network from a fraudulent source (e.g., messages, acknowledgments).
- Content modification - Changes to the contents of a message.
- Sequence modification - Modification of a sequence of messages communicated between parties.
- Timing modification - Delay or replay of messages.

Digital Signature

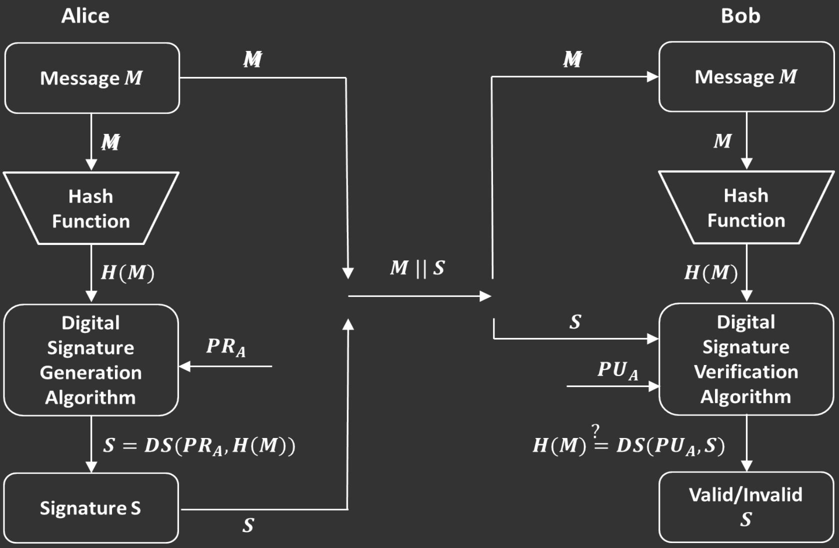
- Masquerade - Insertion of messages into the network from a fraudulent source (e.g., messages, acknowledgments).
- Content modification - Changes to the contents of a message.
- Sequence modification - Modification of a sequence of messages communicated between parties.
- Timing modification - Delay or replay of messages.
- Source repudiation - Denial of transmission of message by the source.

Digital Signature

- Masquerade - Insertion of messages into the network from a fraudulent source (e.g., messages, acknowledgments).
- Content modification - Changes to the contents of a message.
- Sequence modification - Modification of a sequence of messages communicated between parties.
- Timing modification - Delay or replay of messages.
- Source repudiation - Denial of transmission of message by the source.

If two communicating parties do not trust each other, digital signature enables them to communicate with each other securely.

Digital Signature



PR_A - Alice's Private Key PU_A - Alice's Public Key

Digital Signature - Algorithms

- Elgamal Digital Signature Scheme
- NIST Digital Signature Algorithm (DSA)
- Elliptic Curve Digital Signature Algorithm (ECDSA)
- Schnorr Digital Signature Scheme
- RSA-PSS Digital Signature Scheme

Number Theory

What is logarithm?

Logarithm

- The power to which a fixed number (base) must be raised to produce a given number.

$$a^x = b$$
$$x = \log_a b$$

Discrete Logarithm

$$a^x = b \text{ (mod } n)$$
$$x = \log_a b \text{ (mod } n)$$

- If n is a composite number, discrete log is not always possible.

Discrete Logarithm

$$a^x = b \ (\mathsf{mod}\ n)$$
$$x = \log_a b \ (\mathsf{mod}\ n)$$

- If n is a composite number, discrete log is not always possible.
- If n is a prime number, and a is a generator of the group, then discrete log exists.

Discrete Logarithm Problem (DLP)

- If p is a prime number, and g is a generator of $\mathbb{F}_p^*$. Let h be an element of the finite field $\mathbb{F}_p^*$.

Discrete Logarithm Problem (DLP)

- If p is a prime number, and g is a generator of $\mathbb{F}_p^*$. Let h be an element of the finite field $\mathbb{F}_p^*$.
- The Discrete Logarithm Problem (DLP) is the problem of finding an exponent x such that $g^x \equiv h$ (mod p)$)$.

Discrete Logarithm Problem (DLP)

- If p is a prime number, and g is a generator of $\mathbb{F}_p^*$. Let h be an element of the finite field $\mathbb{F}_p^*$.
- The Discrete Logarithm Problem (DLP) is the problem of finding an exponent x such that $g^x \equiv h$ (mod p)$)$.
- The number x is called the discrete logarithm of h base g, i.e., $log_g h$.

Discrete Logarithm Problem (DLP)

- If p is a prime number, and g is a generator of $\mathbb{F}_p^*$. Let h be an element of the finite field $\mathbb{F}_p^*$.
- The Discrete Logarithm Problem (DLP) is the problem of finding an exponent x such that $g^x \equiv h$ (mod p)).
- The number x is called the discrete logarithm of h base g, i.e., $log_g h$.

Given g, x, and p, computing $h = g^x$ (mod p) is feasible.

Discrete Logarithm Problem (DLP)

- If p is a prime number, and g is a generator of $\mathbb{F}_p^*$. Let h be an element of the finite field $\mathbb{F}_p^*$.
- The Discrete Logarithm Problem (DLP) is the problem of finding an exponent x such that $g^x \equiv h \pmod{\text{p}}$.
- The number x is called the discrete logarithm of h base g, i.e., $log_g h$.

Given g, x, and p, computing $h = g^x$ (mod p) is feasible.

Given g, h, and a sufficiently large p (e.g., 1024-bit), finding the value of x is infeasible.

Discrete Logarithm Problem (DLP) - Example-1

Given g, x, and p, computing $h = g^x$ (mod p) is feasible.

- Let $p = 11$, $g = 2$, and $x = 5$, compute h.

