



Blockchain for Fin-Tech

Cristian TOMA, Marius POPA, Alin ZAMFIROIU

IT | Cyber Security Master Program – ISM
Applied Computer Science and Cybersecurity R&D Team – ACS
Department of Economic Informatics & Cybernetics – DICE | DEIC
www.ism.ase.ro | www.acs.ase.ro | www.dice.ase.ro

Make Blockchain Technology working for Fin-Techs



Make Blockchain Technology working for Fin-Techs



Visit us at www.fintech-ho2020.eu

or contact us at info@fintech-ho2020.eu

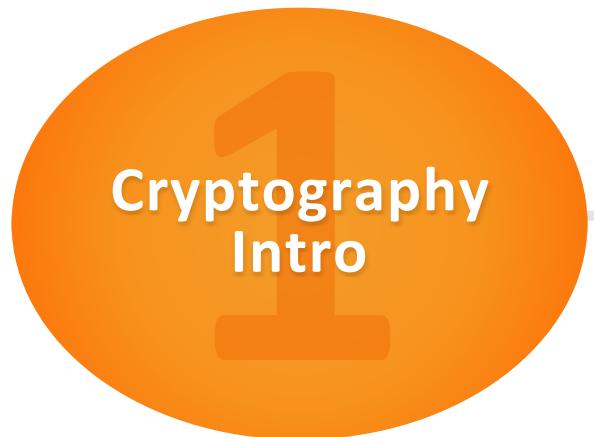
or follow us on [LinkedIn](#)

The development of the materials has also been supported by the project **A FINancial supervision and TECHnology compliance training programme**.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825215 (Topic: ICT-35-2018 Type of action: CSA)

All material presented here reflects only the authors' view. The European Commission is not responsible for any use that may be made of the information it contains.

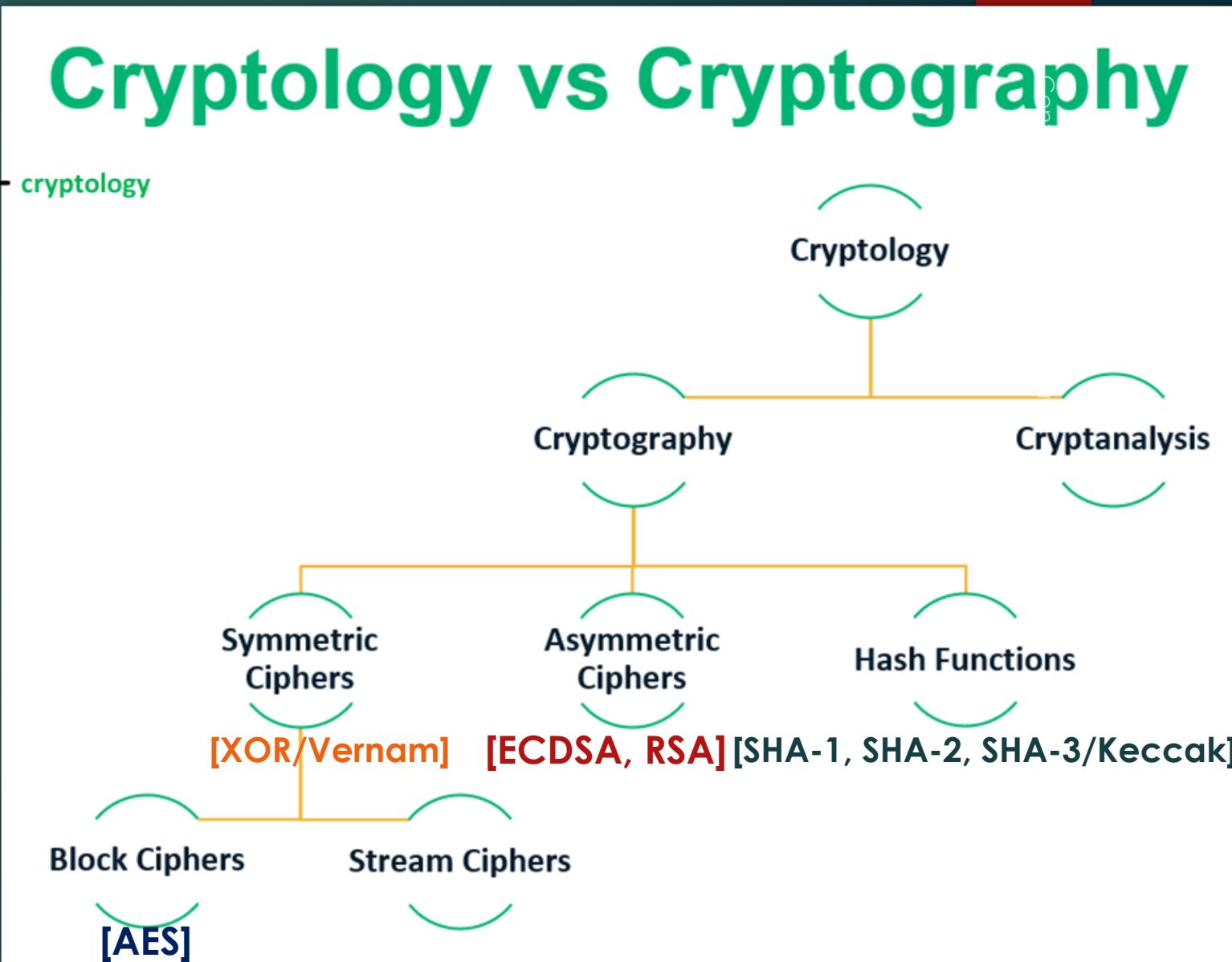
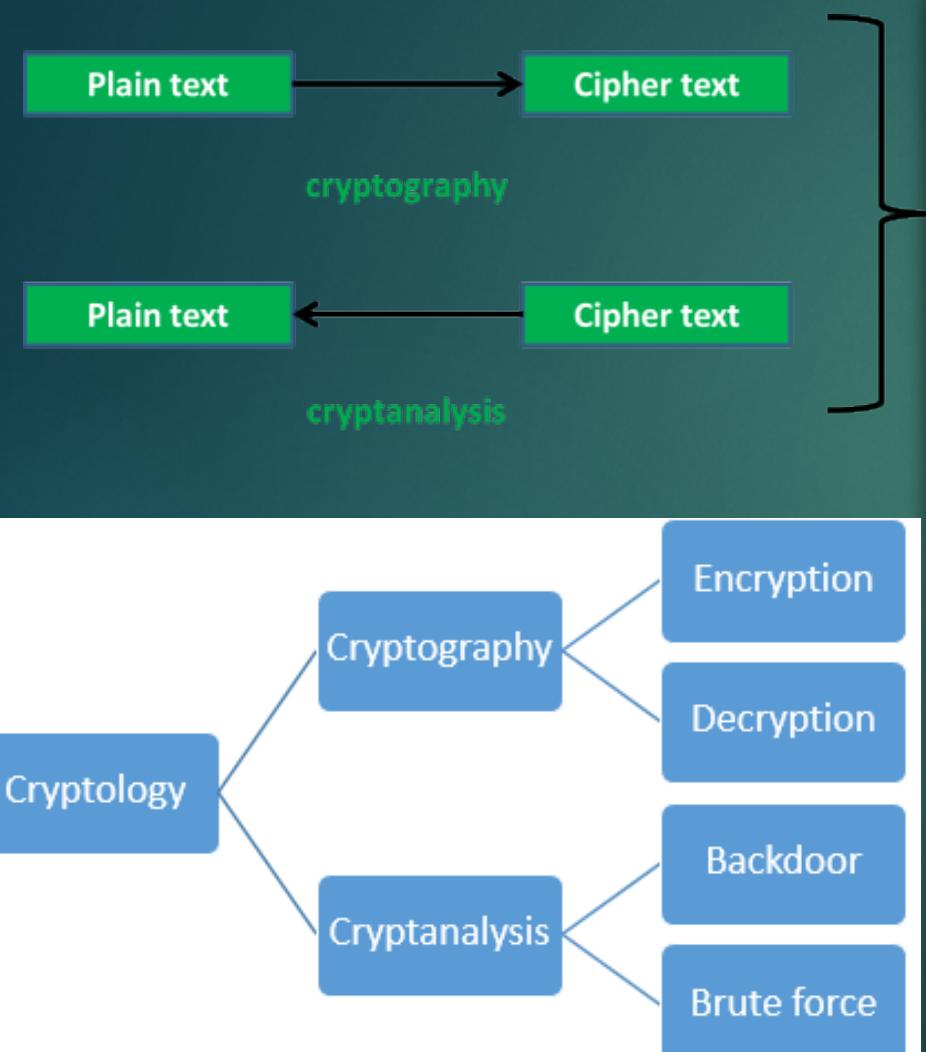
Agenda for the Blockchain in Fin-Tech



www.dice.ase.ro



www.ism.ase.ro



01. Crypto Intro – XOR with symmetric key

6

XOR = Exclusive or or exclusive disjunction is a logical operation that outputs true only when inputs differ (one is true, the other is false).

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

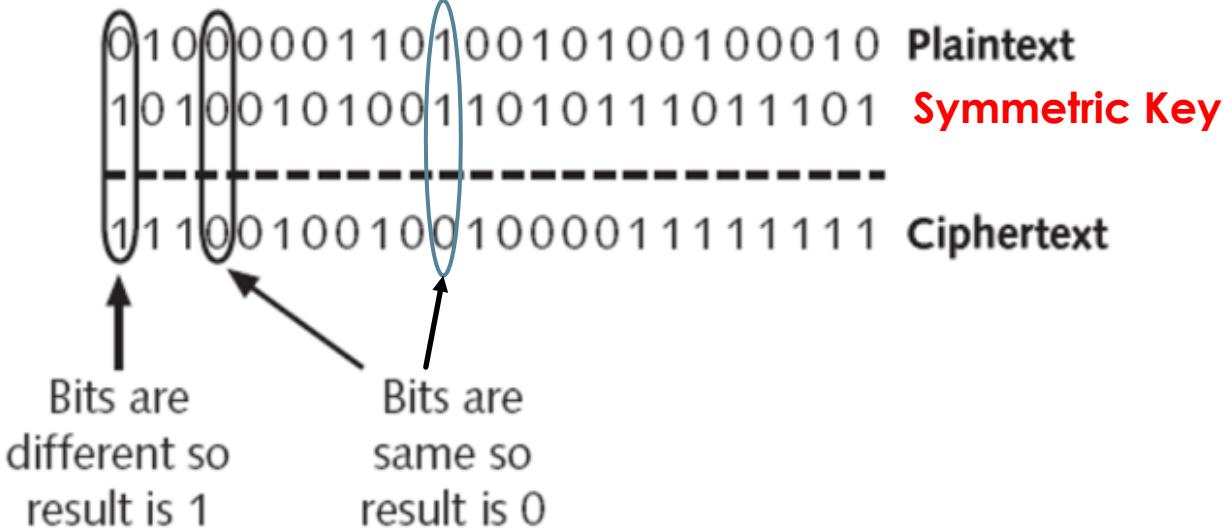
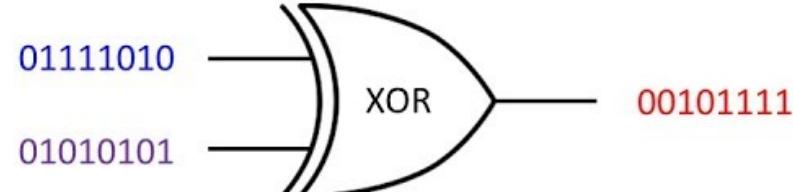
XOR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

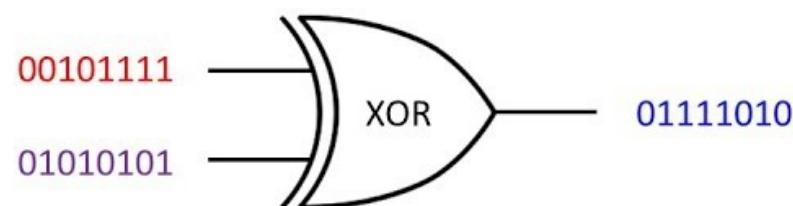
NOT Truth Table

A	B
0	1
1	0

ENCRYPTION



DECRIPTION



Home Insert Draw Design Transitions Anim Mute Start Video Security Participants New Share Pause Share Annotate Remote Control More

Cut Copy Layout Reset Calibri (Body) You are screen sharing Stop Share

Paste Format New Slide Section Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear Save

01. Crypto Intro – XOR with symmetric key

5

XOR = Exclusive or or exclusive disjunction is a logical operation that outputs true only when inputs differ (one is true, the other is false).

AND Truth Table

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR Truth Table

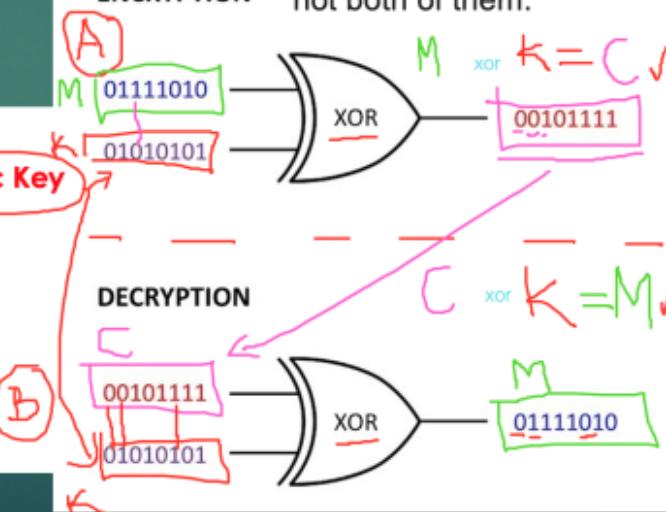
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

NOT Truth Table

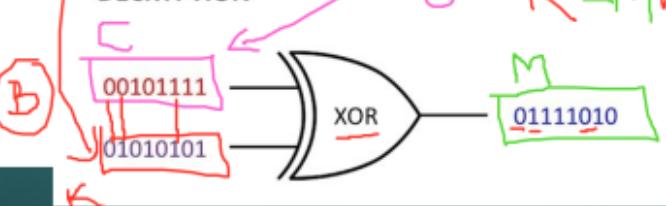
A	B
0	1
1	0

Exclusive disjunction essentially means 'either one, but not both nor none'. In other words, the statement is true if and only if one is true and the other is false. For example, if two horses are racing, then one of the two will win the race, but not both of them.

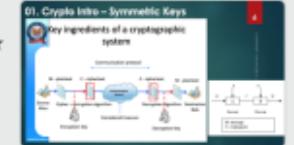
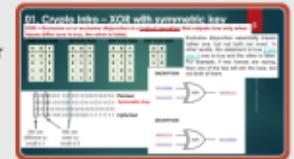
ENCRYPTION



DECRYPTION



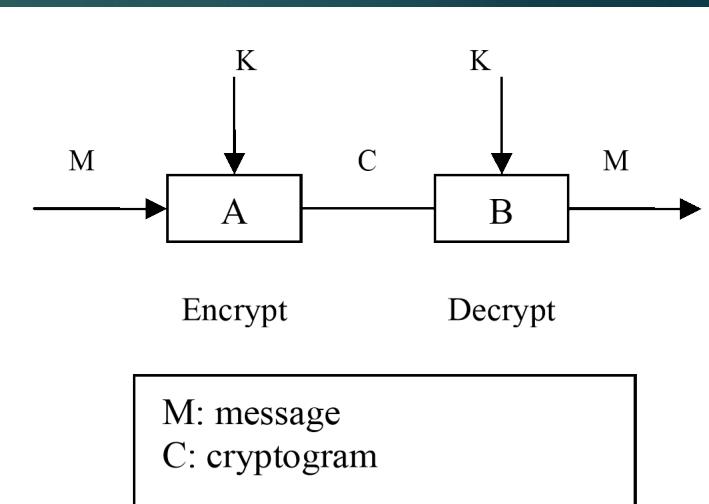
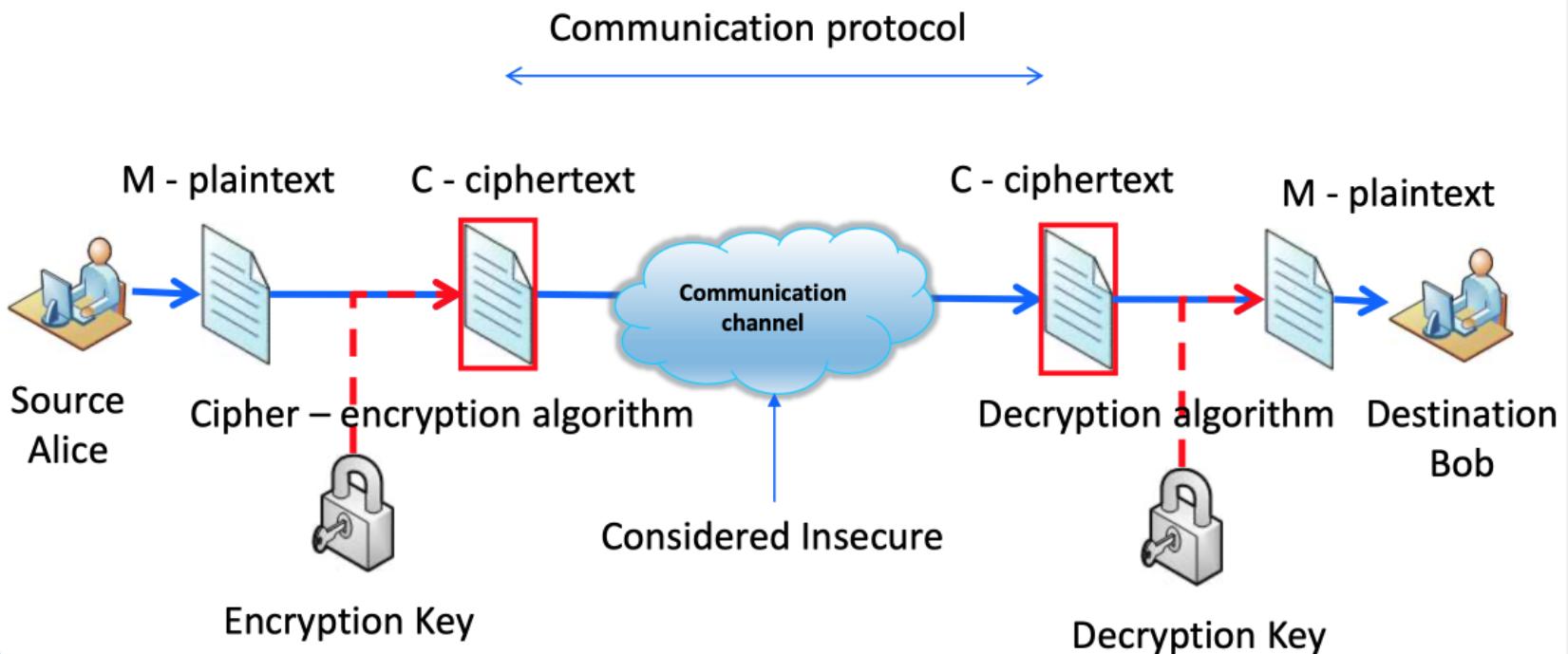
Click to add notes



