CAN304
Computer Systems Security

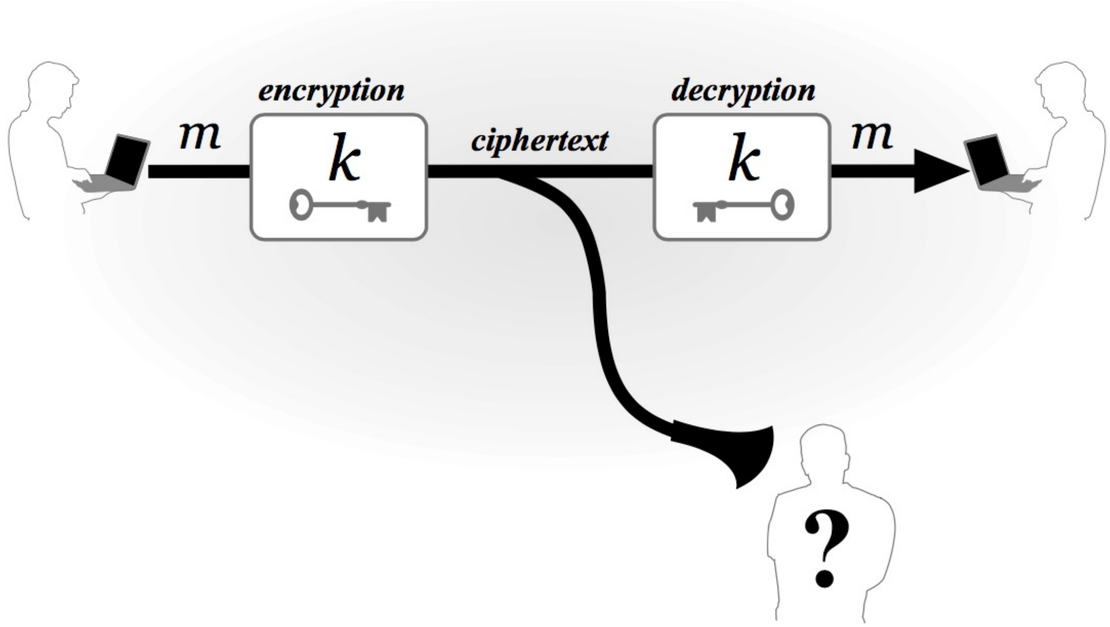
Lecture 3. Fundamentals of cryptography (2)

Week 3: 2022-03-11, 14:00-16:00, Friday

Jie Zhang
Department of Communications and Networking
Email: jie.zhang01@xjtlu.edu.cn
Office: EE522

Review of last week

- Classical and modern cryptography
- Symmetric encryption
  - Block ciphers
  - Cryptographic modes
  - Uses of symmetric cryptography



Learning objectives

- Discuss the use of secure hash functions for message authentication.
- List other applications of secure hash functions.
- Apply secure hash functions in security design and implementations.

Outline

- Message authentication code
- Hash functions and applications

1. Message authentication code

- Message integrity
- Fixed-length MAC
- CBC-MAC
- Secure communications

1.1 Message integrity

Secrecy vs. integrity

- So far, we have been concerned with ensuring secrecy of communication
- What about integrity?
  - I.e., ensuring that a received message originated from the intended party, and was not modified, even if an attacker controls the channel!

A

{transfer \$100 to Jie's account}$_k$

- Is the request issued by A?
- Are the details of the request exactly those intended by A?

How MAC works?

k

m, t

m’, t’

k

m

$t = Mac_k(m)$

$Vrfy_k(m', t') = 1?$

Secrecy vs. integrity

- Secrecy and integrity are orthogonal concerns
  - Possible to have either one without the other

- Encryption can help to discover alteration.
  - If the decrypted message is meaningless, unreadable
  - But is not the purpose of encryption.

Message authentication code (MAC)

- A message authentication code is defined by three algorithms (Gen, Mac, Vrfy):
  - Gen: generates a random key k.
  - Mac: takes as input key k and message $\mathrm{m} \in \{0,1\}^*$; outputs tag t
$$t := \mathrm{Mac_k}(m)$$
  - Vrfy: takes key k, message m, and tag t as input; outputs 1 (“accept”) or 0 (“reject”)

For all m and all k output by Gen,
$\mathrm{Vrfy_k}(\mathrm{m}, \mathrm{Mac_k}(\mathrm{m})) = 1$

1.2 Fixed-length MAC

Intuition?

- We need a keyed function Mac such that
  - Given $\mathrm{Mac}_{\mathrm{k}}(m_1), \mathrm{Mac}_{\mathrm{k}}(m_2), \dots,$
  - ...it is infeasible to predict the value $Mac_k(m)$ for any $m \notin \{m_1, \dots, \}$

- Let Mac be a block cipher

Construction

- Let F be a block cipher

- Construct the following MAC Π:
  - Gen: choose a uniform key k for F
  - $\mathrm{Mac_k}(m)$: output t $\leftarrow \mathrm{F_k}(m)$
  - $\mathrm{Vrfy_k(m, t)}$: output 1 iff $\mathrm{F_k}(m) = t$

Drawbacks?

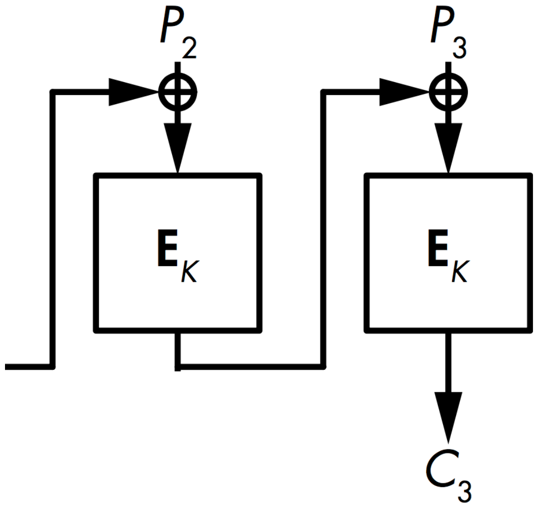
- In practice, block ciphers have short, fixed-length block size
  - E.g., AES has a 128-bit block size
  - So the previous construction is limited to authenticating short, fixed-length messages

