Blockchain Principles and Applications

Amir Mahdi Sadeghzadeh, PhD

Data and Network Security Lab (DNSL)
Trustworthy and Secure AI Lab (TSAIL)

Most slides have been adapted from ECE/COS 470, Pramod Viswanath, Princeton 2024 and CS251, Dan Boneh, Stanford 2023.

Recap

What are Blockchains?

Blockchains are decentralized digital trust platforms

Decentralized system

- No single entity (person/company) is responsible for the smooth operation of the system.
  - Blockchains are peer-to-peer systems
    - each peer has the same prescribed behavior
    - No peer is unique.
    - Peers communicate with each other by exchanging messages
    - Beyond this message exchange, peers function independently of one another.

Trust

- Human success is based on flexible cooperation in large numbers. This requires trust!

Evolution of Trust over human history

Bitcoin: the original blockchain

Cryptocurrency

medium of exchange and store of value

Born during the **2008 Financial Crisis**

Anonymous inventor

pseudonym: Satoshi Nakamoto

Very secure

no attacks, has been live continuously

Bitcoin performance

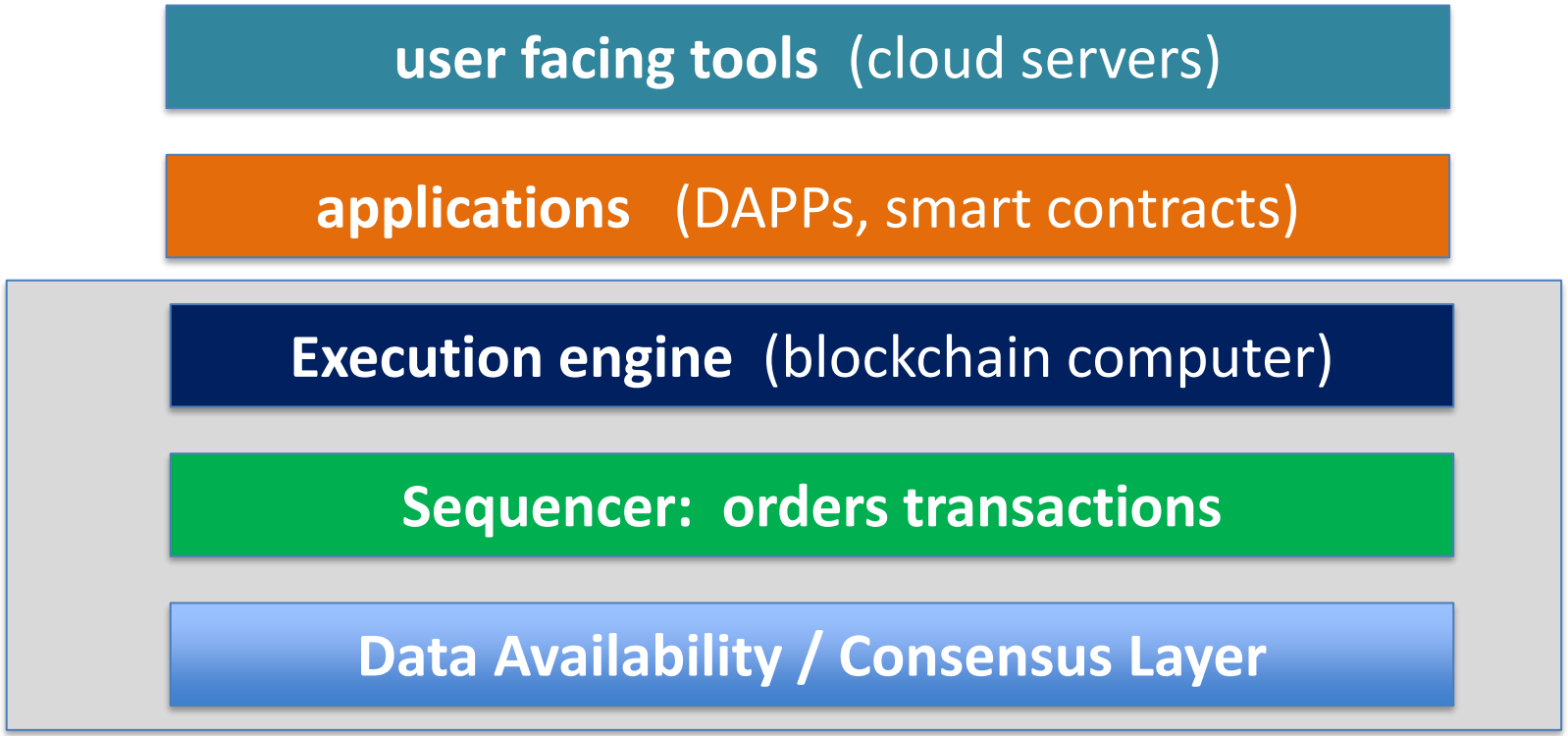
1. Security – 50% adversary
2. Transaction throughput – 7 tx/s
3. Confirmation Latency – hours
4. Energy consumption – medium size country
5. Compute – specialized mining hardware
6. Storage – everyone stores everything
7. Communication – everyone tx/rx everything

Principles of Blockchains

- **Conceptual** Principles
  - Algorithmic designs
  - Byzantine resistance
  - Security analysis
  - Mechanism design and incentive compatibility

- **Engineering** Principles
  - Modular software stack
  - Software engineering
  - Integration and composition of different modules – secure, yet efficient

What is a blockchain?



Consensus layer (informal)

A **public** append-only data structure:

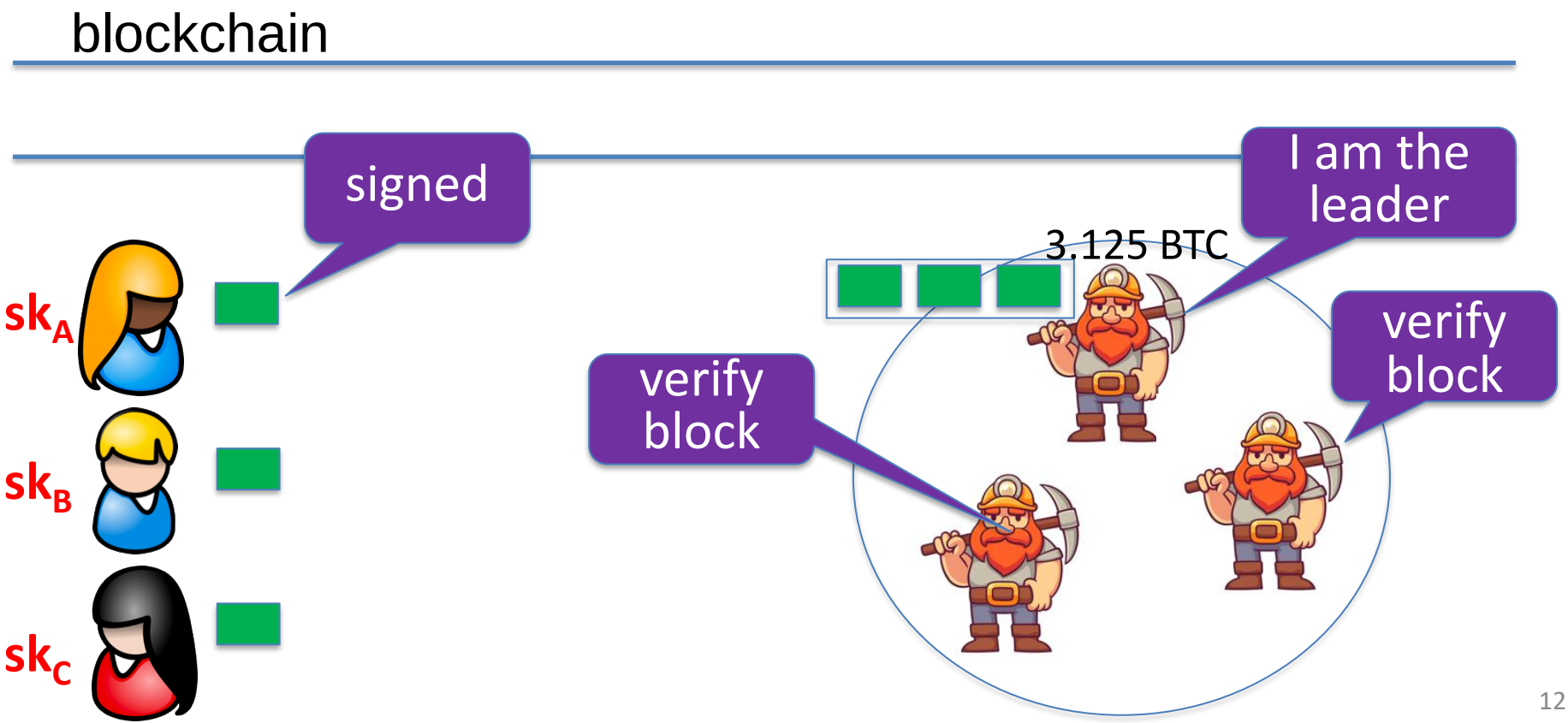
achieved by replication

- **Persistence**: once added, data can never be removed*
- **Safety**: all honest participants have the same data**
- **Liveness:** honest participants can add new transactions
- **Open(?)**: anyone can add data (no authentication)

