Message Integrity and Hash Functions

Sushmita Ruj

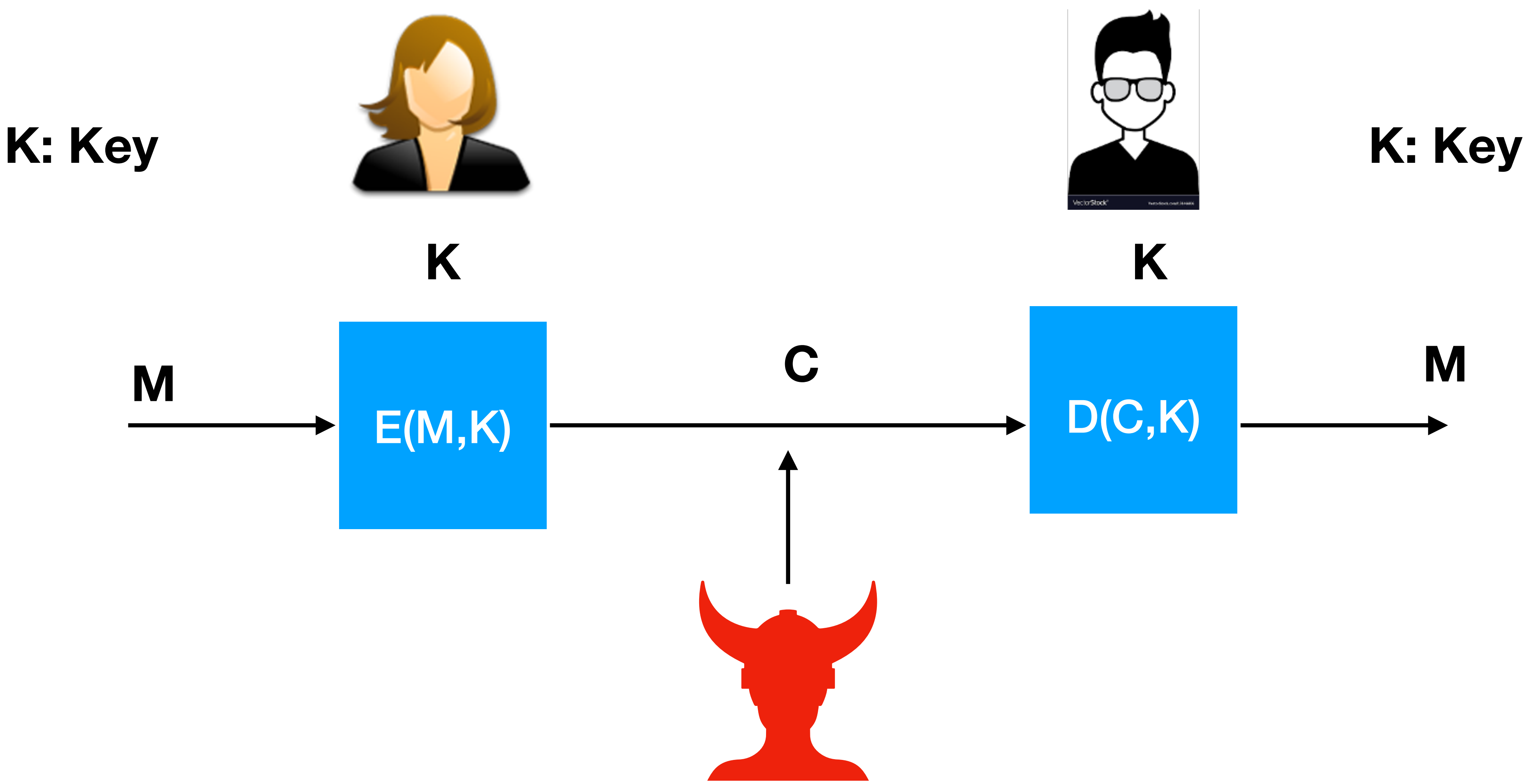
Recap

- Security definitions
- Indistinguishability
- Semantic Security
- Semantic Security of OTP
- Semantic Security of stream ciphers

