Graduate Program in Computer Science, Universidade Federal de Santa Catarina
INE410135 (Blockchain and Cryptocurrencies Technologies)

Notes on Coursera's "Bitcoin and Cryptocurrency Technologies" 1st Lecture
Gustavo Zambonin

Introduction

Currencies of any kind, even in their most primitive forms, need some sense of security to achieve pertinent ownership of valuable goods, and enable users to exchange these accordingly. This concept is generally represented by means of physical money, such as coins, notes and checks, and oversight by a bank. However, transactions nowadays are frequently done in the virtual medium, executed through ATMs, payment cards, personal computers and other devices with digital components. It could be argued that this paradigm shift facilitates the migration of entire currency infrastructures to entirely digital approaches. Yet, various issues arise when taking into account the security of said systems with respect to decentralization of trust, anonymity of transactions, storage of digital coins and various other problems that touch economic and social disciplines.

To efficiently address these situations, a *cryptocurrency* may be established through the use of cryptography to allow secure transactions and creation of new currency units. By design, dissociation from central banking is given special emphasis, therefore making these currencies inherently unregulated. Nevertheless, there has been considerable adhesion from individuals and businesses alike. Ergo, it is reasonable to understand how concepts such as cryptographic hash functions, digital signatures, data structures, distributed ledgers and merge these to construct simple examples of cryptocurrencies.

Cryptographic hash functions

A hash function $\mathcal{H}$ deterministically maps values from a set with possibly infinite size to another, strictly smaller finite set. Hash functions with added properties that make them suitable for use in security applications are called cryptographic hash functions; these can provide assurance of data integrity, even if the data is stored in an insecure place. Outputs of these functions are generally called digests. For any $\mathcal{H}$ to be considered cryptographic, solving the three problems listed below should be computationally infeasible:

(i) Given a digest h, find the original message m such that $\mathcal{H}(m) = h$. $\mathcal{H}$ is said to be *preimage* resistant if this cannot be solved efficiently.

(ii) Given a message m_0, find a message m_1 such that $m_0 \neq m_1$ and $\mathcal{H}(m_0) = \mathcal{H}(m_1)$. $\mathcal{H}$ is said to be *second preimage* resistant if this cannot be solved efficiently.

(iii) Given messages m_0 and m_1 such that $m_0 \neq m_1$, then $\mathcal{H}(m_0) = \mathcal{H}(m_1)$. $\mathcal{H}$ is said to be *collision* resistant if this cannot be solved efficiently.

Note that (ii) and (iii) present a slight difference: in the former, an adversary may not choose the first message, whereas in the latter, it is free to choose any pair of messages. It is also desirable for $\mathcal{H}$ that its mappings happen in such a way that digests have no apparent relation between themselves and their preimages. Furthermore, another suitable characteristic is the avalanche effect, based on the concept of diffusion, stating that any changes in a message m should result in a totally different resulting digest.

Digital signatures

A digital signature scheme is a mathematical construction that enables certain properties to be derived from signed messages. Specifically, it uniquely identifies and therefore *authenticates* the

signer; it preserves the *integrity* of messages, confirming that these were not modified in transit; and it prevents denial from the signer that the message was signed and sent — *non-repudiation*.

These schemes are based on public key cryptography, in which an entity has possession of a key pair, used in a way such that a document signed with one of the keys may only be verified using its correspondent in the pair. Evidently, to prevent impersonation, one of the keys is kept private, while the other is published to enable verification by any party. Digital signature schemes are defined as a triple of algorithms, listed below.

(i) Key pair generation GEN, that takes as input a security parameter $1^n, n \in \mathbb{N}^*$ and outputs a random key pair $(\mathcal{S}_k, \mathcal{P}_k)$.

(ii) Signature generation SIG, that takes as input a message m and the secret key $\mathcal{S}_k$, and outputs a signature σ. A common practice is to first hash the message then sign the resulting digest.

(iii) Signature verification VER, that takes as input m, σ and the public key $\mathcal{P}_k$, and outputs either 0 for a failed verification or 1 for a correct one.

These algorithms are constructed such that signatures on any possible messages are verifiable using any generated keys, that is, $\forall (p, s) \in \text{GEN}(1^n), \forall w \in \{0,1\}^*, \Pr[\text{VER}(p, w, \text{SIG}(s, w)) = 1] = 1$. However, being in accordance to this description is not a sufficient condition for a scheme to be secure, only correct. This notion is achieved in its strongest form when an adversary can forge a valid signature using any message.

Distributed ledgers

A ledger summarizes financial information in the form of debits and credits, showing balances for certain periods of time, allowing visualization and management of transactions. To achieve decentralization, the ledger must be replicated and synchronized across heterogeneous nodes, generally through peer to peer networks. Design of distributed ledgers generally come in the form of graphs, in which nodes containing transaction data are connected by pointers to hashes of other transactions, either linearly (in the case of blockchain), in the form of a directed acyclic graph, or using a binary hash tree, called a Merkle tree. Hence, to falsify a transaction, an adversary would have to forge every transaction before it.

Examples of cryptocurrencies

Summarizing these concepts into an example may be done through the presentation of GoofyCoin. Goofy may create coins whenever needed, and this transaction shall be signed by its private key. Available currency may then be transferred to any recipient with a transaction signed by the coin's owner. Clearly, these transactions are linked through hash pointers. However, this design does not prevent the *double-spending problem*, in which an entity may create transfer transactions haphazardly. To be secure, this cryptocurrency could make use of the blockchain design, that takes in account the history of transactions. Hence, transactions are only valid if coins were really created, were not already consumed, the value being received is equal to the value being given, and shall be signed by all owners of the coins being transacted.