- Next few parts: authenticating long, variable length messages

1.3 CBC-MAC

Recall CBC mode

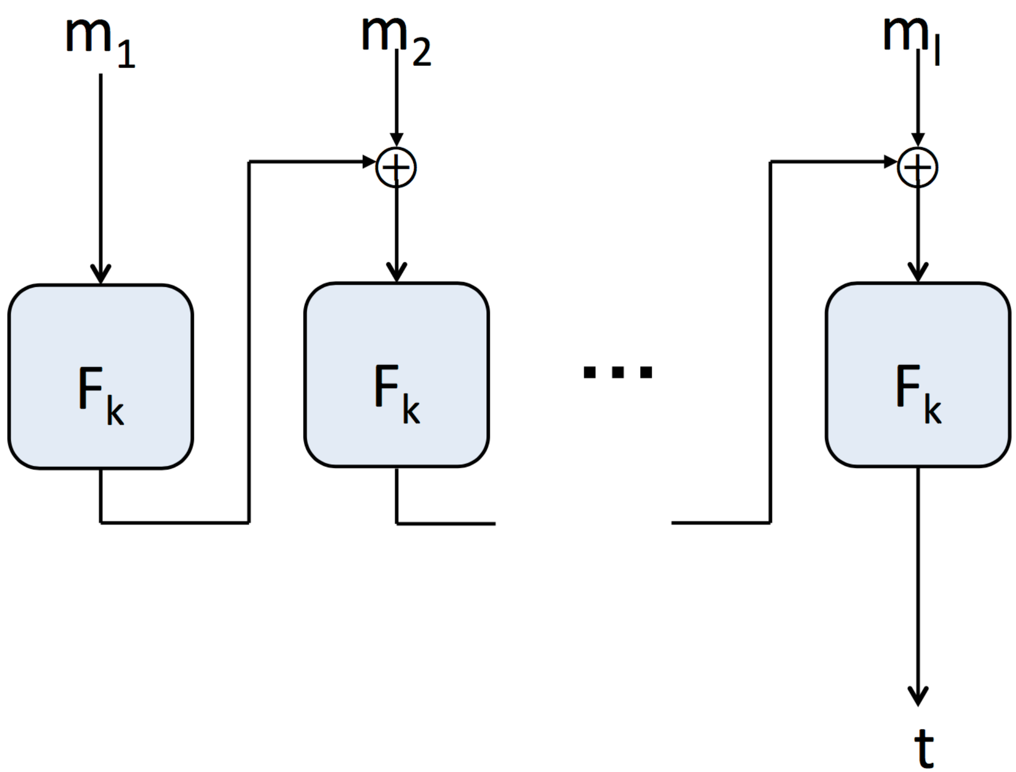
- For block X+1, XOR the plaintext with the ciphertext of block X
  - Then encrypt the result
- Each block's encryption depends on all previous blocks' contents



- What if we only output the last ciphertext block?

(Basic) CBC-MAC

- $m = m_1 m_2 \ldots m_l$



CBC-MAC vs. CBC-mode encryption

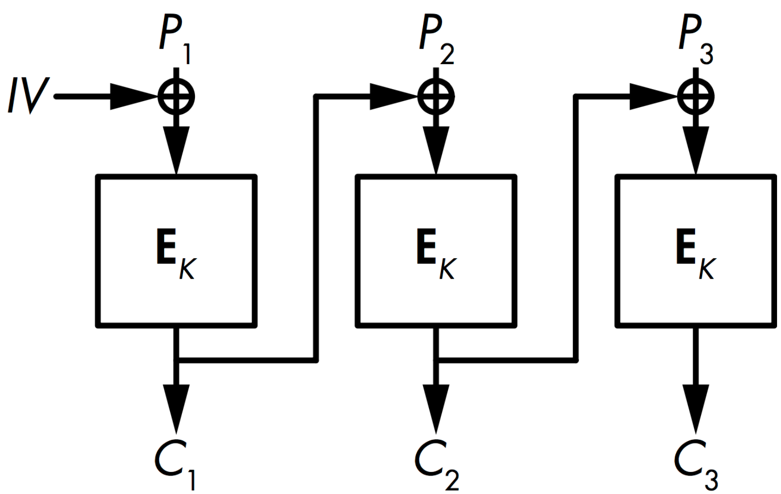
- CBC-MAC is deterministic (no IV)

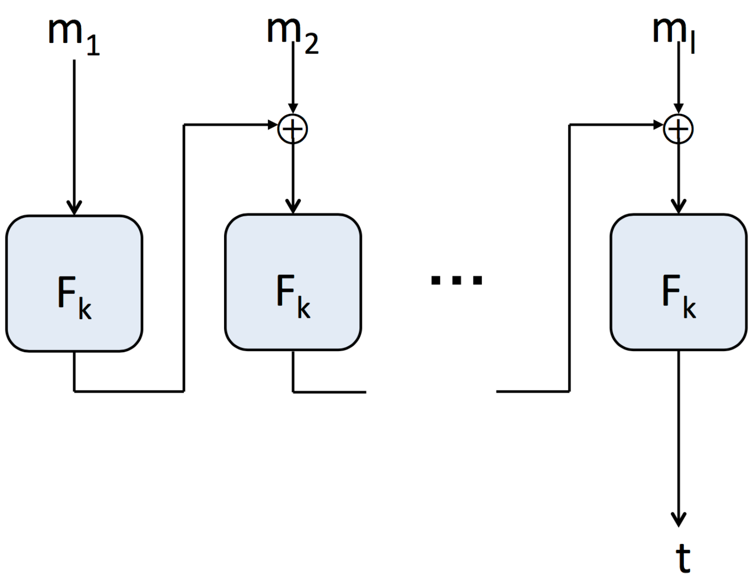
- In CBC-MAC, only the final value is output
  - Verification is done by re-computing the result

- Both of these are essential for security

Can I recover the original data from its MAC tag?

- No



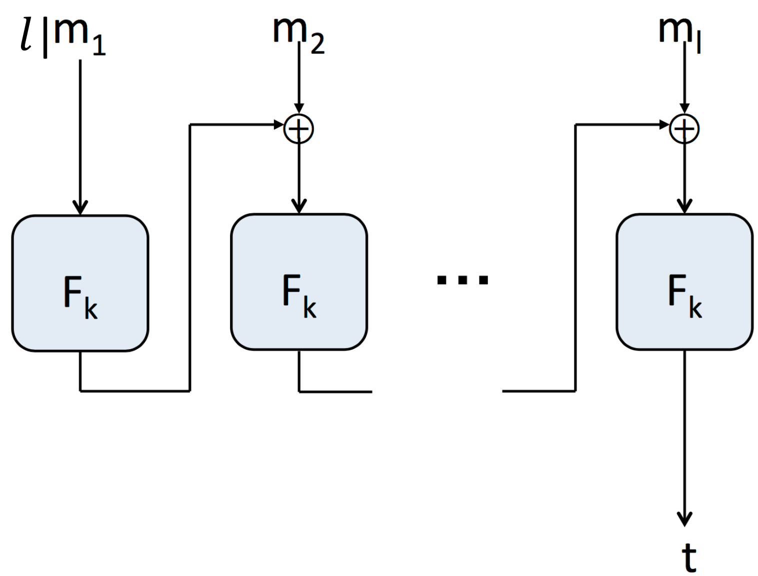


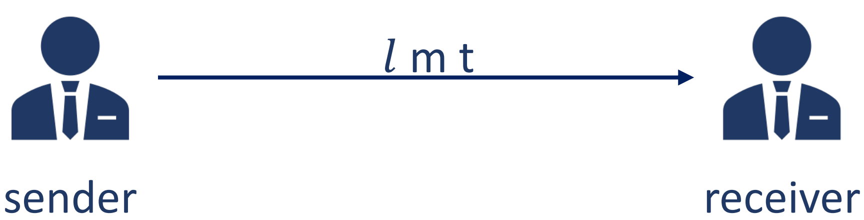
CBC-MAC extensions

- Several ways to handle variable length messages
  - Whose length is not a multiple of the block length
- One of the simplest: prepend the message length before applying (basic) CBC-MAC

CBC-MAC

- $m=m_1m_2 \ldots m_l$
- Suppose length(m_1) < block size





1.4 Secure communication

Secrecy + integrity?