01. Crypto Intro – Symmetric Keys



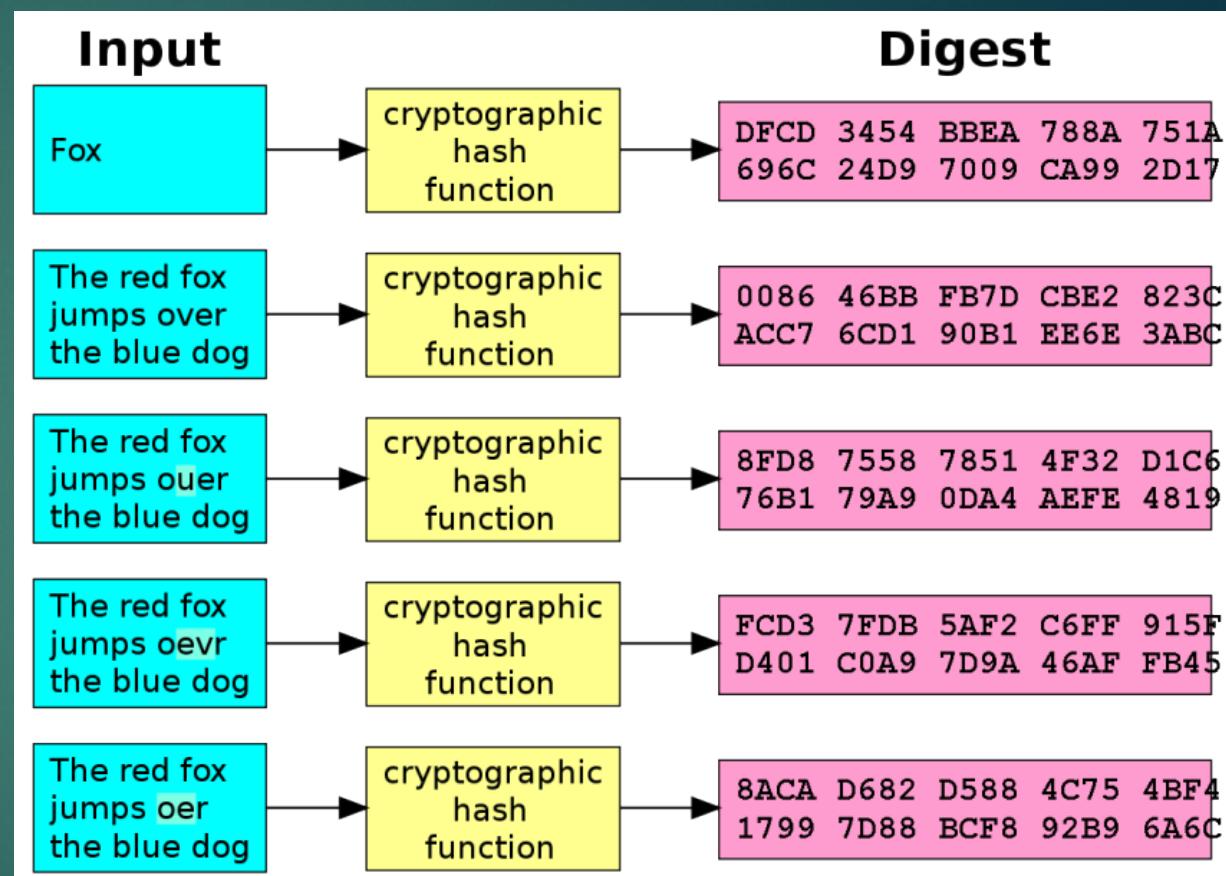
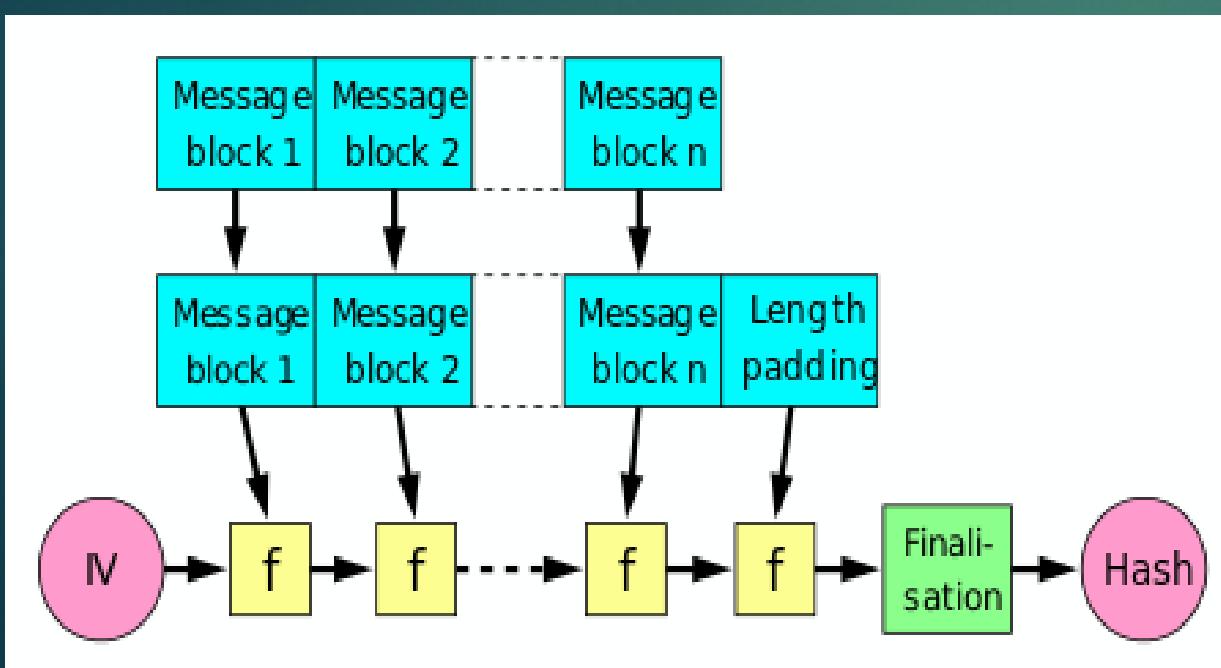
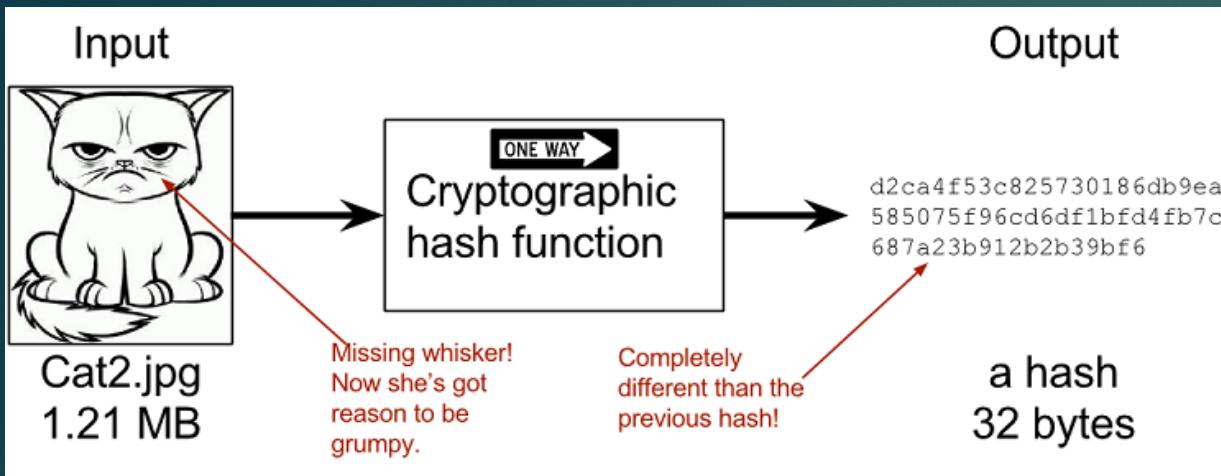
Key ingredients of a cryptographic system



01. Crypto Intro - Hash

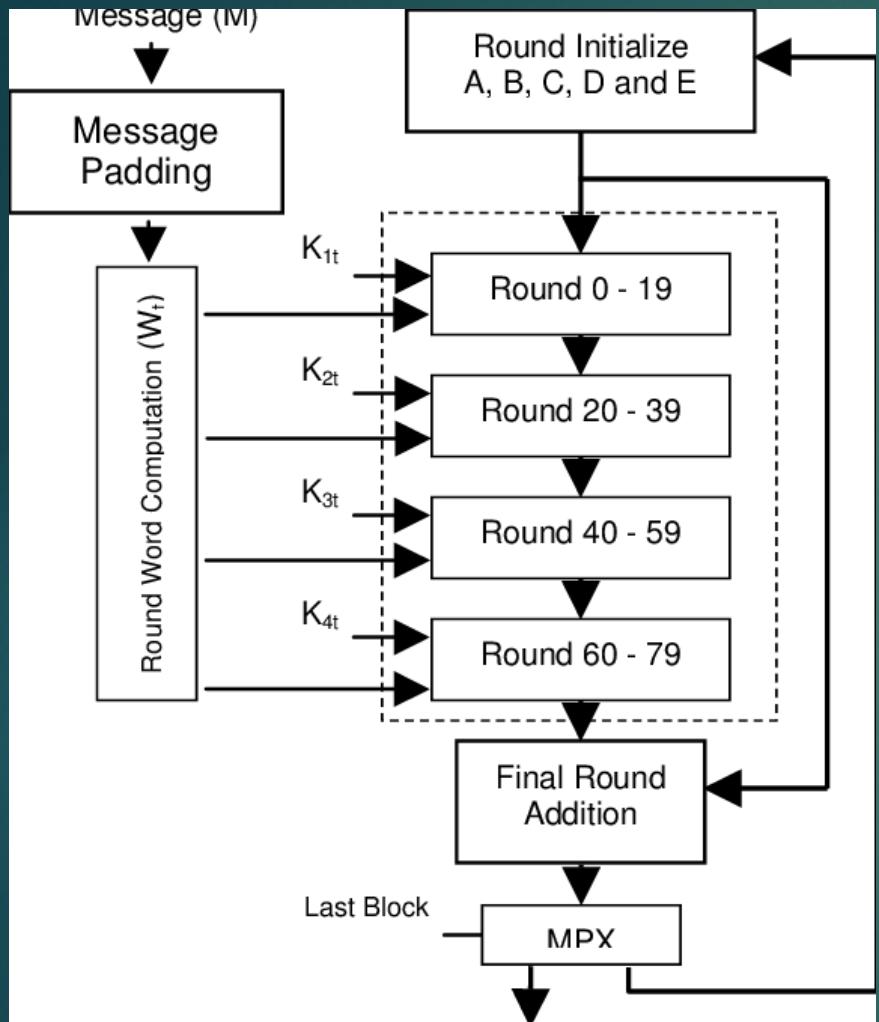
https://en.wikipedia.org/wiki/Cryptographic_hash_function

9



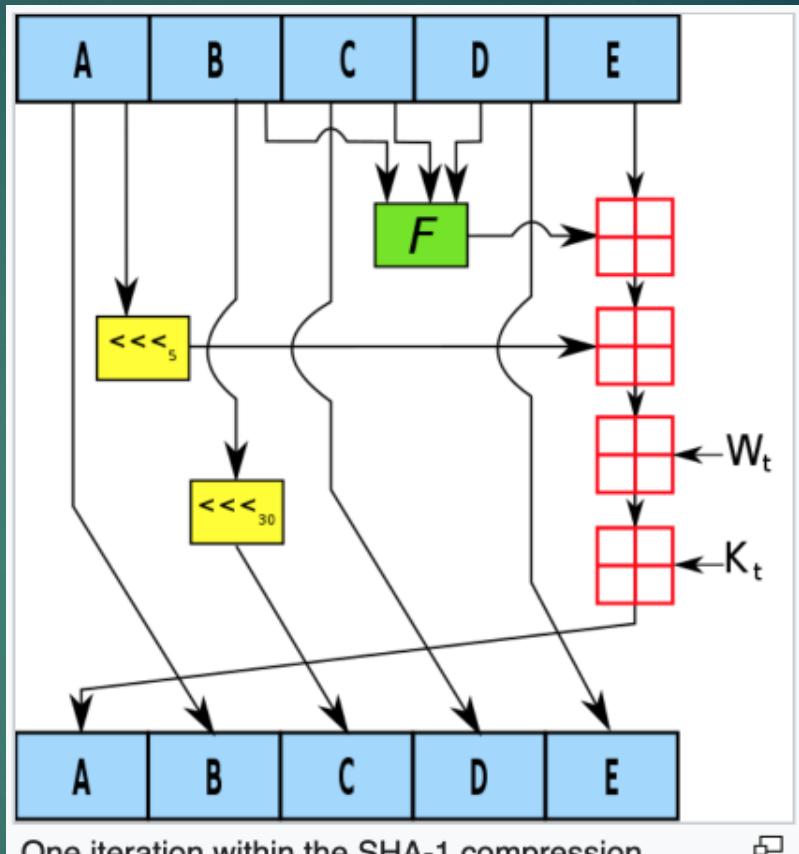
01. Crypto Intro - Hash

10



<https://www.slideshare.net/ruchimaurya487/sha-secure-hashing-algorithm>

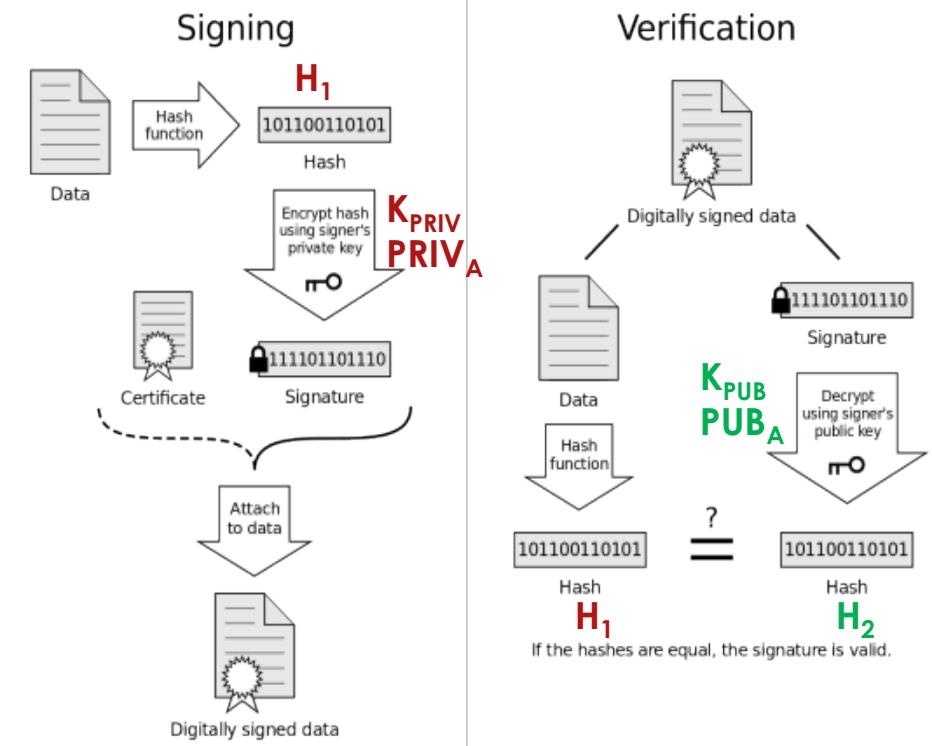
<https://en.wikipedia.org/wiki/SHA-1>



One iteration within the SHA-1 compression function:
A, B, C, D and E are 32-bit **words** of the state;
F is a nonlinear function that varies;
 \lll_n denotes a left bit rotation by n places;
n varies for each operation;
 W_t is the expanded message word of round t;
 K_t is the round constant of round t;
⊕ denotes addition modulo 2^{32} .

01. Crypto Intro – Asymmetric Keys

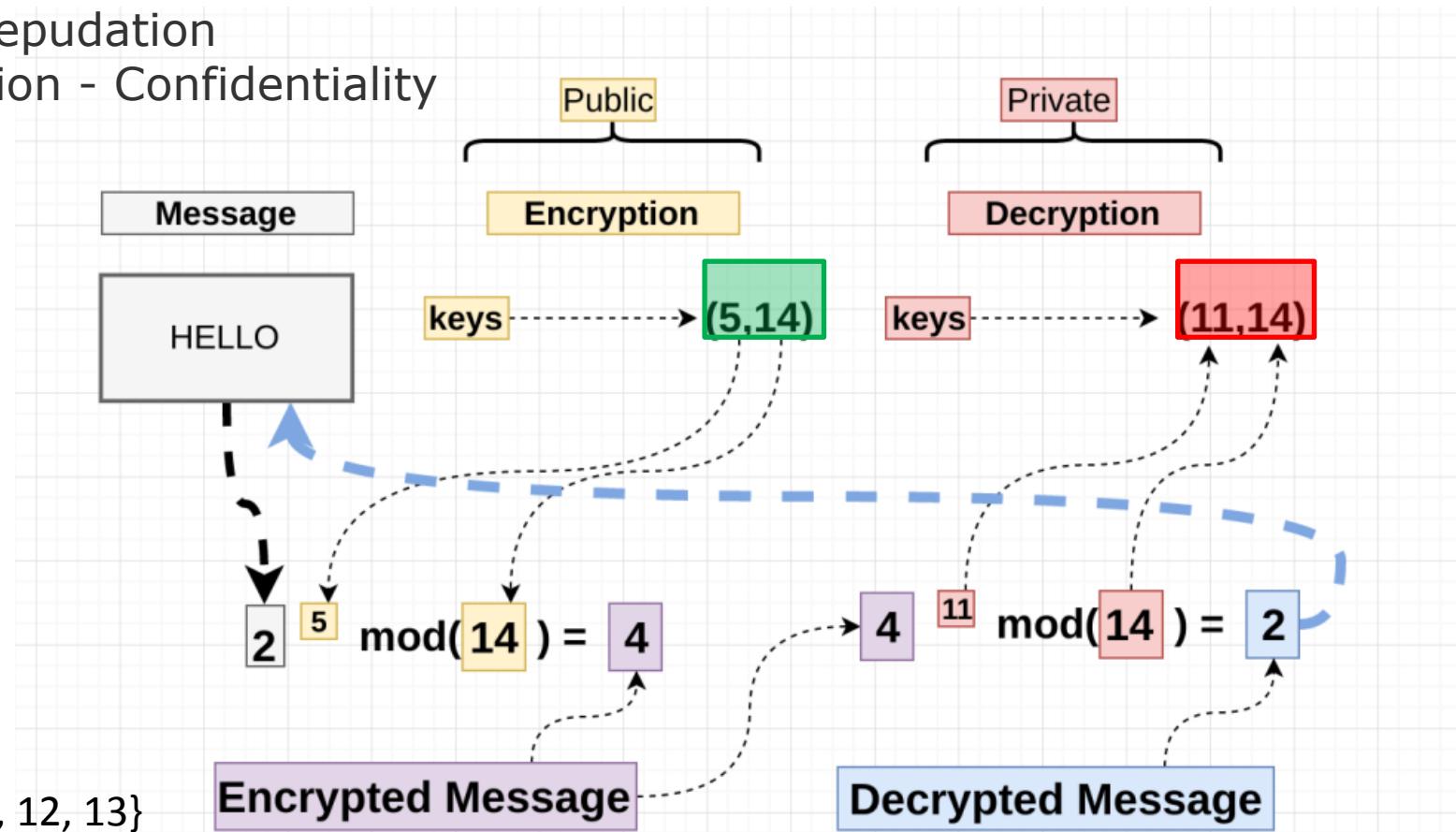
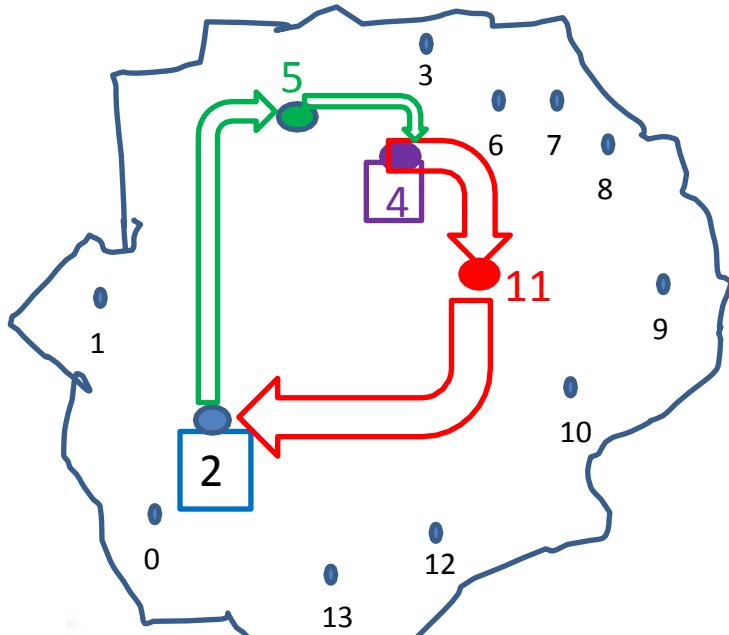
11



Cryptography Elements used in Blockchains

Asymmetric-Key Cryptography - Rivest-Shamir-Adleman (RSA)

- How does RSA work:
 - Signature – Non- Repudation
 - Encryption/Decryption - Confidentiality



Search in Presentation Share

01. Crypto Intro – Asymmetric Keys

9

AUTHENTICATION
if $(H1 == H2)$ the authentication is correct

$(2^6 \bmod 7 = (2^3 \bmod 7)^2 \bmod 7)$

$0, 1, 2, 3, 4, 5, 6$

$M = 2$
 $KPRIV = 3$
 $KPUB = 6$

$(M^KPRIV \bmod 7 = 2^{3 \cdot 2} \bmod 7 = 8 \bmod 7 = 1)$
 $2^6 \bmod 7 = 64 \bmod 7 = [(2^3) * (2^3)] \bmod 7 = [(2^3 \bmod 7) * (2^3 \bmod 7)] = 1 * 1 = 1$

Notes Comments

Slide 9 of 134 English

01. Crypto Intro – RSA simple numeric sample

14

The numeric example is just for the digital signature, is depict also in figure for the electronic signing process:

- given $p=53$ and $q=61$, 2 secret prime numbers of A.
- given $n_A = 53 \cdot 61 = 3233$ the product of these two numbers (if n very big the possibility to find p and q – **meaning to factorize on n** – is very low).
- The indicator of Euler calculated is:
 - $\Phi(n) = (p-1)*(q-1) = 52 \cdot 60 = 3120$.
- Is chosen a secret private key **PRIV_A = 71**; (meaning a number relatively prime with 3120 and in the period $[\max(p,q)+1, n-1]$)
- Is calculated the public key **PUB_A = 791** as multiplied reverse mod 3120 of PRIV_A:
 - $(71 \cdot 791) \bmod 3120 = 1$ i.e **56161 mod 3120 = 1;**

Online modulo calculators:

<https://www.mtholyoke.edu/courses/quenell/s2003/ma139/js/powermod.html>

<https://miniwebtool.com/modulo-calculator/>

01. Crypto Intro – RSA simple numeric sample

15

- Is considered a document whose digest $H(M)=h$ is 13021426. As the value overpasses the length of the module $n_A = 3233$, this digest will be broken in two blocks (1302 and 1426), that A will sign separately, using the **private key of A - $\text{PRIV}_A = (71, 3233) = (\text{EXP_PRIV}_A, \text{MOD}_A)$** :
 - $(1302^{71}) \bmod 3233 = 1984;$
 - $(1426^{71}) \bmod 3233 = 2927;$
- the electronic signature obtained is **$S = 1984\ 2927$** ;
- S obtained is transmitted to the previous step and M (in clear – for this example, there is no confidentiality).
- When receiving, B (B receives the package $S+M$) will calculate $H(M)$ in two modes and will compare them:
- he will calculate $H(M)$ with the **public key of A - $\text{PUB}_A = (791, 3233) = (\text{EXP_PUB}_A, \text{MOD}_A)$** :
 - $(1984^{791}) \bmod 3233 = 1302$
 - $(2927^{791}) \bmod 3233 = 1426;$
 - **So $H_1 = 1302\ 1426$;**
- and will calculate $H(M)$ on the received message:
 $H_2 = H(M) = 1302\ 1426;$

Online modulo calculators:

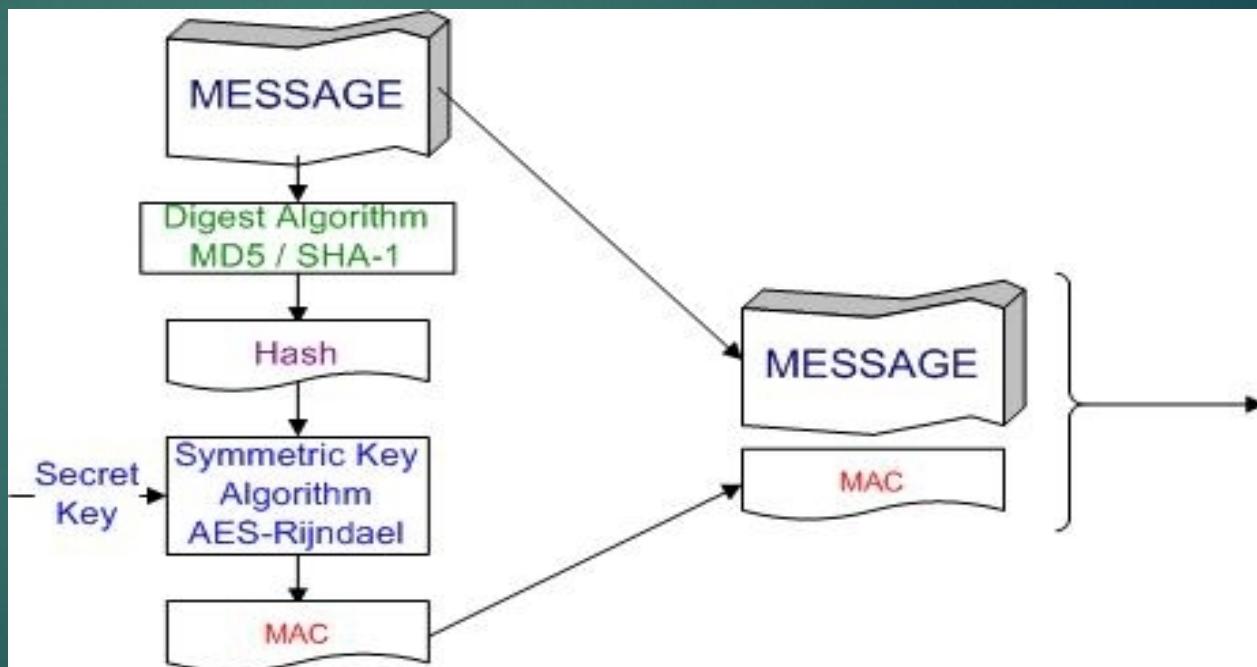
<https://www.mtholyoke.edu/courses/quenell/s2003/ma139/js/powermod.html>

<https://miniwebtool.com/modulo-calculator/>

01. Crypto Intro – HMAC

Wiki: “In [cryptography](#), an **HMAC** (sometimes expanded as either **keyed-hash message authentication code** or **hash-based message authentication code**) is a specific type of [message authentication code](#) (MAC) involving a cryptographic hash function and a secret cryptographic key. As with any MAC, it may be used to simultaneously verify both the data integrity and the authenticity of a message.”