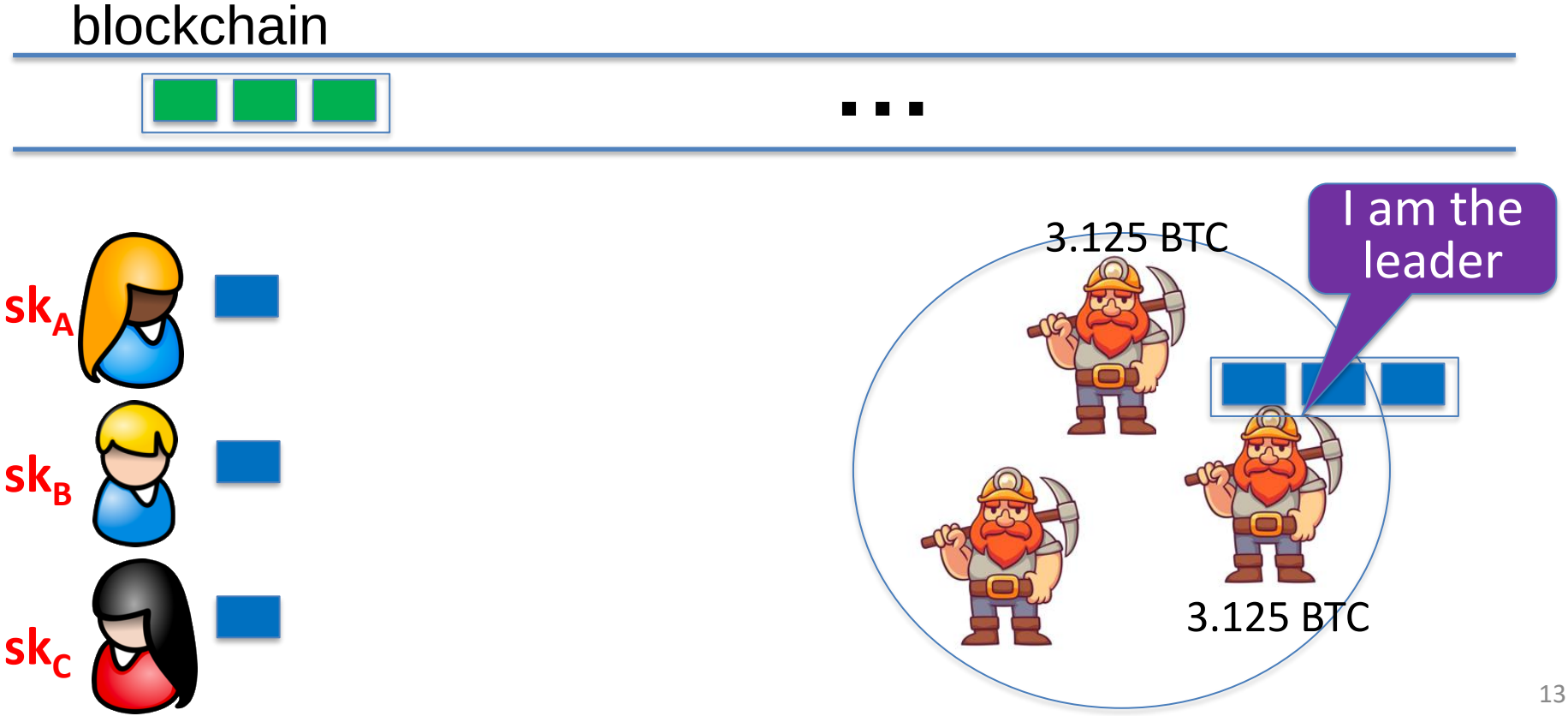
Data Availability / Consensus layer

Introduction (Continue)

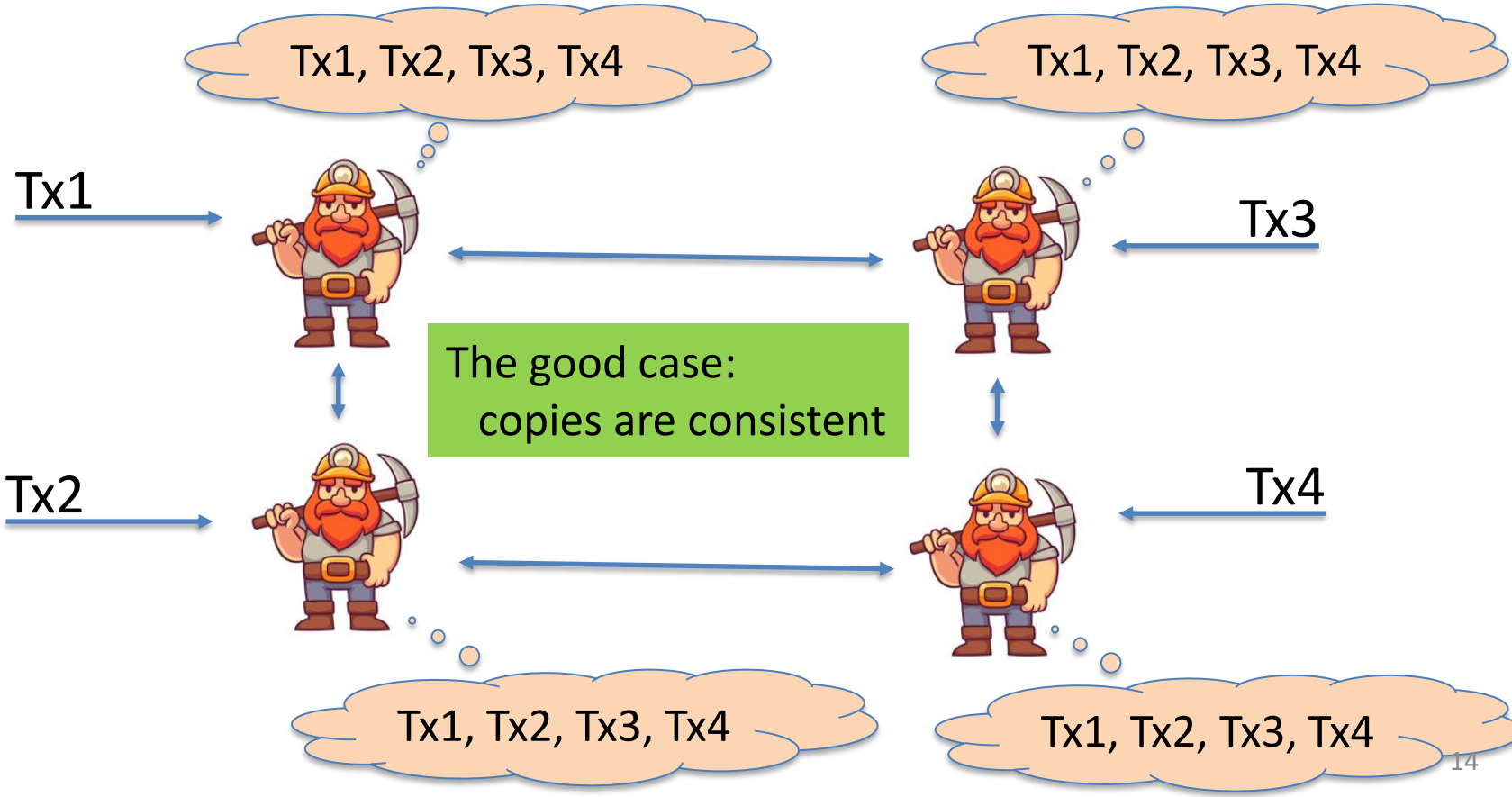
How are blocks added to chain?



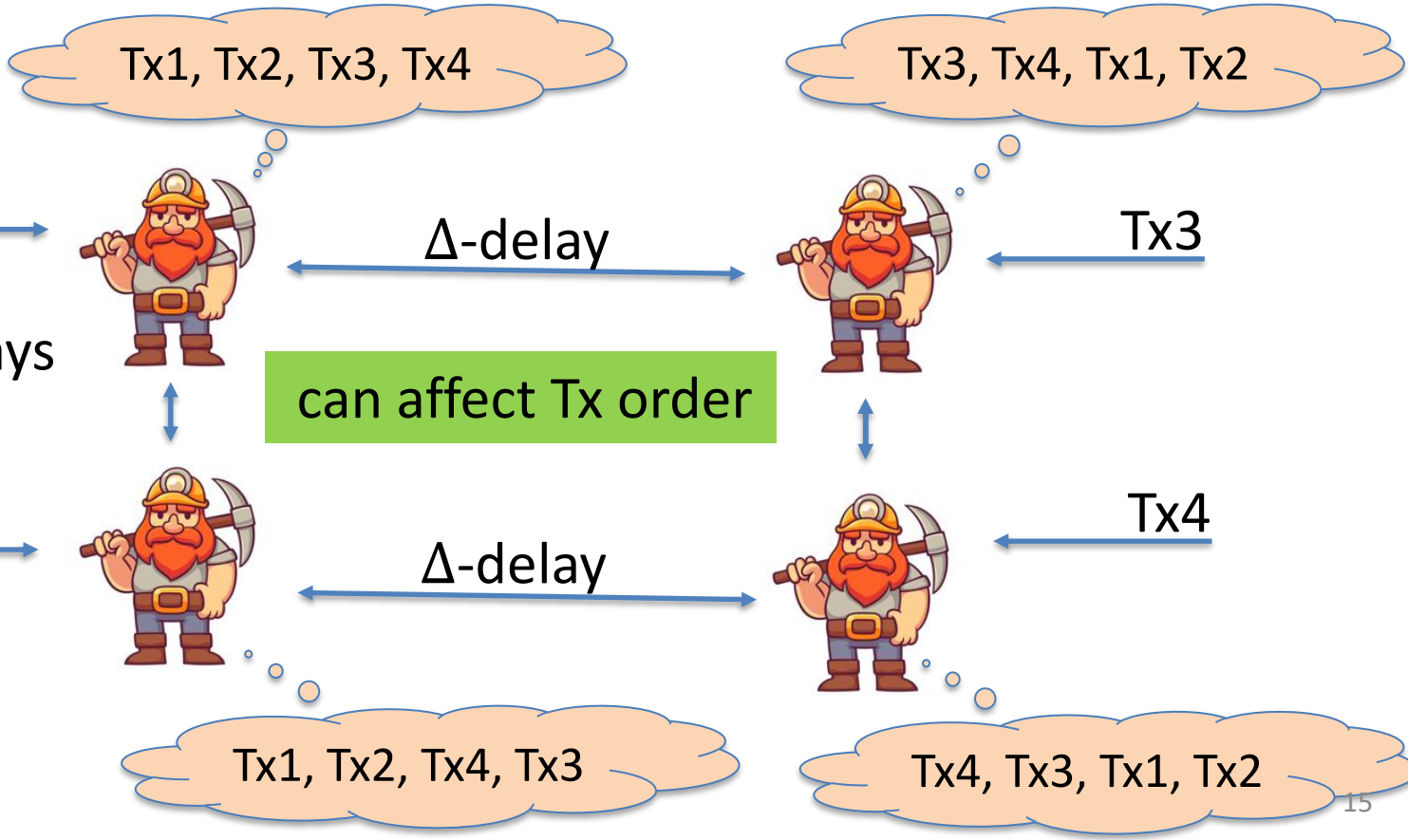
How are blocks added to chain?