- We have shown primitives for achieving secrecy and integrity in the private-key setting

- What if we want to achieve both?

Encrypt and authenticate

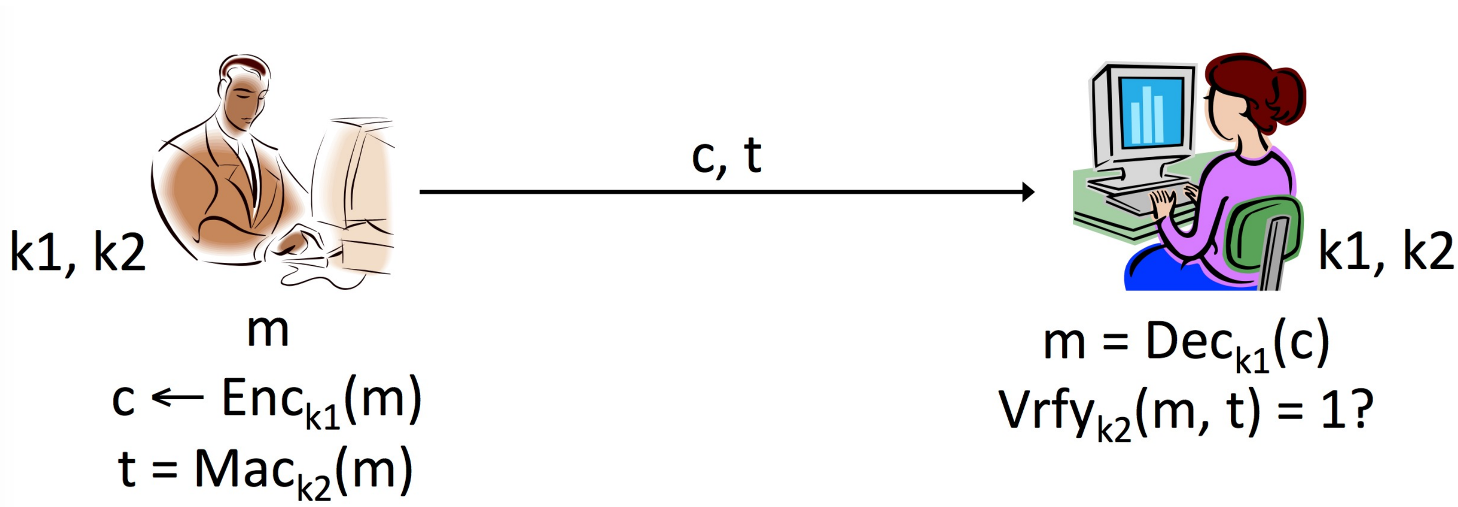
k1, k2 

 k1, k2

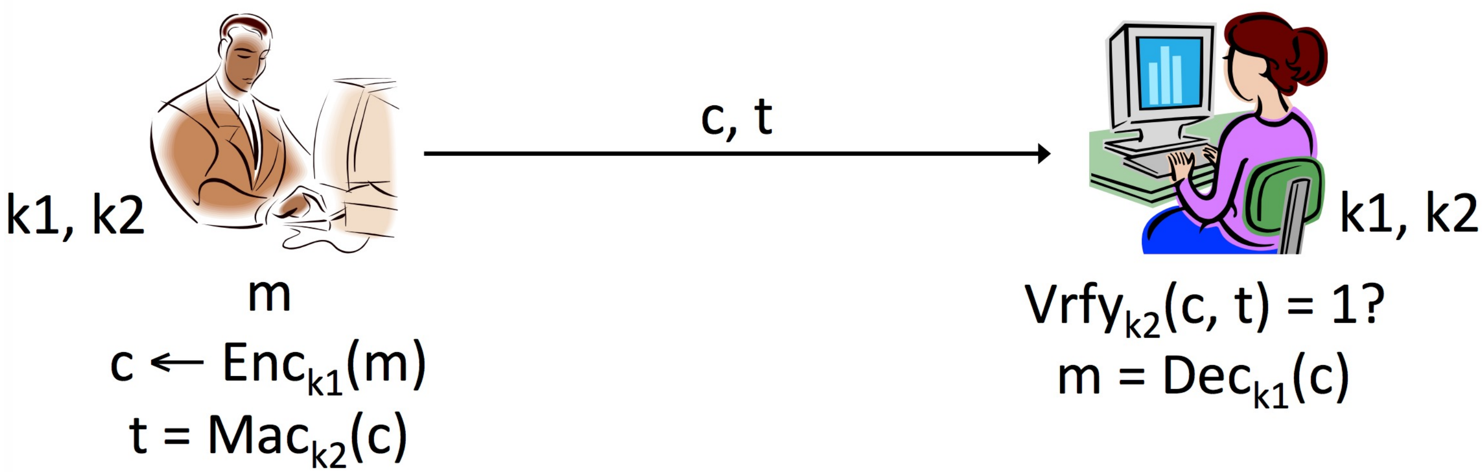
Problems

- The tag t might leak information about m!
  - Nothing in the definition of security for a MAC implies that it hides information about m

- If the MAC is deterministic (as is CBC-MAC), it leaks whether the same message is encrypted twice