“Any cryptographic hash function, such as [SHA-2](#) or [SHA-3](#), may be used in the calculation of an HMAC; the resulting MAC algorithm is termed HMAC-X, where X is the hash function used (e.g. HMAC-SHA256 or HMAC-SHA3-256). The cryptographic strength of the HMAC depends upon the [cryptographic strength](#) of the underlying hash function, the size of its hash output, and the size and quality of the key.”



Message Authentication Code Steps

01. Crypto Intro – HMAC

17

This definition is taken from [RFC 2104](#):

$$\text{HMAC}(K, m) = \text{H} \left((K' \oplus opad) \parallel \text{H} \left((K' \oplus ipad) \parallel m \right) \right)$$

$$K' = \begin{cases} \text{H}(K) & K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases}$$

where

H is a cryptographic hash function

m is the message to be authenticated

K is the secret key

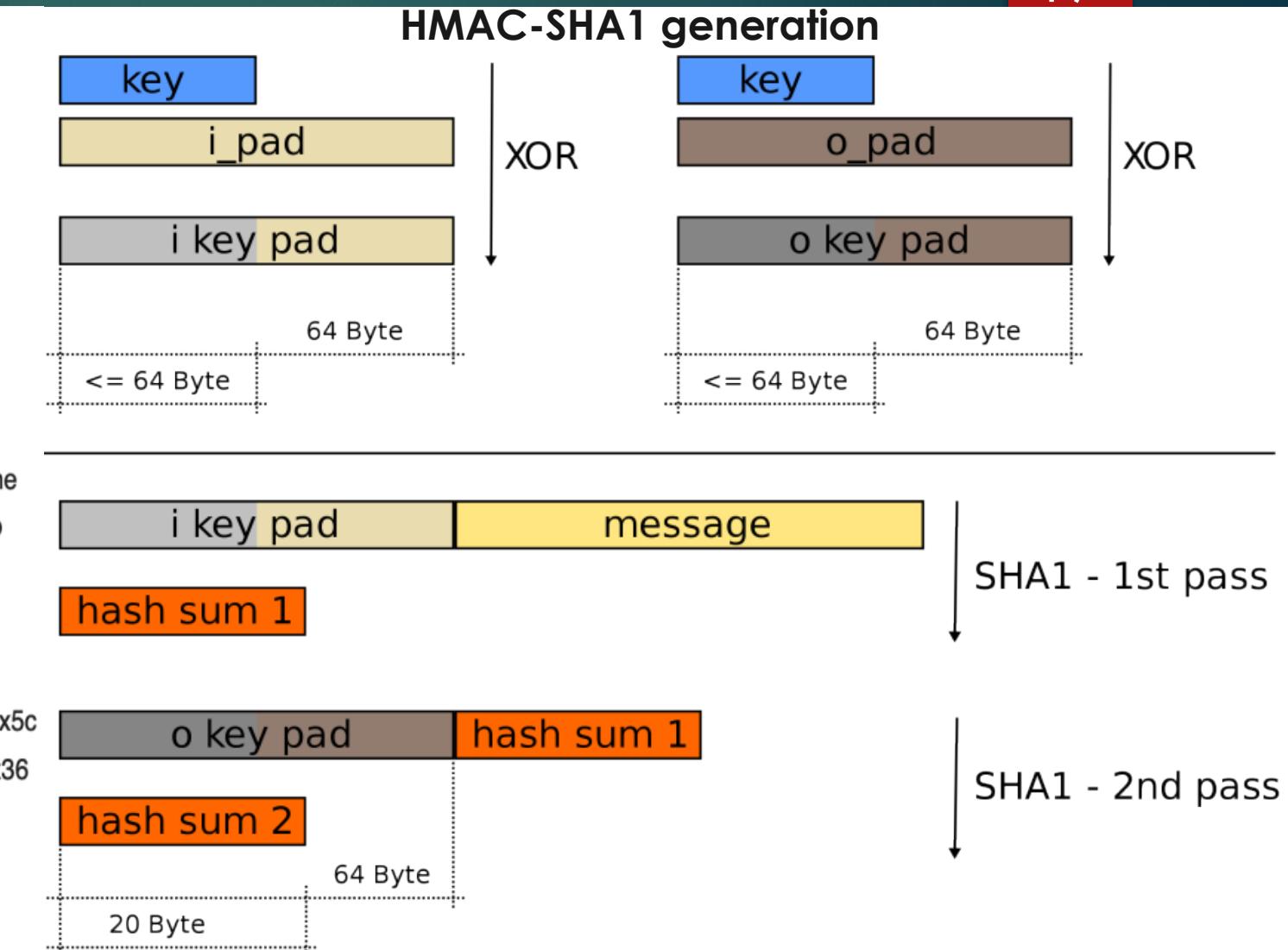
K' is a block-sized key derived from the secret key, K; either by padding to the right with 0s up to the block size, or by hashing down to less than or equal to the block size first and then padding to the right with zeros

|| denotes **concatenation**

\oplus denotes bitwise **exclusive or (XOR)**

opad is the block-sized outer padding, consisting of repeated bytes valued 0x5c

ipad is the block-sized inner padding, consisting of repeated bytes valued 0x36



01. Crypto Intro a little bit more details

What is cryptology?

The meaning

- „*kryptós*“ („hidden, secret“) and „*lógos*“ („writing“, however in this context „*lógos*“ means „study“).
- Cryptology is about techniques and protocols making information available only for authorized persons.
- The fields: **cryptography** and **cryptanalysis**.

Cryptography (*kryptós* and *gráphein*, “to write”)

- Science of encryption systems guaranteeing secure and confidential **storage** and **exchange** of information (e.g. between computers).
- A secure exchange of the encryption keys and **integrity** checking, e.g. for online banking, for electronic elections, or for electronic money.
- Based on (unsolved/difficult) **mathematical problems**.

What is cryptology?

Cryptanalysis (*kryptós* and *analýein*, “to loosen” or “to untie”)

- The counter part to cryptography and studies theories and techniques for **testing** and **breaking** cryptographic methods.
- It tries e.g. to derive information about the original plaintext or the used encryption key by investigating a ciphertext (the result of an encryption process).
- **Maths** and computer science are used (e.g. statistical tests, entropy, analysis of frequency and structure, complexity considerations, brute-force algorithms and much more).

History of cryptography and cryptanalysis

Classic Cryptography

- **Transposition Ciphers** rearrange the order of letters in a message (e.g., 'hello world' becomes 'ehlol owrdl').
- **Substitution Ciphers** replace letters or groups of letters with other letters or groups of letters (e.g., 'fly at once' becomes 'gmz bu podf' by replacing each letter with the one following it in the Latin alphabet).
- Ciphertexts reveal statistical information about the plaintext, and that information can often be used to break the cipher.
- Caesar cipher, Steganography (a message tattooed on a slave's shaved head and concealed under the regrown hair).

History of cryptography and cryptanalysis

Computer Era

- Extensive use of mathematics, including aspects of information theory, computational complexity, statistics, combinatorics, abstract algebra, number theory, and finite mathematics generally.
- Development of digital **computers** and **electronics** helped in cryptanalysis, it made possible much more complex ciphers.
- Many computer ciphers can be characterized by their operation on binary bit sequences (sometimes in groups or blocks), unlike classical and mechanical schemes, which generally manipulate traditional characters (i.e., letters and digits) directly.
- Good modern ciphers have stayed ahead of cryptanalysis; it is typically the case that use of a quality cipher is very efficient.

History of cryptography and cryptanalysis

Advent of Modern Cryptography

- Cryptanalysis of the new mechanical devices proved to be both difficult and laborious.
- Extensive open academic research into cryptography is relatively recent. Cryptography systems based on mathematical problems that are **easy to state** but have been found **difficult to solve**.
- Cryptographic algorithm and system designers must also sensibly consider probable future developments while working on their designs.

Modern cryptography

Symmetric-Key Cryptography

- Encryption methods in which both the sender and receiver share the same key.
- Symmetric key ciphers are implemented as either **block ciphers** or **stream ciphers**.
- Stream ciphers create an arbitrarily long stream of key material, which is combined with the plaintext bit-by-bit or character-by-character, somewhat like the one-time pad.
- Cryptographic **hash functions** take a message of any length as input, and output a short, fixed length hash. For good hash functions, an attacker cannot find two messages that produce the same hash.
- **Message authentication codes** (MACs) are much like cryptographic hash functions, except that a secret key can be used to authenticate the hash value upon receipt.
- Data Encryption Standard (DES), Advanced Encryption Standard (AES), SHA algo family (designed by NSA)

Modern cryptography

Public-Key Cryptography

- A significant disadvantage of symmetric ciphers is the **key management** necessary to use them securely.
- Asymmetric key cryptography use two different but mathematically related keys: a public key and a private key.
- In public-key cryptosystems, the public key may be freely distributed, while its paired private key must remain secret.
- In a public-key encryption system, the public key is used for encryption, while the private or secret key is used for decryption.
- Diffie–Hellman key exchange protocol, a solution that is now widely used in secure communications to allow two parties to secretly agree on a shared encryption key.
- RSA (Ronald Rivest, Adi Shamir, and Len Adleman) algorithm, Elliptic Curve techniques.

Modern cryptography

Public-Key Cryptography

- Used for implementing **digital signature** schemes.
- **Digital signature** is a mathematical scheme for verifying the authenticity of digital messages or documents. Two algorithms: one for signing (secret key is used to process the message or a hash), and one for verification (the matching public key is used to check the validity of the signature).
- Digital signatures are central to the operation of **public key infrastructures** and many network security schemes.
- RSA and DSA are two of the most popular digital signature schemes.

Modern cryptography

Cryptanalysis

- The goal of cryptanalysis is to find some weakness or insecurity in a cryptographic scheme, thus permitting its subversion or evasion.
- Most ciphers, apart from the one-time pad, can be broken with enough computational effort by brute force attack, but the amount of effort needed may be exponentially dependent on the key size.
- Cryptanalytic attacks: **ciphertext-only attack** (access only to the ciphertext), **known-plaintext attack** (access to a ciphertext and its corresponding plaintext), **chosen-plaintext attack** (choose a plaintext and learn its corresponding ciphertext), **gardening** (encouraging a target to use known plaintext in an encrypted message), **man-in-the-middle attack** (gets in between the sender)and the recipient, accesses and modifies the traffic and then forwards it to the recipient).

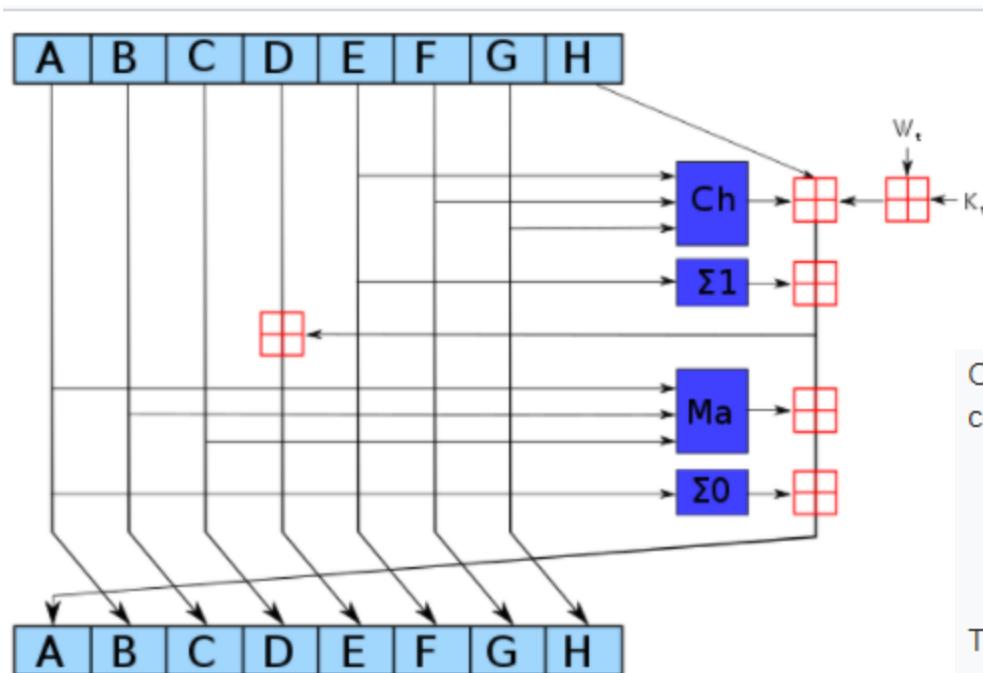
Hash Functions

- A hash function maps a large input set to a smaller output set. The values returned by a hash function are called **hash values**, **hash codes**, **digests**, or simply **hashes**.
- Hash functions are related to (and often confused with) checksums, check digits, fingerprints, lossy compression, randomization functions, error-correcting codes, and ciphers.
- Cryptographic hash algorithms: MD5, SHA-1, SHA-2, SHA-3, RIPEMD-160.

Cryptography Elements used in Blockchains

SHA-2

- SHA-2 (Secure Hash Algorithm 2) is a set of cryptographic hash functions designed by the United States National Security Agency (NSA).



One iteration in a SHA-2 family compression function. The blue components perform the following operations:

$$\text{Ch}(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$$

$$\text{Ma}(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$

$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$

$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

The bitwise rotation uses different constants for SHA-512. The given numbers are for SHA-256.
The red \square is addition modulo 2^{32} for SHA-256, or 2^{64} for SHA-512.

Cryptography Elements used in Blockchains

JCrypTool DEMO - Hash Sensitivity

The screenshot shows the JCrypTool Hash Sensitivity demo interface. The interface includes a toolbar at the top with a question mark icon. Below it, there's a section titled "Hash Sensitivity" with a brief description: "A hash function maps a large input set to a smaller output set. The input is of variable length, however, the output, usually has a small fixed length. Here, it is possible to select a hash function and the representation type of the hash values. The hash values of both input texts are calculated and displayed at once. The difference between the two hash values displayed at the bottom is a measure of the sensitivity of a hash value concerning changes in the input." The "Select a hash function" dropdown is set to "SHA-2 (256 bits)". The "Representation of the hash values" section allows choosing between hexadecimal (selected), decimal, and binary representations. The "Input 1" and "Input 2" sections both contain the same text: "This is the JCrypTool sample file. You can use this file for a fast start with JCrypTool, e.g. by encrypting or digitally signing it using the 'Algorithms' menu, or by applying one of the analysis offered in the..." The "Hash value 1" and "Hash value 2" sections display identical hex values: D9 C1 17 9C B4 13 E8 07 BD A5 31 78 DB 84 52 D9 5A 94 0A 26 9B 41 8E A7 4F FD 84 25 24 4C 7A BF. The "Difference of the two hash values" section shows a large number of zeros followed by a single hash symbol (#). At the bottom, a status message indicates: "0,00% of the bits are different (0 of 256). Longest unchanged bit sequence: Offset 0, length 256." There are also checkboxes for "Underline longest unchanged bit sequences" and "Underline longest changed bit sequences".

Symmetric-Key Cryptography - Advanced Encryption Standard (AES)

- Defined in each of:
 - FIPS PUB 197: Advanced Encryption Standard (AES).
 - ISO/IEC 18033-3: Block ciphers.
- Based on a design principle known as a substitution–permutation network.
- A fixed block size of 128 bits, and a key size of 128, 192, or 256 bits.
- The key size specifies the number of transformation rounds that convert the plaintext into the ciphertext:
 - 10 rounds for 128-bit keys.
 - 12 rounds for 192-bit keys.
 - 14 rounds for 256-bit keys.

Symmetric-Key Cryptography - Advanced Encryption Standard (AES)

High-level description of the algorithm [edit]

1. KeyExpansion – round keys are derived from the cipher key using the [AES key schedule](#). AES requires a separate 128-bit round key block for each round plus one more.
2. Initial round key addition:
 1. AddRoundKey – each byte of the state is combined with a byte of the round key using [bitwise xor](#).
3. 9, 11 or 13 rounds:
 1. SubBytes – a [non-linear substitution](#) step where each byte is replaced with another according to a [lookup table](#).
 2. ShiftRows – a [transposition](#) step where the last three rows of the state are shifted cyclically a certain number of steps.
 3. MixColumns – a [linear mixing](#) operation which operates on the columns of the state, combining the four bytes in each column.
 4. AddRoundKey
4. Final round (making 10, 12 or 14 rounds in total):
 1. SubBytes
 2. ShiftRows
 3. AddRoundKey

Cryptography Elements used in Blockchains

Symmetric-Key Cryptography - Advanced Encryption Standard (AES)

AES key schedule for a 128-bit key

Also define RotWord as a one-byte left circular shift:^[note 6]

$$\text{RotWord}([b_0 \ b_1 \ b_2 \ b_3]) = [b_1 \ b_2 \ b_3 \ b_0]$$

and SubWord as an application of the AES S-box to each of the four bytes of the word:

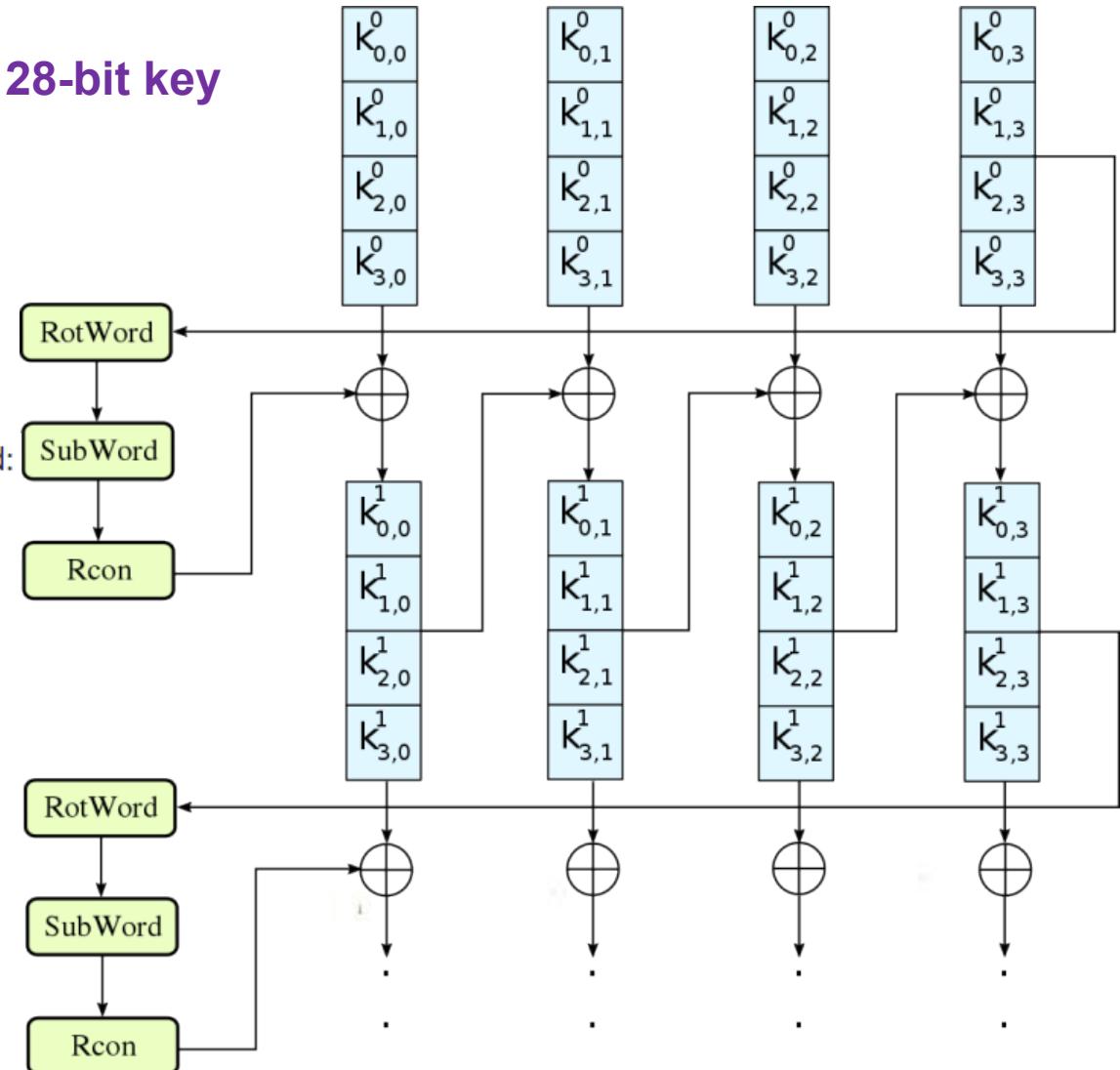
$$\text{SubWord}([b_0 \ b_1 \ b_2 \ b_3]) = [S(b_0) \ S(b_1) \ S(b_2) \ S(b_3)]$$