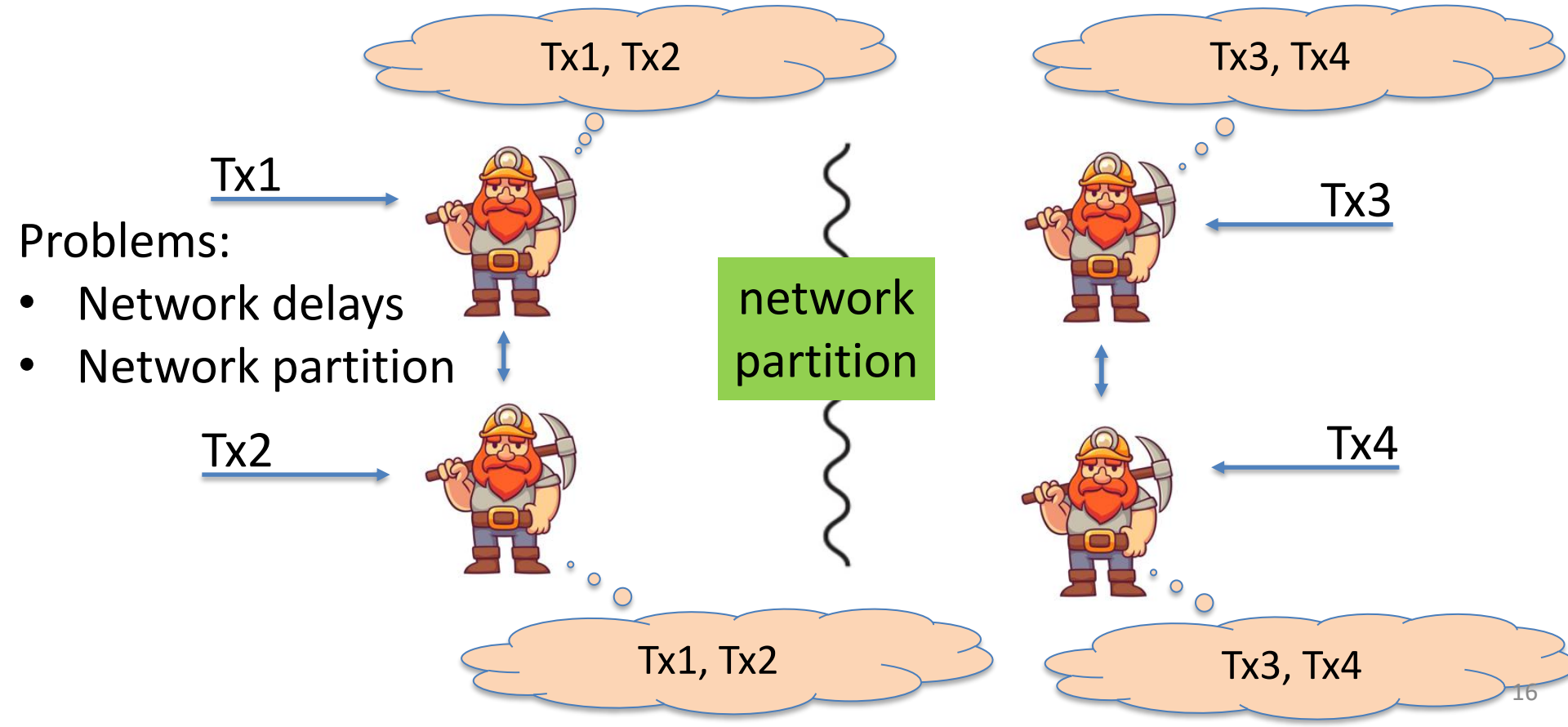
Why is consensus a hard problem?



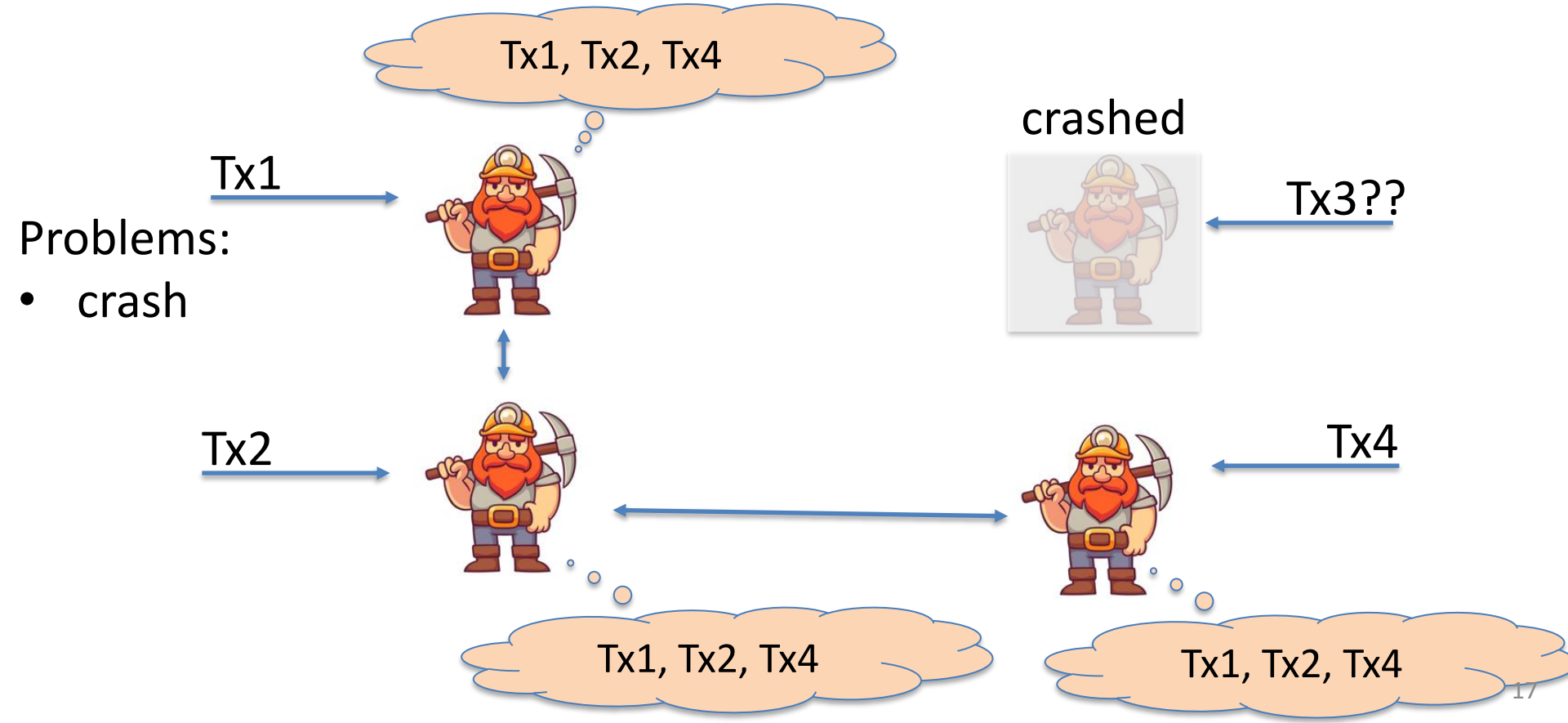
Why is consensus a hard problem?