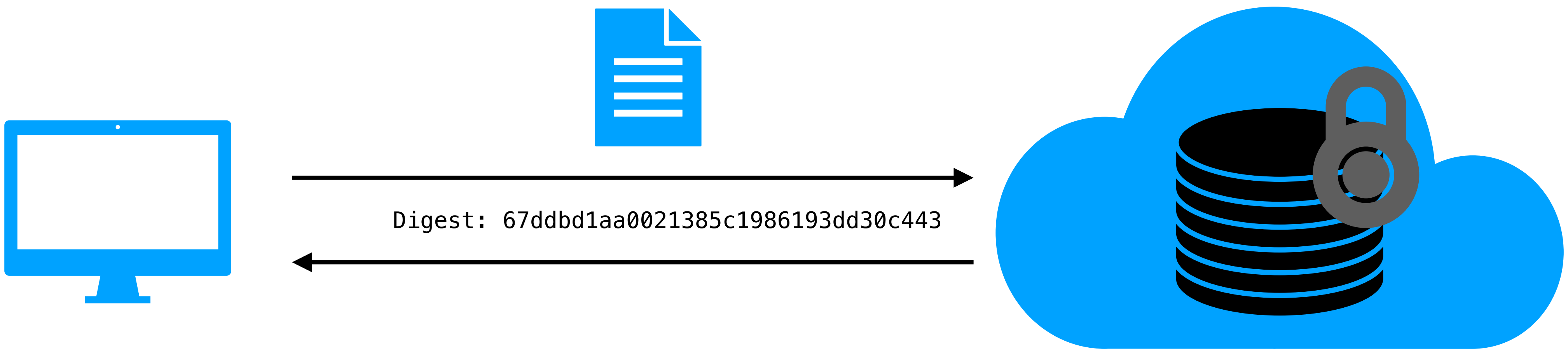
This Lecture

- Hash Functions
- Authentication
- Message Integrity

Encryption : Confidentiality



Integrity

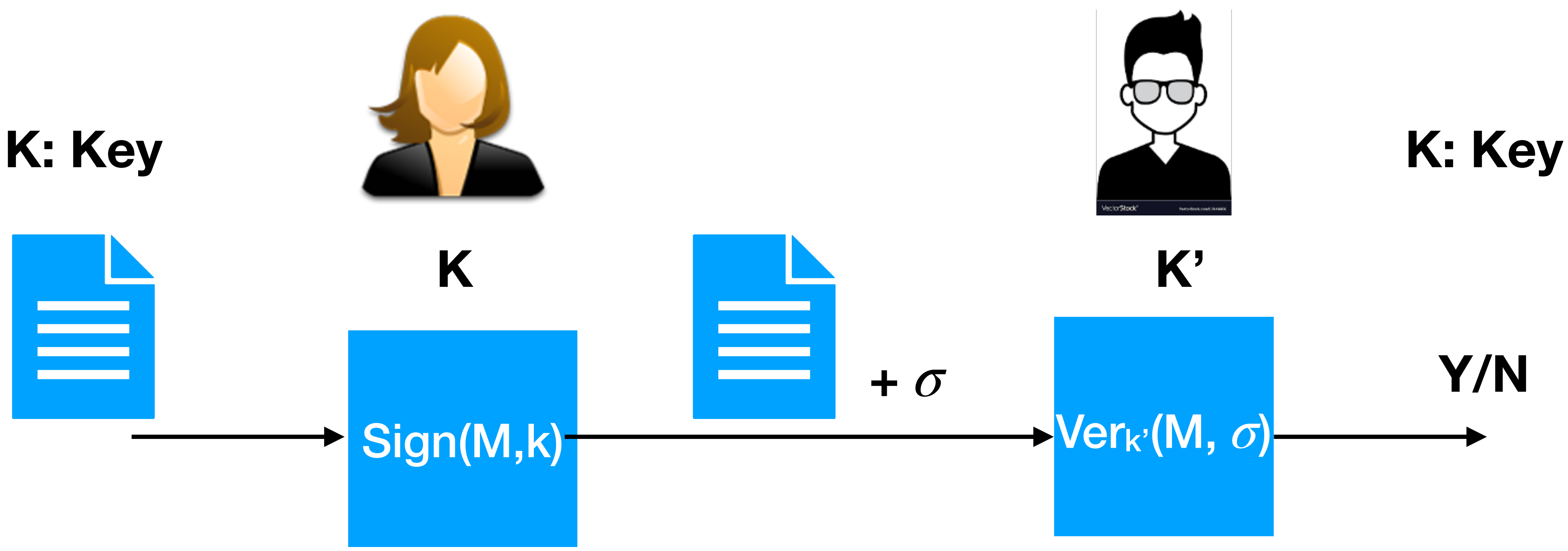


Hash Functions : H

$$H : \{0,1\}^* \rightarrow \{0,1\}^l$$

Week 3

