Cryptographic primitives

University of Athens
Dionysis Zindros, Christos Nasikas

Hash functions

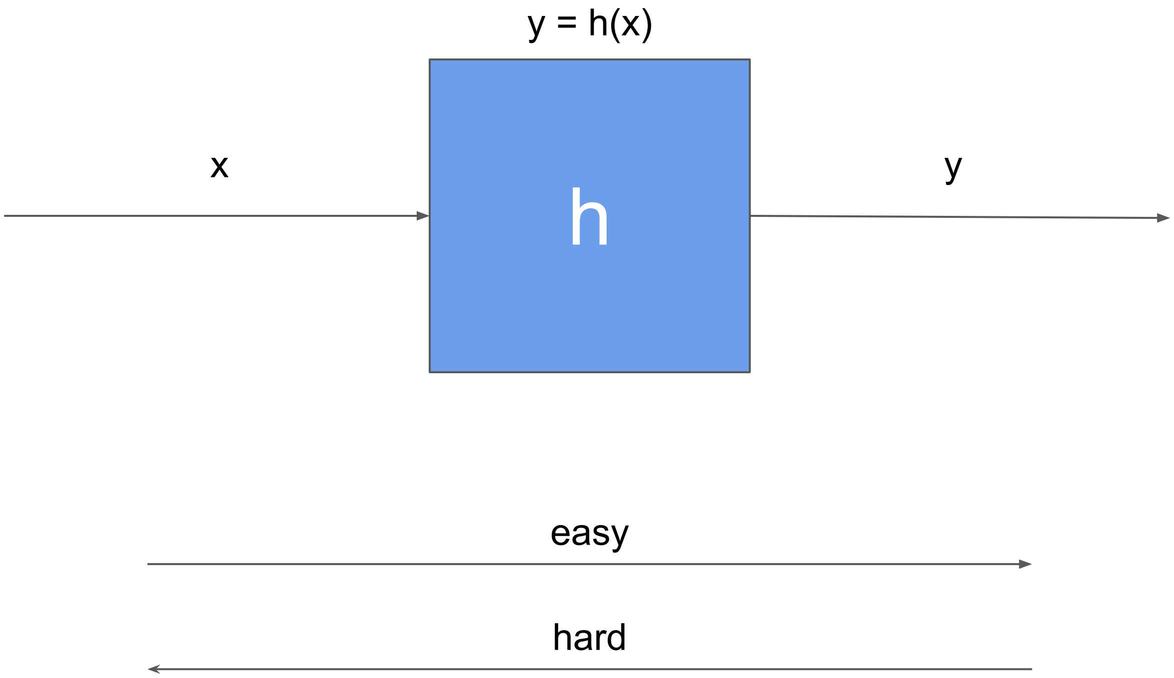
Hash functions

- Map data of arbitrary size to a string of a fixed length
- $h(x) = y$, where $h: \{0, 1\}^* \rightarrow \{0, 1\}^n$
- Various applications: hash tables, proof-of-work, digital fingerprint, file integrity, cache mechanisms, identifiers, time-stamping

Cryptographic Hash Functions

- h(x) = y
- Easy to calculate y, given x
- Practically infeasible to find x from y
- Practically infeasible to alter x without altering y
- Practically infeasible to find different x, x’ that produce the same y

Cryptographic Hash Functions



sha2

- Descendant of sha1
- It is considered safe for cryptographic applications
- SHA224: $\{0,1\}^* \rightarrow \{0,1\}^{224}$
- SHA256: $\{0,1\}^* \rightarrow \{0,1\}^{256}$
- SHA384: $\{0,1\}^* \rightarrow \{0,1\}^{384}$
- SHA512: $\{0,1\}^* \rightarrow \{0,1\}^{512}$

Example of sha256

sha256(‘Hello World!’)

=

7F83B1657FF1FC53B92DC18148A1D65DFC2D4B1FA3D6

77284ADDD200126D9069

‘Reversing’ a hash

- Hashes are designed to be one-way, irreversible
- So, the reversing can be done only by **brute-force**
  - Try all possible strings of length 0, 1, 2, 3 etc…
  - Try words from a dictionary

Hash dictionary attack

```
foreach (word in dictionary){
    if(SHA256(word) == c) {
        return word;
    }
}
```

Hash dictionary attack

```
Σ = {'a', 'b', ..., 'z'};
numdigits = 0;
m = "";
while (SHA256(m) != c) {

   try {
        m = increment(m, Σ, numdigits);

   }
    catch (OutOfBoundsException e) {

       ++numdigits;

       m = repeat(Σ[0], numdigits);
    }

} return m;
```

Commitment schemes

Commitment schemes

- Alice wants to commit to a value “b”
- Alice and Bob do not trust each other
- **Binding**: Bob wants to know that Alice will not change her mind about b
- **Hiding**: Alice does not want to reveal her value beforehand