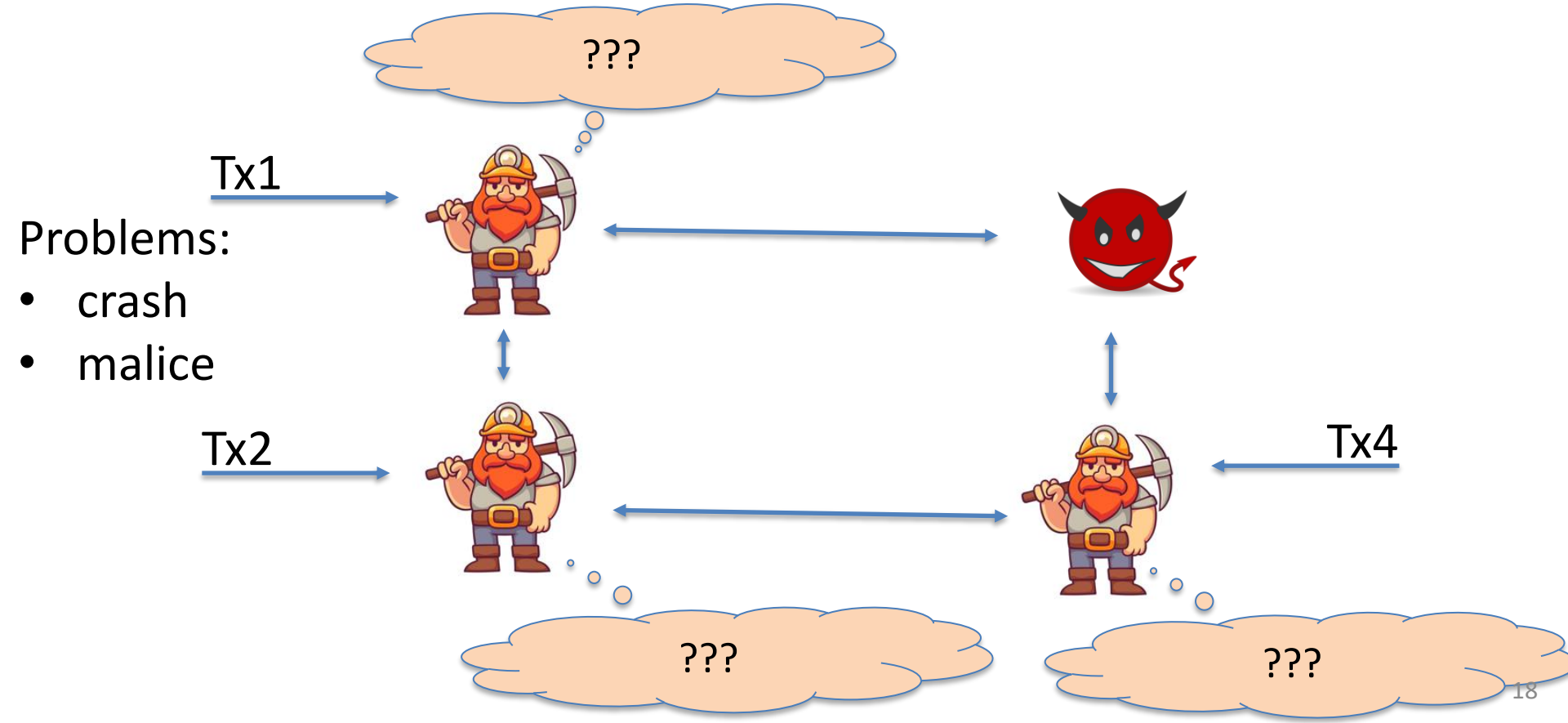
Why is consensus a hard problem?



Why is consensus a hard problem?



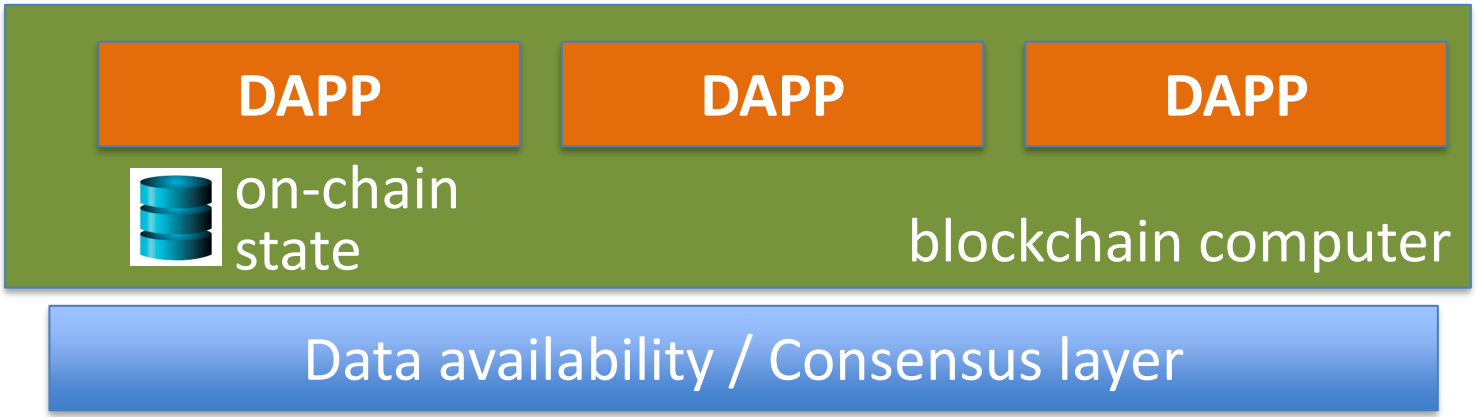
Why is consensus a hard problem?



Next layer: the blockchain computer

Decentralized applications (DAPPs):

- Run on blockchain: code and state are written on chain
- Accept Tx from users ⇒ state transitions are recorded on chain



Next layer: the blockchain computer

Top layer: user facing servers

end user

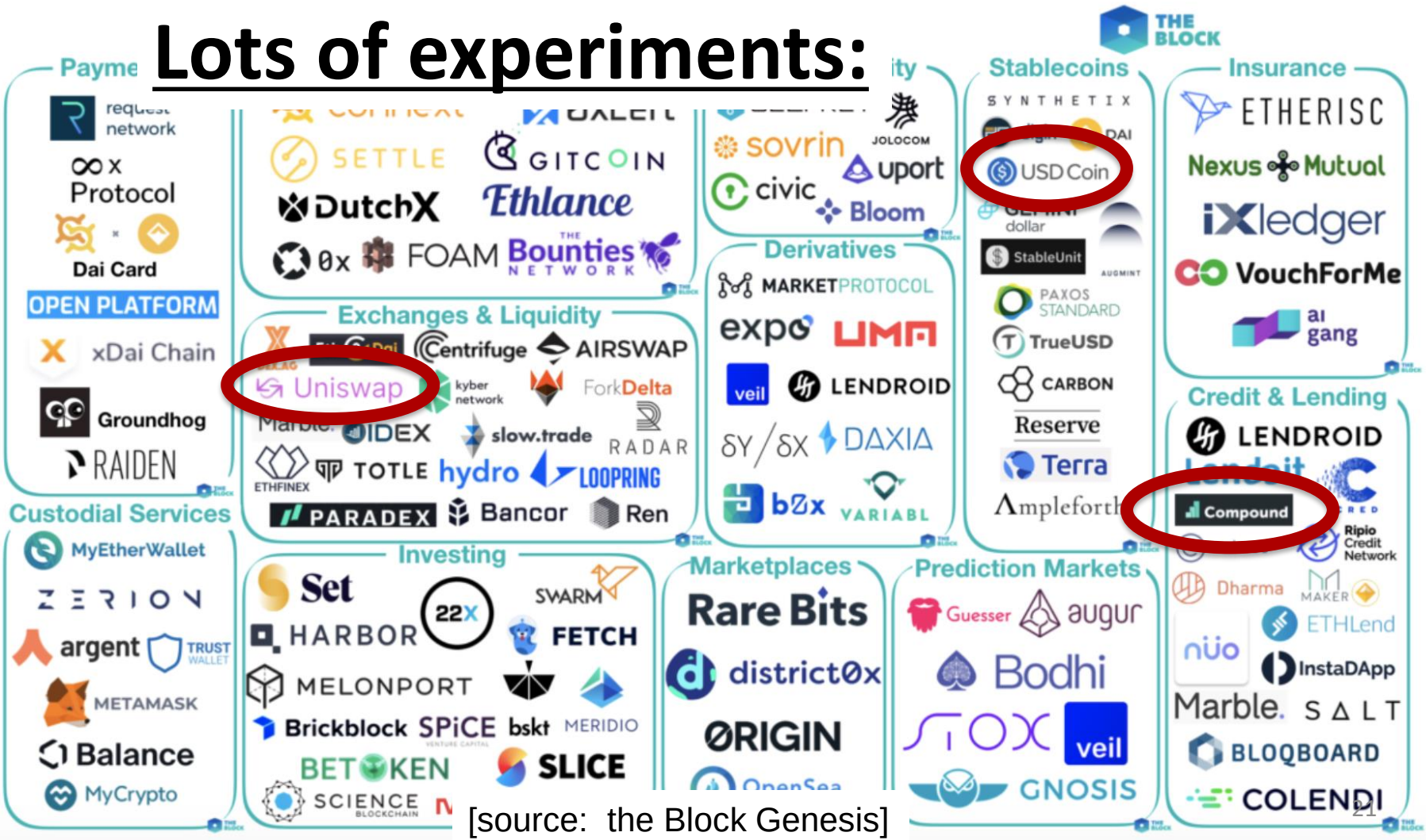
DAPP DAPP DAPP

on-chain state

blockchain computer

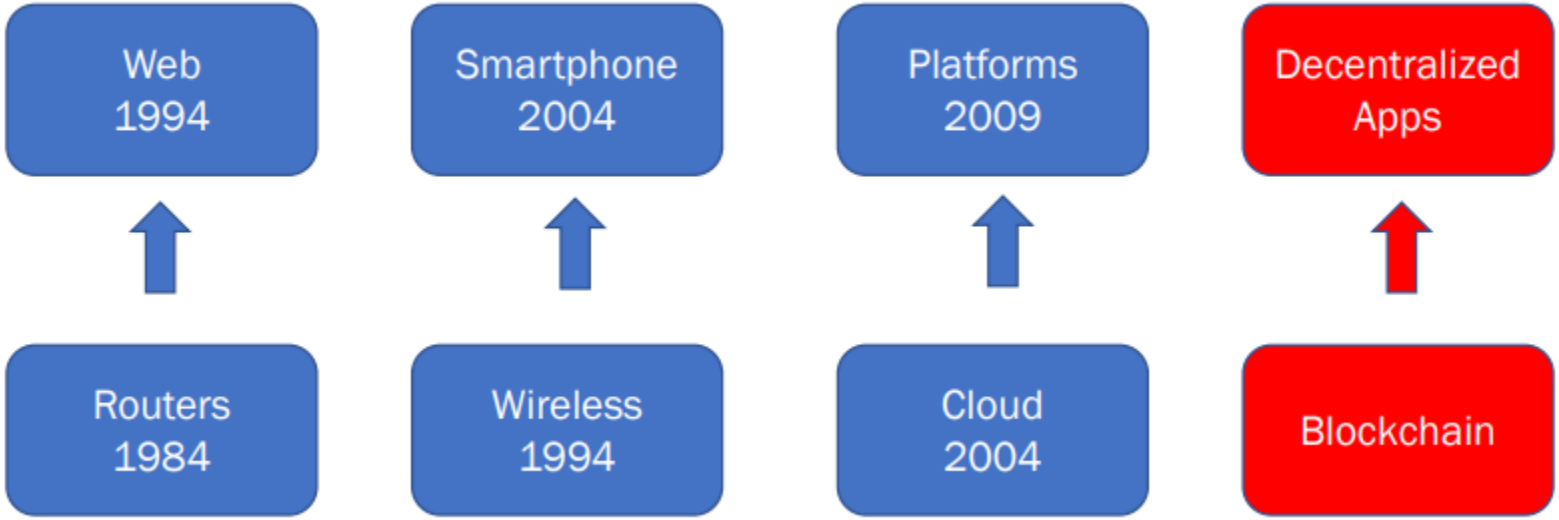
Data availability / Consensus layer

Lots of experiments:



[source: the Block Genesis]

Infrastructure to applications



Security and Cryptography Primitive Review

What is Security?