The round constant $rcon_i$ for round i of the key expansion is the 32-bit word:

$$rcon_i = [rc_i \ 00_{16} \ 00_{16} \ 00_{16}]$$

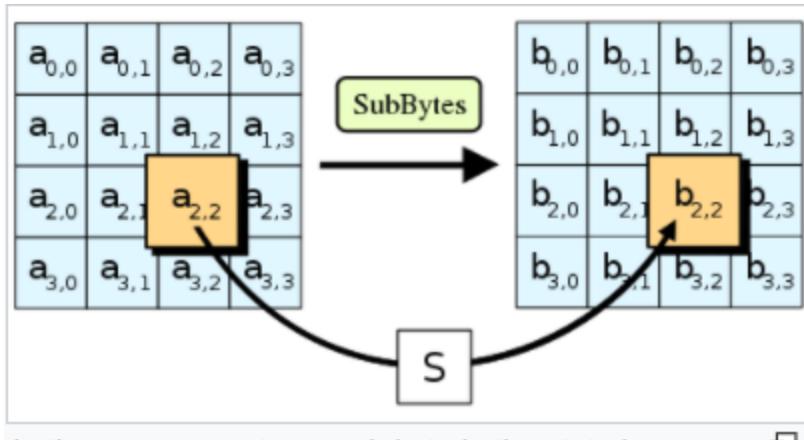
where rc_i is an eight-bit value defined as:

$$rc_i = \begin{cases} 1 & \text{if } i = 1 \\ 2 \cdot rc_{i-1} & \text{if } i > 1 \text{ and } rc_{i-1} < 80_{16} \\ (2 \cdot rc_{i-1}) \oplus 11B_{16} & \text{if } i > 1 \text{ and } rc_{i-1} \geq 80_{16} \end{cases}$$

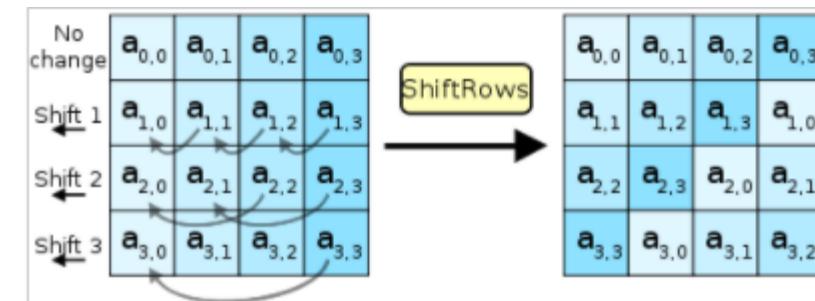


Cryptography Elements used in Blockchains

Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



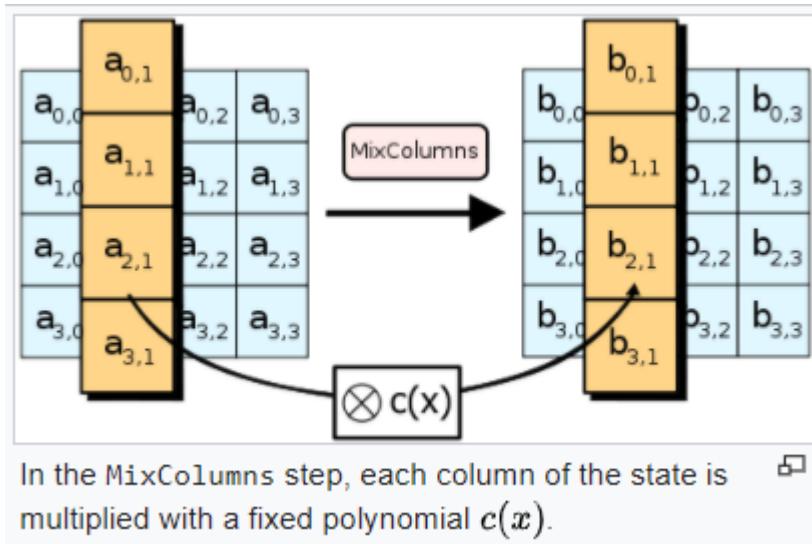
In the SubBytes step, each byte in the state is replaced with its entry in a fixed 8-bit lookup table, S ; $b_{ij} = S(a_{ij})$.



In the ShiftRows step, bytes in each row of the state are shifted cyclically to the left. The number of places each byte is shifted differs incrementally for each row.

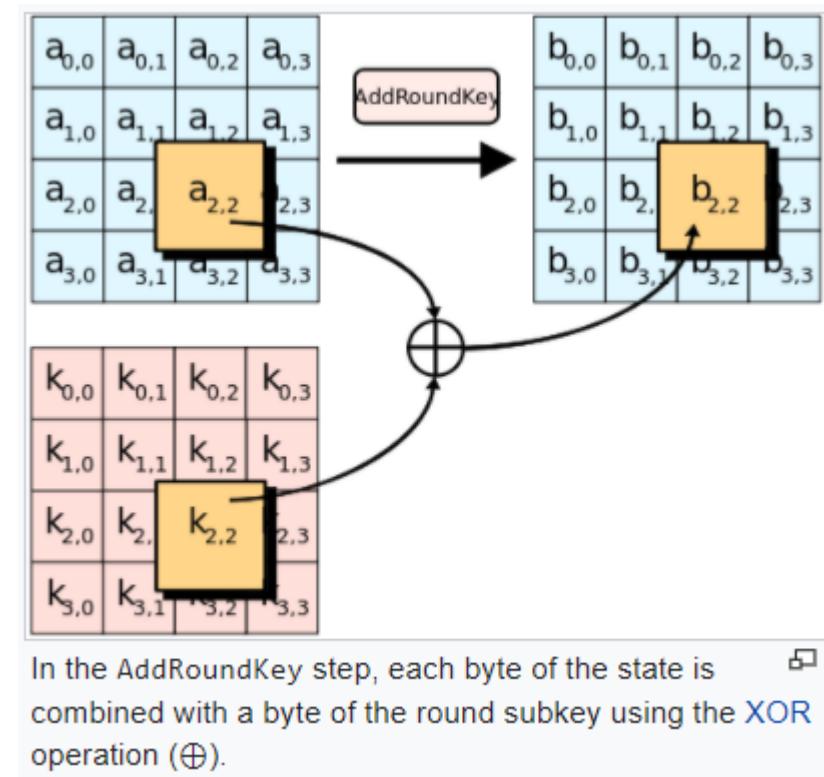
Cryptography Elements used in Blockchains

Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



Multiplication is modulo irreducible polynomial

$$x^8 + x^4 + x^3 + x + 1.$$



Cryptography Elements used in Blockchains

Symmetric-Key Cryptography - Advanced Encryption Standard (AES)

```
Rijndael(State,CipherKey) {  
    KeyExpansion(CipherKey,ExpandedKey) ;  
  
    AddRoundKey(State,ExpandedKey);  
  
    for( i=1 ; i<Nr ; i++ )  
        Round(State,ExpandedKey + Nb*i);  
    FinalRound(State,ExpandedKey + Nb*Nr);  
}  
}  
  
//funcție folosită doar dacă Nk<=6  
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)]) {  
    for(i = 0; i < Nk; i++)  
        W[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);  
  
    for(i = Nk; i < Nb * (Nr + 1); i++)  
    {  
        temp = W[i - 1];  
        if (i % Nk == 0)  
            temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];  
        else if (i % Nk == 4) temp = SubByte(temp);  
        W[i] = W[i - Nk] ^ temp;  
    }  
}
```

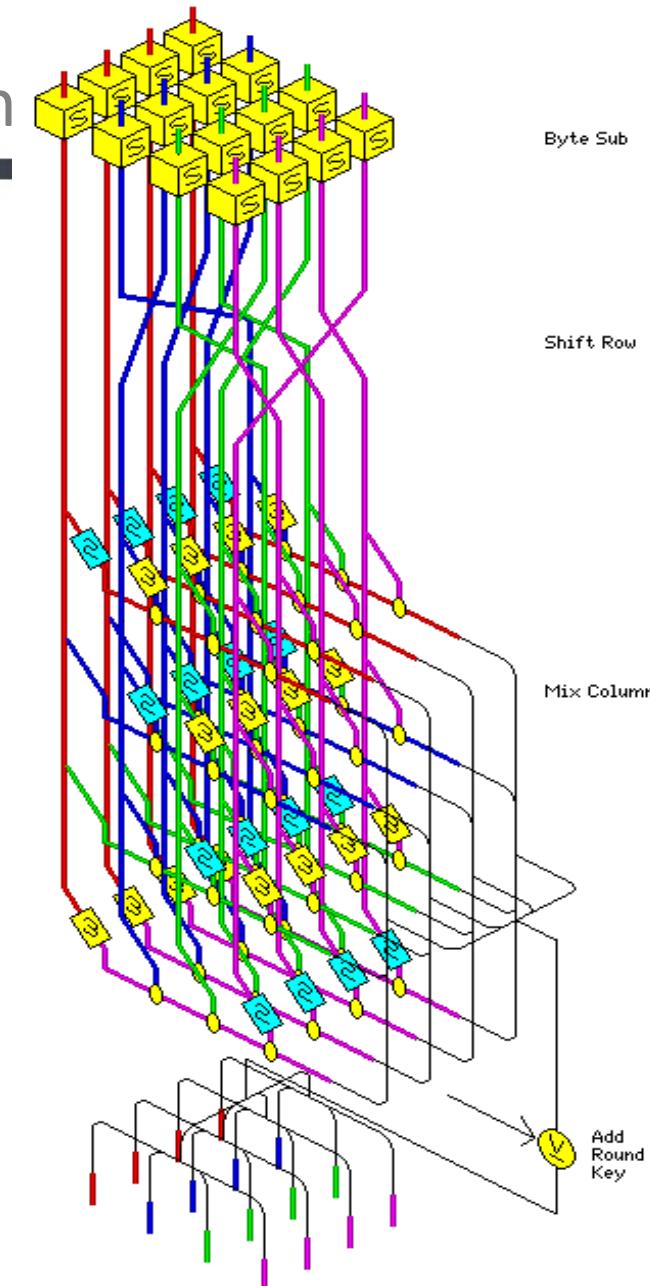
```
/*  
//funcție folosită doar dacă Nk>6  
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)]) {  
    for(i = 0; i < Nk; i++)  
        W[i] = (key[4*i],key[4*i+1],key[4*i+2],key[4*i+3]);  
  
    for(i = Nk; i < Nb * (Nr + 1); i++)  
    {  
        temp = W[i - 1];  
        if (i % Nk == 0)  
            temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];  
        else if (i % Nk == 4) temp = SubByte(temp);  
        W[i] = W[i - Nk] ^ temp;  
    }  
}/*
```

Symmetric-Key Cryptography - Advanced Encryption Standard (AES)

```
Round(State, RoundKey) {  
    ByteSub(State);  
    ShiftRow(State);  
    MixColumn(State);  
    AddRoundKey(State, RoundKey);  
}
```

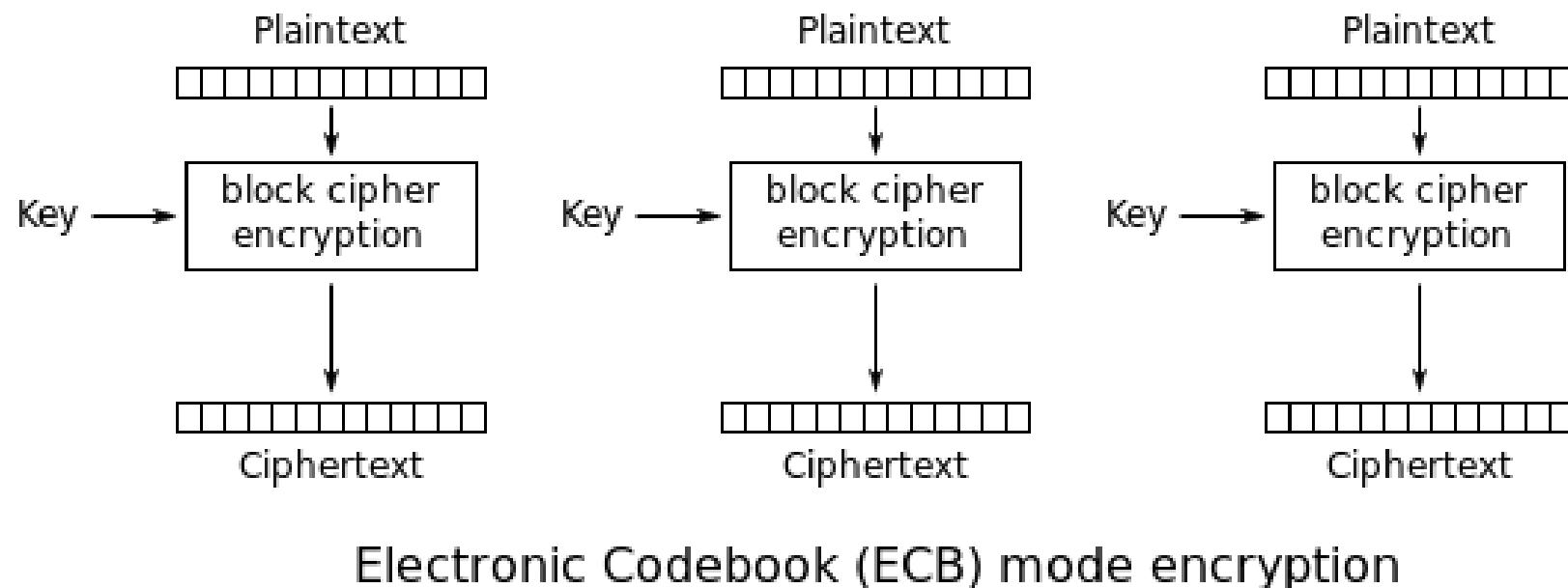
```
FinalRound(State, RoundKey) {  
    ByteSub(State) ;  
    ShiftRow(State) ;  
    AddRoundKey(State, RoundKey);  
}
```

```
AddRoundKey(State, ExpandedKey) {  
    State = (State ^ ExpandedKey);  
}
```

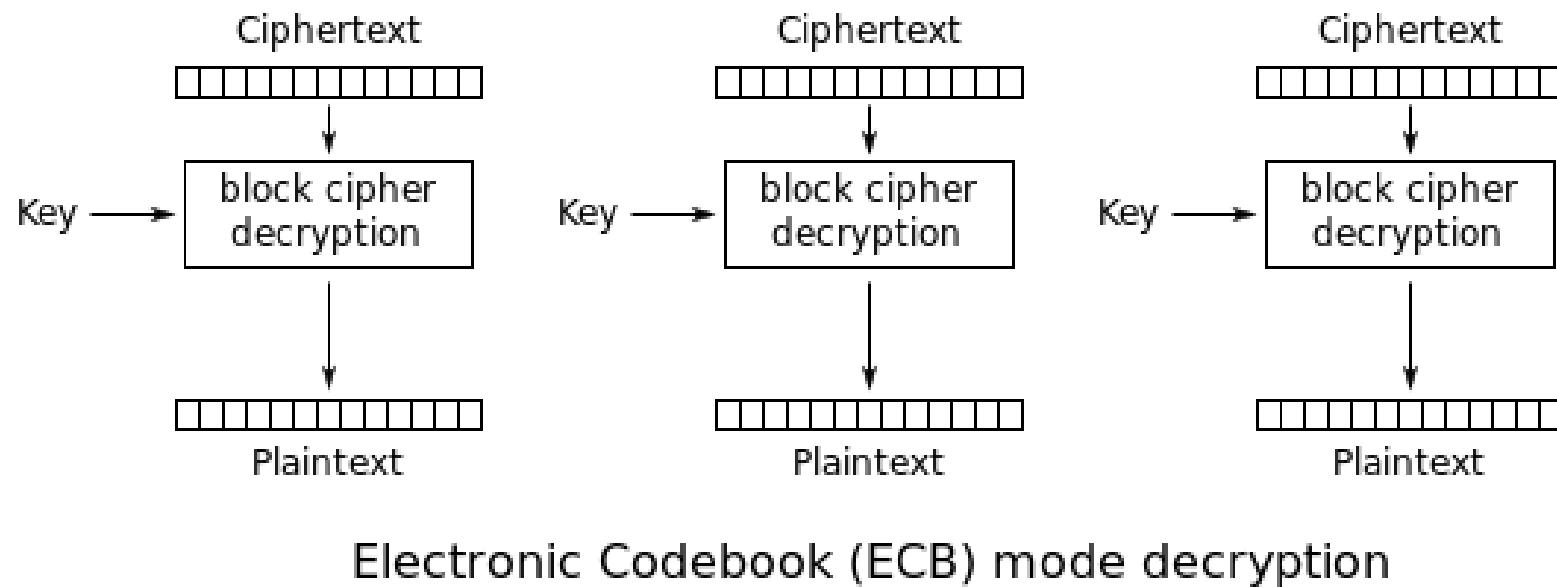


A round in Rijndael algorithm realized by the **Round(State, RoundKey)** function

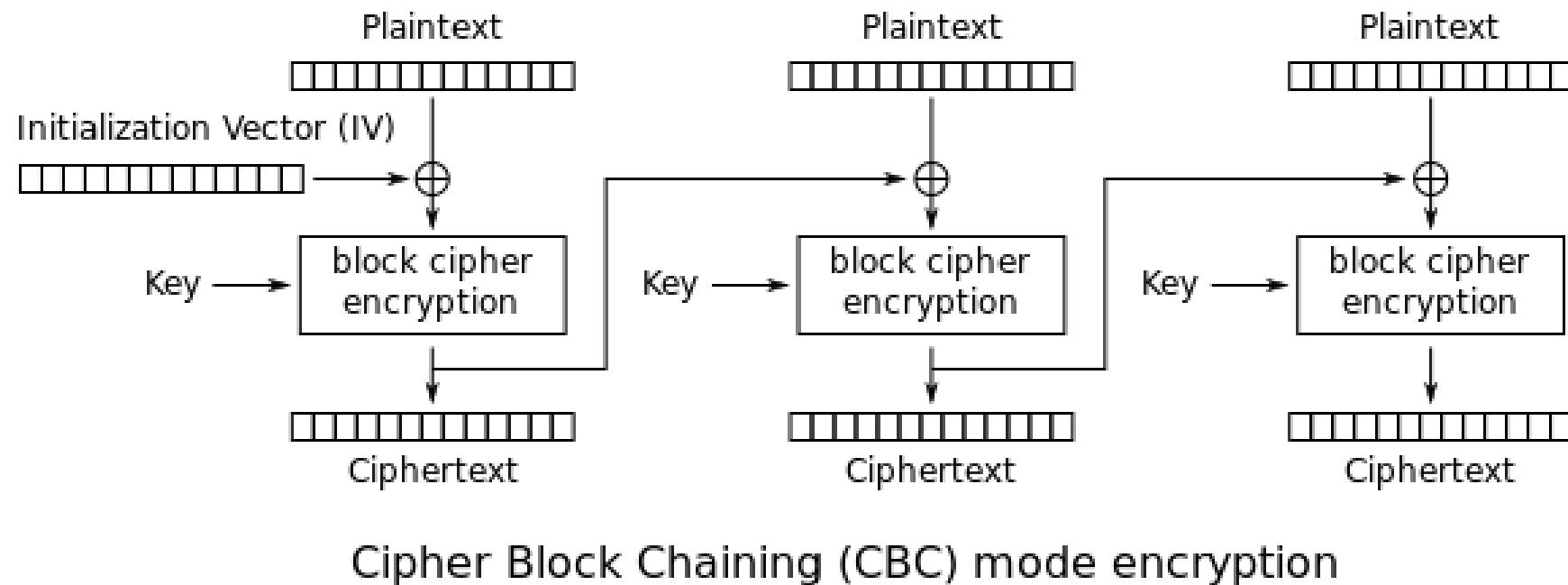
Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



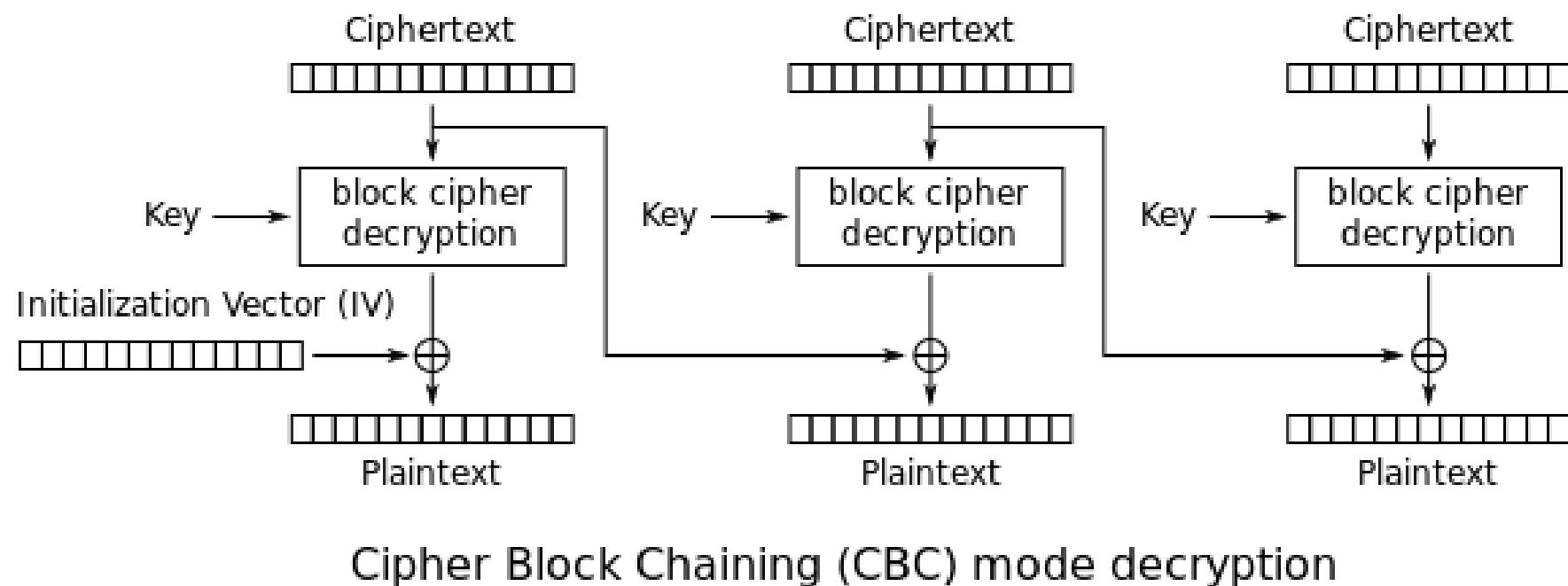
Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



