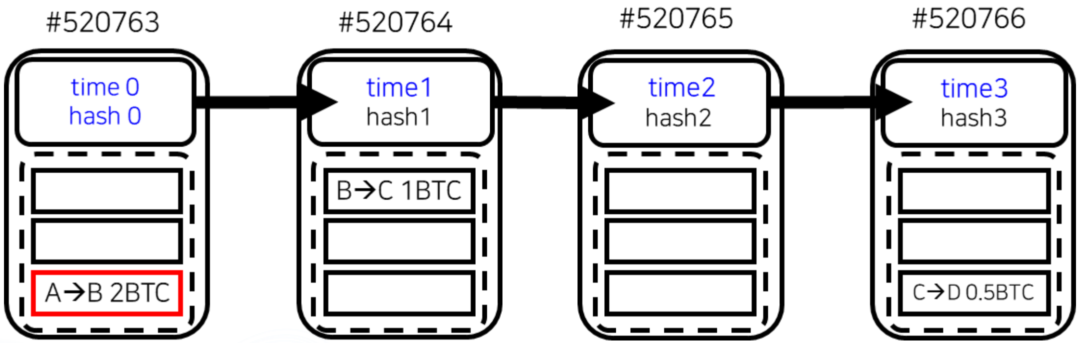
Goal of this lecture note

- How to Put Digital Signature to a Message
- Secure Hash Function

GIST

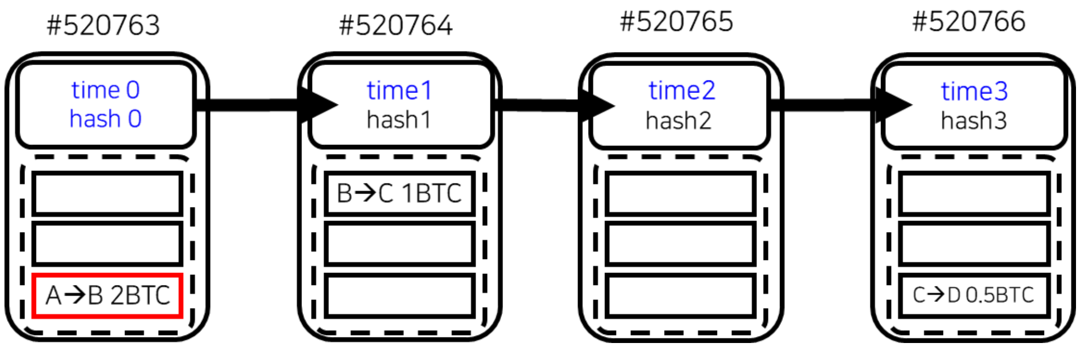
1 How to Put Digital Signature to a Message



- Time 0: A (Sign of A) gives B two coins
- Time 1: B (Sign of B) gives C one coin
- Time 2: Empty
- Time 3: C (Sign of C) gives D 0.5 coin