- Confidentiality
- Integrity
- Availability

Confidentiality (محرمانگی)

- “The property that information is not made available or disclosed to unauthorized individuals, entities, or process [i.e. to any unauthorized system entity].” {definitions from RFC 2828]
- Not the same as privacy

Confidentiality (محرمانگی)

- “The property that information is not made available or disclosed to unauthorized individuals, entities, or process [i.e. to any unauthorized system entity].” {definitions from RFC 2828]
- Not the same as privacy
- Privacy: “the right of an entity (normally a person), acting in its own behalf, to determine the degree to which it will interact with its environment, including the degree to which the entity is willing to share information about itself with others.”
- Privacy is a reason for confidentiality

Integrity (صحت)

- Data integrity: “The property that data has not been **changed**, destroyed, or lost in an unauthorized or accidental manner”

- System integrity: “The quality that a system has when it can perform its **intended function** in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation.”

- Often of more commercial interest than confidentiality

Availability (دسترس‌پذیری)

- “The property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system; i.e. a system is available if it provides services according to the system design whenever users request them.”

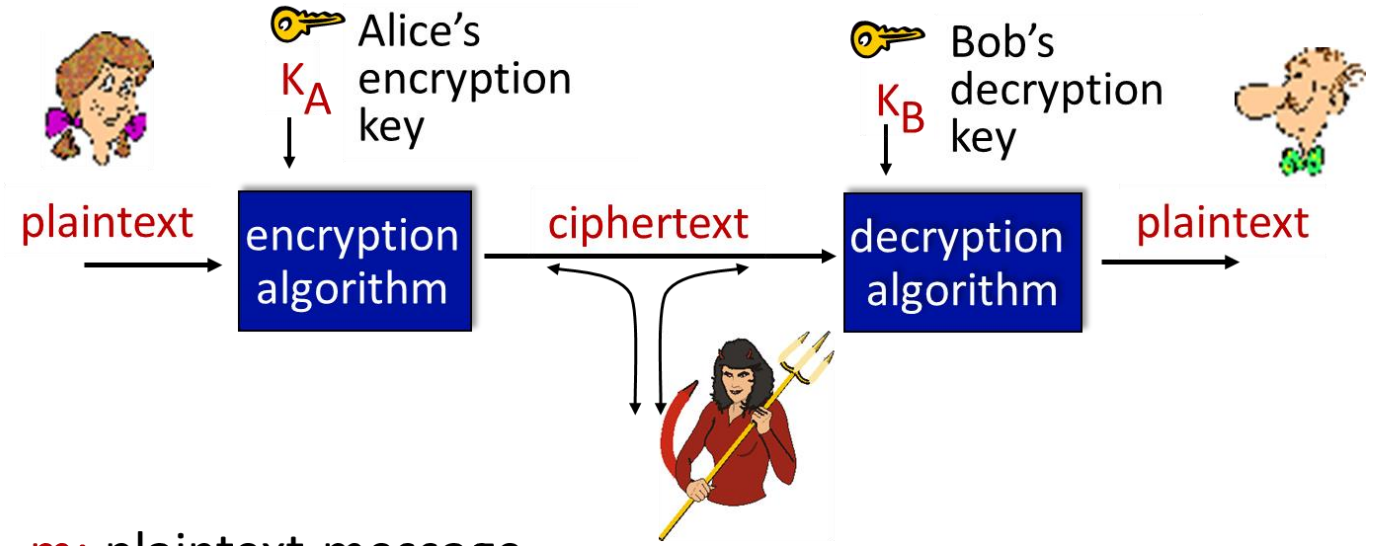
- Turning off a computer provides confidentiality and integrity, but hurts availability. . .

- Denial of service attacks are direct assaults on availability

More Definitions

- Vulnerability (آسیب‌پذیری): An error or weakness in the design, implementation, or operation of a system
- Attack (حمله): A means of exploiting some vulnerability in a system
- Threat (تهدید): An adversary that is motivated and capable of exploiting a vulnerability

The language of cryptography



m: plaintext message

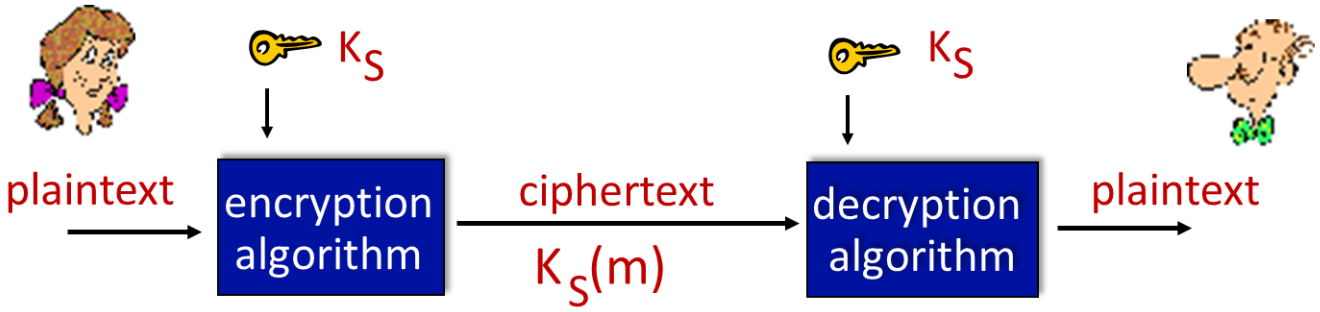
$K_A(m)$: ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Breaking an encryption scheme

- cipher-text only attack:
  Trudy has ciphertext she can analyze
- known-plaintext attack:
  Trudy has plaintext corresponding to ciphertext
- chosen-plaintext attack:
  Trudy can get ciphertext for chosen plaintext
- two approaches:
  - brute force: search through all keys
  - statistical analysis

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K

- *e.g.,* key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST 1993]
- 56-bit symmetric key, 64-bit plaintext input
- DES has 16 rounds
- how secure is DES?
  - DES Challenge: 56-bit-key-encrypted phrase decrypted (brute force) in less than a day
  - no known good analytic attack
- making DES more secure:
  - 3DES: encrypt 3 times with 3 different keys

AES: Advanced Encryption Standard

- NIST standard, replaced DES (Nov 2001)
- processes data in 128-, 192-, or 256-bit blocks
- 128-, 192-, or 256-bit keys
  - 10 rounds for 128-bit keys.
  - 12 rounds for 192-bit keys.
  - 14 rounds for 256-bit keys.
- brute force decryption taking 1 sec on DES, takes 149 trillion years for AES

Confusion and Diffusion

- In cryptography, confusion and diffusion are two properties of the operation of a secure cipher
  - identified by Claude Shannon in his 1945 classified report *A Mathematical Theory of Cryptography*.

Confusion

- Confusion refers to making the relationship between the **key** and the **ciphertext** as complex and involved as possible.
  - The property of confusion hides the relationship between the ciphertext and the key.
- How It's Achieved:
  - Non-linearity: Using non-linear mathematical functions to introduce complexity.
  - Substitution: Substituting bits or blocks of data based on the key.
- Example: S-boxes in the Advanced Encryption Standard (AES) provide confusion.

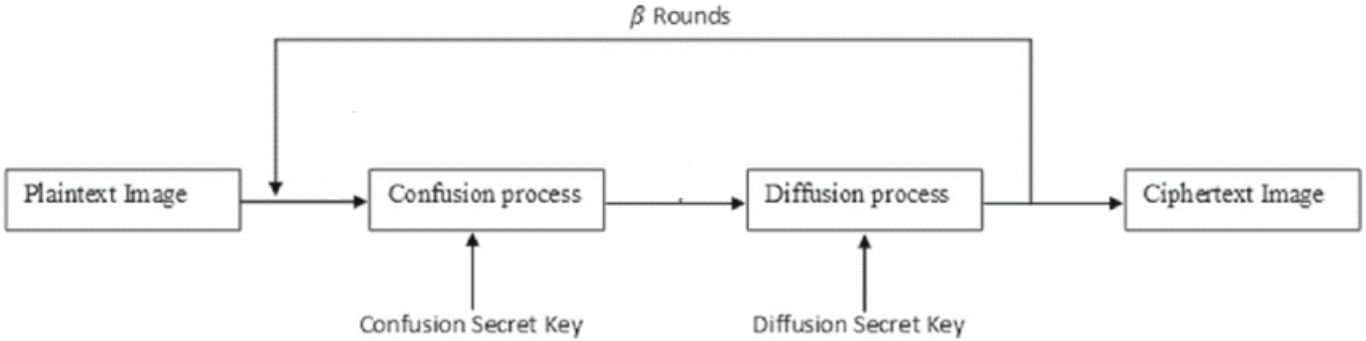
Diffusion

- Diffusion refers to the property that the **redundancy in the statistics** of the plaintext is **dissipated in the statistics of the ciphertext**.
  - In other words, the non-uniformity in the distribution of the individual letters (and pairs of neighboring letters) in the plaintext should be redistributed into the non-uniformity in the distribution of much larger structures of the ciphertext, which is much harder to detect.
  - In a cipher with good diffusion, if one bit of the plaintext is changed, then the ciphertext should change completely, in an unpredictable or pseudorandom manner.