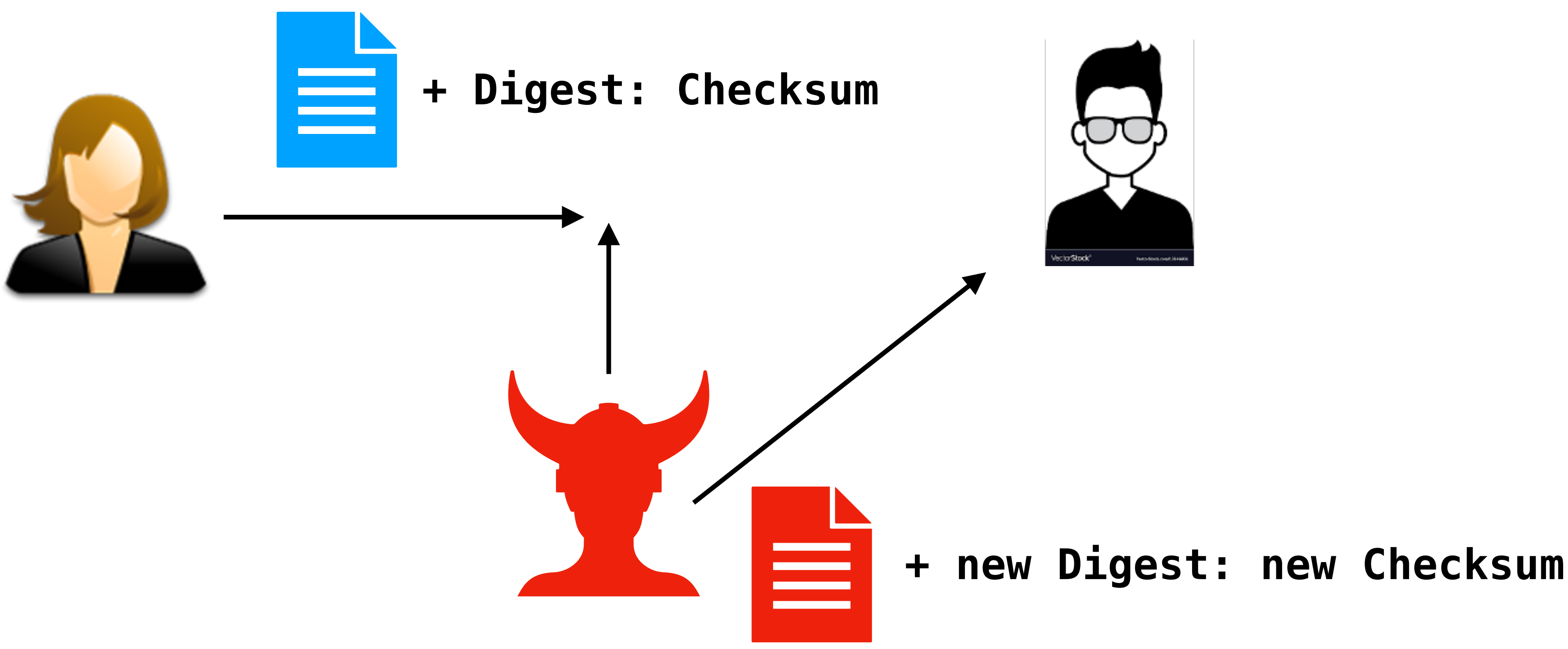
Digital Signatures



Week 7

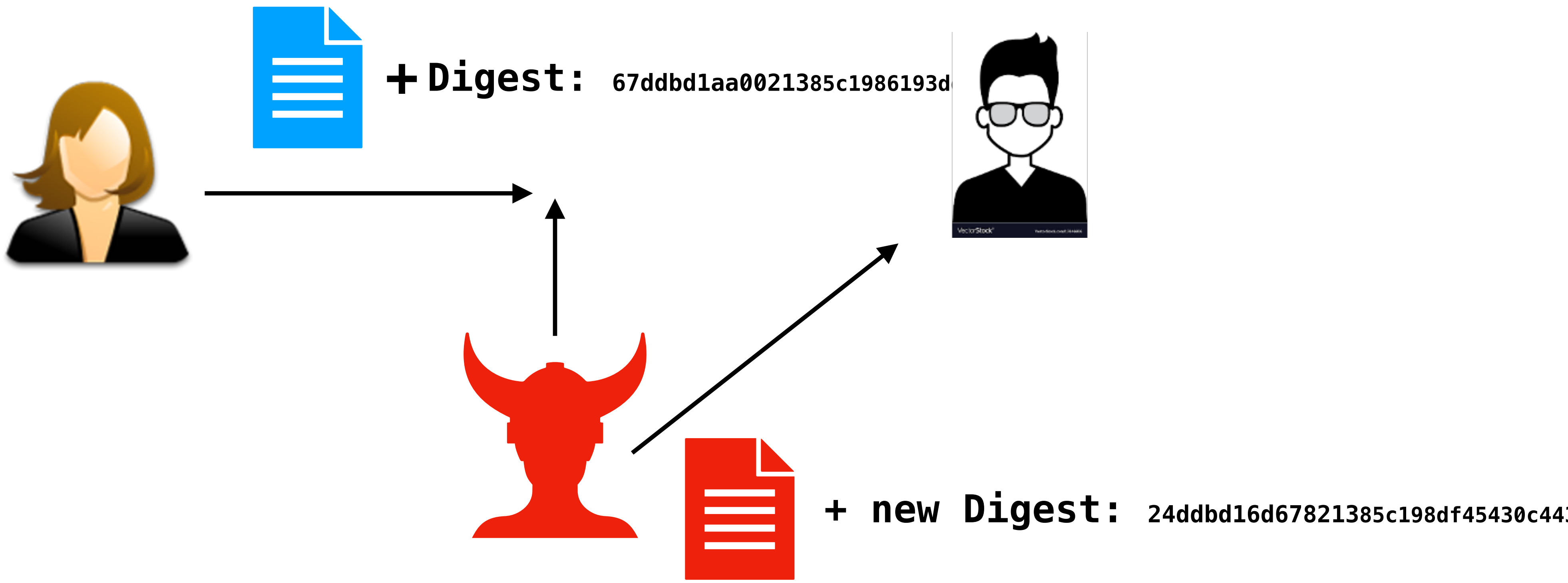
Message Integrity

Message Integrity



Adversary can hijack the message, create new message and new digest, Bob checks message, satisfied that it hasn't been modified