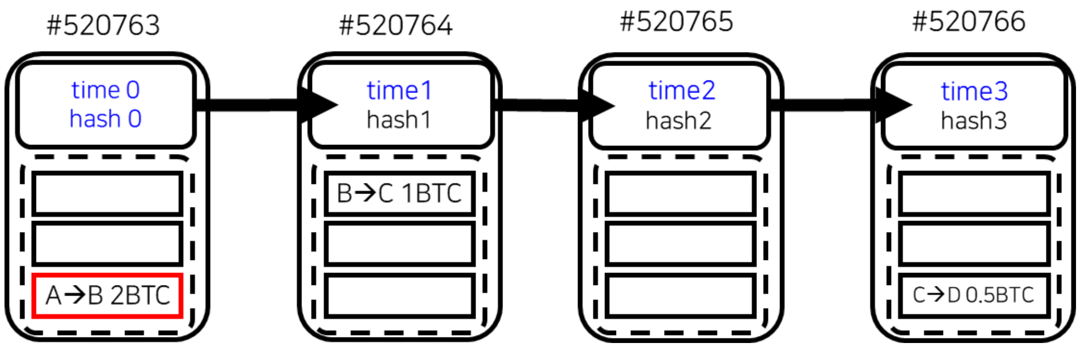
GIST

1 How to Put Digital Signature to a Message



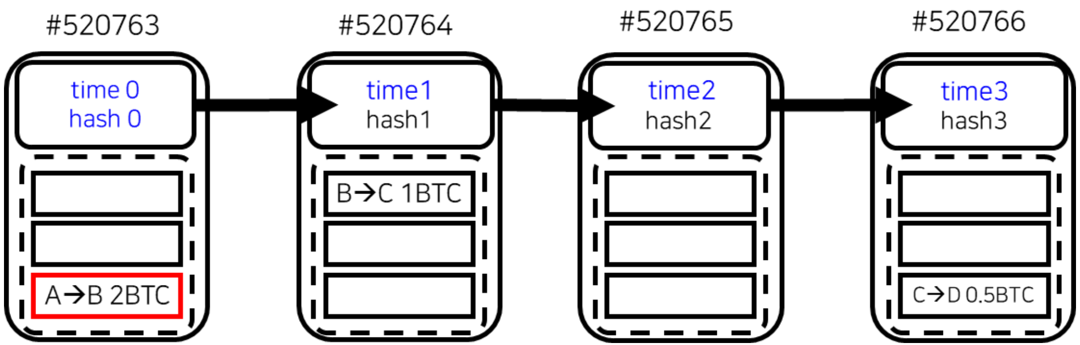
- This is one of the essential charts for understanding how a message transfer to someone can work as a value transfer using a Blockchain.

How to Put Digital Signature to a Message



- Namely, the sender shall put his digital signature in order to show the ownership of his coin.

1 How to Put Digital Signature to a Message



- Now we aim to show an example how a digital signature is created.

GIST

1 How to Put Digital Signature to a Message

- Public key and private key
  - In cryptography, any person can create as many number of pairs of keys.
  - Each pair comes with a public key and a private key.
  - Encryption
    - With one key, a message can be locked.
  - Decryption
    - With the other key, the locked message can be unlocked.
  - Alice can send Bob a private message.

1 How to Put Digital Signature to a Message

- Generation of a key pair
  - Consider two individuals, Alice and Bob.

Alice generates her keys, Pub_A and Pri_A

Bob does the same, Pub_B and Pri_B

  - Each person keeps the private key in secret, while lets the public key widely known.
  - Using them, one can send a private message and put a digital signature to it.

GIST

1 How to Put Digital Signature to a Message

- Encryption and Decryption
  - Define a message m.
  - Define a pair of functions, $\mathrm{ENC}(\ )$and $\mathrm{DEC}(\ )$.
  - These functions are publicly known functions.
  - Cyphered message or encrypted message is created with $\mathrm{ENC}(\ )$, i.e.,

$$y = \mathbf{ENC}(\boldsymbol{m}, \mathrm{Pub_B})$$

  - Cyphered message can only be deciphered using $\mathrm{Pri_B}$, i.e.,

$$\boldsymbol{m} = \mathbf{DEC}(\mathbf{y}, \mathrm{Priv_B})$$

GIST

1 How to Put Digital Signature to a Message

- RSA Example of ENC and DEC functions
  - Let e, m and n be known positive integers. Is it easy to find d?

$$(m^e)^d = m \bmod n \quad \text{--- (1)}$$

- Once $\boldsymbol{d}$ known, it is easy to check

$$(m^d)^e = m \bmod n \quad \text{--- (2)}$$

- Let $\boldsymbol{d}$ be private key and $\boldsymbol{e}$ public key.

✓ Modulo란?
대상 숫자의 나머지를 구하는 연산

- Ex) Modulo-3: 5%3=5-3*floor(5/3) = 2.

GIST

1 How to Put Digital Signature to a Message

- EX1 Alice would like to send a private message "I love you Bob." to Bob.

	Private key d	Public key e
Alice	d_A	e_A
Bob	d_B	e_B

- Alice encrypts her message m with Bob's public key, i.e., y = ENC(m, e_B).
- The encrypted message y is transferred to Bob
- Only can Bob decipher encrypted Alice's message, i.e., m = DEC(y, d_B).

GIST

1 How to Put Digital Signature to a Message

- EX2 Alice attaches a digital signature to her encrypted message m sent to Bob.
  - Alice hashes her message m and get $h(m)$.
  - She puts her signature to the digital message m.
  - The digital signature to her message is

$$Sign(m)=h(m)^{d_A}$$

  - Alice uses her pri_key d_A to generate $Sign(m)$.
  - Using Alice's pub_key e_A , Bob recovers $h(m)$ via (2).