Diffusion

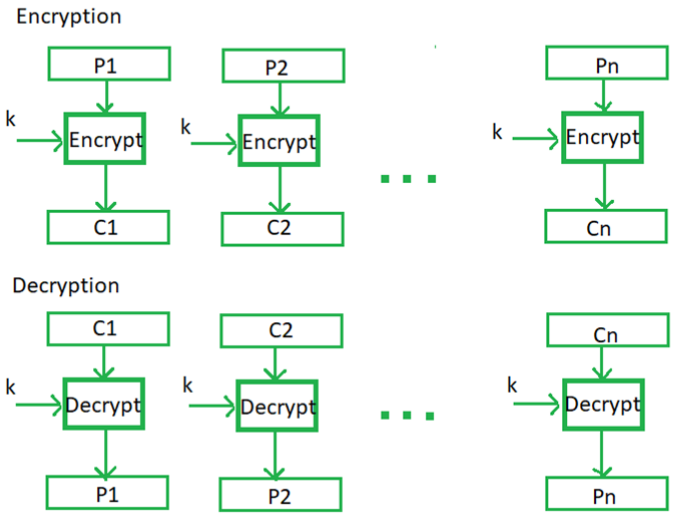
- How It's Achieved:
  - Permutation: Rearranging bits or blocks of data.
  - Mixing: Applying mathematical operations that mix data.
  - Generally speaking, Linear operation.
- Example: Permutation layers in block ciphers contribute to diffusion.

Encryption Scheme

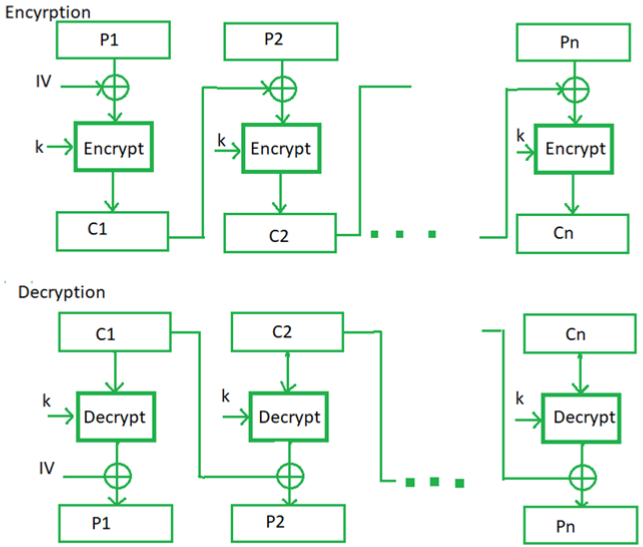


Block Cipher modes of Operation

- Electronic Code Book (ECB)



- Cipher Block Chaining (CBC)



Public Key Cryptography

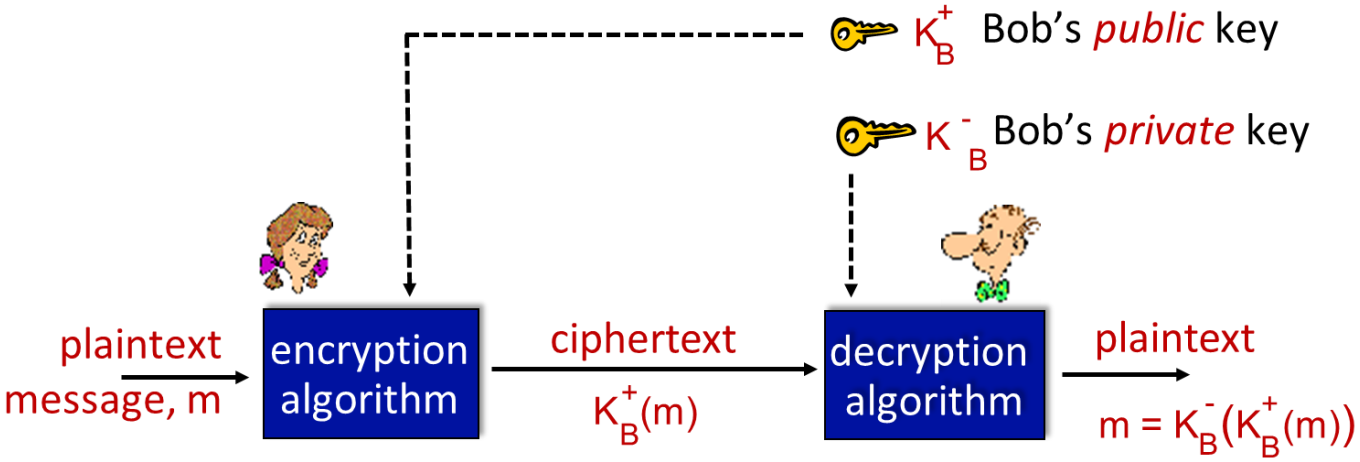
symmetric key crypto:

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?
- Message integrity, authentication?

public key crypto

- *radically* different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

Public Key Cryptography



Public key cryptography revolutionized 2000-year-old (previously only symmetric key) cryptography!

- similar ideas emerged at roughly same time, independently in US and UK (classified)

Public key encryption algorithms

requirements:

1. need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

2. given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

One-Way Functions for Public-Key Cryptography

- One-way Function
  - One-way functions are easy to compute but it is very difficult to compute their inverse functions.
    - Thus, having data x it is easy to calculate f(x) but, on the other hand, knowing the value of f(x) it is quite difficult to calculate the value of x.
  - Modular exponentiation is a common example of a one-way function in RSA cryptography.
    - Computing $c = m^e \ mod \ n$ is easy, but finding m given c, e, and n is computationally difficult.

Euler's phi function

- Euler's phi function $\phi(n)$ counts the positive integers up to a given integer n that are relatively prime to n.
  - In other words, it is the number of integers k in the range $1 \leq k \leq n$ for which the greatest common divisor gcd(n, k) is equal to 1.

$$\varphi(n) = p_1^{k_1-1}(p_1-1)\,p_2^{k_2-1}(p_2-1)\cdots p_r^{k_r-1}(p_r-1),$$

where $n = p_1^{k_1} p_2^{k_2} \cdots p_r^{k_r}$ is the prime factorization of n (that is, $p_1, p_2, \ldots, p_r$ are distinct prime numbers).

Euler's theorem

- Fermat's little theorem
  - For any prime number p, and for any number n, 0 < n < p, $n^{p-1} \equiv 1 \bmod p$.

- Euler's theorem generalizes Fermat's theorem to the case where the modulus is not prime.
  - if p is a positive integer and n, p are coprime (n <p), then $n^{\phi(p)} \equiv 1 \bmod p$ where ϕ(p) is the Euler's phi function .
  - $5^{\phi(12)} mod\ 12 = 5^4\ mod\ 12 = 625\ mod\ 12 = 1$

Prerequisite: modular arithmetic

- a mod n = remainder of a when divide by n
- facts:

  $[(a \bmod n) + (b \bmod n)] \bmod n = (a+b) \bmod n$

  $[(a \bmod n) - (b \bmod n)] \bmod n = (a-b) \bmod n$

  $[(a \bmod n) * (b \bmod n)] \bmod n = (a*b) \bmod n$
- thus

  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: a=14, n=10, d=2:

  $(a \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$

  $a^d = 14^2 = 196 \quad a^d \bmod 10 = 6$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number

example:

- m= 10010001. This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key

1. choose two large prime numbers *p, q.* (e.g., 1024 bits each)

2. compute *n = pq*, $\phi(n) = z = (p-1)(q-1)$

3. choose *e (*with *e<z)* that has no common factors with z (*e, z* are "relatively prime").

4. choose *d* such that *ed-1* is exactly divisible by *z*. (in other words: *ed* mod *z = 1*).

5. *public* key is *(n,e)*. *private* key is *(n,d)*.

K_B^+ K_B^-

RSA: encryption, decryption

0. given (*n,e*) and (*n,d*) as computed above

1. to encrypt message *m* (*<n*), compute

$$c = m^e \bmod n$$

2. to decrypt received bit pattern, *c*, compute

$$m = c^d \bmod n$$

$$m = (\underbrace{m^e \bmod n}_{c})^d \bmod n$$

Why does RSA work?

- must show that $c^d \bmod n = m$, where $c = m^e \bmod n$
- thus,

$c^d \bmod n = (m^e \bmod n)^d \bmod n$

$= m^{ed} \bmod n$

$= m^{az+1} \bmod n$

$= m^{az} \,.\, m^1 \bmod n$

$= (m^{az} \bmod n.\ m \bmod n) \bmod n$

$= ((m^z)^a \bmod n.\ m) \bmod n$

$= ((m^z \bmod n)^a \bmod n.\ m) \bmod n$ #Euler's theorem

$= (1^a \bmod n.\ m) \bmod n = 1.m \bmod n = m$

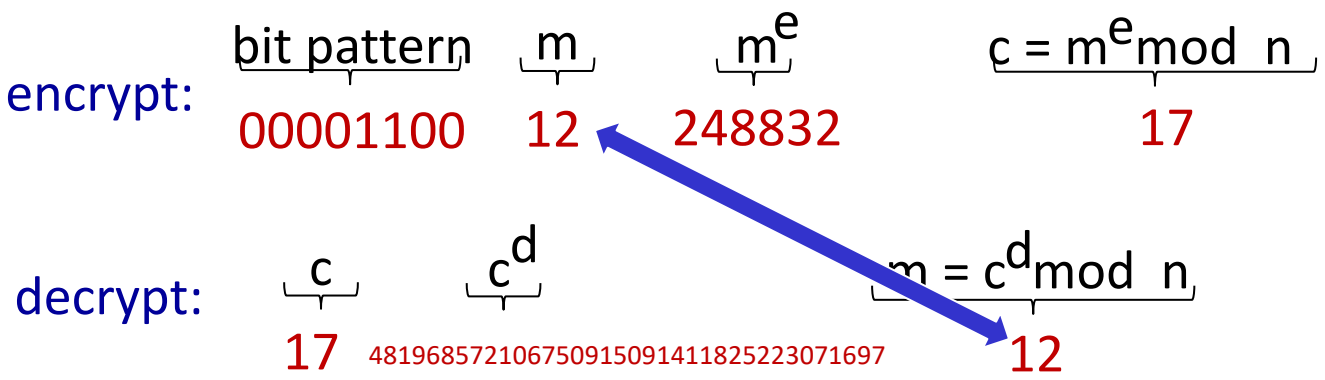
RSA example:

Bob chooses *p=5, q=7*. Then *n=35, z=24*.

e=5 (so *e, z* relatively prime).

d=29 (so *ed-1* exactly divisible by z).

encrypting 8-bit messages.