Message Integrity

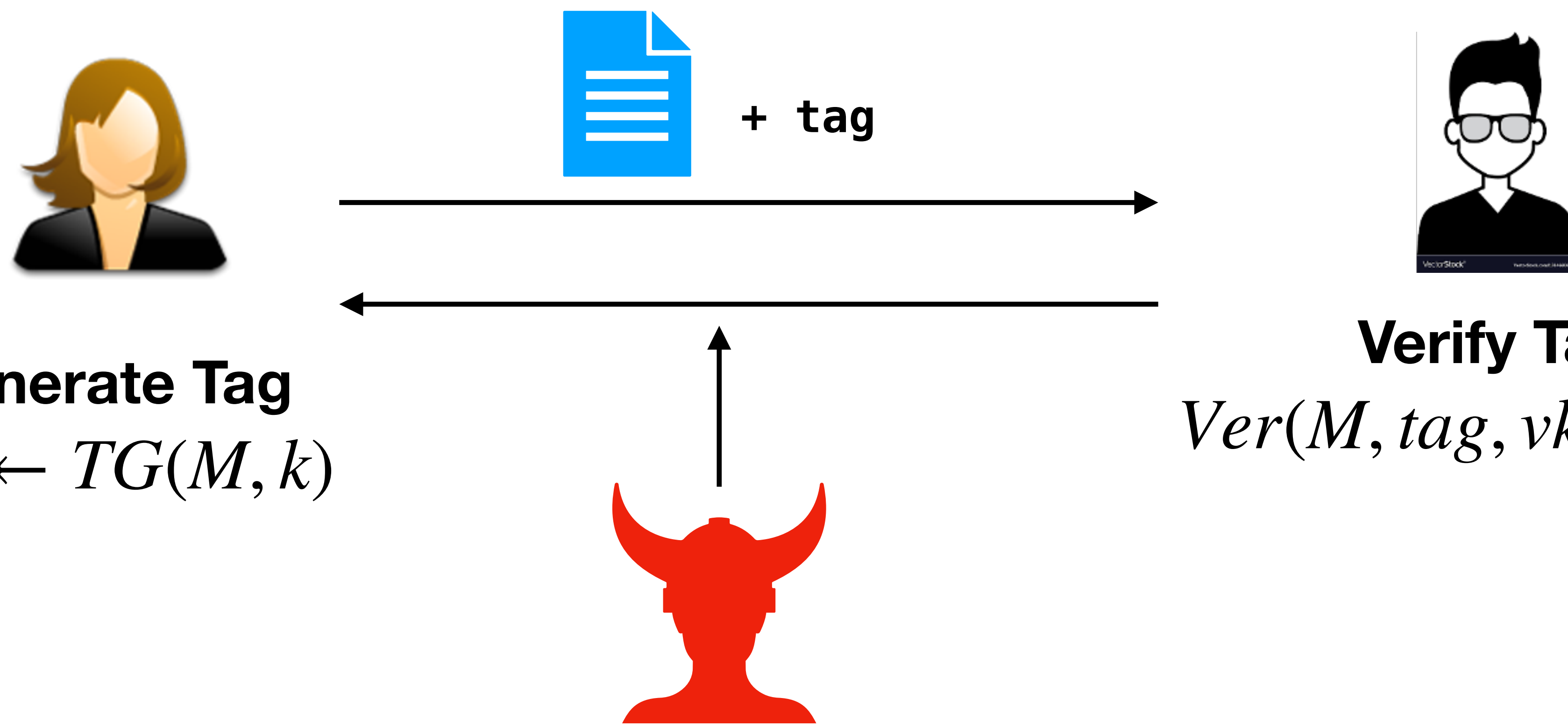


Adversary can hijack the message, create new message and new digest, Bob checks message, satisfied that it hasn't been modified

Use a keyed function, such that the key is known only to Alice and Bob

Message authentication code

Message Integrity



Generate Tag

$tag \leftarrow TG(M, k)$

Verify Tag

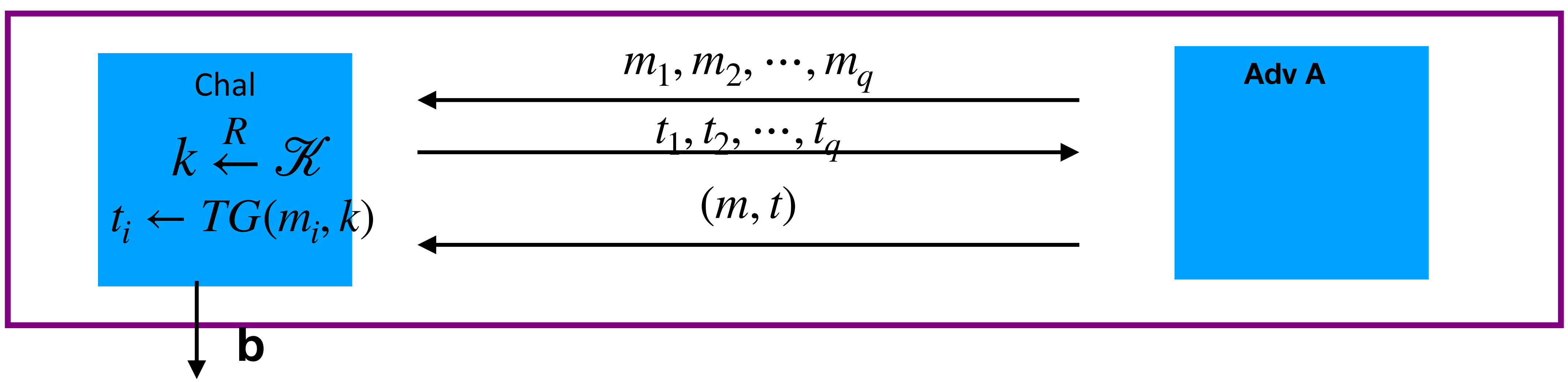
$Ver(M, tag, vk) \rightarrow 0/1$

Security: Eve does not know k, so can't generate a new tag on a new message

Secure Macs

- Choose message attack: Attacker observers message $(m_1, t_1), (m_2, t_2)\cdots, (m_q, t_q)$
- Attacker's goal: Generate a new message/tag pair, $(\hat{m}, \hat{t})$, s.t. $\hat{m} \neq m_1, m_2, \cdots, m_q$ - **Existential forgery**
- **Loosely we say that the tag is unforgable**

Security of Mac



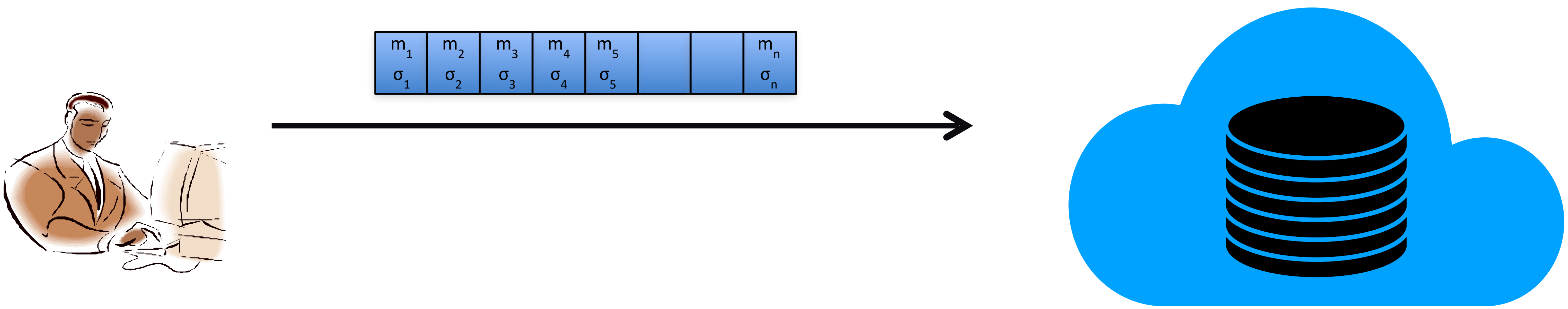
b=1 if Ver(k,m,t) = `yes' and $(m,t) \notin \{ (m_1,t_1) , \ldots , (m_q,t_q) \}$
b=0 otherwise