Encrypt then authenticate

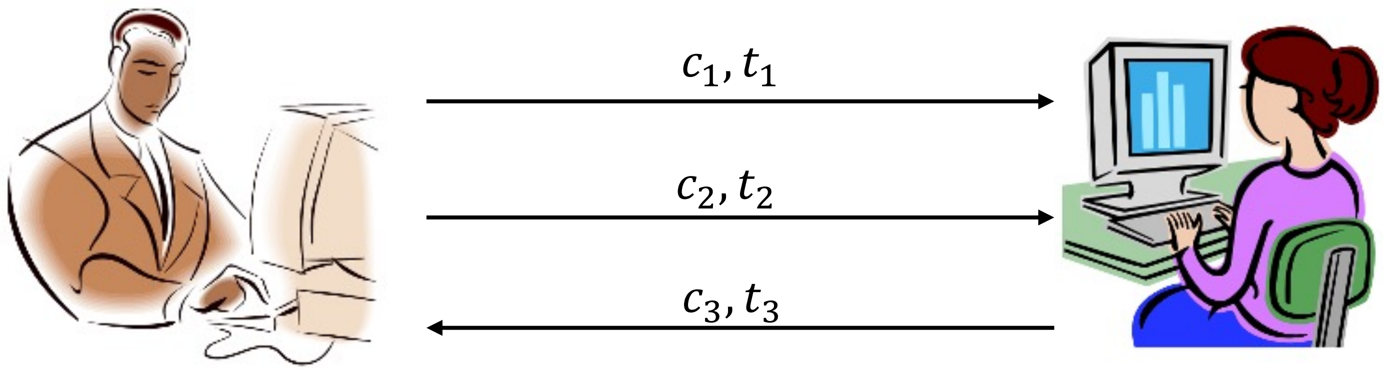


Secure sessions

- Consider parties who wish to communicate securely over the course of a session
  - “Securely” = secrecy and integrity
  - “Session” = period of time over which parties are willing to maintain state

- Can use authenticated encryption...

Any attacks?