RSA: another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

result is the same!

RSA: another important property

Why $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$?

follows directly from modular arithmetic:

$$(m^e \bmod n)^d \bmod n = m^{ed} \bmod n$$
$$= m^{de} \bmod n$$
$$= (m^d \bmod n)^e \bmod n$$

Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard

Prime-counting function

- The prime-counting function $\pi(x)$ is the function counting the number of prime numbers less than or equal to some real number x.
  - It was conjectured in the end of the 18th century by Gauss and by Legendre to be approximately

Input

$$\frac{2^{1024}}{\log(2^{1024})}$$

Decimal approximation

$$2.532737276080075837450112514401885703366568528844967434378 8\ldots \times 10^{305}$$

Input

$$2^{128}$$

Result

340 282 366 920 938 463 463 374 607 431 768 211 456

Scientific notation

$$3.402823669209384634633746074317682114 56 \times 10^{38}$$

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key crypto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_S

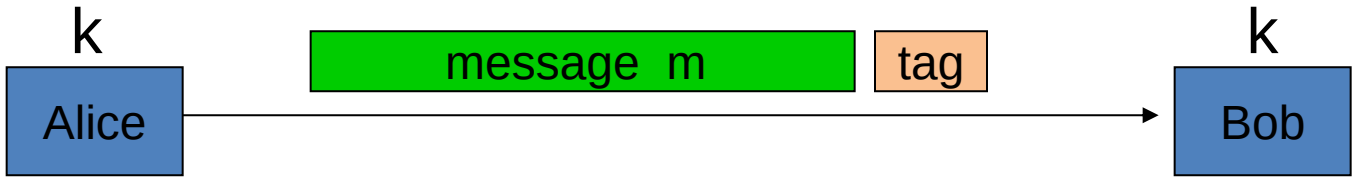
- Bob and Alice use RSA to exchange a symmetric session key K_S
- once both have K_S, they use symmetric key cryptography

Homomorphic Property

- In mathematics and cryptography, a homomorphism is a function that preserves certain algebraic operations.
  - RSA exhibits a homomorphic property with respect to multiplication.
  - If you encrypt two plaintexts and then multiply their ciphertexts, the result is equivalent to encrypting the product of the plaintexts.

  - $m_3 = m_1 . m_2$
  - $c_3 = m_3^e \ mod \ n = (m_1 . m_2)^e \ mod \ n = m_1^e . m_2^e \ mod \ n$
  - $= (m_1^e \ mod \ n . m_2^e \ mod \ n) \ mod \ n = (c_1 . c_2) \ mod \ n$
  - $c_3 = c_1 . c_2 \ mod \ n$

Message Authentication Codes (MACs)



Generate tag:
$tag \leftarrow S(k, m)$

Verify tag:
$V(k, m, tag) \stackrel{?}{=}$ `yes'

Def: **MAC** I = (S,V) defined over (K,M,T) is a pair of algs:

- S(k,m) outputs t in T
- V(k,m,t) outputs `yes' or `no'

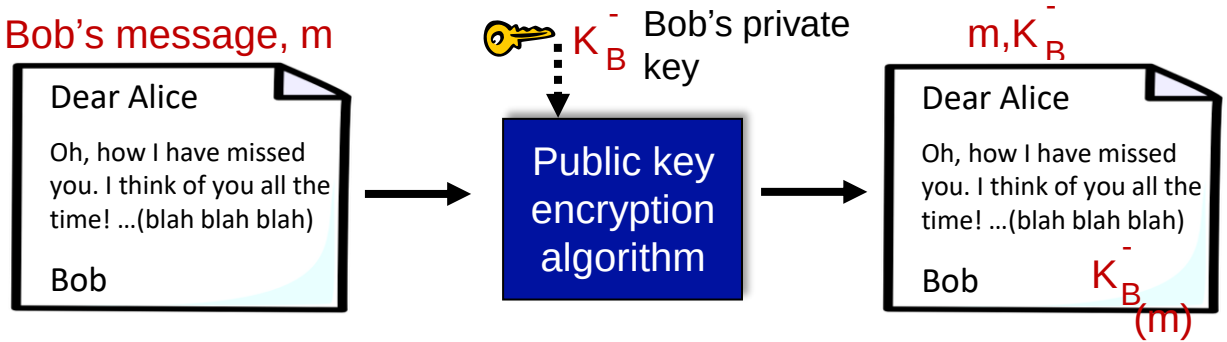
Digital Signatures: The Problem

- Consider the real-life example where a person pays by credit card and signs a bill; the seller verifies that the signature on the bill is the same with the signature on the card
- Contracts, they are valid if they are signed.
- Can we have a similar service in the electronic world?

Digital signatures

cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document: he is document owner/creator.
- *verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document
- simple digital signature for message m:
  - Bob signs m by encrypting with his private key K_B, creating "signed" message, $K_B^-(m)$



Digital signatures

- suppose Alice receives msg m, with signature: m, $K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key

Alice thus verifies that:

- Bob signed m
- no one else signed m
- Bob signed m and not m'

non-repudiation:

- ✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

Security properties of digital signature

- Message authentication
  - When the verifier validates the digital signature using public key of a sender, he is assured that signature has been created only by sender who possess the corresponding secret private key and no one else.
- Data Integrity
  - In case an attacker has access to the data and modifies it, the digital signature verification at receiver end fails. The modified data and the output provided by the verification algorithm will not match. Hence, receiver can safely deny the message assuming that data integrity has been breached.
- Non-repudiation
  - Since it is assumed that only the signer has the knowledge of the signature key, he can only create unique signature on a given data. Thus, the receiver can present data and the digital signature to a third party as evidence if any dispute arises in the future.

Adversarial Goals

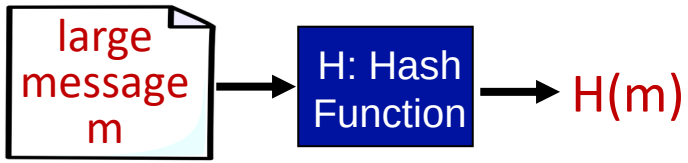
- Total break
  - adversary is able to find the secret (victim's private key) for signing, so he can forge then any signature on any message.
- Selective forgery
  - adversary is able to create valid signatures on a message chosen by someone else, with a significant probability.
- Existential forgery
  - adversary can create a pair (message, signature), s.t. the signature of the message is valid.

Attack Models for Digital Signatures

- Key-only attack
  - Adversary knows only the verification function, including victim's public key (which is supposed to be public).
- Known message attack
  - Adversary knows a list of messages previously signed by victim.
- Chosen message attack
  - Adversary can choose what messages wants victim to sign, and he knows both the messages and the corresponding signatures.

Message digests

- computationally expensive to public-key-encrypt long messages
- Goal: fixed-length, easy- to-compute digital "fingerprint" and extra layer of security
  - apply hash function H to m, get fixed size message digest, H(m)



Hash function properties:

- produces fixed-size msg digest (fingerprint)
- given message digest *x*, computationally infeasible to find *m* such that *x = H(m)*

Internet checksum: poor crypto hash

Internet checksum has some properties of hash function:

- produces fixed length digest (16-bit sum) of message
- is many-to-one

but given message with given hash value, it is easy to find another message with same hash value:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

message	ASCII format
I O U 9	49 4F 55 39
0 0 . 1	30 30 2E 31
9 B O B	39 42 D2 42
	B2 C1 D2 AC

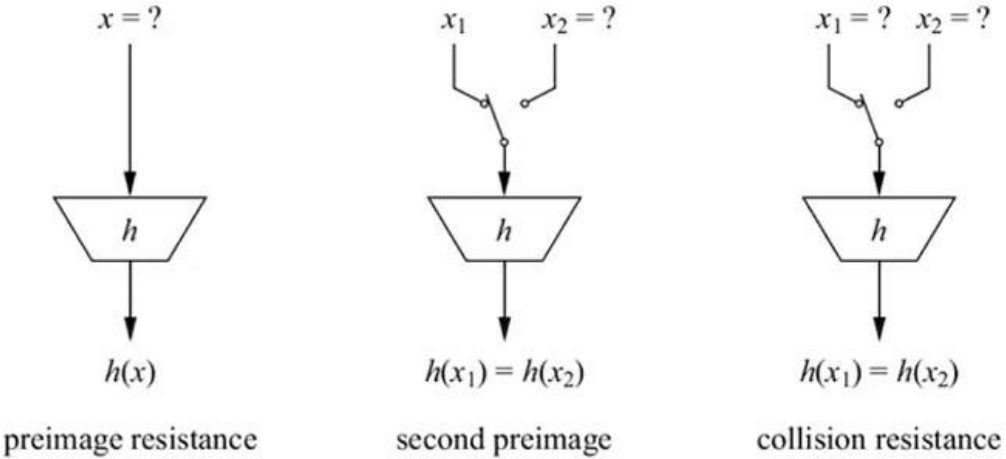
different messages but identical checksums!