Def: I=(TG, Ver) is a **secure MAC** if for all "efficient" adversary A:

$\text{Adv}_{\text{MAC}}[A,I]$ = Pr[Chal. outputs 1] is "negligible."

Intuitively: Tag Should be long and random, else easy to guess

Data Auditing



σ_i is the tag (unforgeable)

Question: How to efficiently verify that server has not tampered with the data Without downloading the data

Construction of MAC

- Secure PRFs yield secure MACs
- Let $F: K \times X \rightarrow Y$ be a secure PRF
- Construct a MAC I = (TG,V), such that
- TG(k,m) = F(m,k)
- Ver(m,t,k) = 1 if t=TG(k,m) and 0, o.w.

Security Proof

Theorem If **$F: K \times X \longrightarrow Y$** is a secure PRF and $1/|Y|$ is negligible (i.e. $|Y|$ is large) then I_F is a secure MAC.

Proof

In particular, for every efficient MAC adversary A attacking I_F

there exists an efficient PRF adversary B attacking F s.t.:

$$\mathrm{Adv}_{\mathrm{MAC}}[A, I_F] \leq \mathrm{Adv}_{\mathrm{PRF}}[B, F] + 1/|Y|$$

$\Rightarrow$ I_F is secure as long as $|Y|$ is large, say $|Y| = 2^{80}$.

Proof in Boneh-Shoup, do as an exercise

Examples of MAC

- AES can be used (mentioned last class), however it is small
- Convert small PRF to Large PRFs
- CBC-MAC: Used in banking – ANSI X9.9, X9.19, FIPS 186-
- HMAC: Internet Protocols, like SSH, IPSec, etc

Construction of Mac

PRFs

CBC-MAC, ECBC-MAC, CMAC : commonly used with AES (e.g. 802.11i)