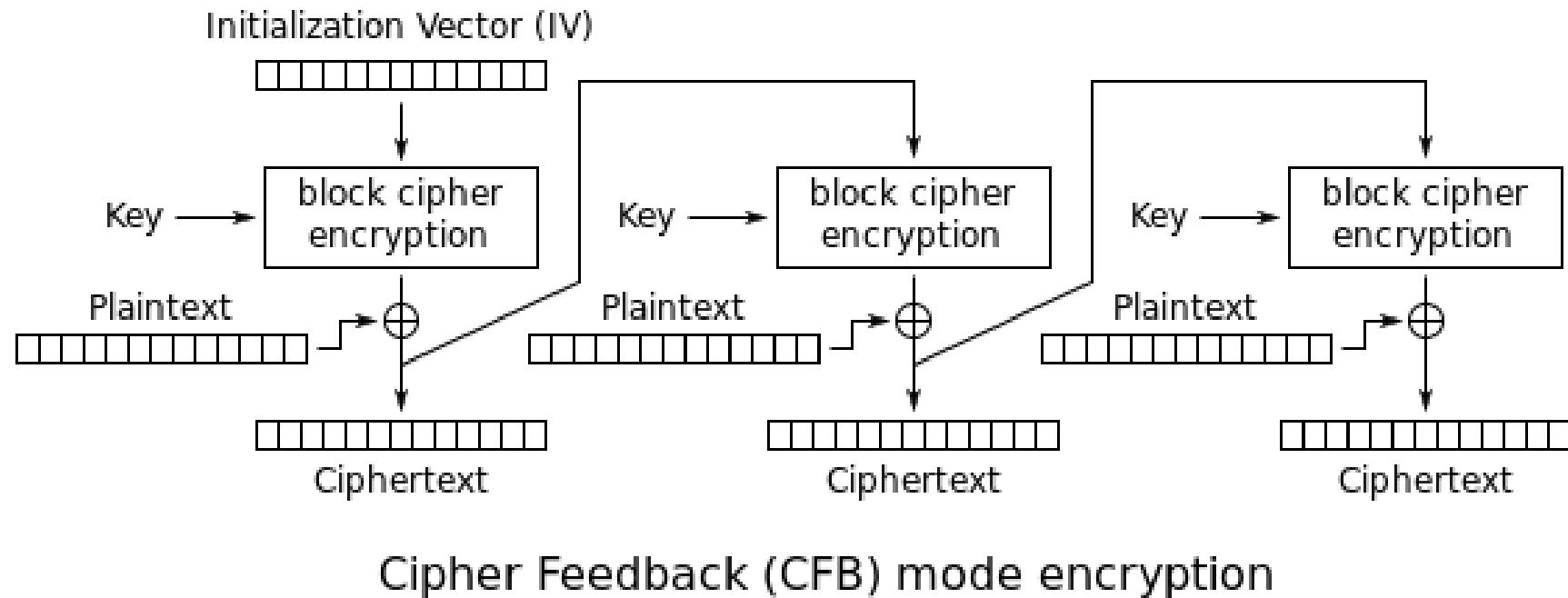
Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



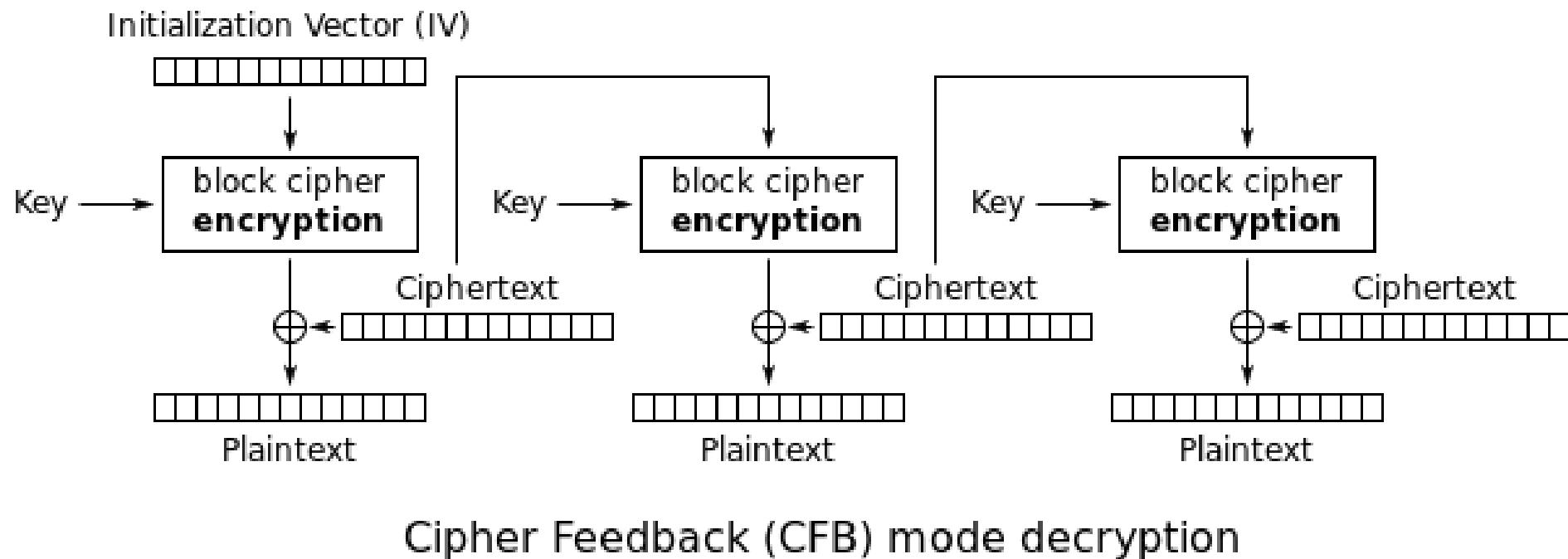
Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



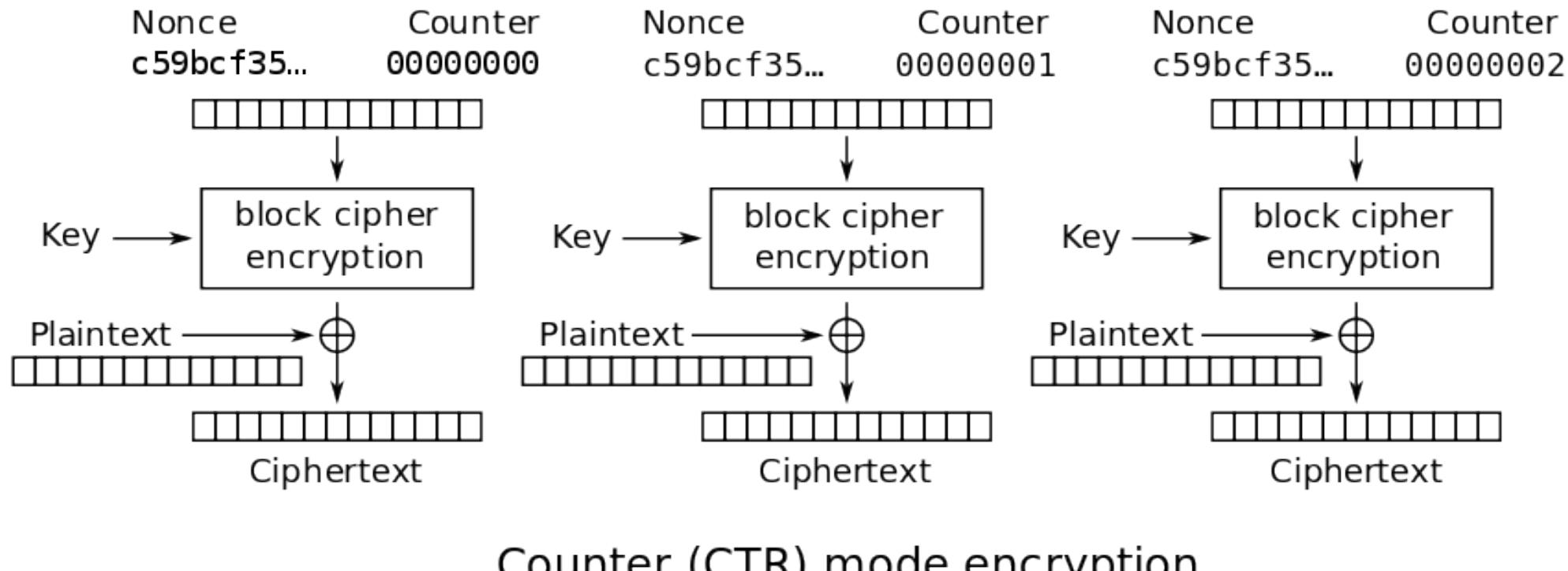
Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



Symmetric-Key Cryptography - Advanced Encryption Standard (AES)

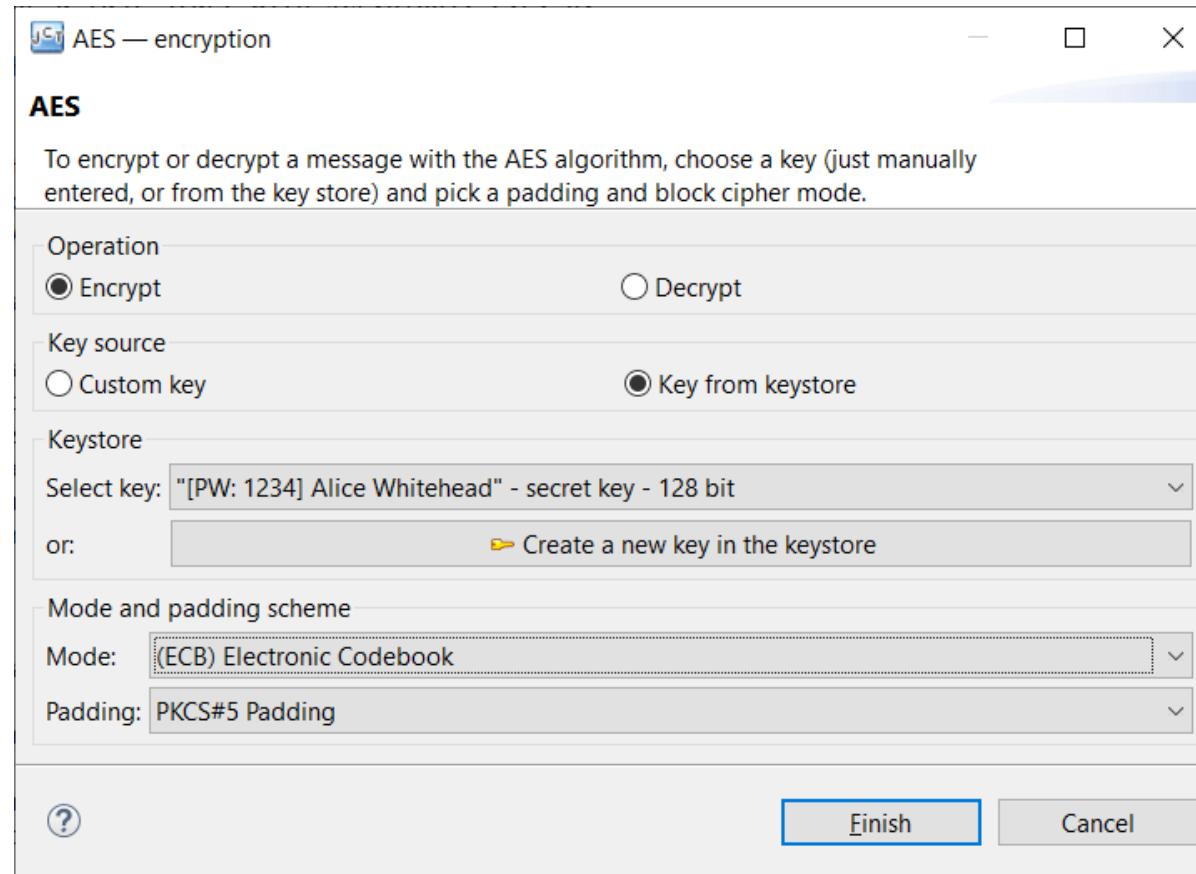


Symmetric-Key Cryptography - Advanced Encryption Standard (AES)



Cryptography Elements used in Blockchains

JCrypTool DEMO – AES Encryption



Asymmetric-Key Cryptography - Rivest-Shamir-Adleman (RSA)

- First public-key cryptosystems.
- Widely used for encryption and signing.
- RSA key generation:
 - 1 - Generate two primes p and q and compute their product N .
 - 2 - Pick e such that $\gcd(e, \phi(N)) = 1$, where $\phi(n) = (p - 1)(q - 1)$.
 - 3 - Compute d such that $d = e^{-1} \bmod \phi(N)$.

Public Key = $\langle e, N \rangle$

Private Key = $\langle d, N \rangle$

Asymmetric-Key Cryptography - Rivest-Shamir-Adleman (RSA)

- RSA encryption / decryption:

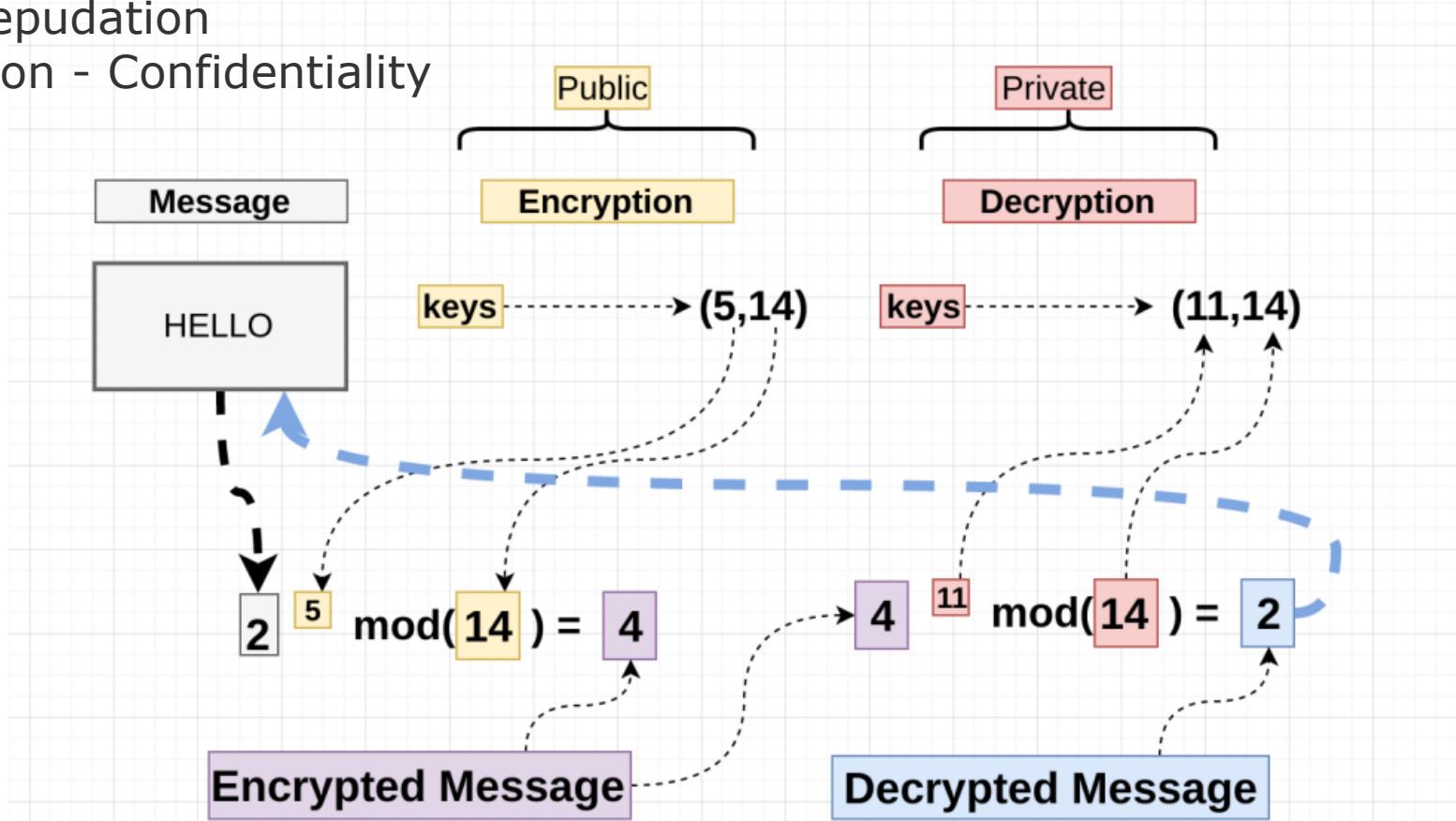
Encryption: Given a plaintext M and the public key $\langle e, N \rangle$, compute the ciphertext $C = M^e \text{ mod } N$.

Decryption: Given a ciphertext C and the private key $\langle d, N \rangle$, compute the plaintext $M = C^d \text{ mod } N$.

Cryptography Elements used in Blockchains

Asymmetric-Key Cryptography - Rivest-Shamir-Adleman (RSA)

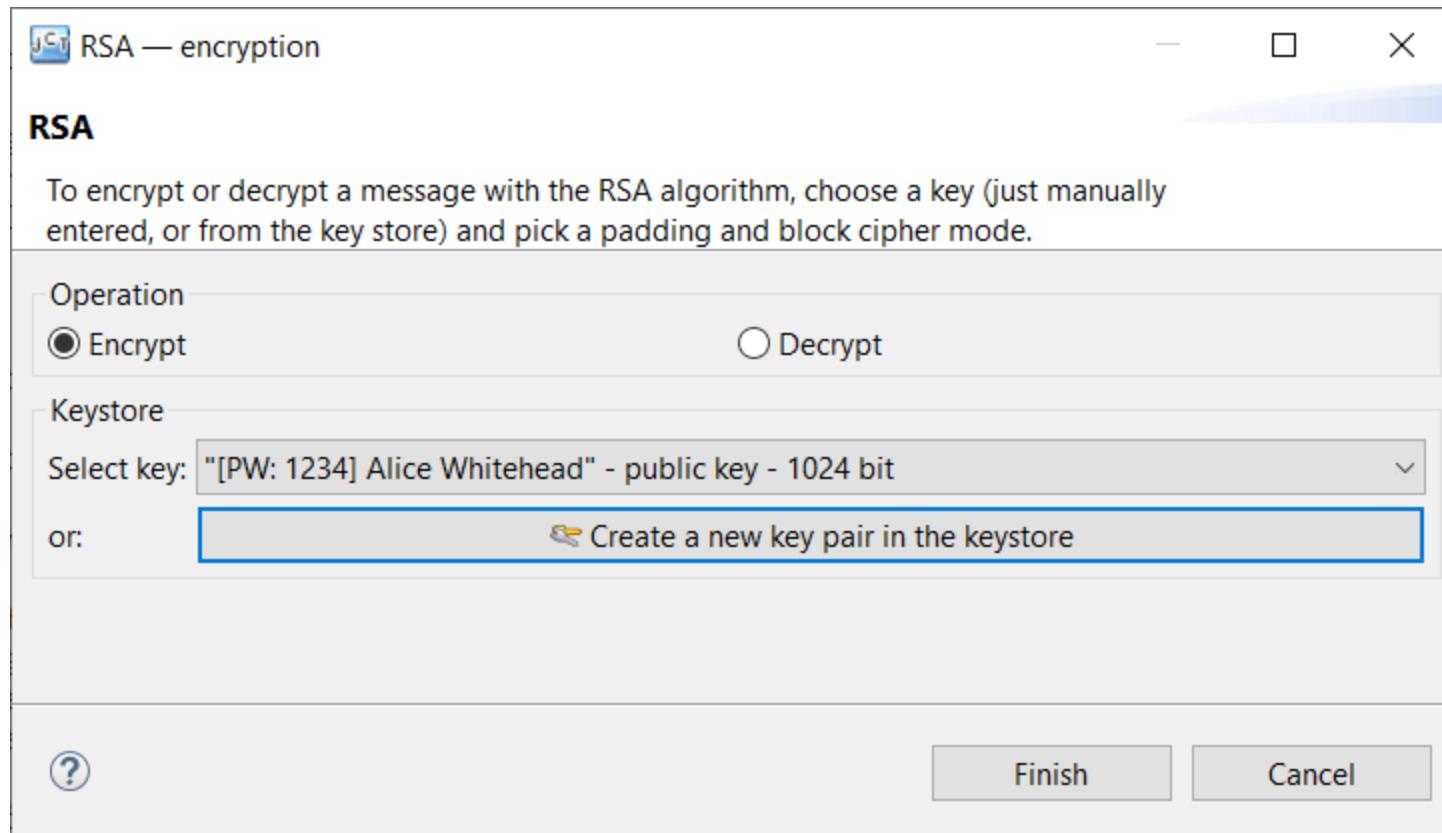
- How does RSA work:
 - Signature – Non- Repudation
 - Encryption/Decryption - Confidentiality



Cryptography Elements used in Blockchains

JCrypTool DEMO – RSA Encryption

Default Perspective > Algorithms > Asymmetric > RSA



Cryptography Elements used in Blockchains

JCrypTool DEMO – RSA Encryption

Default Perspective > Visuals > Extended RSA Cryptosystem

Extended RSA Cryptosystem (including identities and multi-prime RSA)

This visualization demonstrates the usage of RSA for message encryption. You can incorporate different identities, like „Alice Whitehat”, and encrypt messages for other identities like „Bob Whitehat” which only he is able to decrypt again. „Bob Whitehat” can do the same with messages for „Alice Whitehat”. You can also add other identities to see how RSA message encryption works with multiple participants.