1 How to Put Digital Signature to a Message

- EX2 Alice attaches a digital signature to her encrypted message m sent to Bob
  - Using the Alice's message m deciphered from Ex1), Bob generates its hash, $h(m)$.
  - Bob checks if the two hash values match.

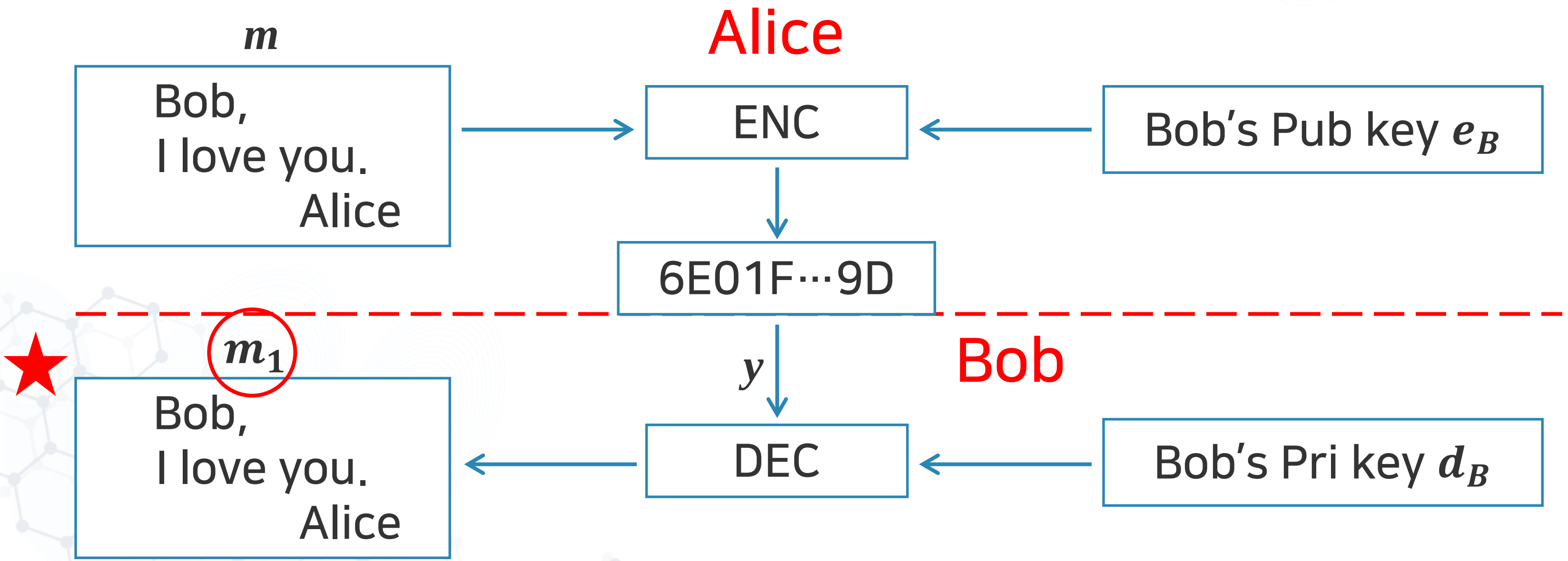
1 How to Put Digital Signature to a Message

- Note
  - Bitcoin does not use RSA but Elliptic Curve Signatures.
  - Our purpose of showing how to put digital signature to a message can be served with RSA as well.

"We may use RSA because it is more familiar to us."