NMAC : basis of HMAC

PMAC: a parallel MAC

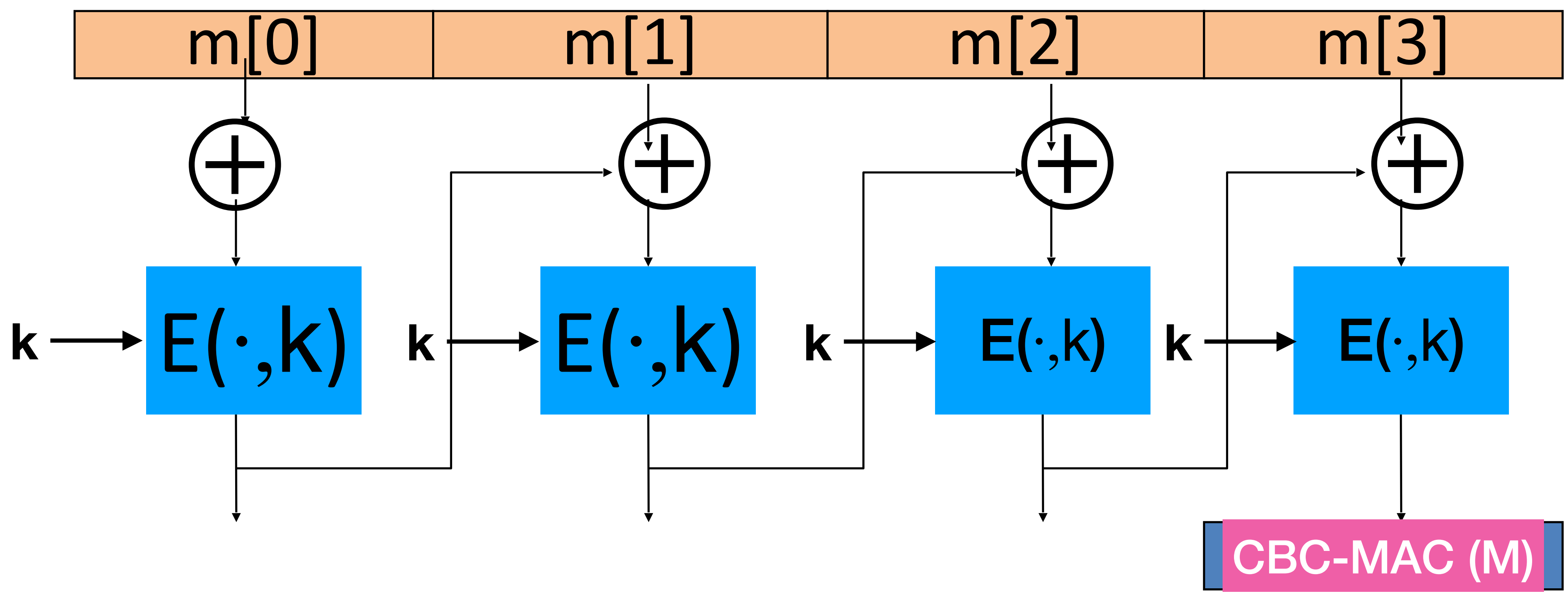
Randomised MAC

Carter-Wegman MAC: built from a fast one-time MAC

CBC-MAC

$$E : \{0,1\}^l \times \{0,1\}^m \rightarrow \{0,1\}^m$$

$IV \in \{0,1\}^n$



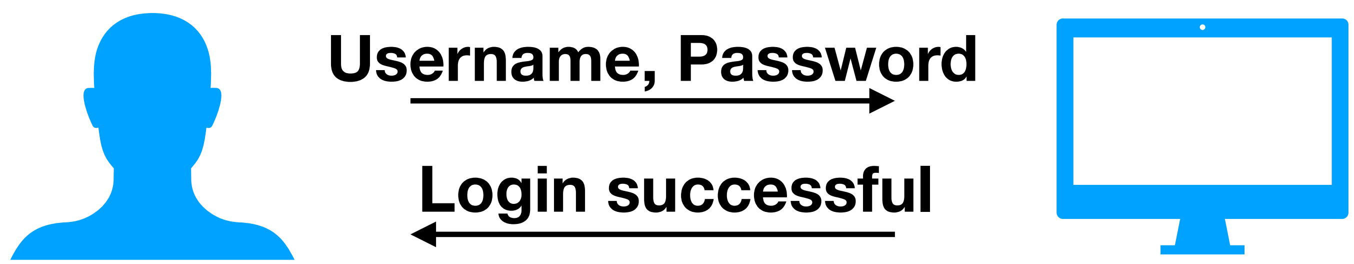
Security of the Block-cipher implies security of MAC

Hash Functions

Hash Function-Overview

- Integrity: Collision-resistance
  - H(m) allows verification of m
  - Application: Blockchains, file distributions etc.
- Confidentiality: One-way functions (OWF)
  - OWF-hash H(m) does not reveal m
- Pseudo-randomness:
  - If m is sufficiently random, then H(m) is pseudorandom

Applications

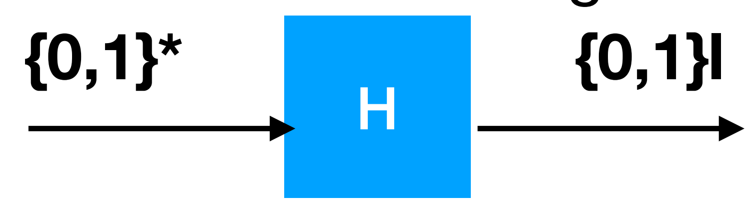


Given H(Password), cannot retrieve password

Username	H(password)

What is a Hash Function?

- $H: \{0,1\}^* \rightarrow \{0,1\}^l$: Given variable length input produces a fixed length output (also called msg digest or fingerprint)



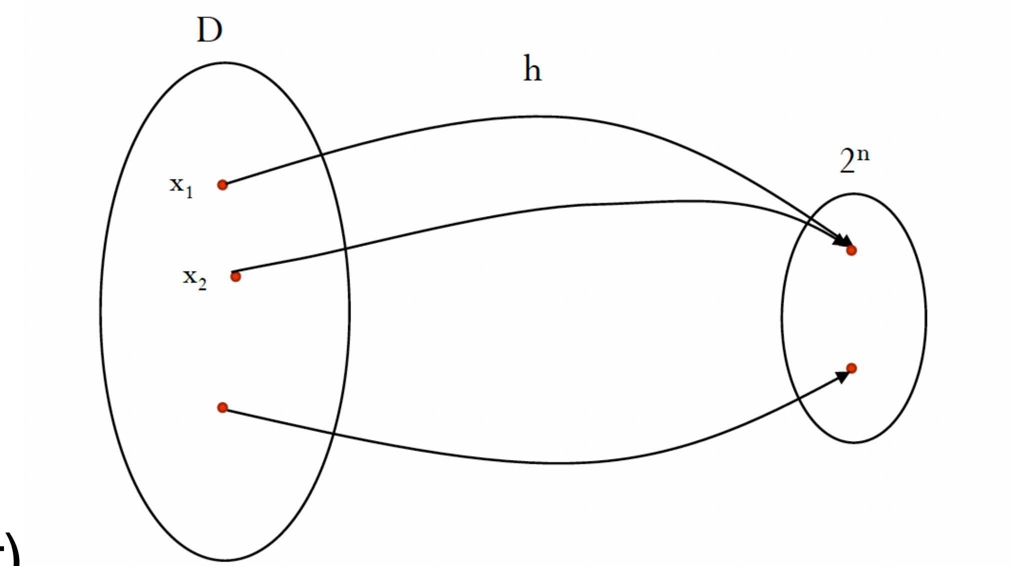
Properties of a hash function

- Given *x*, *y=H(x)* is easy to compute
- Given *y= H(x)*, *x* is computationally infeasible to compute (One-wayness)
- It is computationally infeasible to find two strings *x,x’* (*x≠x’*), such that *H(x)* = *H(x’)* (collision resistance)
- Given *y=H(x)*, it is difficult to find *x≠x’*, such that *H(x)* = *H(x’)* (second preimage resistance)

- Output cannot be too short: One can find collisions by random search (“birthday attack”)
- For any input, the output should be “random”; cannot find *(x,y)*, s.t. *x* is short and *y=H(x)*, except by picking *x* and evaluating *H(x)*.

Collision Resistant Hash Functions

Let H: M →T be a hash function (|M| >> |T|)



A **collision** for H is a pair m_0 , $m_1 \in M$ such that:

$$H(m_0) = H(m_1) \quad \text{and} \quad m_0 \neq m_1$$

A function H is **collision resistant** if for all (explicit)

$Adv_{CR}[A,H]$ = Pr[A outputs collision for H]

is "neg".