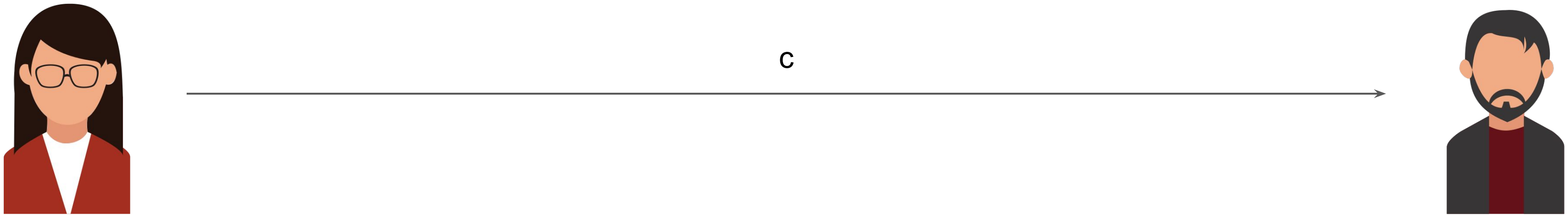
Commitment schemes: phases

- Commitment phase:
  - Alice chooses and commits to a value
  - Sends a secret to Bob
  - **Hiding:** Bob cannot use the secret to find the value of Alice
- Reveal phase:
  - Bob learns the initial value Alice chose
  - **Binding:** Bob confirms that Alice didn’t change her value using the secret

Commitment schemes: using a hash

- Commitment phase:
    - Alice computes **c = H(b)**
    - Send **c** to Bob
    - Bob cannot reverse **c** to get **b**

Commitment schemes: commitment phase



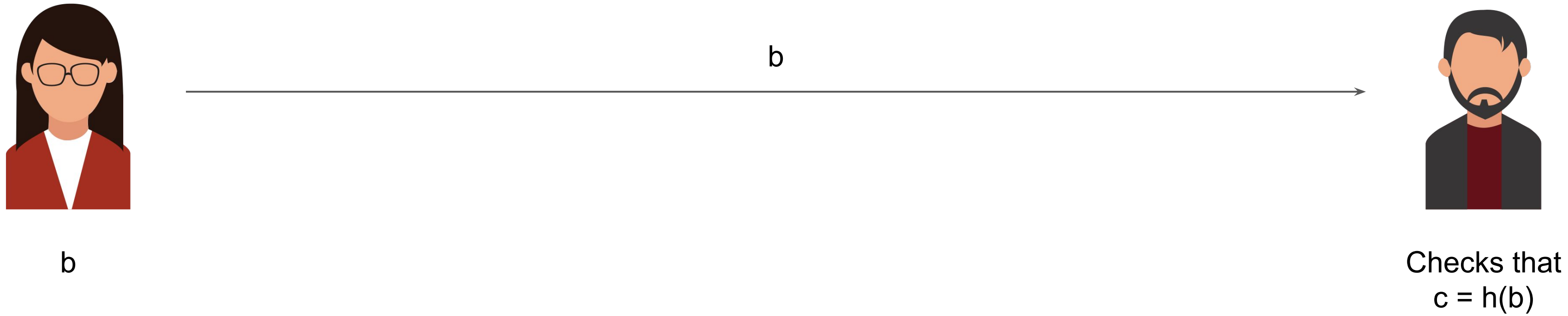
Computes $c = h(b)$

c

Commitment schemes: using a hash

- Reveal phase:
    - Alice sent **b** to Bob
    - Bob checks that H(b) = c

Reveal phase: reveal phase



Proof of work

- Cryptographic primitive that describes the proof of executed CPU cycles
- Alice wants to prove that Bob will dedicate his CPU power only for her
- How can she do that ?

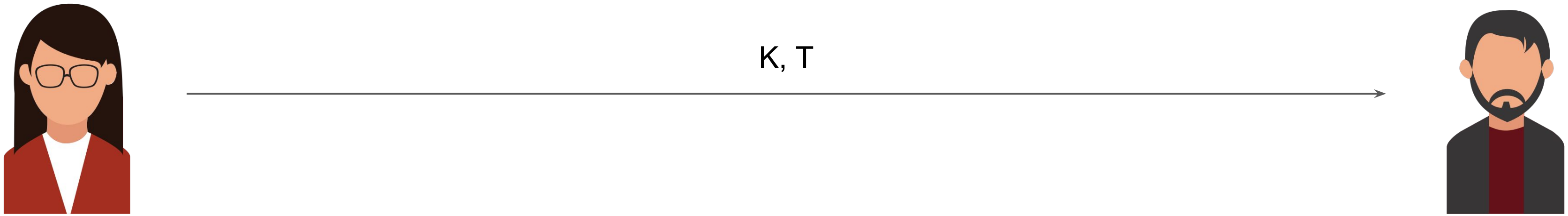
Proof of work

- Alice chooses a random *salt* K, big enough (i.e. 128 bits)
- Alice sends to Bob the salt K, and a target parameter T.
- The target parameter defines the amount of work to be done

