Blockchain and Cryptocurrencies

Chapter 1.3: Basic Tools — Cryptography and Digital Signatures

Prof. Dr. Peter Thiemann

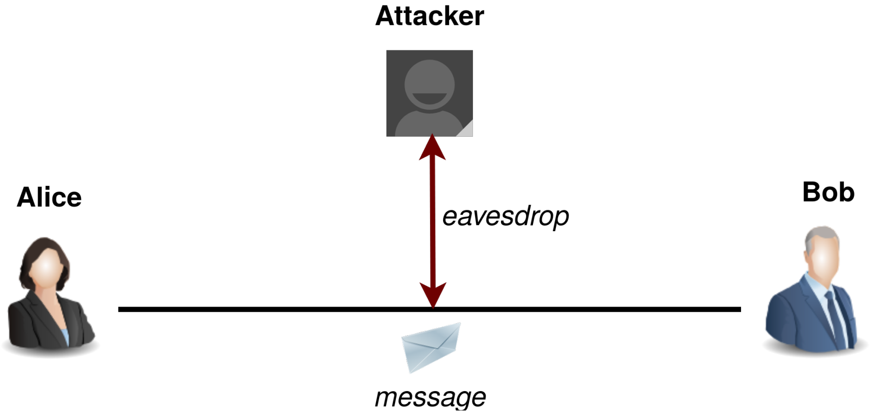
Albert-Ludwigs-Universität Freiburg, Germany

SS 2020

Data Encryption

Goal

Preserve confidentiality of data exchanged between communication partners.



Contents

Symmetric Encryption

Symmetric Encryption

- Two algorithms

$$\begin{aligned} \text{encode} &: \text{key} \times \text{plaintext} \to \text{ciphertext} \\ \text{decode} &: \text{key} \times \text{ciphertext} \to \text{plaintext} \end{aligned}$$

- $\mathbf{decode}(k, \mathbf{encode}(k, m)) = m$, for all plaintext messages m

$\Rightarrow$ All communication partners need to have the same key.

$\Rightarrow$ Weakness: if the key is exposed, confidentiality is compromised.

Asymmetric Encryption

Public Key Cryptography

- Keys come in pairs: public key (pk) and secret key (sk)
- The public key is freely available, e.g. in a public directory with the name of its owner
- The secret key is private to the owner and is never communicated

Asymmetric Encryption

Public Key Cryptography

- Keys come in pairs: public key (pk) and secret key (sk)
- The public key is freely available, e.g. in a public directory with the name of its owner
- The secret key is private to the owner and is never communicated

Asymmetric Encryption

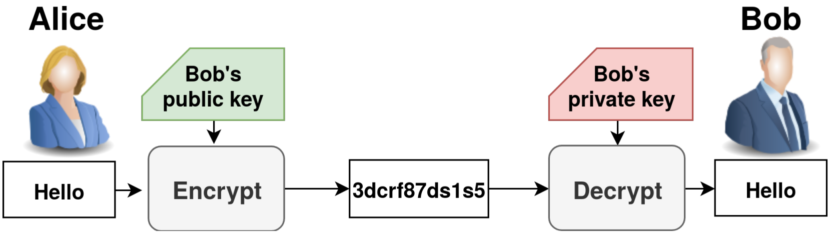
- Let key = (pk, sk)
- Two algorithms

$$\begin{aligned} \text{encode} &: \text{pk} \times \text{plaintext} \to \text{ciphertext} \\ \text{decode} &: \text{sk} \times \text{ciphertext} \to \text{plaintext} \end{aligned}$$

- $\mathbf{decode}(sk, \mathbf{encode}(pk, m)) = m$, for all plaintext messages m

Public Key Cryptography

Anyone can encrypt messages using the public key, but only the holder of the paired secret key can decrypt.