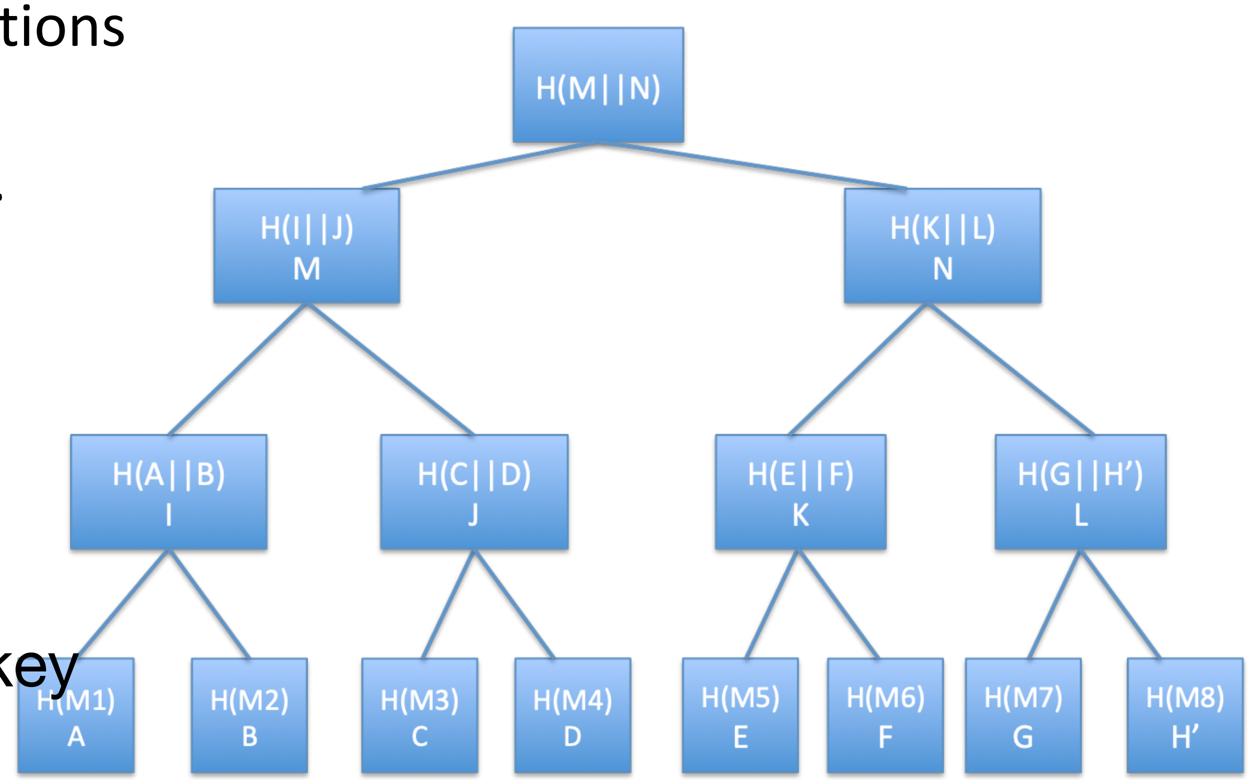
Example: SHA-256 (outputs 256 bits)

Pasword Management

- Simple Password manager
- Salt Based Password Manager
- Rainbow Tables
- Password checker: Set membership and Bloom filters

