Blockchain and cryptocurrencies

Contents

1. Preliminary concepts at the basis of Blockchain [2],[3]

   1.1. Introduction to the cryptography concepts used in Blockchain [3]

   - Cryptography services (confidentiality, authentication, integrity, non-repudiation)
   - Public and private key cryptography
   - Elliptic curve cryptography
   - Hash functions
   - Elliptic curve digital signature algorithm (ECDSA)

   1.2. Distributed systems and decentralization

1 Preliminary cryptographic concepts

1.1 Hash functions

A hash function is a fuction that maps an arbitrary long input string to a fixed length output string. Let h refer to an hash function of length n:

$$h\colon \{0,1\}^* \to \{0,1\}^n$$

m is usually called "the message", while d is usually called "the digest" and it can be seen as a compact representation of m. The length of d is the length of the hash.

Hash functions are usually used to provide data integrity and they're also used to construct other cryptographic primitives such as MACs and digital signatures.

1.1.1 Desired properties

An hash function should ideally meet these properties:

- **Computational efficiency**: given m, it must be easy to compute $d = h(m)$
- **Preimage resistance** (also called **one-way property**): given $d = h(m)$, it must be computationally infeasible computing m (m is the preimage)
- **Weak collision resistance** (also called **2nd preimage resistance**): given m_1 and $d_1 = h(m_1)$, it must be computationally infeasible finding a $m_2 \neq m_1$ so that $h(m_2) = d_1$
- **Strong collision resistance**: it must be computationally infeasible finding paris of distinct and colliding messages. Two messages $m_1 \neq m_2$ collide when $h(m_1) = h(m_2)$.
- **Avalanche effect**: changing a single bit of m should cause every bit of $d = h(m)$ to change with probability $P = 0.5$

1.1.2 Examples of hash functions

- **MD5**: published in 1991, it's a 128-bit hash function that was used for file integrity checks. Today it's considered unsecure and it shouldn't be used anymore.
- **Secure Hash algorithm 1 (SHA-1)**: 160-bit hash function that was used in SSL and TLS implementations. Today is considered unsecure and it's deprecated.
- **SHA-2**: family of SHA functions which includes SHA-256, SHA-384 and SHA-512. SHA-256 is currently used in several parts of the Bitcoin network.
- **SHA-3**: latest family of SHA functions, it is a NIST-standardized version of Keccak, which uses a new approach called "sponge construction" instead of the Merkle-Damgard transformation previously used. This family includes SHA3-256, SHA3-384 and SHA3-512.

1.1.3 Design of SHA-256

1.1.4 Message Authentication Codes (MACs)

A MAC is an hash function which uses a key and which can therefore be used to provide both integrity and authentication (proof of origin). Authentication is based on a key pre-shared between the sender and the receiver. The receiver can verify both integrity and authentication of a message by computing the MAC function of the message and comparing it with the one received from the sender: if they are the same then integrity and authentication are confirmed (note that it is assumed that only the sender and the receiver know the key).

MAC functions can be constructed using block ciphers or hash functions:

- in the first approach, block ciphers are used in the Cipher block chaining mode (CBC mode): the MAC of a message will be the output of the last round of the CBC operation. The length of MAC in this case is the same as the block length of the block cipher used to generate it.
- In the second approach they key is hashed with the message using a certain construction scheme. The most simple ones are *suffix-only* and *prefix-only*, which however are weak and vulnerable:
  - suffix-only: $d = MAC_k(m) = h(m|k)$, where h is an hash function
  - prefix-only: $d = MAC_k(m) = h(k|m)$, where h is an hash function

1.2 Digital signature

Digital signatures are used to associate a message with the entity from which the message has been originated. They provide the same service as MACs (authentication and non-repudiation) plus the non-repudiation.

Digital signature is based on public key cryptography: Alice can sign a message by encrypting it using its private key. Usually however, for efficiency and security reasons, Alice doesn't encrypt the message but its digest (hash of the message). Figure 1 shows how a generical digital signature function works.

An example of digital signature algorithms are RSA and ECDSA.