Identity management

Create new identity Show/Hide identity Delete identity

Alice Whitehat Bob Whitehat

Actions:

- Encrypt and send message
- Receive and decrypt message
- Manage keys
- Attack public key

Encrypt and send message
Subject:

Message: Encrypted message (hexadecimal representation):

Recipient:

Select key:

Encrypt message Send message

Explanations

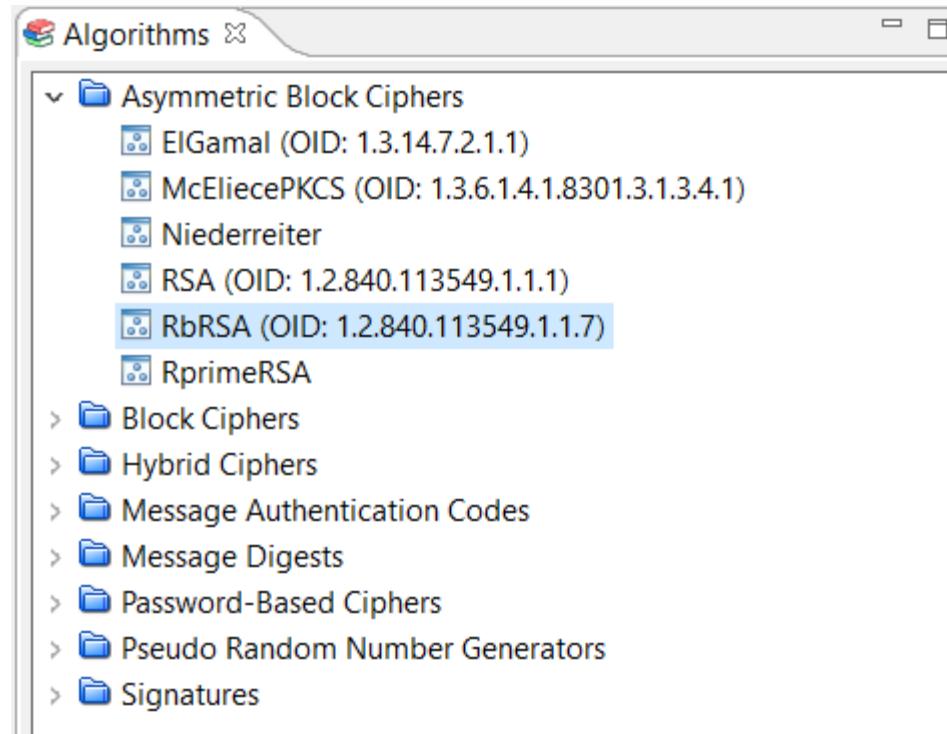
Action: Receive and decrypt message

For decrypting a message, the parameters N and d are needed. To get more information about the parameters, see the tab „My keys“ in „Manage keys“.

How to encrypt and send a message:

- 1) Select an arbitrary message from the message store.
- 2) Select your private key and enter the password for that key.
- 3) Press the button „Decrypt message“. The message can only be decrypted correctly if the right key was chosen.
- 4) Press the button „Delete message“ to delete the encrypted message from the message store (only available for real recipient).

JCrypTool DEMO – RSA Encryption Algorithms Perspective > Algorithms tab



Asymmetric-Key Cryptography - Elliptic-curves

- Based on the algebraic structure of elliptic curves over finite fields.
- Allows smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security.
- Cryptographic schemes:

Several discrete logarithm-based protocols have been adapted to elliptic curves, replacing the group $(\mathbb{Z}_p)^\times$ with an elliptic curve:

- The [Elliptic Curve Diffie–Hellman](#) (ECDH) key agreement scheme is based on the [Diffie–Hellman](#) scheme,
- The [Elliptic Curve Integrated Encryption Scheme](#) (ECIES), also known as Elliptic Curve Augmented Encryption Scheme or simply the Elliptic Curve Encryption Scheme,
- The [Elliptic Curve Digital Signature Algorithm](#) (ECDSA) is based on the [Digital Signature Algorithm](#),
- The deformation scheme using Harrison's p-adic Manhattan metric,
- The [Edwards-curve Digital Signature Algorithm](#) (EdDSA) is based on [Schnorr signature](#) and uses [twisted Edwards curves](#),
- The [ECMQV](#) key agreement scheme is based on the [MQV](#) key agreement scheme,
- The [ECQV](#) implicit certificate scheme.

Asymmetric-Key Cryptography - Elliptic-curves

- Standard documents for defining standard curves (named curves):
 - NIST, Recommended Elliptic Curves for Government Use 
 - SECG, SEC 2: Recommended Elliptic Curve Domain Parameters 
 - ECC Brainpool (RFC 5639 ), ECC Brainpool Standard Curves and Curve Generation 

Asymmetric-Key Cryptography - Elliptic-curves

Domain parameters

- Agreement on all the elements defining the elliptic curve ---> the domain parameters of the scheme.
- They are (p, a, b, G, n, h) for prime case, and (m, f, a, b, G, n, h) for binary case.
- The **field** is defined by **p** (prime) case and the pair of **m** and **f** (binary case).
- **Constants a** and **b** used in its defining equation.
- **Cyclic subgroup** is defined by its generator (base point) **G**.
- The **order** of **G**, that is the smallest positive number **n** such that $nG = \{\text{the point at infinity of the curve, and the identity element}\}$. It is normally prime.
- The number $h = \frac{1}{n}|E(\mathbb{F}_p)|$ is an integer.

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the real number space

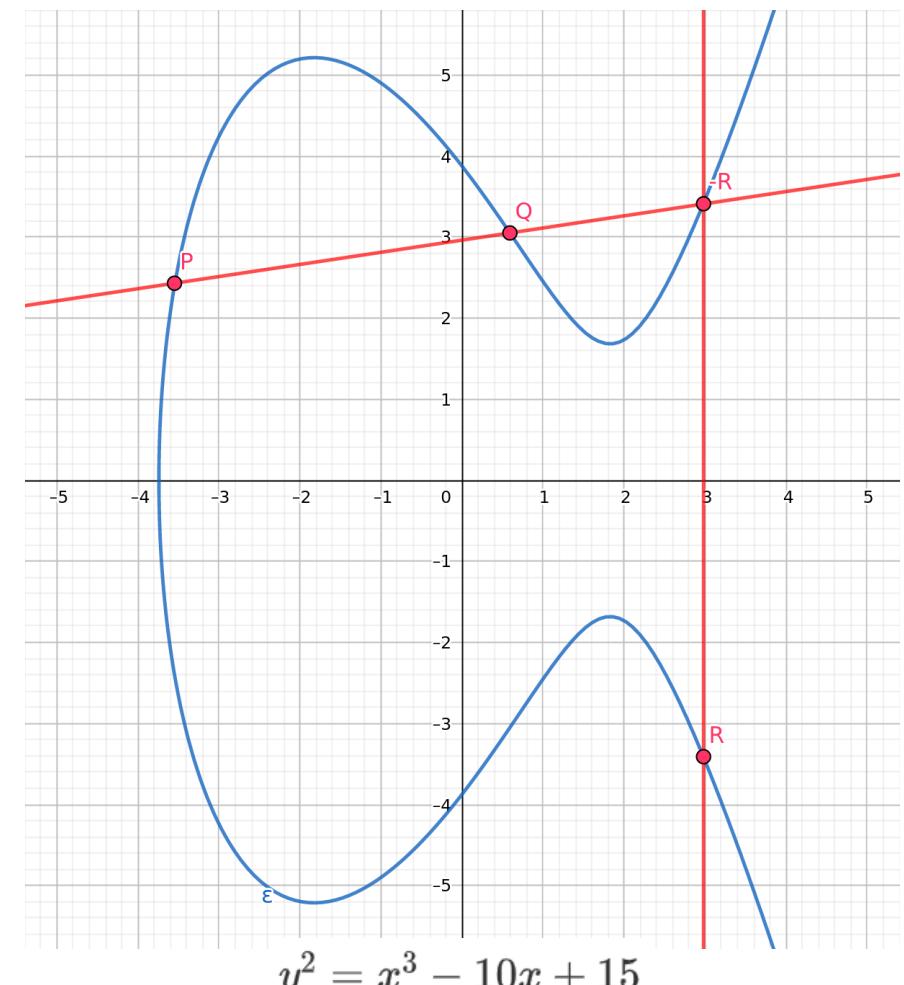
- Equation:

$$y^2 = x^3 + a \times x + b$$

Where a , b , x and y are real numbers.

- Point addition:

1. Select a point P and a point Q .
2. Draw a line through P and Q and continue until you cross the curve for the third time (P and Q being the first two times). This point is $-R$.
3. Point R is the point $-R$ with inverted y value (mirrored on the x axis).



Asymmetric-Key Cryptography - Elliptic-curves

Curves in the real number space

- Three simple formulas:

$$m = \frac{Q_y - P_y}{Q_x - P_x}$$

$$R_x = m^2 - P_x - Q_x$$

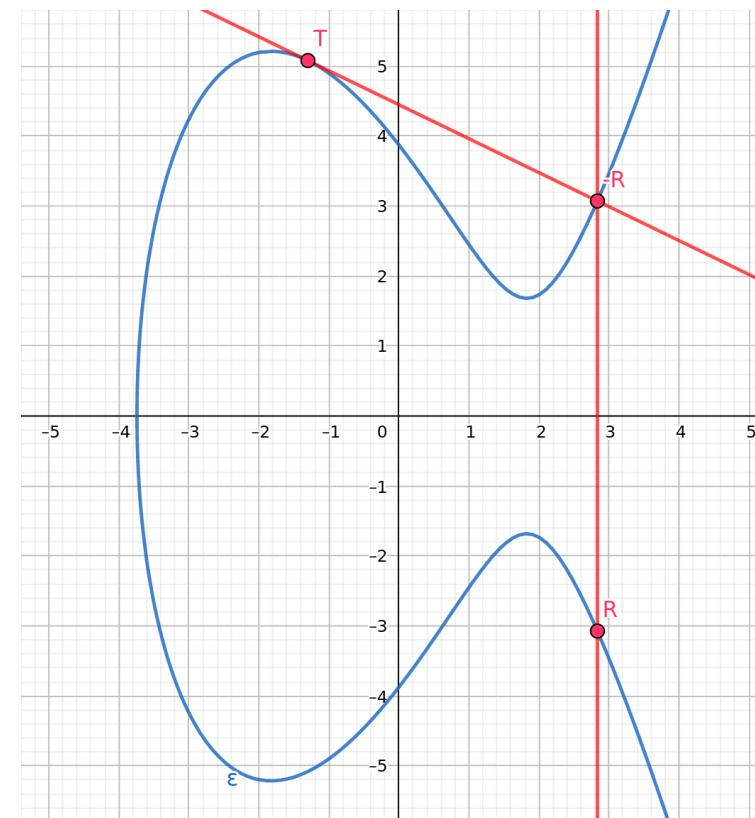
$$R_y = -P_y + s(P_x - R_x)$$

- Point doubling and multiplication:
 - Multiply points by adding them up.
 - Double a point calculate a tangent for that point

$$m = \frac{3x^2 + a}{2y}$$

$$R_x = m^2 - 2x$$

$$R_y = -y + m(x - R_x)$$



$$y^2 = x^3 - 10x + 15$$

Cryptography Elements used in Blockchains

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the real number space

- Three simple formulas:

$$m = \frac{Q_y - P_y}{Q_x - P_x}$$

$$R_x = m^2 - P_x - Q_x$$

$$R_y = -P_y + s(P_x - R_x)$$

- Point doubling and multiplication:

- Multiply points by adding them up.

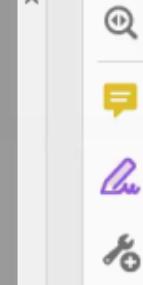
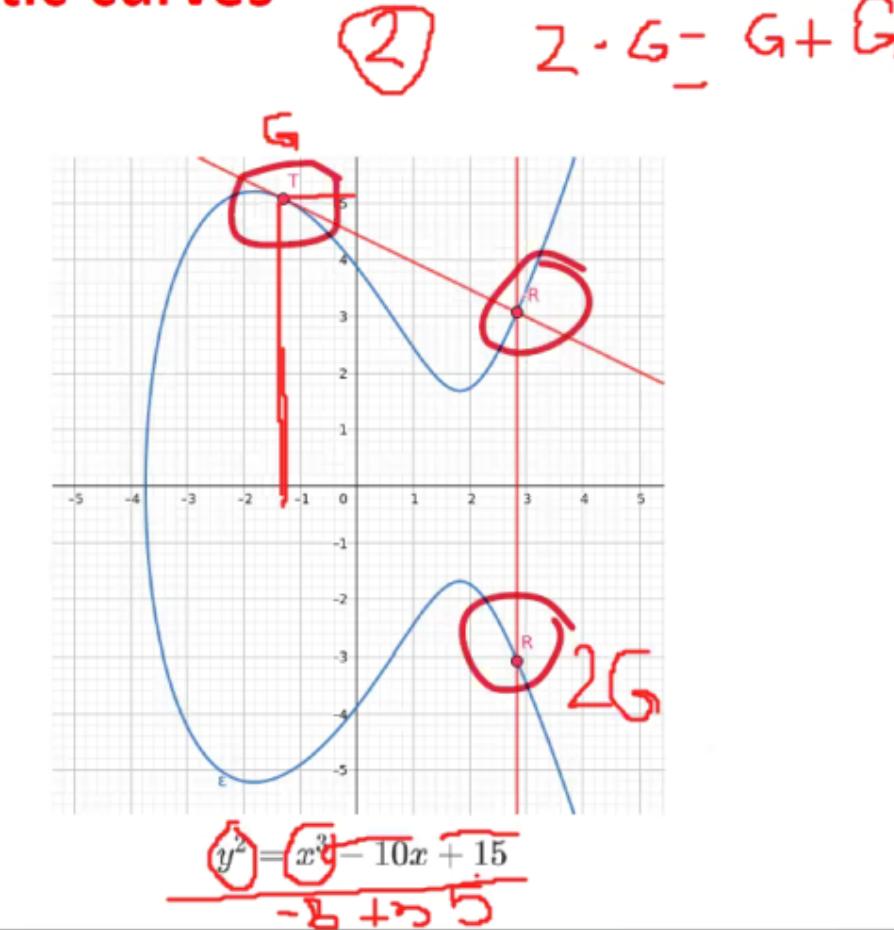
- Double a point calculate a tangent for that point

$$m = \frac{3x^2 + a}{2y}$$

$$R_x = m^2 - 2x$$

$$R_y = -y + m(x - R_x)$$

Source: <https://www.cryptool.org/en/jct/>



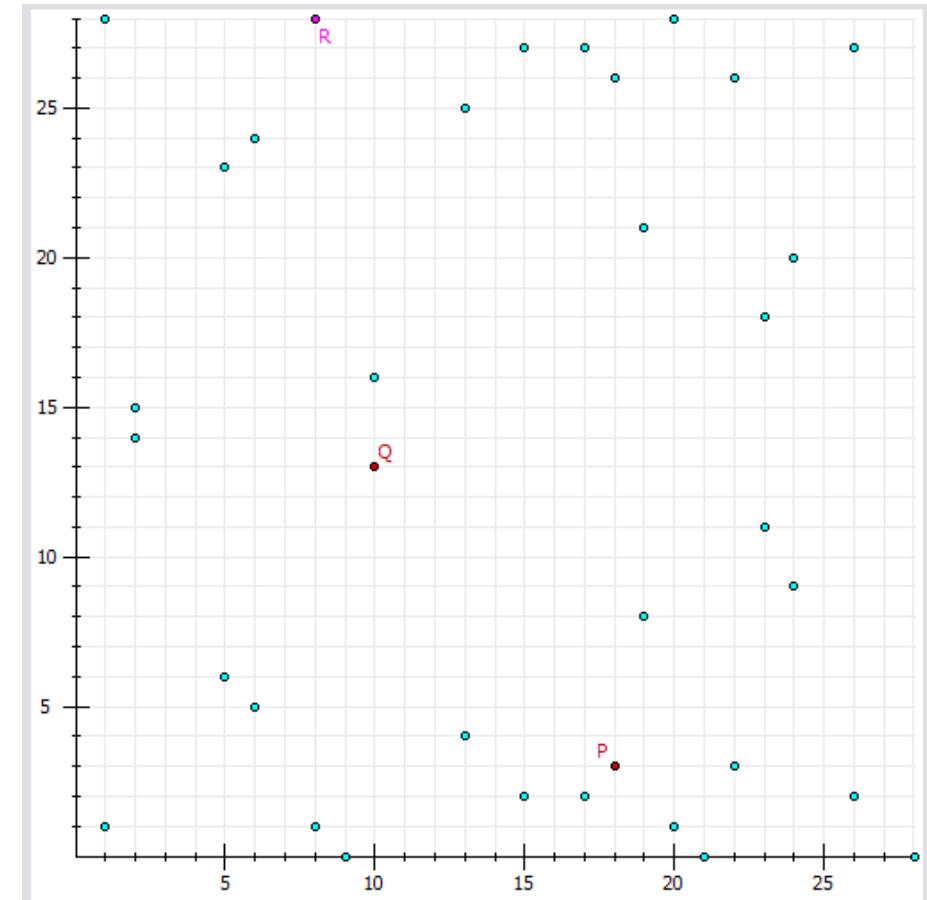
Asymmetric-Key Cryptography - Elliptic-curves

Curves over finite fields F_p

- Equation: $y^2 = x^3 + ax + b \pmod{p}$

where a and b are real numbers parameters of EC.

- There are a maximum of p^2 points on the curve.
- Each point (x, y) is element of the group of points in F_p where p is a prime number. Not all points are valid ones.
- The form of the elliptic curve is not recognizable anymore



An elliptic curve over F_{29}

Asymmetric-Key Cryptography - Elliptic-curves

Curves over finite fields \mathbb{F}_p

- Point addition:
 - Very similar to curves with real numbers
$$m = (Q_y - P_y) \cdot (Q_x - P_x)^{-1} \mod p$$
$$R_x = m^2 - P_x - Q_x \mod p$$
$$R_y = -P_y + m(P_x - R_x) \mod p$$
- Point doubling and multiplication:
 - Formulas for point doubling
$$m = (3x^2 + a) \cdot (2y)^{-1} \mod p$$
$$R_x = m^2 - 2x \mod p$$
$$R_y = -y + m(x - R_x) \mod p$$

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the field \mathbb{F}_{2^m}

- Equation $y^2 + xy = x^3 + ax^2 + b$.
- Additional parameters:
 - Polynomial / bit length m
 - a and b for the elliptic curve
 - Generator G : an irreducible primary polynomial with a length of $m + 1$. Irreducible polynomials have similar properties as prime numbers. They cannot be divided by other polynomials. G is used to generate points on the curve
 - Reduction polynomial $f(x)$: a polynomial through which is divided for the modulo calculation

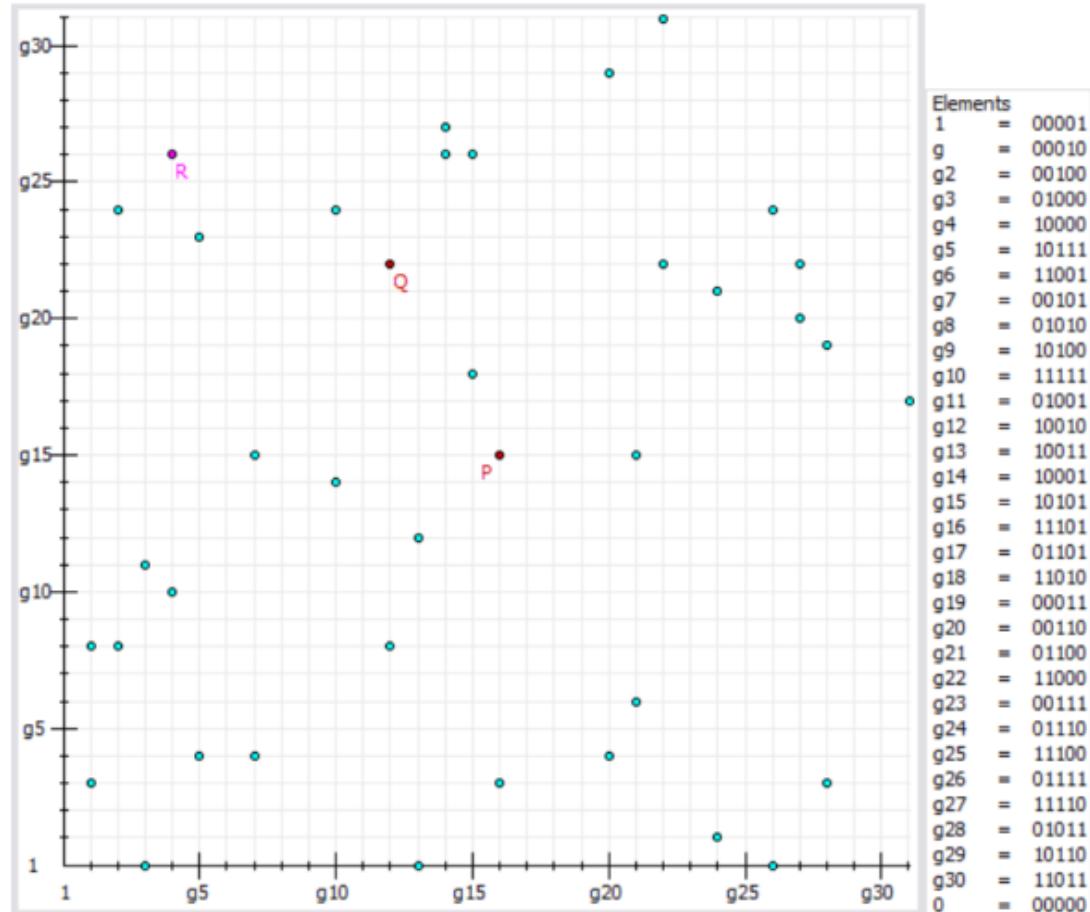
Cryptography Elements used in Blockchains

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the field \mathbb{F}_{2^m}

- Rules and definitions:

- Given a point $P = (x, y)$, its negative is $-P(x, y) := (x, x + y)$.
- There exists a neutral element O for which applies: $-O := O$, and for each point P of the elliptic curve applies $P + O := O + P := P$.