k_1, k_2

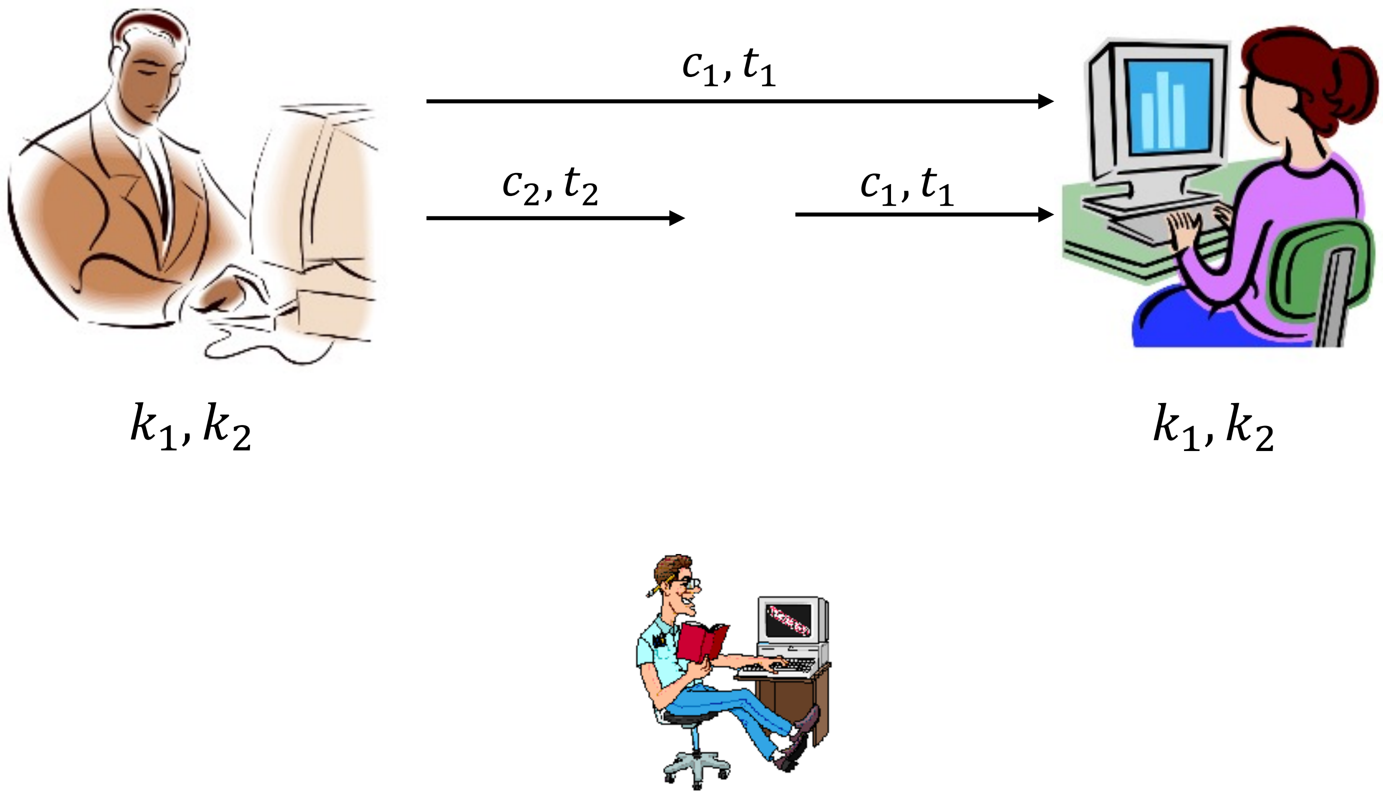
k_1, k_2

$c_1 = Enc_{k_1}(m_1),$
$t_1 = MAC_{k_2}(c_1)$

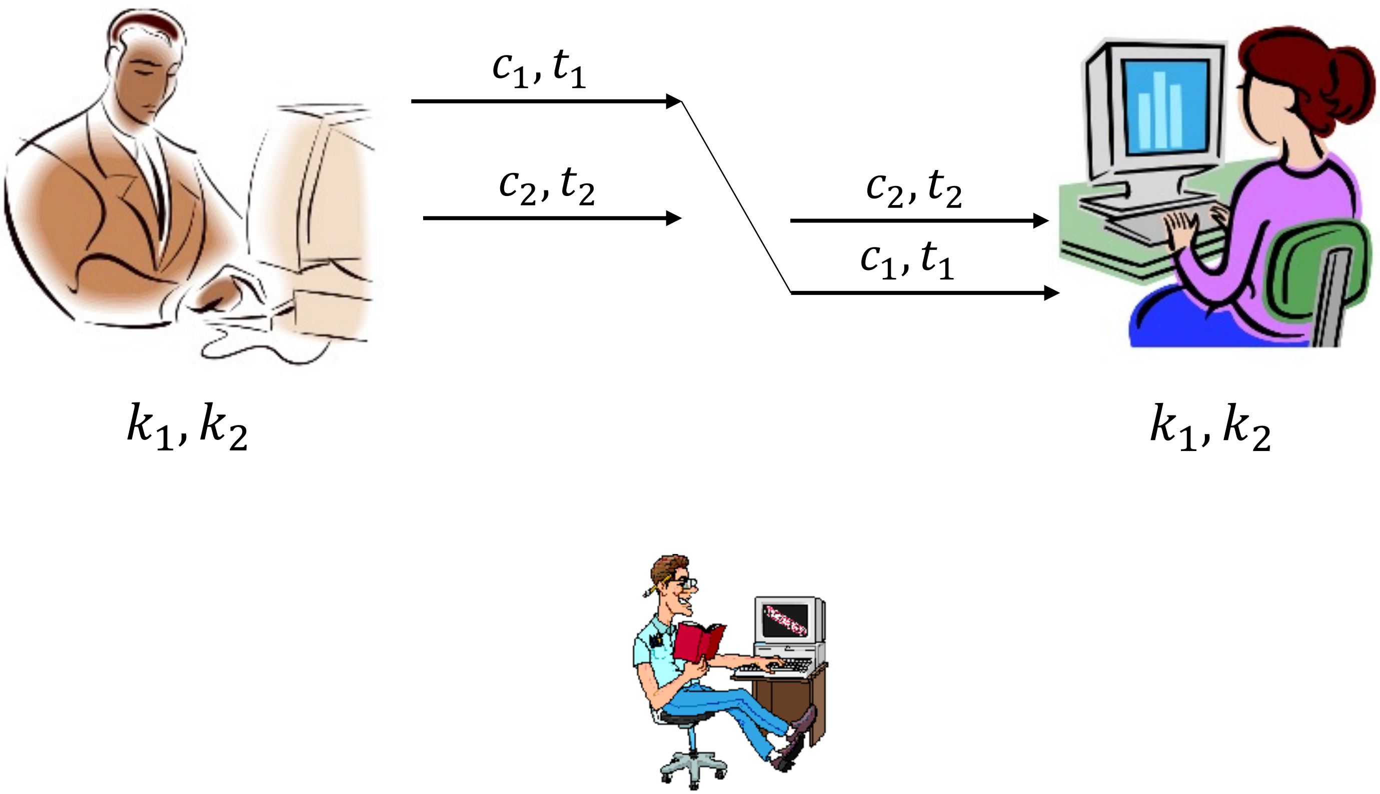
$c_2 = Enc_{k_1}(m_2),$
$t_2 = MAC_{k_2}(c_2)$

$c_3 = Enc_{k_1}(m_3),$
$t_3 = MAC_{k_2}(c_3)$

Replay attack

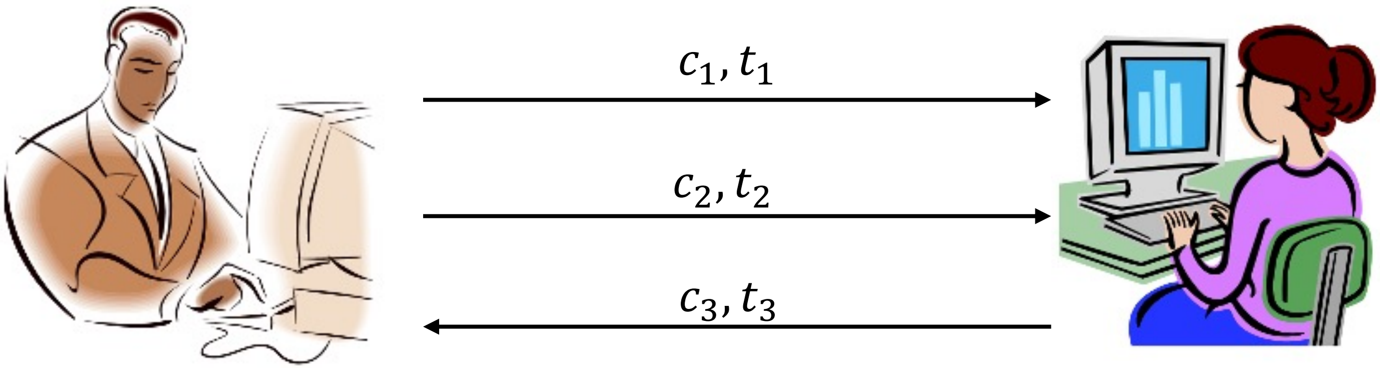


Re-ordering attack



Secure sessions

- These attacks (and others) can be prevented using counters and identities



k_1, k_2 $\quad\quad$ k_1, k_2

$c_1 = Enc_{k_1}("Bob"|m_1|1)$,
$t_1 = MAC_{k_2}(c_1)$

$c_2 = Enc_{k_1}("Bob"|m_2|2)$,
$t_2 = MAC_{k_2}(c_2)$

$c_3 = Enc_{k_1}("Alice"|m_3|1)$,
$t_3 = MAC_{k_2}(c_3)$

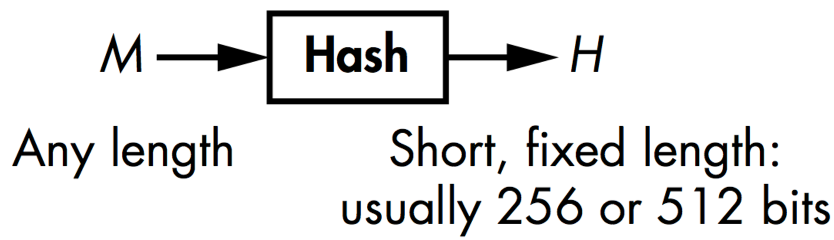
2. Hash functions and applications

- Hash functions
- HMAC
- Merkle tree
- Bitcoin

2.1 Hash functions

Hash functions

- (Cryptographic) hash function: maps arbitrary length inputs to a short, fixed-length digest



- Can define keyed or unkeyed hash functions
  - Formally, keyed hash functions are needed
  - In practice, hash functions are unkeyed
  - (We will work with unkeyed hash functions, and be less formal)

Properties of cryptographic hash functions

- Let $H: \{0,1\}^* \rightarrow \{0,1\}^n$ be a hash function

- Preimage resistance
  - one-way
- Second preimage resistance
  - weak collision resistant
- Collision resistance
  - strong collision resistance

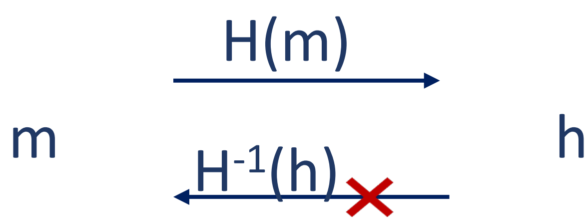
Preimage resistance

- Let $H: \{0,1\}^* \rightarrow \{0,1\}^n$ be a hash function

- A preimage of a given hash value, h, is any message, M, such that

$$H(M) = h$$

- Preimage resistance
  - Given a random hash value, it is computationally infeasible to find a preimage of that hash value
  - One-way function



Second preimage resistance

- Let $H: \{0,1\}^* \to \{0,1\}^n$ be a hash function

- Given x, it is computationally infeasible to find $x' \neq x$ such that $H(x) = H(x')$

- Aka. weak collision resistant

Collision resistance

- Let $H: \{0,1\}^* \to \{0,1\}^n$ be a hash function
- A collision is a pair of distinct inputs x, x' such that $H(x) = H(x')$
- H is collision-resistant if it is computationally infeasible to find a collision in H.
- Aka. strong collision resistance.