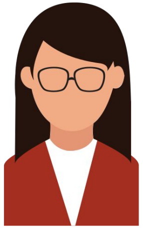
Proof of work

- Bob calculates x such that
  - $H(K \| x) < T$
- x is the *nonce*
- Bob sends x to Alice
- Alice validates that $H(K \| x) < T$

Proof of work: Request

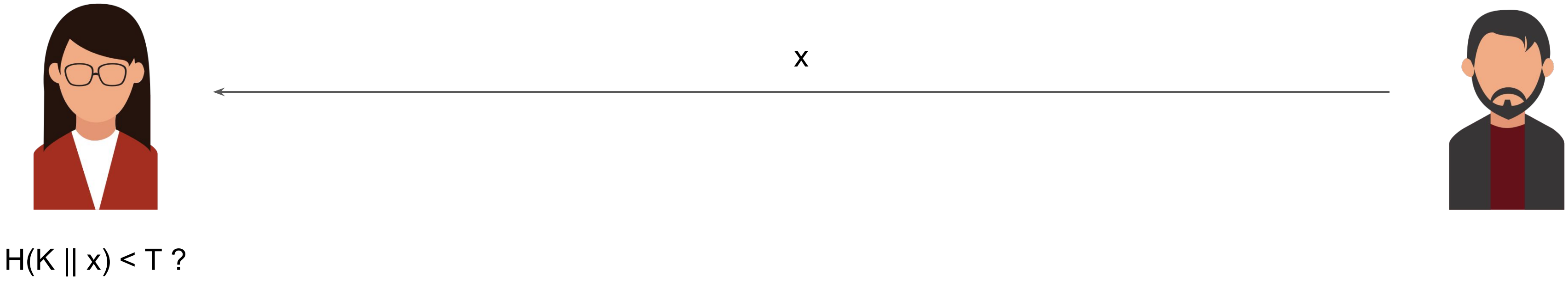


Proof of work: Work



$H(K \| x) < T$?

Proof of work: Proof



Proof of work: Algorithm

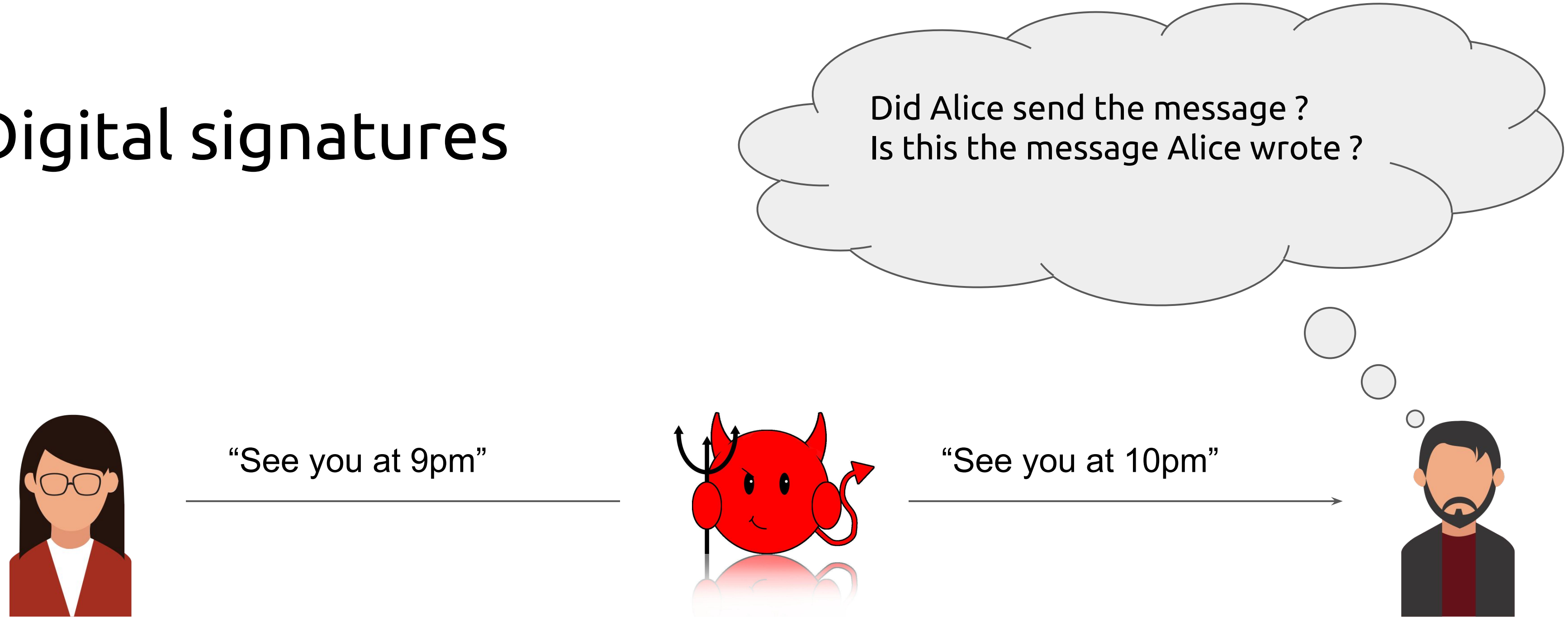
```
do {
    x = rand();
} while (H(K || x) >= T);
return x;
```