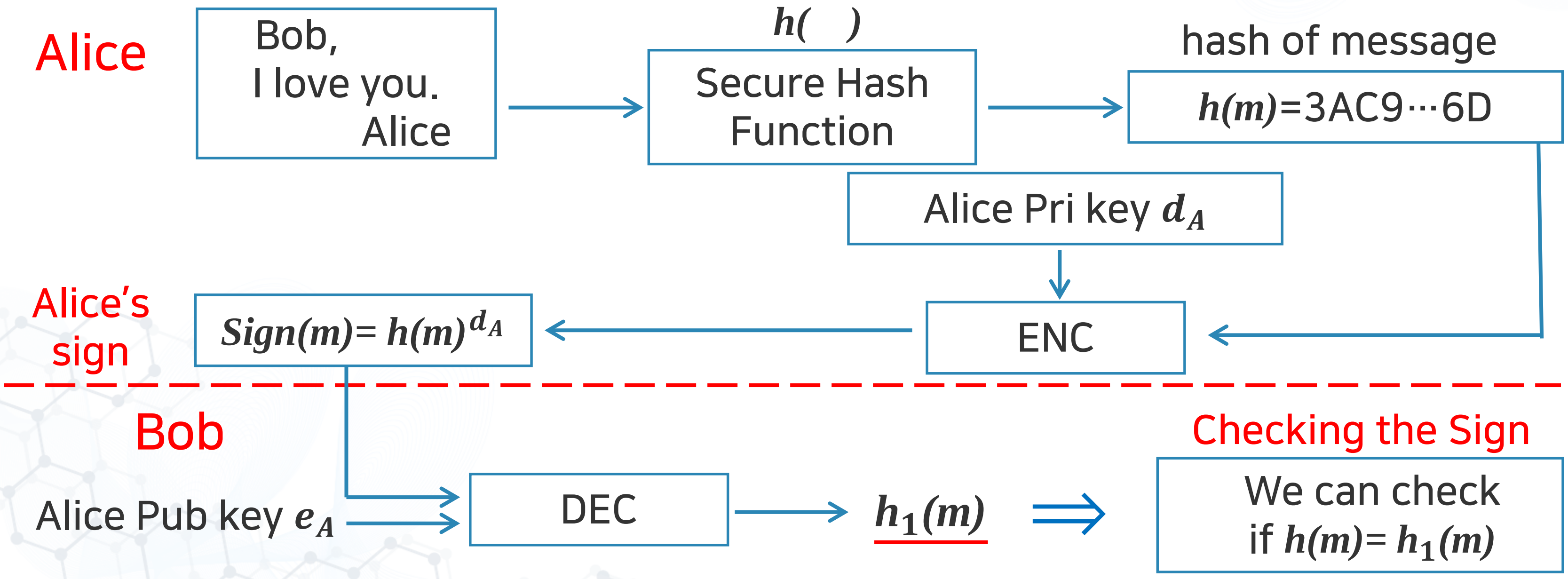
How to Put Digital Signature to a Message

- Alice sends a private message to Bob.



How to Put Digital Signature to a Message

- Digital Sign and Validation



1 How to Put Digital Signature to a Message

- Let m be "A → B 2 BTC"
  - Alice sends the message m .
  - Alice attaches her digital signature to it.
  - Together it shall look like:

    A → B 2BTC Sign_by_A

  - Ownership verification is done by checking the sign.
  - Balance can be checked by looking at the address A.
  - Transaction is complete if this is recorded into the book.

1 How to Put Digital Signature to a Message

- Anonymity
  - A is an address of Bitcoin made from a public key of Alice.
  - B is an address of Bitcoin made from a public key of Bob.

2 Secure Hash Functions

- What is a Hash Function?
  - Definition
    - A hash function is a function, represented with H(input) = output, which takes a text message as its input and gives as its output a fixed number of binary bits.

2 Secure Hash Functions

- What is a Hash Function?
  - Bitcoin uses the Secure Hash Function 256.
  - The length of output bit string is 256.
  - The input to a hash function is a text message or a file.

Ex Input-Output of Hash function H

- Message = [Bob, I love you. Alice.]
- H(Message) = [2FE442157E2025AB75F3856F09238E2CD78A3B 396BC25F128B95D04AD6252634]
- A string of 64 hexadecimals or a string of 256 bits.

GIST

2 Secure Hash Functions

- Conditions for Good Hash Function
  - One way
    With a little change in the input, the output is completely different.
    - Input distance has no relation to output distance.
  - Collision free
    Given $y = H(x)$, finding $x_1 \neq x$ such that $H(x_1) = y$ shall be almost impossible!
  - Collision free stronger
    Finding an input pair of different messages x and x_1 which leads to $H(x) = H(x_1)$ shall be almost impossible!