Generic hash-function attacks

- What is the best "generic" collision attack on a hash function $H: \{0,1\}^* \to \{0,1\}^n$?
  - size of $\{0,1\}^*$: arbitrarily large
  - size of $\{0,1\}^n$: $2^n = 2\times2\times\cdots\times2$

- If we compute $H(x_1), \ldots, H(x_{2^n+1})$, we are guaranteed to find a collision
  - Is it possible to do better?
  - guaranteed: 100%

“Birthday” attacks

- How many hashes do we need to compute to find a collision with a large probability, such as 50%?

- Related to the so-called birthday paradox
  - How many people are needed to have a 50% chance that some two people share a birthday?

Theorem

- When the number is $O(N^{1/2})$, the probability of a collision is 50%
  - Birthdays: 23 people suffice!
  - Hash functions: $O(2^{n/2})$ hash function-evaluations
    - much better than $2^n + 1$

Hash functions in practice

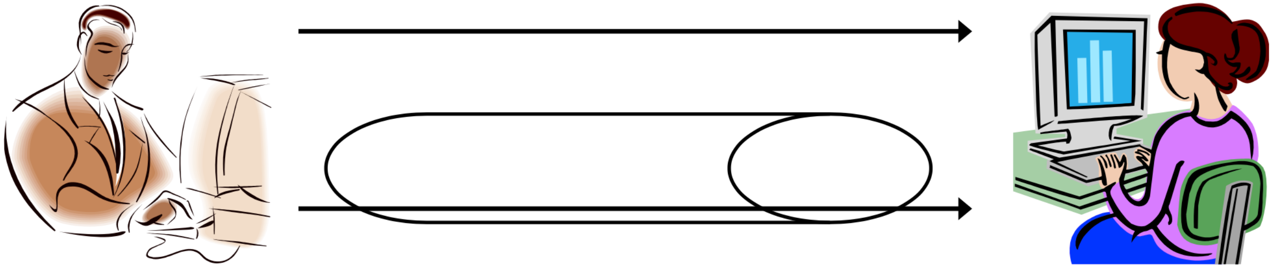
- MD5
  - Developed in 1991
  - 128-bit output length
  - Collisions found in 2004, should no longer be used
- SHA-1
  - Introduced in 1995
  - 160-bit output length
  - Theoretical analyses indicate some weaknesses
  - Very common; current trend to migrate to SHA-2

Hash functions in practice

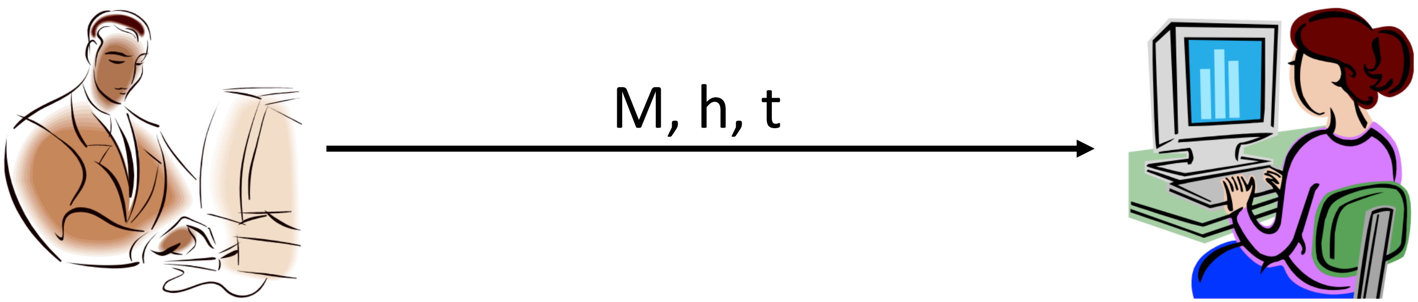
- SHA-2
  - 224-bit, 256-bit, 384-bit or 512-bit output lengths
  - No known significant weaknesses
- SHA-3/Keccak
  - Result of a public competition from 2008-2012
    - 64 submissions
    - 51 entered the first round
  - Very different design than SHA family
  - Supports 224, 256, 384, and 512-bit outputs

2.2 HMAC

Intuition…

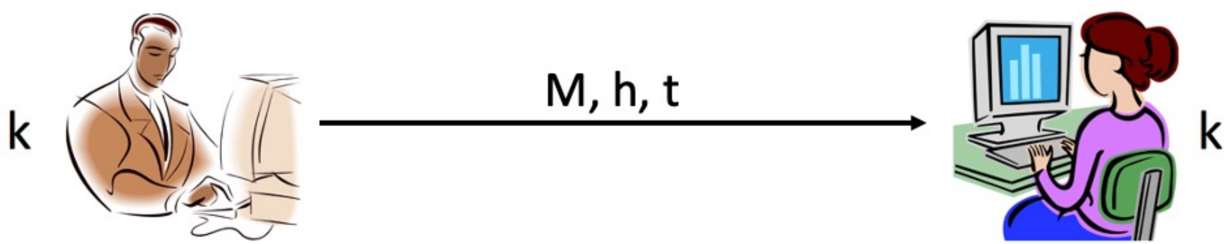


Hash-and-MAC



Instantiation?

- Hash function + block cipher-based MAC?
  - Need to implement two crypto primitives



M
$h = H(M)$
$t = \mathrm{Mac}_k(h)$

$h = H(M)$
$\mathrm{Vrfy}_k(h, t) = 1?$

HMAC

- Constructed entirely from (certain type of) hash functions
  - MD5, SHA-1, SHA-2
  - Not SHA-3