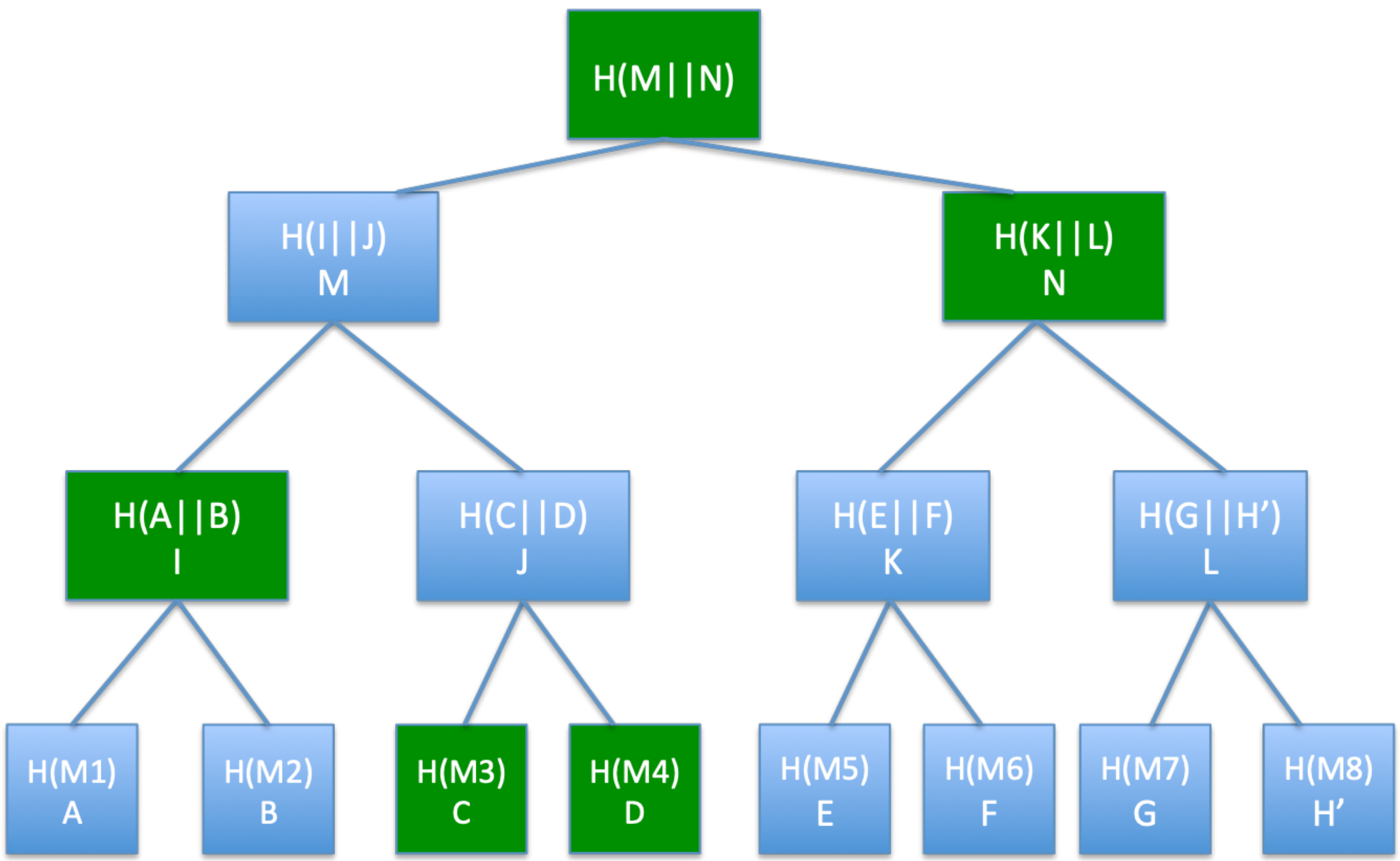
(Magnificent)Merkle Tree

- Devised and patented by Ralph Merkle as a part of a signature scheme in 1979.
- Collision resistant property of hash functions
- Allows efficient and secure verification of the contents of a large data structure.
- Used in Bitcoins, Ethereum, IPFS, etc.



Merkle is also famous for Merkle Puzzles, that laid the foundation for public key cryptography. read about it.

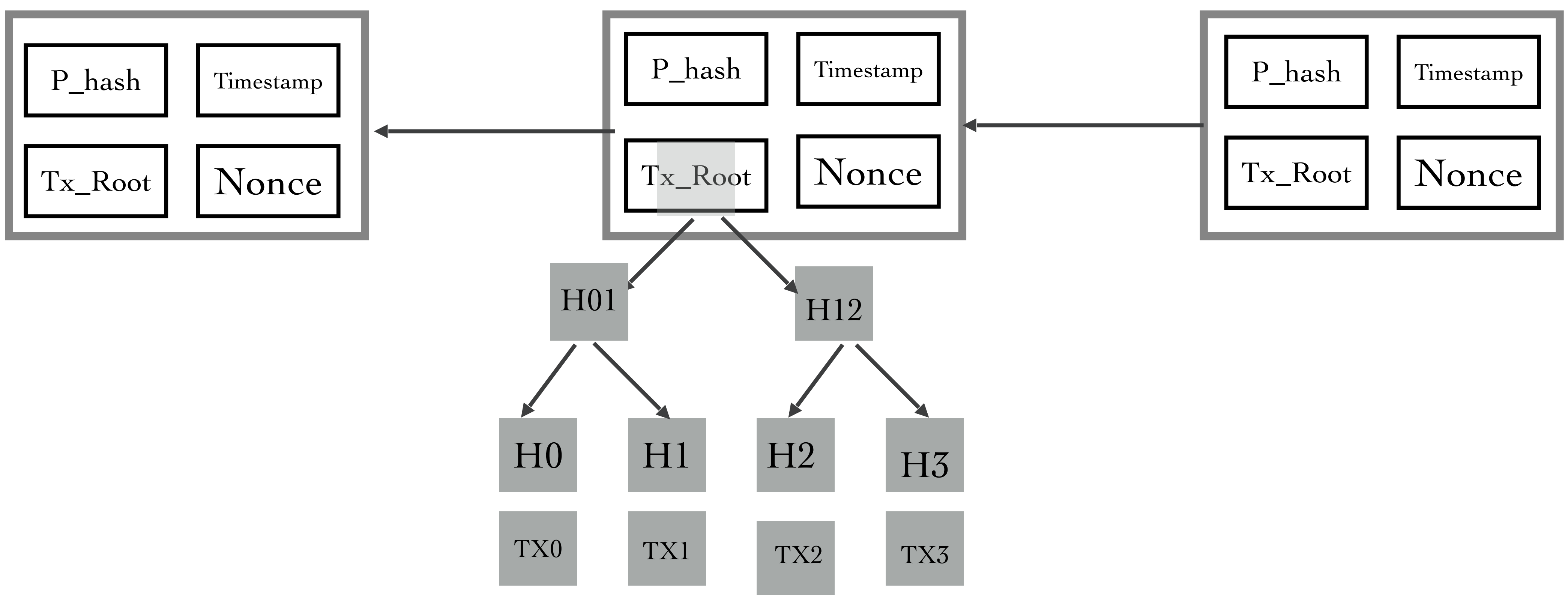
Merkle Tree



To check D, Proof = <H(M4), H(M3), H(A||B), H(K||L), H(M||N)> should match with root.

Proof size log(n), n is the number of blocks

Blockchain



P_hash
Timestamp
Tx_Root
Nonce
P_hash
Timestamp
Tx_Root
Nonce
P_hash
Timestamp
Tx_Root
Nonce
H01
H12
H0
H1
H2
H3
TX0
TX1
TX2
TX3

Next Lecture

- Construction of Collision Resistant Hash Function
- Merkle-Damgard construction (SHA-1, SHA-256)
- Sponge Construction (Keccak/SHA-3)
- HMAC
- Random Oracles and Hash Functions

Thank you