- x_1 = [Bob, I love you. Alice.]
- $H(x_1)$ = [2FE442157E2025AB75F3856F09238E2CD78A3B 396BC25F128B95D04AD6252634]

- Illustration of Onewayness

Ex1

- x_1 = [Bob, I love you. Alice.]
- $H(x_1)$ = [2FE442157E2025AB75F3856F09238E2CD78A3B 396BC25F128B95D04AD6252634]

Ex2

- x_2 = [Bob, I love you. Alice]
- $H(x_2)$ = [B1316ED8BA74AD416C8E966574CD584AD447B8 11B722FB9230C71B047C71B825]

2 Secure Hash Functions

- Illustration of Onewayness

  Ex3

  - x_3 = [Bob, I loved you. Alice.]
  - $H(x_3)$ = [BFDDB00446539D8CF8ECC712E3A8144EDF41A771C0F96560E9EDE3E576CD8FBF]

2 Secure Hash Functions

- Tiny difference → Big difference
  - Note that there is a very small difference between x_1 and x_2.
  - But the difference in the output is huge.
  - This property can be utilized to spot out a tiny alteration made to an original input file.
  - A tiny unnoticeable alteration, and thus is difficult to be detected by human eyes, but can be magnified into easily discernable hash difference.

2 Secure Hash Functions

- INPUT-OUTPUT of SHA256 $\boldsymbol{H}(\ )$

INPUT: Message or File

OUTPUT: digest, hash values (256 binary bits or 64 Hexadecimals)

$$\boldsymbol{x} \in \mathbf{X} \longrightarrow \boxed{\boldsymbol{H(x)}} \longrightarrow \boldsymbol{y} \in \mathbf{Y}$$

x_1=[Bob, I love you. Alice.]

$H(x_1)$ =
[2FE442157E2025AB75F3856F092
38E2CD78A3B396BC25F128B95D0
4AD6252634]

2 Secure Hash Functions

- INPUT-OUTPUT of SHA256 $\boldsymbol{H}(\ )$

INPUT: Message or File

OUTPUT: digest, hash values (256 binary bits or 64 Hexadecimals)

$$\boldsymbol{x} \in \mathbf{X} \longrightarrow \boxed{\boldsymbol{H(x)}} \longrightarrow \boldsymbol{y} \in \mathbf{Y}$$

x_2=[Bob, I love you. Alice]

$H(x_2)$ = [B1316ED8BA74AD416C8E966574CD584AD447B811B722FB9230C71B047C71B825]

2 Secure Hash Functions

- INPUT-OUTPUT of SHA256 ***H*( )**

INPUT: Message or File

OUTPUT: digest, hash values (256 binary bits or 64 Hexadecimals)

$$\boldsymbol{x} \in X \longrightarrow \boldsymbol{H(x)} \longrightarrow \boldsymbol{y} \in Y$$

x_3=[Bob, I loved you. Alice.]

$H(x_3)$ =
[BFDDB00446539D8CF8ECC712E3A8
144EDF41A771C0F96560E9EDE3E57
6CD8FBF]

GIST

2 Secure Hash Functions

- SHA-256 Algorithm

https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

SHA-256(M):
  (* Let M be the message to be hashed *)
  for each 512-bit block B in M **do**
    $W = f_{exp}(B)$;
    (* Initialize the registers with the constants. *)
    $a = H_0$; $b = H_1$; $c = H_2$; $d = H_3$; $e = H_4$; $f = H_5$; $g = H_6$; $h = H_7$;
    for $i = 0$ **to** 63 **do**
      (* Apply the 64 rounds of mixing. *)
      $T_1 = h + \Sigma_1(e) + f_{if}(e, f, g) + K_i + W_i$;
      $T_2 = \Sigma_0(a) + f_{maj}(a, b, c)$;
      $h = g$; $g = f$; $f = e$; $e = d + T_1$; $d = c$; $c = b$; $b = a$; $a = T_1 + T_2$;
    (* After all the rounds, save the values in preparation of the next data block. *)
    $H_0 = a + H_0$; $H_1 = b + H_1$; $H_2 = c + H_2$; $H_3 = d + H_3$;
    $H_4 = e + H_4$; $H_5 = e + H_5$; $H_6 = e + H_6$; $H_7 = e + H_7$;
  (* After all 512-bit blocks have been processed, return the hash. *)
  return concat(H_0, H_1, H_2, H_3, H_4, H_5, H_6, H_7);