- Construction

$$\text{HMAC: } S(k,m) = H(k \oplus opad || H(k \oplus ipad || m))$$

  - ipad: 00110110
  - opad: 01011100

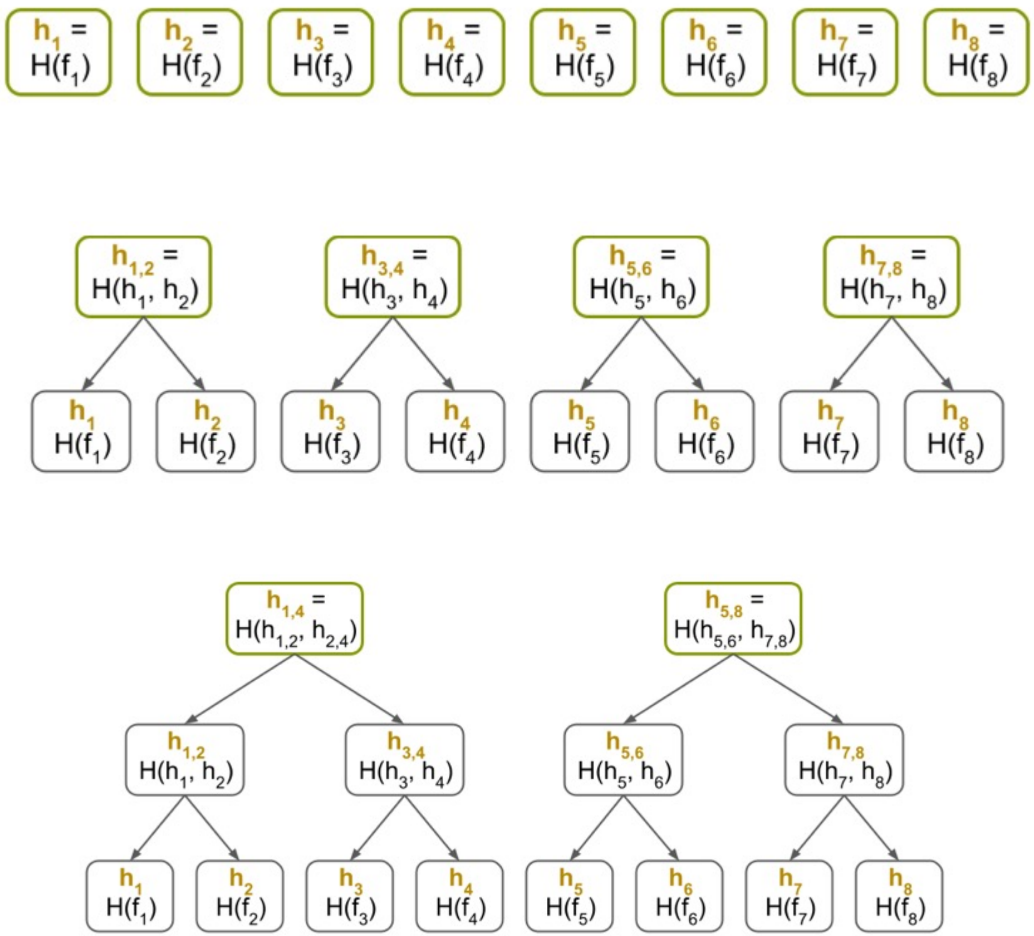
2.3 Merkle tree

Merkle tree

- Merkel tree
- $h = MHT(x_1, x_2, \dots, x_n)$
- Merkle proof

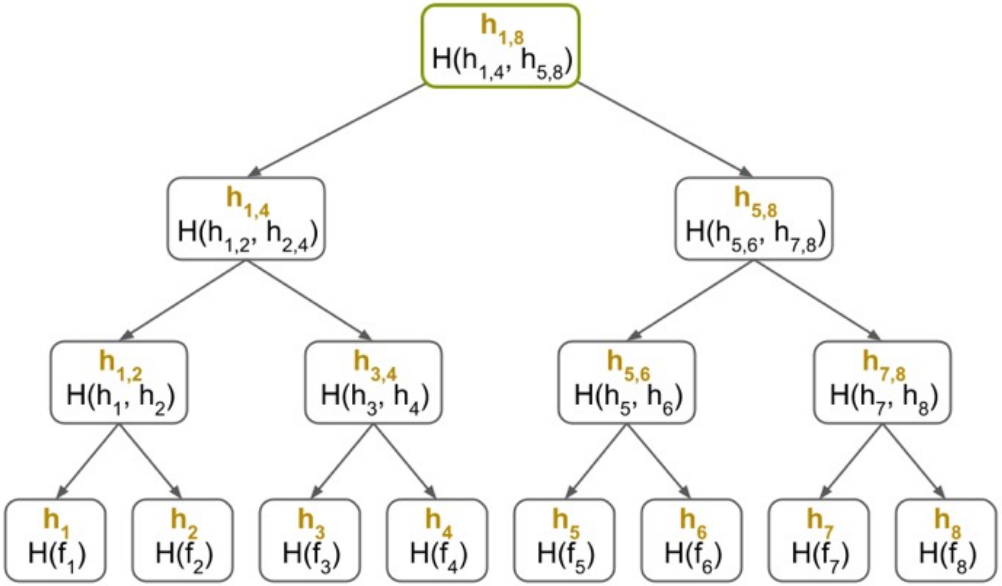
Construction of Merkle tree

- Suppose we have 8 files $(f_1, \dots, f_8)$, H is collision resistant hash function



Construction of Merkle tree

- Suppose we have 8 files $(f_1, \dots, f_8)$, H is collision resistant hash function



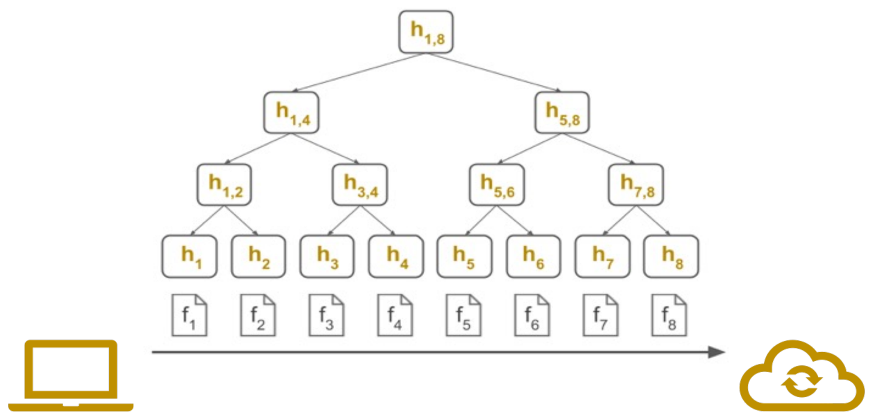
$$h_{1,8} = MHT(f_1, \dots, f_8)$$

Merkle proof

- Suppose we have 8 files $(f_1, \ldots, f_8)$, H is collision resistant hash function
- Merkle tree: $h_{1,8} = MHT(f_1, \ldots, f_8)$

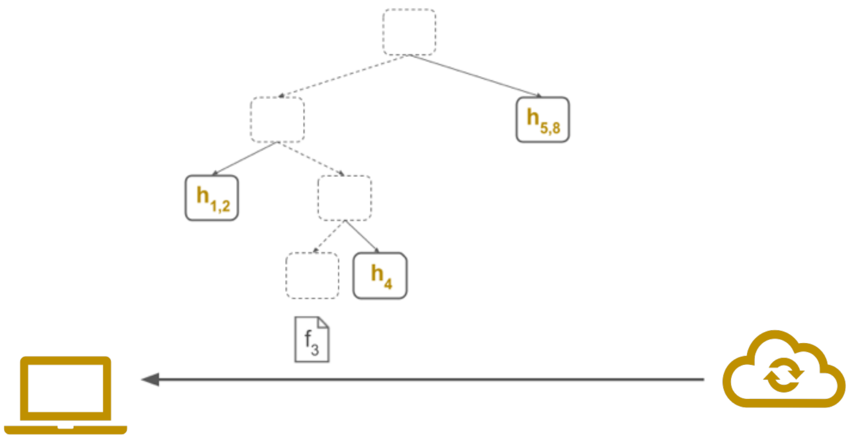
Whether f_3 is still stored in the cloud, and not changed?