An elliptic curve over \mathbb{F}_{2^5} with the parameters:
 $y^2 + xy = x^3 + g9 \cdot x^2 + g3$ and the generator $G = 110111$



40 / 54



85%



Cryptography Elements used in Blockchains

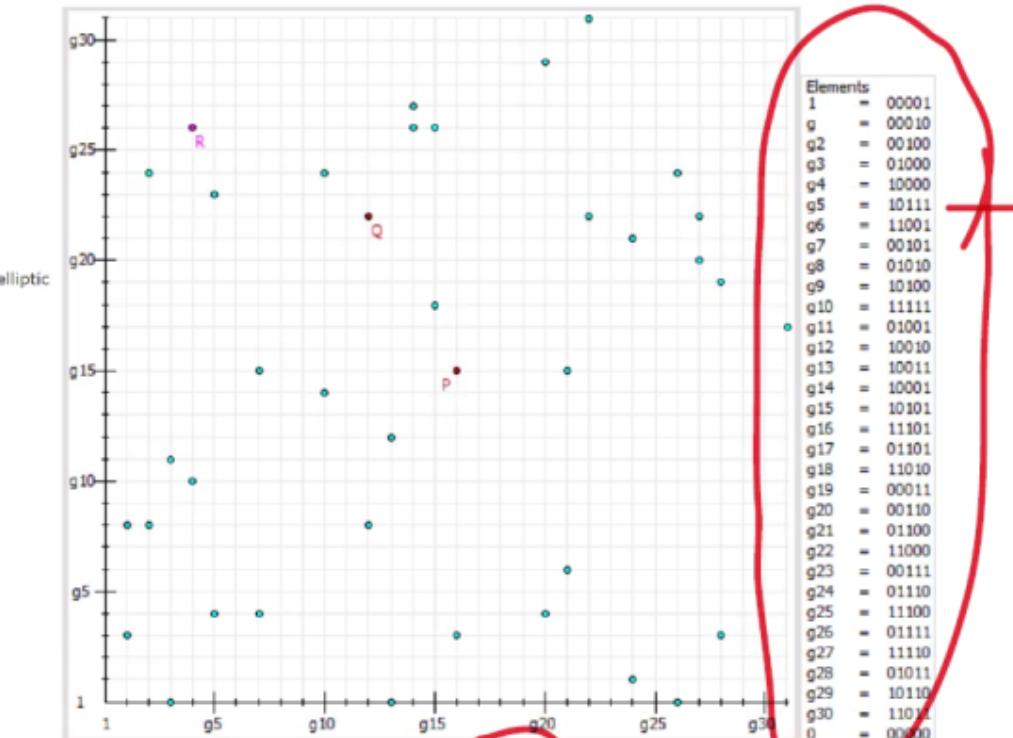
Asymmetric-Key Cryptography - Elliptic-curves

Curves in the field \mathbb{F}_{2^m}

- Rules and definitions:

- Given a point $P = (x, y)$, its negative is $-P(x, y) := (x, x + y)$.
- There exists a neutral element O for which applies: $-O := O$, and for each point P of the elliptic curve applies $P + O := O + P := P$.

0000 0010
0000 0100



An elliptic curve over \mathbb{F}_{2^5} with the parameters:

$$y^2 + xy = x^3 + g9 \cdot x^2 + g3 \text{ and the generator } G = 110111$$

Source: <https://www.cryptool.org/en/jct/>

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the field \mathbb{F}_{2^m}

- Calculations
 - **Addition** - simple bitwise XOR, also called "carryless addition"

	Polynomial	Bitstring
p	$x^8+x^4+x^2+1$	100010101
q	x^4+x+1	000010011
$p+q$	x^8+x^2+x	100000110

- **Subtraction** - exactly the same operation as the addition, a bitwise XOR

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the field \mathbb{F}_{2^m}

- Calculations
 - Multiplication - single term polynomials (monomial), left shift of k bits

Polynomial	Bitstring	Polynomial	Bitstring
$(x^8 + x^4 + x^2 + 1) \cdot x$	$100010101 \cdot 10$	$(x^8 + x^4 + x^2 + 1) \cdot x^4$	$100010101 \cdot 10000$
$(x^9 + x^5 + x^3 + x)$	1000101010	$(x^{12} + x^8 + x^6 + x^4)$	1000101010000

- Multiplication – for two polynomials results from the distributive law

Polynomial	Bitstring
$(x^8 + x^4 + x^2 + 1) \cdot (x^4 + x + 1) =$	$100010101 \cdot 10011$
$(x^8 + x^4 + x^2 + 1) \cdot x^4 +$	1000101010000
$(x^8 + x^4 + x^2 + 1) \cdot x +$	1000101010
$(x^8 + x^4 + x^2 + 1) \cdot 1$	100010101
<hr/>	
$(x^{12} + x^9 + x^6 + x^5 + x^3 + x^2 + x + 1)$	1001001101111

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the field \mathbb{F}_{2^m}

- Calculations
 - Division – done with remainder for applying normal polynomial division rules

Polynomial	Bitstring
$\begin{array}{r} (x^8 + x^4+x^2 + 1) / (x^4 + x + 1) = x^4 + x \\ \hline -(x^8+x^5+x^4) \\ x^5 + x^2 \\ - (x^5 + x^2+x) \\ \hline x+1 \text{ Remainder} \end{array}$	$\begin{array}{r} 100010101/10011 = 10010 \\ \hline -(10011) \\ 00010010 \\ - (10011) \\ 000011 \text{ Remainder} \end{array}$

Asymmetric-Key Cryptography - Elliptic-curves

Curves in the field \mathbb{F}_{2^m}

- Point addition

$$m = (Q_y + P_y) \cdot (Q_x + P_x)^{-1} \mod f(x)$$

$$R_x = m^2 + m + P_x + Q_x + a \mod f(x)$$

$$R_y = m(P_x + R_y) + R_x + P_y \mod f(x)$$

- Point doubling

$$m = P_x + P_y \mod f(x)$$

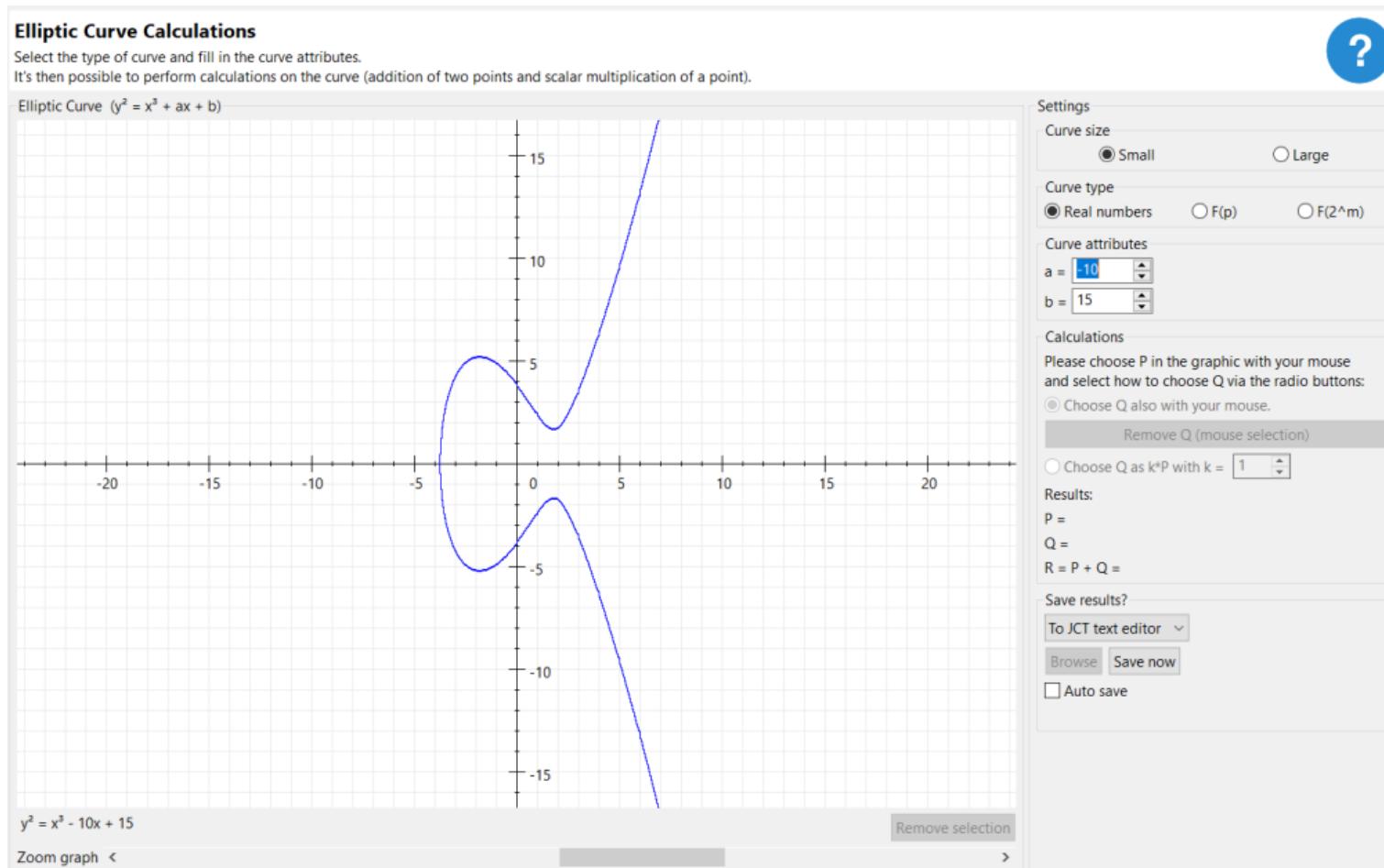
$$R_x = m^2 + m + a \mod f(x)$$

$$R_y = P_x^2 + R_x \cdot (m + 1) \mod f(x)$$

Cryptography Elements used in Blockchains

JCrypTool DEMO – ECC

Default Perspective > Visuals > Elliptic Curve Calculations



Digital Signature Schemes and Algorithms - DSA

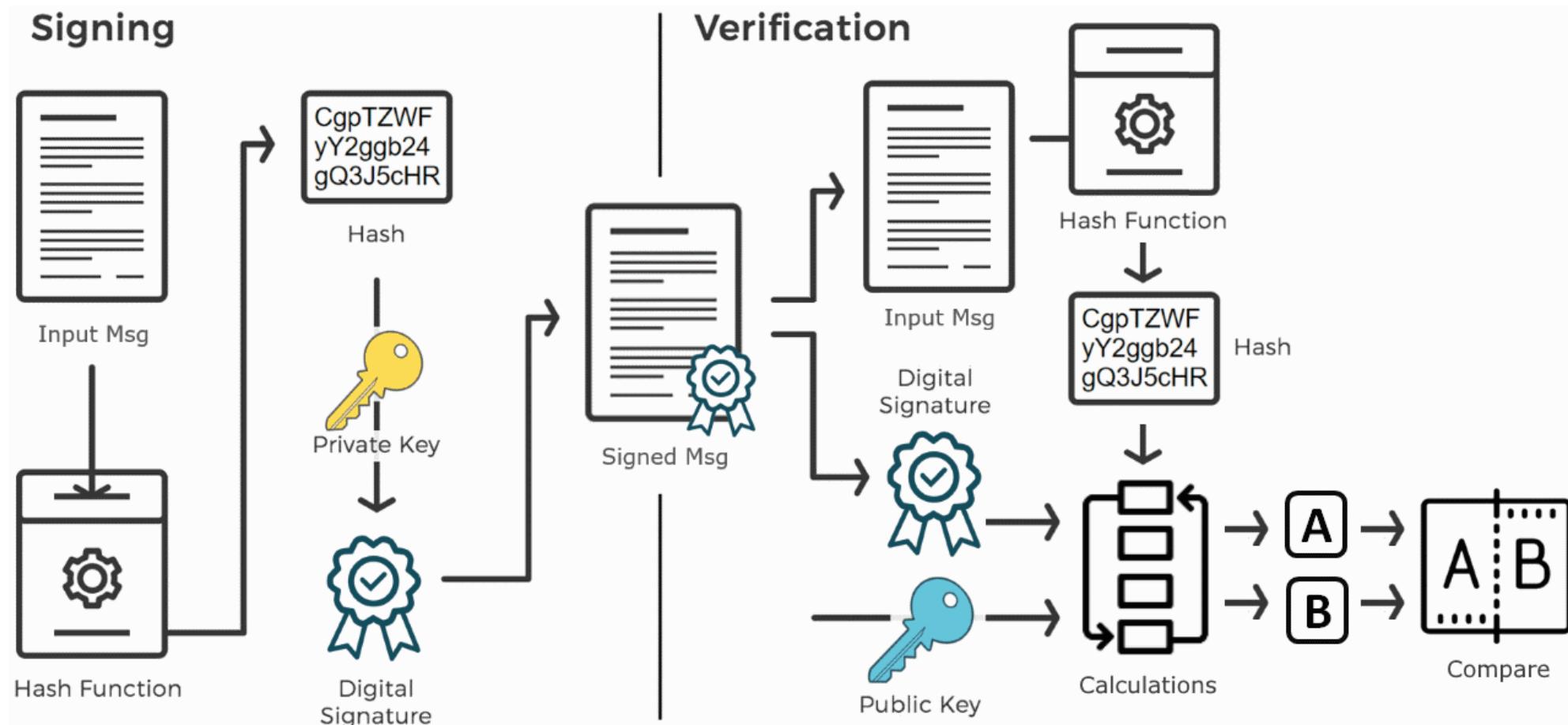
- A cryptographically secure standard for digital signatures (signing messages and signature verification).
- Based on the math of the modular exponentiations and discrete logarithms and the difficulty of the discrete logarithm problem (DLP).
- **DSA signing:**
 - Compute a **message hash**.
 - Generate a **random integer** k and computes the signature.
 - The **signature** is a pair of integers {r, s}, where r is computed from k and s is computed using the message hash + the private key exponent + the random number k.
 - Due to randomness, the signature is **non-deterministic**.

Digital Signature Schemes and Algorithms - DSA

- **DSA signature verification:**
 - Based on the message **hash** + the **public key exponent** + the **signature {r, s}**.

Cryptography Elements used in Blockchains

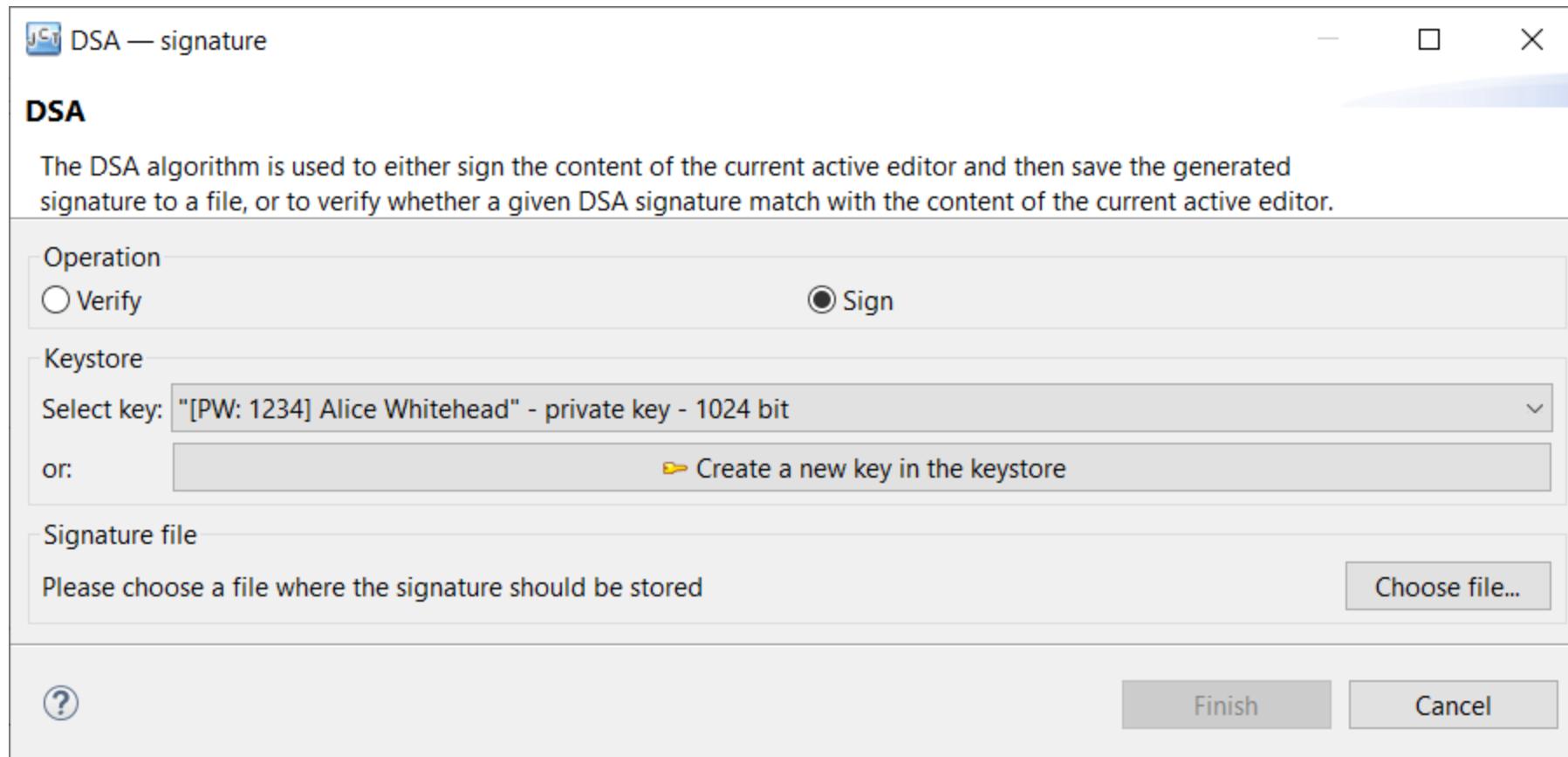
Digital Signature Schemes and Algorithms - DSA



Cryptography Elements used in Blockchains

JCrypTool DEMO – DSA

Default Perspective > Algorithms > Signature > DSA



Digital Signature Schemes and Algorithms - ECDSA

- Variant of the Digital Signature Algorithm (DSA) which uses elliptic curve cryptography.
- Needed EC parameters:

Parameter	
CURVE	the elliptic curve field and equation used
G	elliptic curve base point, a point on the curve that generates a subgroup of large prime order n
n	integer order of G , means that $n \times G = O$, where O is the identity element.
d_A	the private key (randomly selected)
Q_A	the public key (calculated by elliptic curve)
m	the message to send

Digital Signature Schemes and Algorithms - ECDSA

- Sign a message m by the sender:

1. Calculate $e = \text{HASH}(m)$. (Here HASH is a **cryptographic hash function**, such as **SHA-2**, with the output converted to an integer.)
2. Let z be the L_n leftmost bits of e , where L_n is the bit length of the group order n . (Note that z can be *greater* than n but not *longer*.^[1])
3. Select a **cryptographically secure random** integer k from $[1, n - 1]$.
4. Calculate the curve point $(x_1, y_1) = k \times G$.
5. Calculate $r = x_1 \bmod n$. If $r = 0$, go back to step 3.
6. Calculate $s = k^{-1}(z + rd_A) \bmod n$. If $s = 0$, go back to step 3.
7. The signature is the pair (r, s) . (And $(r, -s \bmod n)$ is also a valid signature.)

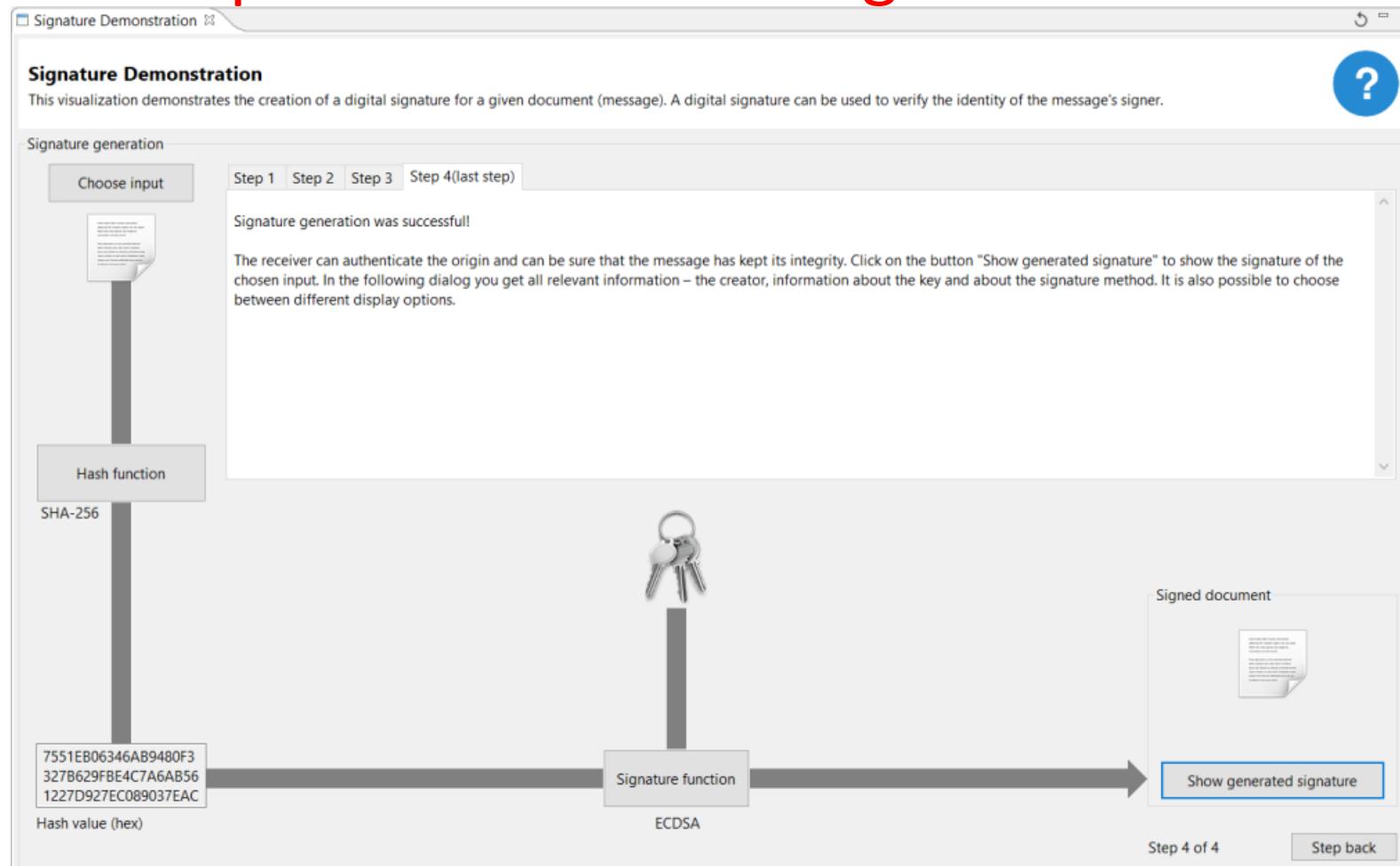
Digital Signature Schemes and Algorithms - ECDSA

- Verify signature by the receiver:
 - Check the public key is a valid elliptic curve point
 1. Check that Q_A is not equal to the identity element O , and its coordinates are otherwise valid
 2. Check that Q_A lies on the curve
 3. Check that $n \times Q_A = O$
 - Check the DSA signature
 1. Verify that r and s are integers in $[1, n - 1]$. If not, the signature is invalid.
 2. Calculate $e = \text{HASH}(m)$, where HASH is the same function used in the signature generation.
 3. Let z be the L_n leftmost bits of e .
 4. Calculate $u_1 = z s^{-1} \pmod{n}$ and $u_2 = r s^{-1} \pmod{n}$.
 5. Calculate the curve point $(x_1, y_1) = u_1 \times G + u_2 \times Q_A$. If $(x_1, y_1) = O$ then the signature is invalid.
 6. The signature is valid if $r \equiv x_1 \pmod{n}$, invalid otherwise.