Secure Hash Function

- A secure hash function is a mathematical algorithm that takes an input (or 'message') and returns a fixed-size string of characters
  - It is designed to be a one-way function, meaning it's computationally infeasible to reverse the process (i.e., find the original input from the hash value)

- **Characteristics of Secure Hash Functions**
  1. **Pre-image Resistance**: It should be extremely difficult to find an input that corresponds to a given hash value.
  2. **Collision Resistance**: It should be computationally infeasible to find two different inputs that produce the same hash value.
  3. **Avalanche Effect**: A small change in the input should result in a significantly different hash value.
  4. **Second Pre-image Resistance**: Given a hash value, it should be computationally infeasible to find a different input that produces the same hash value.

Secure Hash Function



Avalanche Effect

Ivan	EBEC404B6897056F95C343D7E7D2A12E081ADC36413163A03CE3248E9D312C97
ivan	CD0B9452FC376FC4C35A60087B366F70D883FC901524DAF1F122FBD319384F6A

https://academy.moralis.io/blog/what-is-hashing-a-complete-guide-to-hashing

Hash function algorithms

- MD5 hash function widely used (RFC 1321)
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x

- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Commitment Scheme Using Hash Functions

- A commitment scheme allows one party (the "committer") to commit to a value, keeping it hidden from others, and later reveal the committed value.

- **Components of a Commitment Scheme**
  1. **Commitment Phase**:
     1. The committer selects a value "x" and a random value "r."
     2. Compute a commitment C by hashing both "x" and "r": C = Hash(x || r).
  2. **Reveal Phase**:
     1. At a later time, the committer reveals both "x" and "r."
     2. Others can verify the commitment by hashing "x" and "r" and comparing it to the original commitment.

Commitment Scheme

Commitments {*commit*, *open*, *verify*}

A prover $\mathcal{P}$ hides a secret in the commit phase and opens it to a verifier $\mathcal{V}$ in the open phase.

Commit phase

$\mathcal{P}$ computes and sends *com* to $\mathcal{V}$:
r = *random*()
com = *commit*(*secret*, *r*)

Hiding

$\mathcal{V}$ cannot find clues of (*secret*, *r*) from *com* at the commit phase.

After some confirmations...

Open phase

$\mathcal{P}$ *open* *secret* and r to $\mathcal{V}$.
$\mathcal{V}$ *verify*(*com*, *secret*, *r*) = *T* / *F*

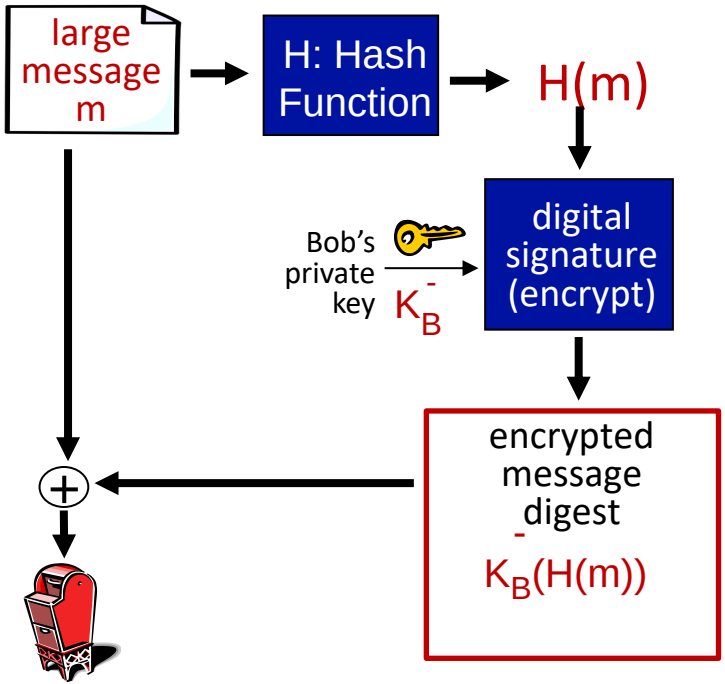
Binding

$\mathcal{P}$ cannnot open (*secret'*, *r'*) =/= (*secret*, *r*) such that
T = *verify*(*com*, *secret'*, *r'*)

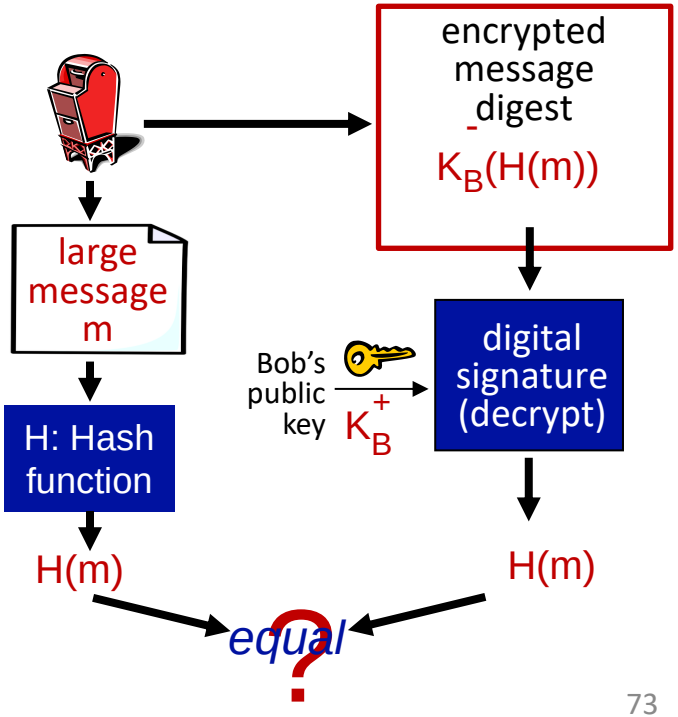
https://kyle-crypto.medium.com/cryptographic-commitments-55ccc4f297c9

Digital signature = signed message

Bob sends digitally signed message:

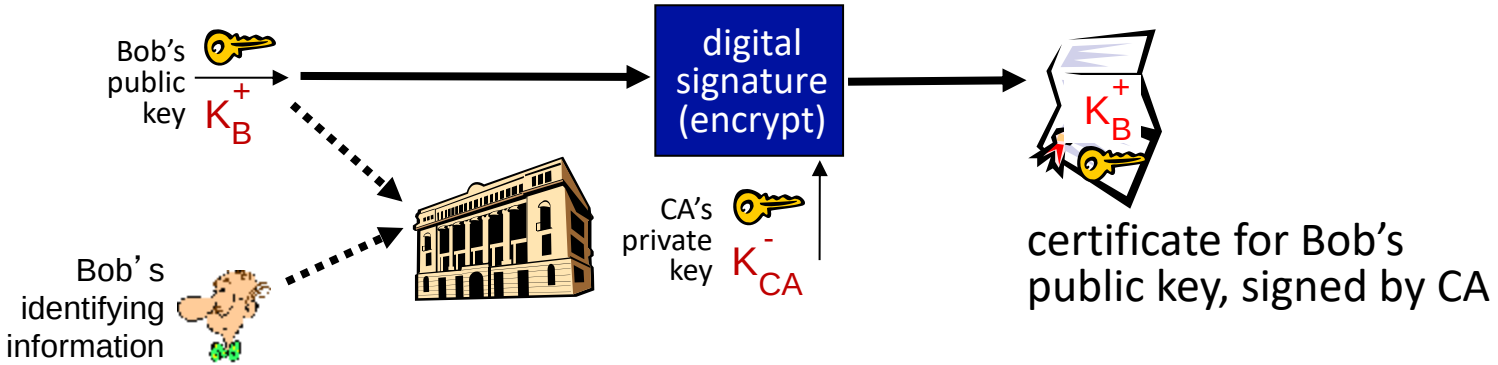


Alice verifies signature, integrity of digitally signed message:



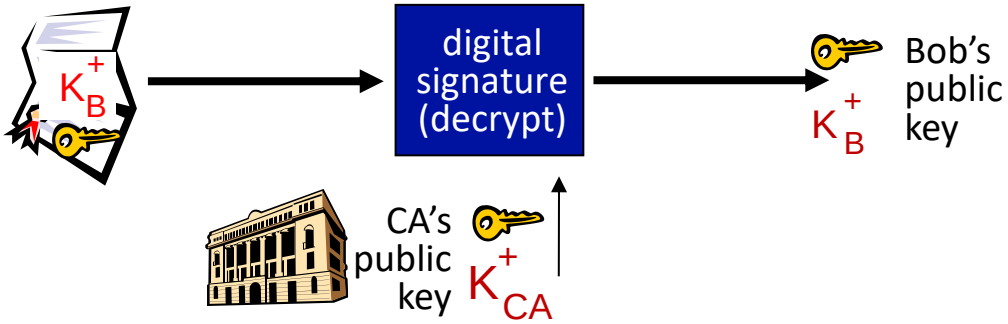
Public key Certification Authorities (CA)

- certification authority (CA): binds public key to particular entity, E
- entity (person, website, router) registers its public key with CE provides "proof of identity" to CA
  - CA creates certificate binding identity E to E's public key
  - certificate containing E's public key digitally signed by CA: CA says "this is E's public key"



Public key Certification Authorities (CA)

- when Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere)
  - apply CA's public key to Bob's certificate, get Bob's public key



Resources

- ECE/COS 470, Pramod Viswanath, Princeton 2024
- CS251, Dan Boneh, Stanford 2023
- Computer Networking: A Top-Down Approach, Jim Kurose, Keith W. Ross, Chapter 8