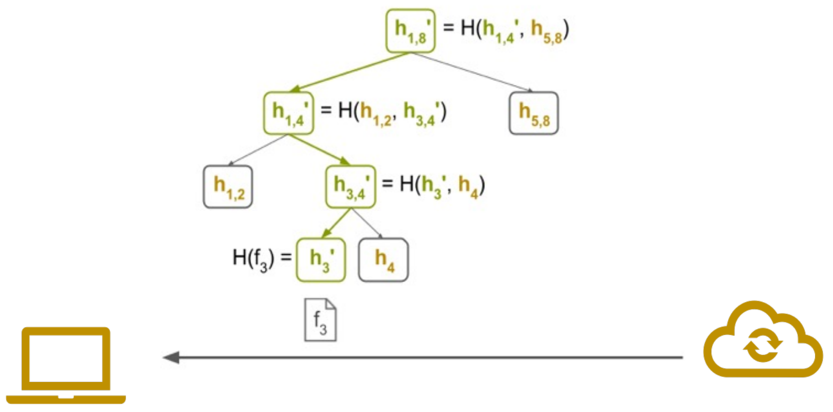
$h_{1,8}$



Merkle proof

- Integrity of f_3
- download $f_3, h_4, h_{1,2}, h_{5,8}$





2.4 Bitcoin

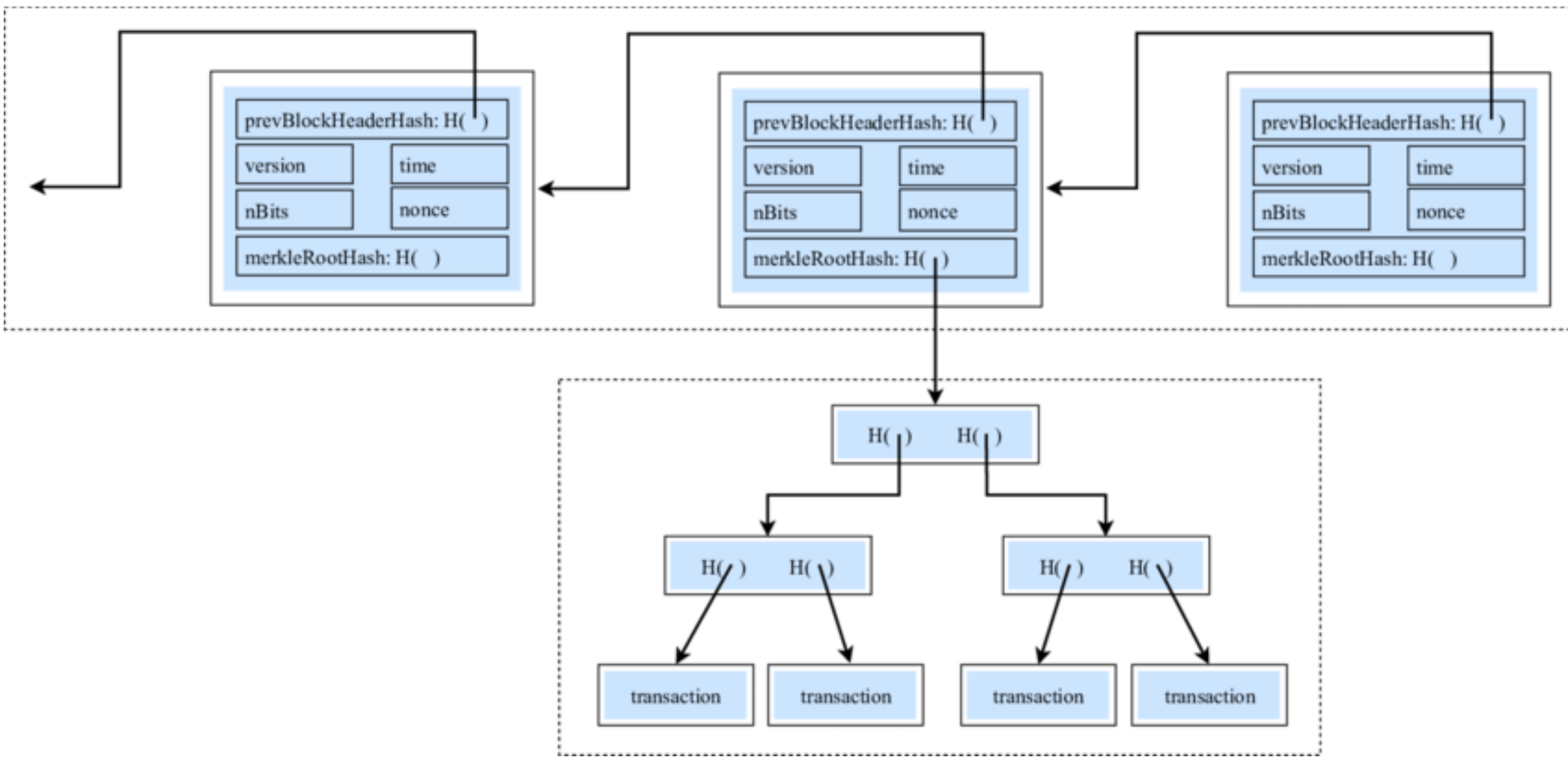
Bitcoin

- Bitcoin is the first and most widely recognized cryptocurrency.

- Bitcoin is based on the ideas laid out in a 2008 whitepaper titled Bitcoin: A Peer-to-Peer Electronic Cash System.
  - Satoshi Nakamoto

Block chain

- Block chain
  - Ledger of past transactions



Mining

- Mining is the process of adding transaction records to Bitcoin's public ledger of past transactions.
- Mining is also the mechanism used to introduce Bitcoins into the system.

```
block_header = version + previous_block_hash + merkle_root + time + target_bits + nonce
for i in range(0, 2**32):
    if sha256(sha256(block_header)) < target_bits:
        break
    else:
        continue
```

Summary

- Message authentication code
  - Message integrity
  - Fixed-length MAC
  - CBC-MAC
  - Secure communications
- Hash functions and applications
  - Hash functions
  - HMAC
  - Merkle tree
  - Bitcoin