Proof of work

- Very important to blockchain consensus mechanisms
- More details in the following lectures...

Digital signatures

Digital signatures



Digital Signatures

- **Integrity**: The signed message was not *altered* in transit
- **Authentication**: The signature was created by a known sender
- **Non-repudiation**: The sender cannot deny having signed the message

Digital Signatures

- The sender produce a key pair (vk, sk)
- sk: signing key
- vk: verification key

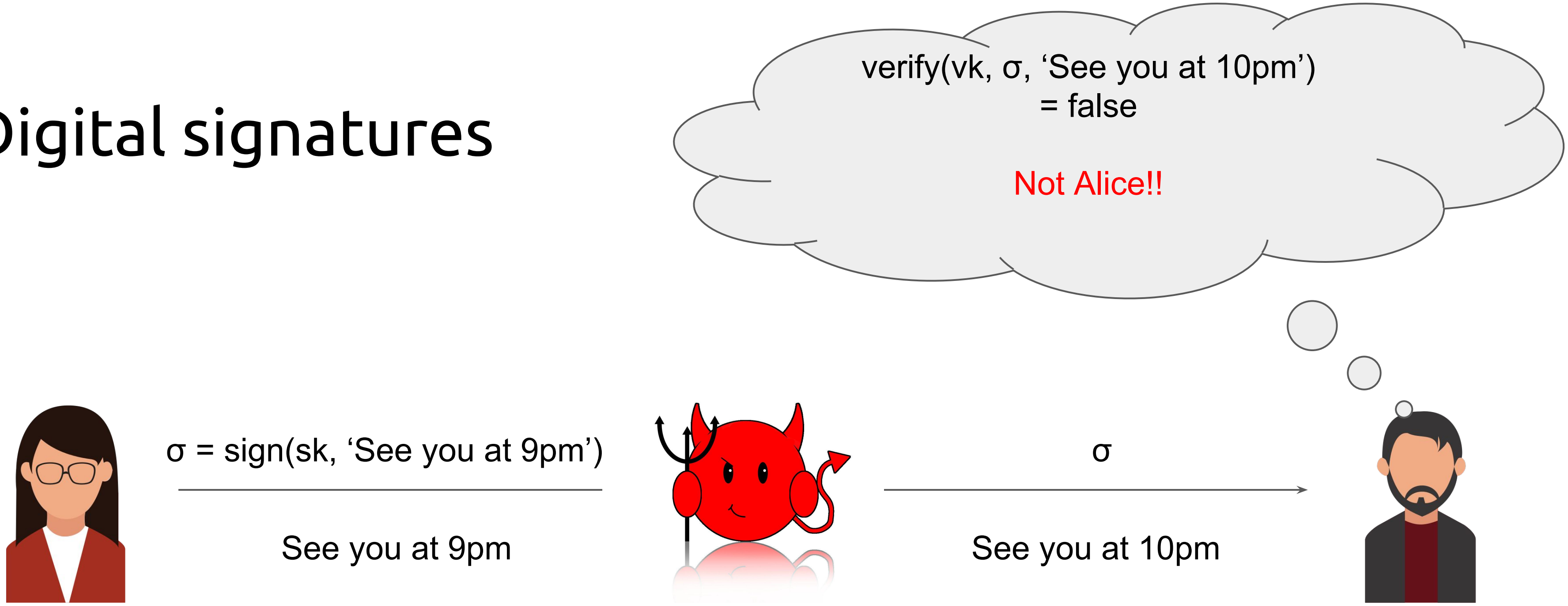
Digital Signatures

- Verification key is **public**
  - publicly verifiable
  - transferable signatures
- Signing key is private

Digital Signatures

- $\sigma = \text{sign}(sk, m)$
- $\text{verify}(vk, \sigma, m) = \{0, 1\}$

Digital signatures



Digital Signatures

- Digital signatures are very important to blockchain systems
- A valid signature implies ownership
- More details in the following lectures...