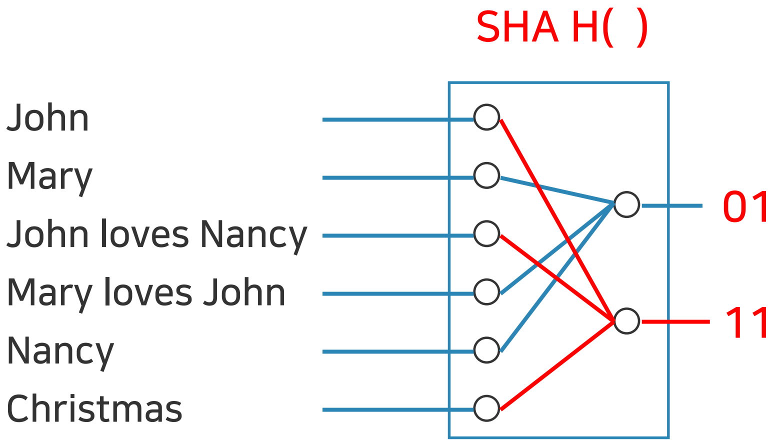
Algorithm 1.3: THE SHA-256 ALGORITHM.

National
Institute of
Standard and
Technology
미국 국립 표준 연구소

Wouter Penardand and Timvan Werkhoven, "On the Secure Hash Algorithm family," *Cryptography in Context*, 2002

2 Secure Hash Functions

- Collision free



Note

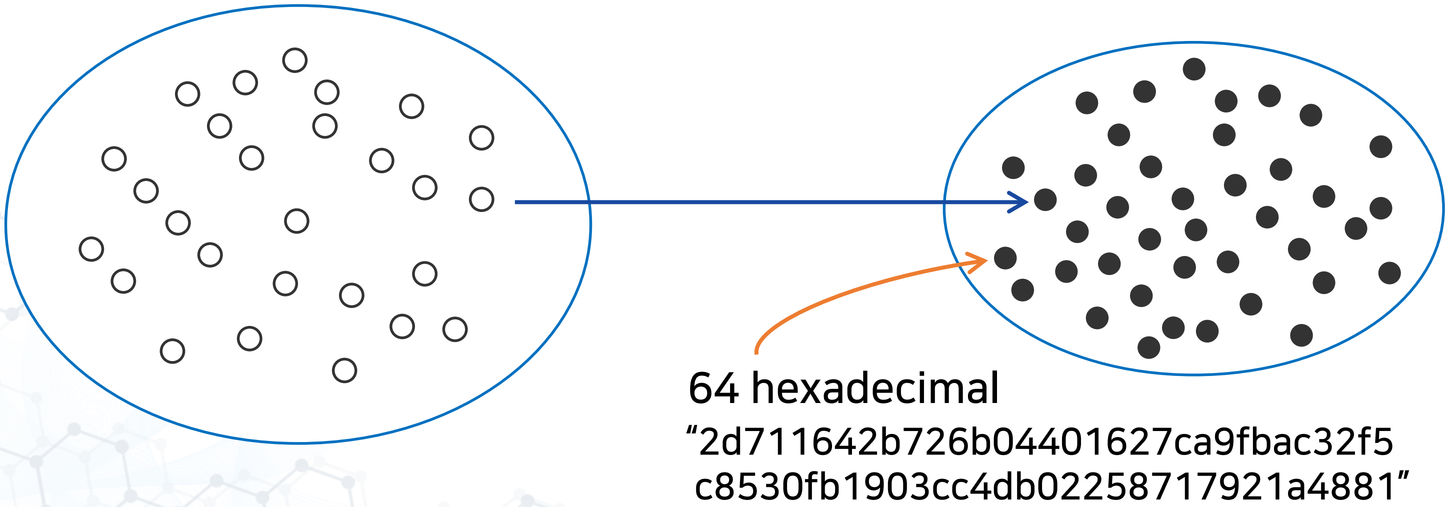
H(John)
=H(Christmas)
=H(John loves Nancy)

- Collision free implies there surely are collisions but one can hardly encounter one.

2 Secure Hash Functions

- Input-Output of SHA-256, i.e., $H(x) = y$

$X := \{x | x$ is a message up to 1 Mbyte in size$\}$ $\quad$ $Y := \{y | y$ is a 256bit string$\}$



2 Secure Hash Functions

- Cardinality of the Input file set
  - Bitcoin allows an input file whose size is up to a 1Mbyte.
  - What is the cardinality of the set of all possible input file sets?
    - All possible input files can be enumerated from small files to large files, such as noting, 0, 1, 10, 11, 100, 101, 110, 111, ⋯.
  - Thus, there are $2^{8000000}$ different files.
  - The cardinality of x is about $10^{2400000}$.

GIST

2 Secure Hash Functions

- Cardinality of Output Hash Set
  - Each input file produces a 256 bit output.
  - The cardinality of the set of all output hash values is $2^{256} \sim 10^{77}$.
  - For each y in $\mathcal{Y}$, how many input files x in X are there such that each $H(x) = y$?