⇒ To send a message to Bob, Alice needs to obtain Bob's pk

- Bob decodes with his private sk

⇒ Weakness: if Bob's sk is exposed, then confidentiality of all messages (past and future) encoded for Bob is compromised

Public Key Cryptosystem

Definition

- A **key pair** consists of a public key and a secret key.
- **keygenerator** is a probabilistic polynomial time algorithm. It computes a key pair (pk, sk) from an input of size k.
- **encode** is a probabilistic polynomial time algorithm. It computes a ciphertext $c = \mathbf{encode}(sk, m)$ from a message m.
- **decode** is a polynomial time algorithm. Given a ciphertext $c = \mathbf{encode}(pk, m)$, it computes $m' = \mathbf{decode}(sk, c)$ such that $m = m'$.

Example: RSA

keygenerator

- for input of size k, choose two prime numbers p, q such that $n = pq$ has k bits
- choose $1 < e < (p-1)(q-1)$ such that $\mathbf{gcd}(e, (p-1)(q-1)) = 1$
- calculate $1 < d < (p-1)(q-1)$ such that $de \equiv 1 \mod (p-1)(q-1)$.
- the public key is the pair (n, e)
- the secret key is d

Example: RSA

keygenerator

- for input of size k, choose two prime numbers p, q such that $n = pq$ has k bits
- choose $1 < e < (p-1)(q-1)$ such that $\mathbf{gcd}(e, (p-1)(q-1)) = 1$
- calculate $1 < d < (p-1)(q-1)$ such that $de \equiv 1 \mod (p-1)(q-1)$.
- the public key is the pair (n, e)
- the secret key is d

Remark

- Choices must be random
- Certain choices of p, q, and e are easy to break

Example: RSA, II

Encode

- Given the public key (n, e) and message $0 \leq m < n$
- Compute $c = \textbf{encode}((n, e), m) = m^e \mod n$

Example: RSA, II

Encode

- Given the public key (n, e) and message $0 \leq m < n$
- Compute $c = \textbf{encode}((n, e), m) = m^e \mod n$

Decode

- Given the public key (n, e), secret key d, and ciphertext c.
- let $\textbf{decode}((n, d), c) = c^d \mod n$.

This is a decoding because $ed = 1 \mod (p-1)(q-1)$ means that there exists some $l \in N$ such that $ed = 1 + l(p-1)(q-1)$. Hence

$$(m^e)^d = m^{ed} = m^{1+l(p-1)(q-1)} = m \cdot m^{l(p-1)(q-1)} = m \mod n$$

by Fermat's theorem.

RSA, III

Remark

- RSA is special in that encoding and decoding are the same function.
- Other schemes have different encoding and decoding functions.
- Bitcoin uses a different scheme based on elliptic curves

Contents

Properties of a Signature

Signature (https://en.wikipedia.org/wiki/Signature)

A signature (from Latin: signare — ["to mark", "to stamp", "to designate",] "to sign"[, "to seal"]) is a handwritten (and often stylized) depiction of someone's name, nickname, or even a simple "X" or other mark that a person writes on documents as a proof of identity and intent.

Properties of a Signature

Signature (https://en.wikipedia.org/wiki/Signature)

A signature (from Latin: signare — ["to mark", "to stamp", "to designate",] "to sign"[, "to seal"]) is a handwritten (and often stylized) depiction of someone's name, nickname, or even a simple "X" or other mark that a person writes on documents as a proof of identity and intent.

Functions of a Signature

"[to] permanently affix to a document a person's uniquely personal, undeniable self-identification as physical evidence of that person's personal witness and certification of the content of all, or a specified part, of the document." [Wikipedia]

- proof that signer has seen the content of the document
- integrity of the document (signature physically attached)
- signature is difficult to forge, but easy to verify for anyone

Properties of a Signature

Signature (https://en.wikipedia.org/wiki/Signature)

A signature (from Latin: signare — ["to mark", "to stamp", "to designate",] "to sign"[, "to seal"]) is a handwritten (and often stylized) depiction of someone's name, nickname, or even a simple "X" or other mark that a person writes on documents as a proof of identity and intent.

