Chapter 4: Cryptography.

Cryptography is one of Ethereum's foundational technologies, and is a branch of math, it can be used to prove a secret, without revealing the secret.

One of the cryptography used in Ethereum is the PKC - Public Key Cryptography used in private and public keys.

Ethereum transactions require a valid digital signature to be included in the blockchain. Anyone with access to a public key has access to the ether the account holds.

EOAs are backed by private keys unlike contracts.

Public key cryptography is also called assymetric cryptography.

For cryptography, it is quite simple doing the forward process, but the backward process is near impossible, unless you're given some sort of magic clue, hence functions like these are called the "trapdoor function".

But in the case of cryptography based off arithmetic on an elliptic curve, it produces the "discrete logarithm problem" and there is no known trapdoor, hence, this is the one that Ethereum uses.

The public key is created from the private key and, the public key makes the account accessible, but the private key gives the owner the control over the account as it is used to sign transactions from the account, these signatures are called "digital signatures".

Digital signatures can be created to sign any message. How? The elliptic curve cryptography makes a way for the transaction data and the private key to be combined to create a code, and that code can only be produced with the knowledge of the private key. That code is called the digital signature.

The transaction is valid if the digital key matches the transaction data and the public key that access is being requested. This verification does not involve the private key.

Making the private key, get a 256 bit number and hash it with sha256 or keccak256.

An Ethereum public key is a point on an elliptic curve, meaning it is a set of x and y coordinates that

satisfy the elliptic curve equation.

The math K = k * G where k is the private key and G is the generator point, a constant point and K is the resulting public key.

The generator point is specified as part of the secp256k1 standard, because the generator point is always the same for all Ethereum users, a private key k multiplied with G will always result in the same public key K.

A point in an elliptic curve, G can be multiplied with another point k, to produce another point, K, we might think dividing K with G will give us k, but that is not how it works in elliptic curve cryptography, getting k from K, is as difficult as trying all possible values of k.

Elliptic curve cryptography is a type of asymmetric or public key cryptography based on the discrete logarithm problem as expressed by addition and multiplication on the points of an elliptic curve.

Ethereum uses a specific elliptic curve and set of mathematical constants, as defined in a standard called "secp256k1", established by the US National Institute of Standards and Technology (NIST).

y^2 * mod p = ((x ^ 3) + 7) * mod p
Mod p == Fp, where p (prime order of the elliptic curve).

p = 2^256 – 2^32 – 2^9 – 2^8 – 2^7 – 2^6 – 2^4 – 1,

Hashes are one way, meaning that, knowing the hash doesn't guarantee knowing the input data.

Determinism, Verifiability, Non-correlation, Irreversibility, Collision protection are the properties of a good hashing function.

Ethereum uses the Keccak256 cryptographic hashing algorithm.

The last 20 bytes of the Keccak256 hash of the public key derived from K = k * G is the Ethereum address. The 0x prefix on them eg 0x1234567...888 means they're 40 in length.

Using the Address without checksum, I can generate an address with the right checksum.

The Keccak256 hash of a lowercase address is the checksum hash.