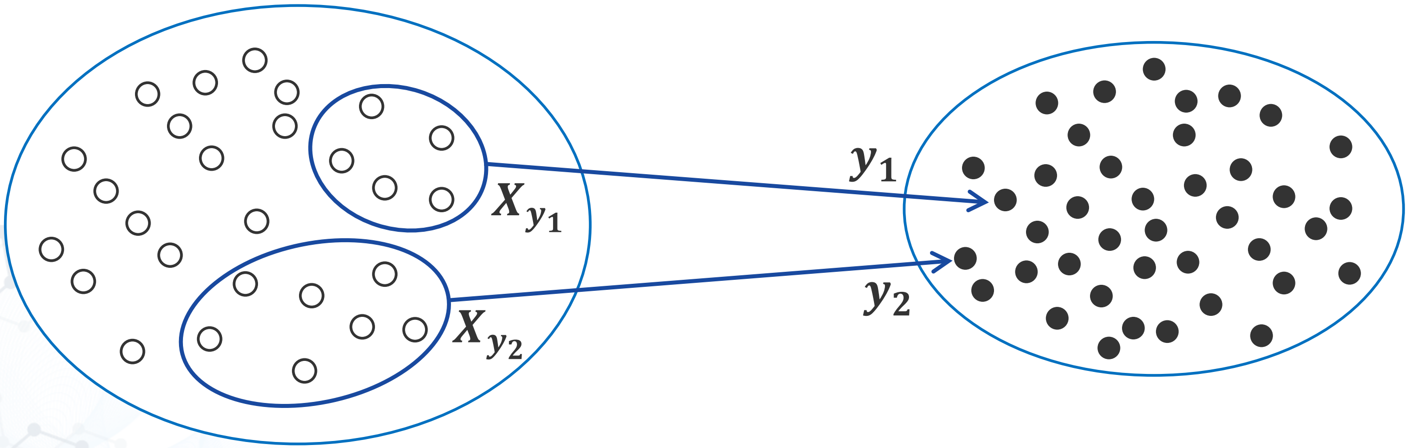
GIST

2 Secure Hash Functions

- Preimage of y is a subset of X

$X_y := \{x \in X | H(x) = y\}$

$Y := \{y | \text{y is a 256bit string}\}$



2 Secure Hash Functions

- Size of Input Set per Hash Output
  - What is the average size of the input file set whose element leads to the same hash output?
  - For each output hash y in y , the preimage of y can be defined as

$$X_y := \{x : H(x) = y\}$$

  - WLOG, assume the same size for any y_1 and y_2 :

$$|X_{y_1}| = |X_{y_2}|$$

GIST

2 Secure Hash Functions

- There are 2^{256} preimage sets.
  - There are 2^{256} distinct y's in y.
  - There are 2^{256} preimages of y in X.
  - These are mutually non-overlapping sets.
  - The size of a preimage of a point y is

$$log_{10}|X_y| = log_{10}\frac{|X|}{|Y|} = 2400000 - 77 = 2399923.$$

2 Secure Hash Functions

- Collisions are abound, but can you find one?
  - Collisions must occur, even abundantly.
  - Consider any two different files x_1 and x_2 in X_y, i.e., the two hashes are the same $H(x_1) = H(x_2)$
  - For any file x_3 in X but not in X_y, we note,

$$H(x_3) \neq y$$

  - What do you mean by Collision Free then?

2 Secure Hash Functions

- What is the meaning of Collision Free?

Small problem

- Suppose the input x is a file of size up to 1 Kilobyte and the SHA output is truncated to 10 bit.
- Bob has found that the input file x_0 has the hash value y_0.

(a) What is the size of the input file set?

(b) What is the size of the output file set?

(c) Bob selects a file x_1 at random from his desktop computer, size smaller than 1 Kilobyte, and runs it thought the truncated to the first 10 bit, say SHA-10.

What is the probability that this output is the same as the first output y_0?