Functions of a Signature

"[to] permanently affix to a document a person's uniquely personal, undeniable self-identification as physical evidence of that person's personal witness and certification of the content of all, or a specified part, of the document." [Wikipedia]

- proof that signer has seen the content of the document
- integrity of the document (signature physically attached)
- signature is difficult to forge, but easy to verify for anyone

Digital Signature

should enjoy analogous properties

Digital Signature

Definition

A **digital signature scheme** consists of the following three algorithms:

- $(sk, pk) :=$ generateKeys($keysize$) probabilistic
  - sk: secret key
  - pk: public verification key
- sig $:=$ sign(sk, $message$) probabilistic
- isValid $:=$ verify(pk, $message$, sig)

Two properties

- verify(pk, $message$, sign(sk, $message$)) $==$ true
- signatures are existentially unforgeable

The Unforgeability Game

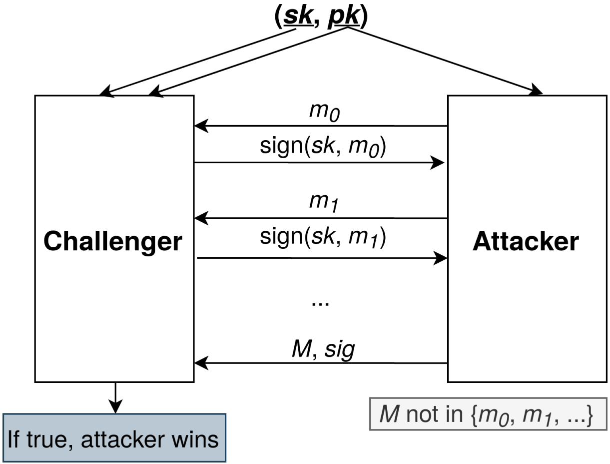


Figure: The attacker and challenger play the unforgeability game.

Digital Signatures

Practical Concerns

- (many) signature algorithms are probabilistic: a good source of randomness is essential
- a limit on the message size
  - sign the hash of the message, rather than the massage itself
  - sign the root of a hash list or hash tree: the signature covers, or protects, the whole structure

Digital Signatures

Practical Concerns

- (many) signature algorithms are probabilistic: a good source of randomness is essential
- a limit on the message size
  - sign the hash of the message, rather than the massage itself
  - sign the root of a hash list or hash tree: the signature covers, or protects, the whole structure

Elliptic Curve Digital Signature Algorithm (ECDSA)

- Bitcoin: signature scheme based on the standard elliptic curve secp256k1, estimated to provide 128 bits of security
- some quantitative information:
  - Secret key: 256 bits
  - Public key, uncompressed: 512 bits (compressed: 257 bits)
  - Message to be signed: 256 bits
  - Signature: 512 bits

Contents

Public Key as Identity

Public Keys as Identities

- if *sig* verifies under a public key *pk*, think of it as *pk* stating the message
- an identity consists of a key pair (sk, pk)
  - the public key *pk* is the public identity (in practice, the hash of the public key)
  - the secret key *sk* lets you speak on behalf of the identity *pk*

Public Key as Identity

Public Keys as Identities

- if *sig* verifies under a public key *pk*, think of it as *pk* stating the message
- an identity consists of a key pair (sk, pk)
  - the public key *pk* is the public identity (in practice, the hash of the public key)
  - the secret key *sk* lets you speak on behalf of the identity *pk*

Decentralized Identity Management

- new identities can be created any time and as often as needed
- **assumption**: good source of randomness required to ensure that no two keys are the same and others' keys are unguessable ($\rightarrow$ cryptographic pseudorandom function)
- no need for central user registry
- nobody knows who you are (anonymity and privacy)
- **but** transactions may reveal behavior and connections ($\rightarrow$ pseudonymity)

Thanks!