Discrete Logarithm Problem (DLP) - Example-1

Given g, x, and p, computing $h = g^x$ (mod p) is feasible.

- Let $p = 11$, $g = 2$, and $x = 5$, compute h.

$$h = g^x \text{ (mod p)}$$
$$h = 2^5 \text{ (mod 11)}$$
$$h = 10$$

Discrete Logarithm Problem (DLP) - Example-2

Given g, h, and a sufficiently large p (e.g., 1024-bit), finding the value of x is infeasible.

$$h = g^x \text{ (mod p)}$$

- Let $p = 11$, $g = 2$, and $h = 10$, find x.

Discrete Logarithm Problem (DLP) - Example-2

Given g, h, and a sufficiently large p (e.g., 1024-bit), finding the value of x is infeasible.

$$h = g^x \text{ (mod p)}$$

- Let $p = 11$, $g = 2$, and $h = 10$, find x.

$10 \stackrel{?}{=} 2^1$ (mod 11) - No
$10 \stackrel{?}{=} 2^2$ (mod 11) - No
$10 \stackrel{?}{=} 2^3$ (mod 11) - No
$10 \stackrel{?}{=} 2^4$ (mod 11) - No
$10 \stackrel{?}{=} 2^5$ (mod 11) - Yes

There is no efficient way but to try all possible combinations until the answer is found. - A Discrete Logarithm Problem

Discrete Logarithm Problem (DLP) - Example-3

Given g, h, and a sufficiently large p (e.g., 1024-bit), finding the value of x is infeasible.

$$h = g^x \text{ (mod p)}$$

- Let $p = 131$, $g = 2$, and $h = 3$, find x.

Elgamal Digital Signature Scheme

Elgamal Digital Signature Scheme

- Public parameters

Elgamal Digital Signature Scheme

- Public parameters
  - Primitive root (generator) α and a prime number p.

Elgamal Digital Signature Scheme

- Public parameters
  - Primitive root (generator) α and a prime number p.
- Key generation

Elgamal Digital Signature Scheme

- Public parameters
  - Primitive root (generator) α and a prime number p.
- Key generation
  - Choose a random integer X_A where $1 < X_A < p - 1$.

Elgamal Digital Signature Scheme

- Public parameters
  - Primitive root (generator) α and a prime number p.
- Key generation
  - Choose a random integer X_A where $1 < X_A < p - 1$.
  - Compute, $Y_A = \alpha^{X_A} \bmod p$.

Elgamal Digital Signature Scheme

- Public parameters
  - Primitive root (generator) α and a prime number p.
- Key generation
  - Choose a random integer X_A where $1 < X_A < p - 1$.
  - Compute, $Y_A = \alpha^{X_A} \bmod p$.

X_A is a private-key, and Y_A is a public-key.

Elgamal Digital Signature Scheme

- Public parameters
  - Primitive root (generator) α and a prime number p.
- Key generation
  - Choose a random integer X_A where $1 < X_A < p - 1$.
  - Compute, $Y_A = \alpha^{X_A} \bmod p$.

X_A is a private-key, and Y_A is a public-key.

Is is feasible for an adversary to obtain the private key?

Elgamal Digital Signature Scheme

- Signature generation - To sign a message m where $0 \leq m \leq p-1$,

Elgamal Digital Signature Scheme

- Signature generation - To sign a message m where $0 \leq m \leq p-1$,
    - Choose a random number r where $1 \leq r \leq p-1$, and $GCD(r, p-1) = 1$.

Elgamal Digital Signature Scheme

- Signature generation - To sign a message m where $0 \leq m \leq p-1$,
  - Choose a random number r where $1 \leq r \leq p-1$, and $GCD(r, p-1) = 1$.
  - Compute, $S_1 = \alpha^r \bmod p$.

Elgamal Digital Signature Scheme

- Signature generation - To sign a message m where $0 \leq m \leq p-1$,
  - Choose a random number r where $1 \leq r \leq p-1$, and $GCD(r, p-1) = 1$.
  - Compute, $S_1 = \alpha^r \bmod p$.
  - Compute, $S_2 = r^{-1}(m - X_A \cdot S_1) \bmod (p-1)$.

Elgamal Digital Signature Scheme

- **Signature generation** - To sign a message m where $0 \leq m \leq p-1$,
  - Choose a random number r where $1 \leq r \leq p-1$, and $GCD(r, p-1) = 1$.
  - Compute, $S_1 = \alpha^r \bmod p$.
  - Compute, $S_2 = r^{-1}(m - X_A \cdot S_1) \bmod (p-1)$.

The signature consists of the pair (S_1, S_2).

Elgamal Digital Signature Scheme

- Signature verification

Elgamal Digital Signature Scheme

- Signature verification
  - Compute, $V_1 = \alpha^m \bmod p$.

Elgamal Digital Signature Scheme

- Signature verification
  - Compute, $V_1 = \alpha^m \bmod p$.
  - Compute, $V_2 = (Y_A)^{S_1} \cdot (S_1)^{S_2} \bmod (p)$.

Elgamal Digital Signature Scheme

- Signature verification
  - Compute, $V_1 = \alpha^m \bmod p$.
  - Compute, $V_2 = (Y_A)^{S_1} \cdot (S_1)^{S_2} \bmod (p)$.

If $V_1 = V_2$, the signature is valid.

References

References

- Arvind Narayanan, Edward Felten, Steven Goldfeder, Joseph Bonneau, Andrew Miller, Bitcoin and Cryptocurrency Technologies, Princeton University Press, 2016.
- William Stallings, Cryptography and Network Security: Principles and Practice, Prentice Hall, 2017.

Thank You.