GIST

2 Secure Hash Functions

- Solution 1
  - The set sizes are

$$s_X := log_{10}|X| = log_{10} 2^{8000} = 8e3 \times 0.3010 \sim 2.40e3$$

$$s_Y := log_{10} 2^{10} = 10 \times 0.3010 = 3.01$$

$$s_{X_y} := log_{10} \frac{|X|}{|Y|} = s_X - s_Y \sim 2400 - 3 = 2397$$

2 Secure Hash Functions

- Solution 2
  - Let p_c^1 be the prob. of selecting $x_1 \neq x_0$ leading to hash collision.

$$p_c^1 := Pr\{x_1 : H(x_1) = H(x_0)\} = Pr\{x_1 \in X_y\}$$

$$= \frac{|X_y| - 1}{|X| - 1} \approx \frac{1}{|Y|}$$

2 Secure Hash Functions

- Solution 3
  - Suppose there were no hash collisions for two x_0 and x_1. Now select another file x_2.
  - Let p_c^2 be the prob. that the hash of x_2 is equal to either of the two previous hashes, leading to hash collision. Find it.

$$p_c^2 := Pr\{x_2 : H(x_2) = H(x_0)\} \cup \{x_2 : H(x_2) = H(x_1)\}$$

$$= Pr\{x_2 \in X_y\} + Pr\{x_2 \in X_{y_1}\}$$

$$= \frac{2(|X_y| - 1)}{|X| - 2} \approx \frac{2}{|Y|}$$

GIST

2 Secure Hash Functions

- Solution 3
  - Suppose there were no hash collisions up to three selections of files x_0 , x_1 and x_2 . Now select another file x_3. Let p_c^3 be the prob. that the hash of x_3 is equal to any of the three previous hashes, leading to hash collision. Find it.

$$p_c^3 = Pr\{x_3 \in X_y\} + Pr\{x_3 \in X_{y_1}\} + Pr\{x_3 \in X_{y_2}\}$$

$$= \frac{3(|X_y| - 1)}{|X| - 3} \approx \frac{3}{|Y|}$$

GIST

2 Secure Hash Functions

- Solution 4
  - Suppose there were no hash collisions up to m selections of files x_0 , x_1 and x_{m-1}. Now select an m-th file x_m
  - Let p_c^m be the prob. that the hash of x_m is equal to any of the previous hashes, leading to hash collision. Find it.

$$p_c^m = Pr\{x_m \in X_y\} + \cdots + Pr\{x_m \in X_{y_{m-1}}\}$$

$$= \frac{m(|X_y| - 1)}{|X| - m} \approx \frac{m}{|Y|}$$

2 Secure Hash Functions

- Solution 5
  - The hash collision probability increases as m grows, i.e.,

$$p_c^1 = \frac{1}{1024}$$

$$p_c^2 = \frac{2}{1024}$$

$$p_c^3 = \frac{3}{1024}$$

$$\dots$$

$$p_c^{512} = \frac{512}{1024}$$

GIST

2 Secure Hash Functions

- What is the meaning of Collision Free?

Large problem

- Here the input x is a file of size up to 1 Megabyte.
- Bob has found that the input file x_0 has the hash value y_0
    - (a) What is the size of the input file set?
    - (b) He selects a file x_1 at random from his desktop computer and runs it thought SHA-256. What is the probability that this output is the same as the first output y_0?

2 Secure Hash Functions

- Solution
  - The collision probability is so small no matter how many files are selected.

$$p_c^m = Pr\{x_m \in X_y\} + \cdots + Pr\{x_m \in X_{y_{m-1}}\}$$

$$= \frac{m(|X_y| - 1)}{|X| - m} \approx \frac{m}{|Y|} = \frac{m}{10^{77}}$$

2 Secure Hash Functions

- Bitcoin hash cycles per second is huge.
  No collision thus far for 10 years?
  - Bitcoin hash power has reached 10^20 cycles/sec.
    Suppose it's been that way for the past 10 years.
  - What is the probability of collision occurred?
  - Given m=O(10^{29}) distinct hashes generated
    in 10 years,

$$p_c^m = \frac{m}{|Y|} = \frac{10^{29}}{10^{77}} = 10^{-48}$$

GIST

2 Secure Hash Functions

- What is the meaning of Collision Free?
  - Size of the hash output set is so huge.
  - One knows there are large number of collisions, but one cannot come across any collision.
  - How larger is this number 10^{77}
    - The number of cells in a human body is O(10^{13}).
    - The number of cells in all human body is O(10^{23}) .
    - The number of stars in the observable universe is O(10^{22}) .
    - The number of atoms in the observable universe is O(10^{80}) .
    - https://en.wikipedia.org/wiki/Large_numbers