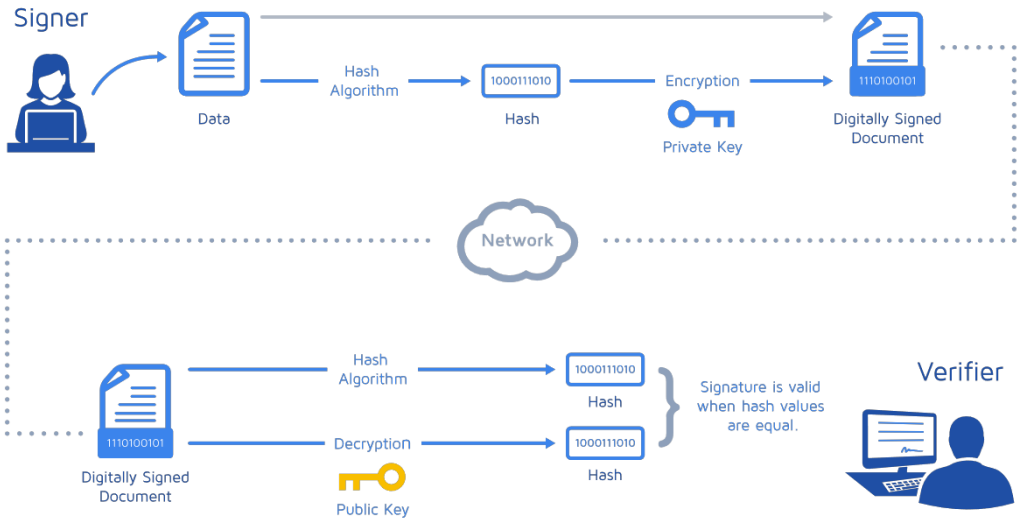

Figure 1: digital signature signing and verification scheme

1.3 Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA is a variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography.

1.3.1 Key pair generation

1. Define an elliptic curve E with modulus P, coefficients a and b and a generator point G that forms a cyclic group of order p, with p prime
2. Choose a random integer d so that $0 < d < q$

3. Compute the public key B so that $B = dA$

The public key is the sextuple $K_{pb} = (p, a, b, q, A, B)$, while the private key is the value of d randomly chosen in Step 2: $K_{pr} = d$

1.3.2 Signing a message

1. Choose an ephemeral key K_e, where $0 < K_e < q$. It should be ensured that Ke is truly random and no two signatures have the same key because otherwise the private key can be calculated
2. Compute $R = K_e A$
3. Initialize a variable r with the x coordinate value of the point R
4. The signature on the message m can be calculated as follow:

$$S = (h(m) + dr)K_e^{-1} \mod q$$

2 First

asdasd

2.1 test

References

[1] A.M. Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, 2017. ISBN: 9781491954362. Available at: https://books.google.it/books?id=MpwnDwAAQBAJ.

[2] J.J. Bambara et al. *Blockchain: A Practical Guide to Developing Business, Law, and Technology Solutions*. McGraw-Hill Education, 2018. ISBN: 9781260115864. Available at: https://books.google.it/books?id=z5hIDwAAQBAJ.

[3] I. Bashir. *Mastering Blockchain*. Packt Publishing, 2017. ISBN: 9781787125445. Available at: https://books.google.it/books?id=dMJbMQAACAAJ.

[4] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. "Cryptocurrencies Without Proof of Work". In: *Financial Cryptography and Data Security*. Ed. by Jeremy Clark et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 142–157. ISBN: 978-3-662-53357-4.

[5] Kyle Croman et al. "On Scaling Decentralized Blockchains". In: *Financial Cryptography and Data Security*. Ed. by Jeremy Clark et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 106–125. ISBN: 978-3-662-53357-4.

[6] Ethan Heilman, Foteini Baldimtsi, and Sharon Goldberg. "Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions". In: *Financial Cryptography and Data Security*. Ed. by Jeremy Clark et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 43–60. ISBN: 978-3-662-53357-4.

[7] G. Karame and E. Androulaki. *Bitcoin and Blockchain Security*. Artech House information security and privacy series. Artech House, 2016. ISBN: 9781630810139. Available at: https://books.google.it/books?id=b%5C_nwjwEACAAJ.

[8] Amitabh Saxena, Janardan Misra, and Aritra Dhar. "Increasing Anonymity in Bitcoin". In: *Financial Cryptography and Data Security*. Ed. by Rainer Böhme et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 122–139. ISBN: 978-3-662-44774-1.