Cryptography Elements used in Blockchains

JCrypTool DEMO – DSA

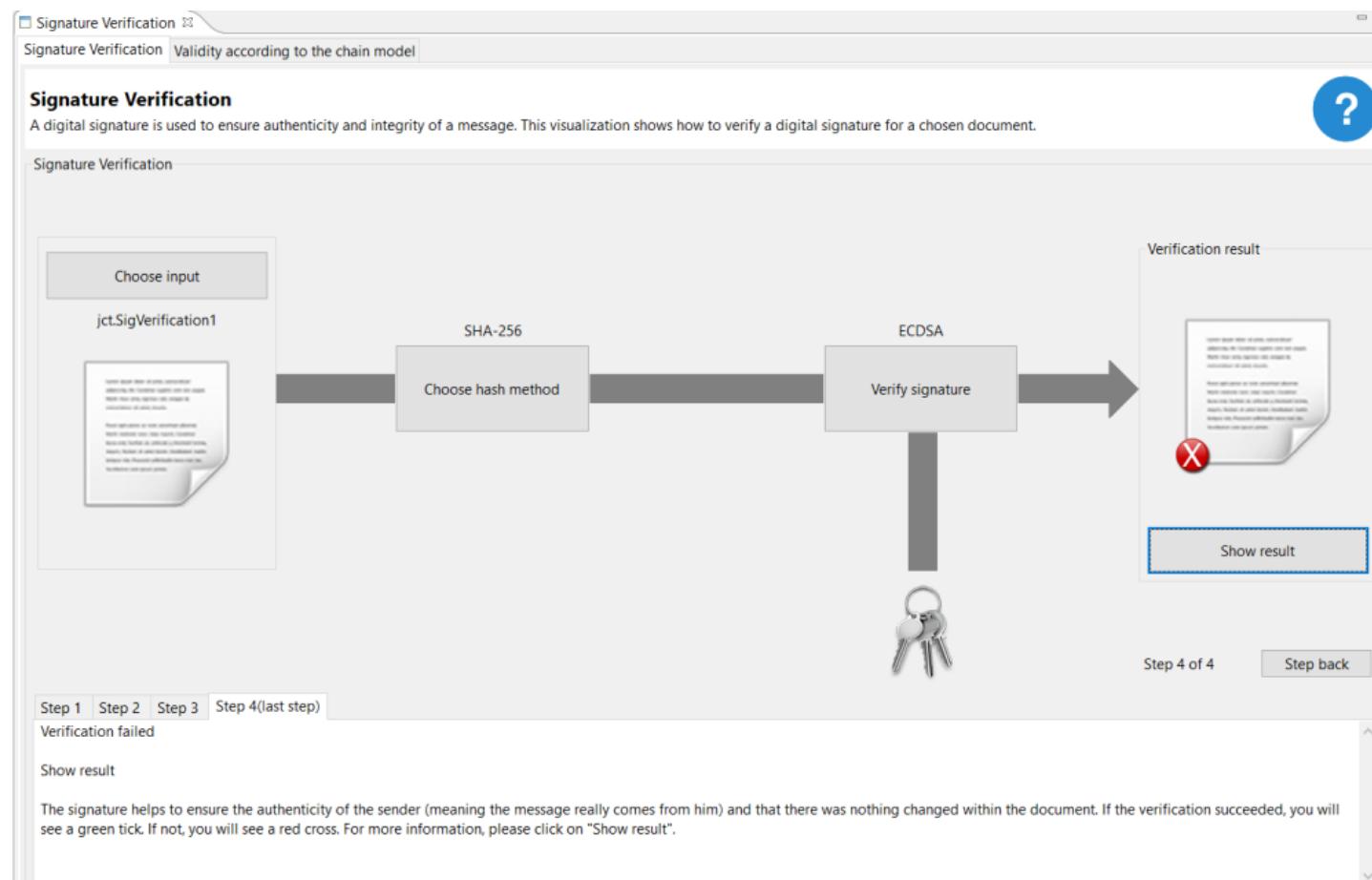
Default Perspective > Visuals > Signature Demonstration



Cryptography Elements used in Blockchains

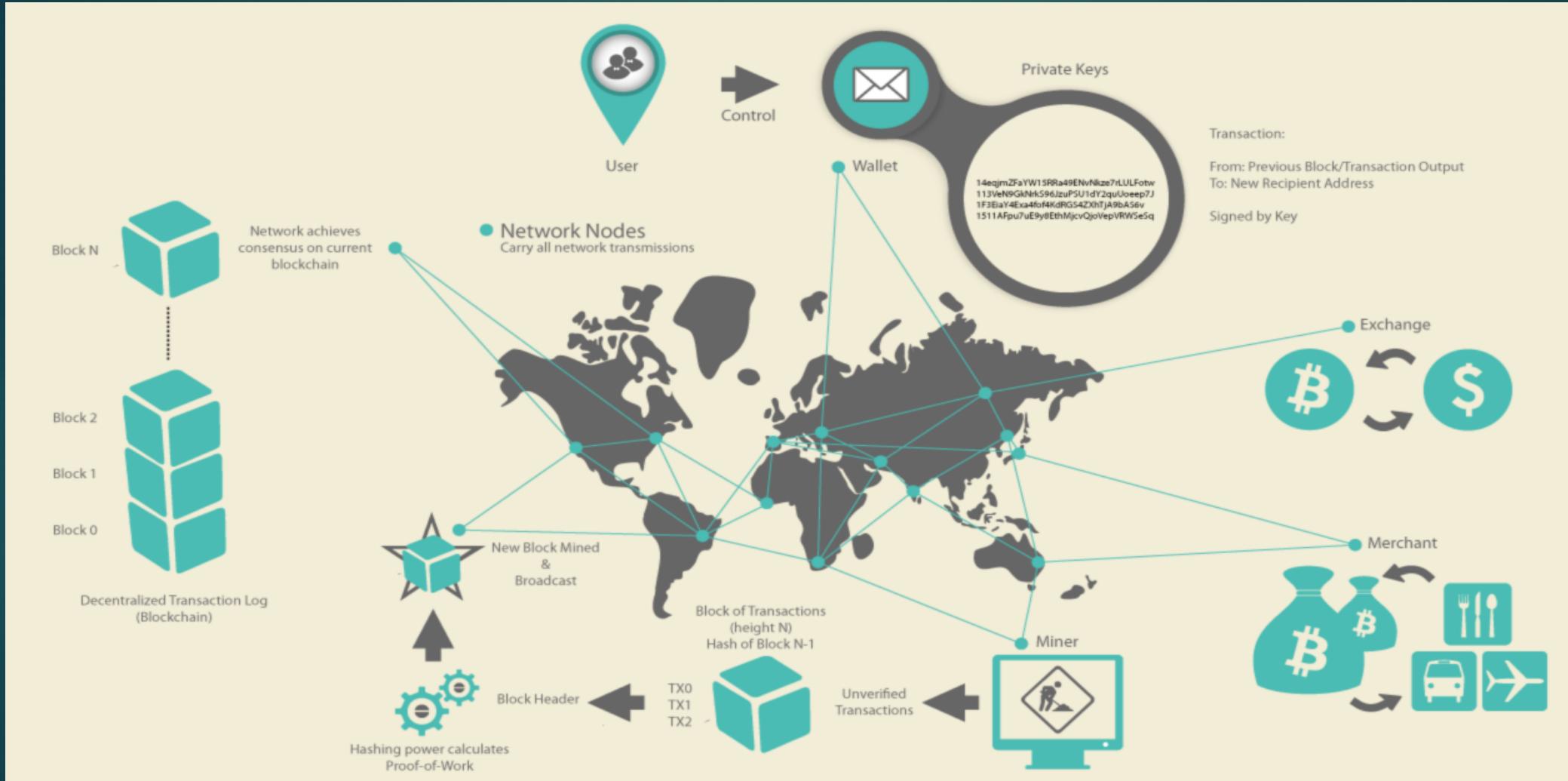
JCrypTool DEMO – DSA

Default Perspective > Visuals > Signature Verification



02. Bitcoin | Ethereum Blockchain Technology

77



Confidential – www.ism.ase.ro | www.acs.ase.ro

02. Bitcoin | Ethereum Blockchain Technology

78

Transaction View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fb8a57286c345c2f2

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)



1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA
- (Unspent) 0.015 BTC
1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK -
(Unspent) 0.0845 BTC

97 Confirmations

0.0995 BTC

Summary

Size 258 (bytes)

Received Time 2013-12-27 23:03:05

Included In 277316 (2013-12-27 23:11:54 +9
Blocks minutes)

Inputs and Outputs

Total Input 0.1 BTC

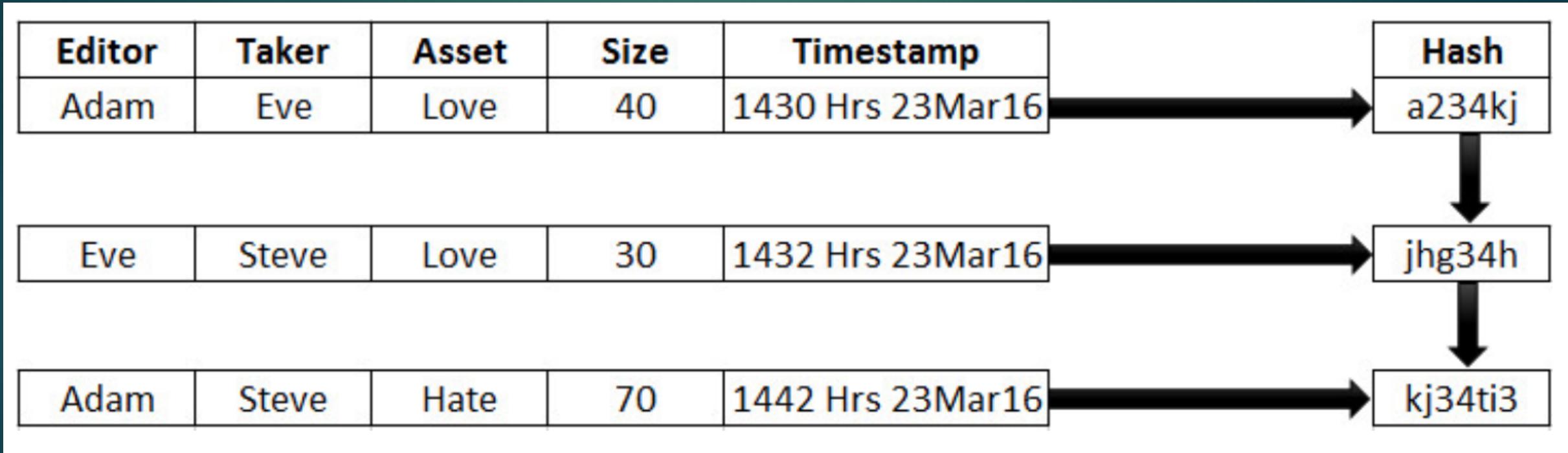
Total Output 0.0995 BTC

Fees 0.0005 BTC

Estimated BTC Transacted 0.015 BTC

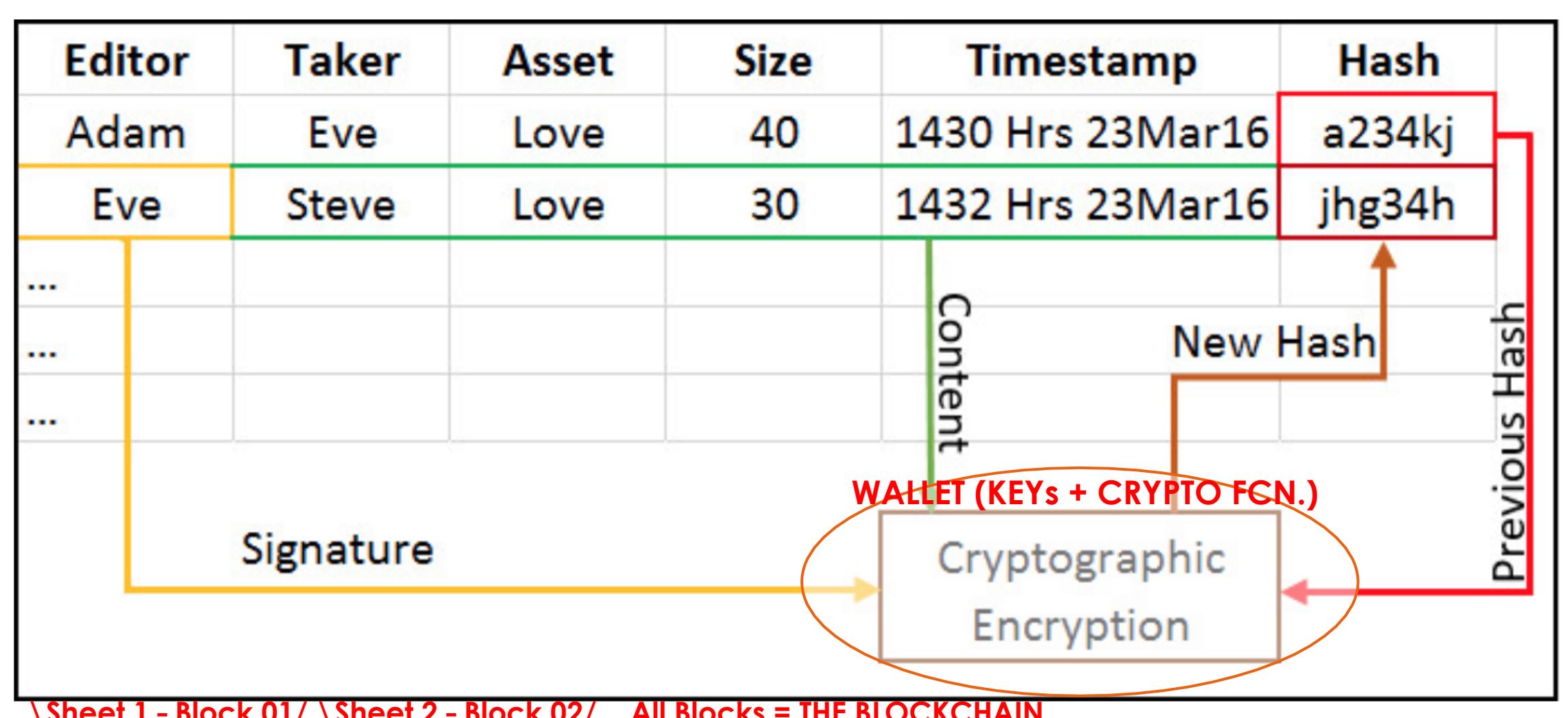
02. Bitcoin / Ethereum Blockchain Technology

Bitcoin / Ethereum != Blockchain; // They use blockchain technology



02. Bitcoin / Ethereum Blockchain Technology

((Bitcoin / Ethereum / crypto currencies) != Blockchain); // They use blockchain technology



\Sheet 1 - Block 01/ \Sheet 2 - Block 02/ ... All Blocks = THE BLOCKCHAIN

<https://www.linkedin.com/pulse/introduction-blockchain-bankers-excel-sheet-analogy-jayaram-firm>

02. Google Sheet DEMO

81

```
// https://docs.google.com/spreadsheets/
```

```
// Create Blank Sheet and Tools->Script Editor
```

```
//=getSha1(CONCATENATE(A4,B4,C4,D4,E4))
```

```
function getSha1(input) {
  var rawHash = Utilities.computeDigest(Utilities.DigestAlgorithm.SHA_1, input);
  var txtHash = "";

  for (var j = 0; j < rawHash.length; j++) {
    var hashVal = rawHash[j];

    if (hashVal < 0)
      hashVal += 256;

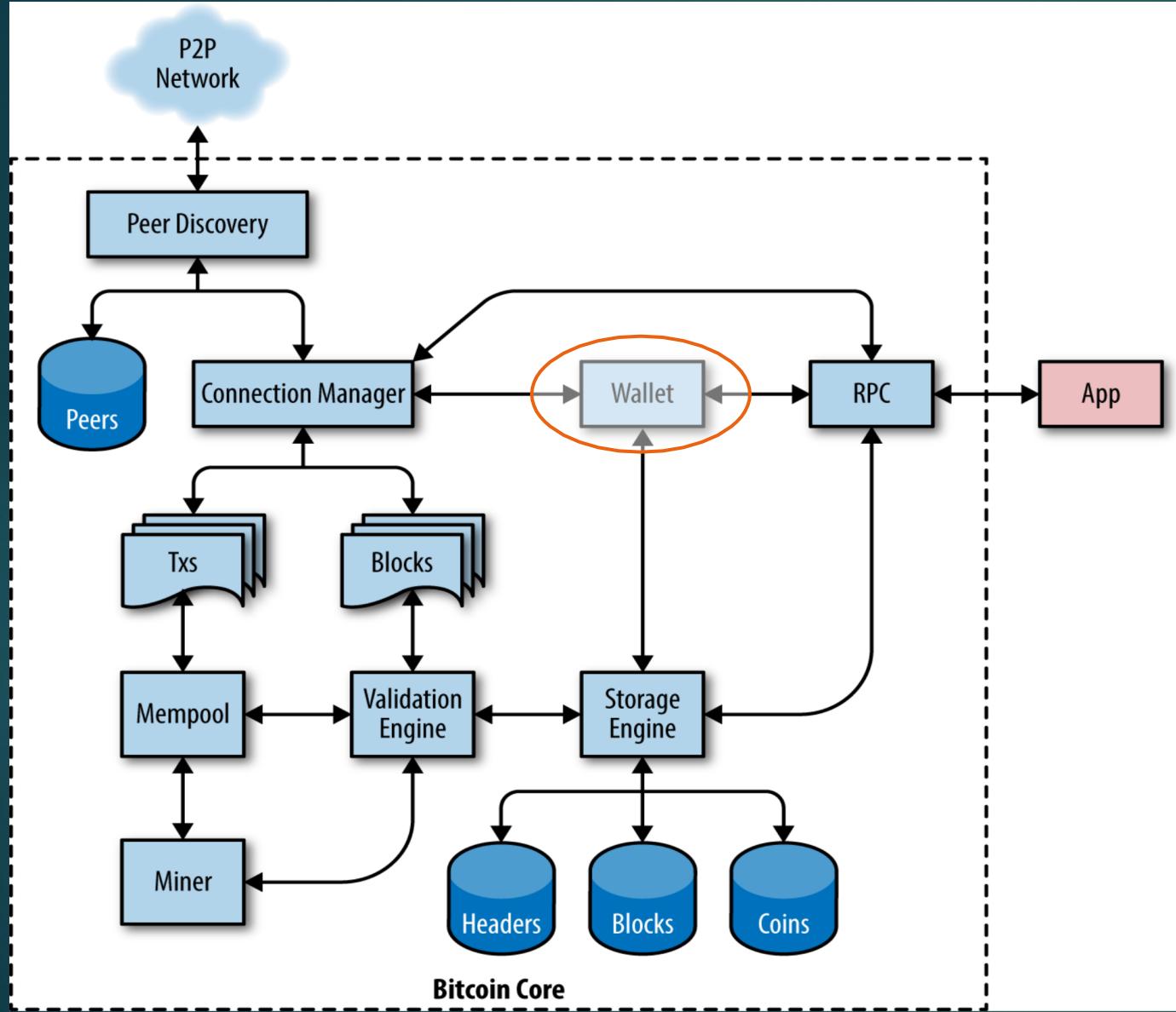
    if (hashVal.toString(16).length == 1)
      txtHash += "0";

    txtHash += hashVal.toString(16);
  }
  return txtHash;
}
```

<https://docs.google.com/spreadsheets/d/1jSLoPvsW6-L2BcTsOW9tULBop0thN1J05Fdt28QFWcA/edit?usp=sharing>

No	WalletSRC	WalletDST	DateTime	Amount	SHA-1 Row	SHA-1 Final
1	W01	W02	20210216110607	2	a6c168d7f2af54e611Bea10fcbe5985da6db95c8	
2	5 W02	W04	20210216110607	3	3b153c1724f1535deca2aa917613bf16240bed	febdd3e72d862f7ac0bad9bb02e776228eabff1s9
3	3 W02	W05	20210216110908	1.5	44741692955a56f455620f11042e05b4bcab0100	f11204dd7aaef93717976409b9c7e2c04ca83d

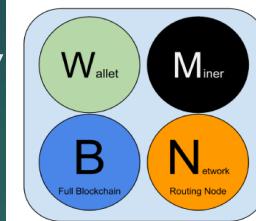
02 Bitcoin | Ethereum Blockchain Technology



Copyright: Andreas M. Antonopoulos. "Mastering Bitcoin, 2nd Edition", Pub

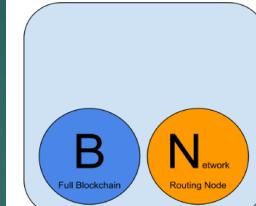
Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



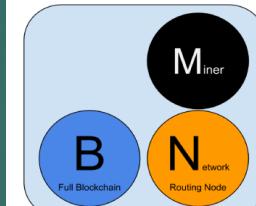
Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



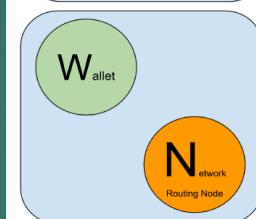
Solo Miner

Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



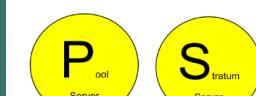
Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.



Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.



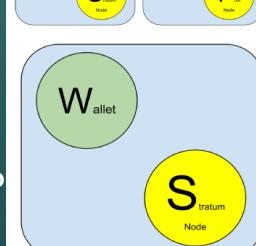
Mining Nodes

Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.



Lightweight (SPV) Stratum wallet

Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.



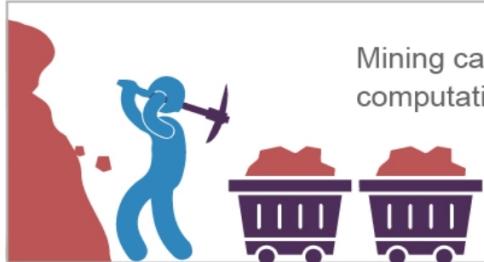
02. Blockchain P2P Network – Consensus Algorithm

83

Proof of Work

VS

Proof of Stake



Mining capacity depends on computational power



Validating capacity depends on the stake in the network



Miners receive block rewards to solve a cryptographic puzzle



Validators do not receive a block reward, instead, they collect transaction fees as reward



Hackers would need to have a computer powerful than 51% of the network to add a malicious block, leading to 51% attack

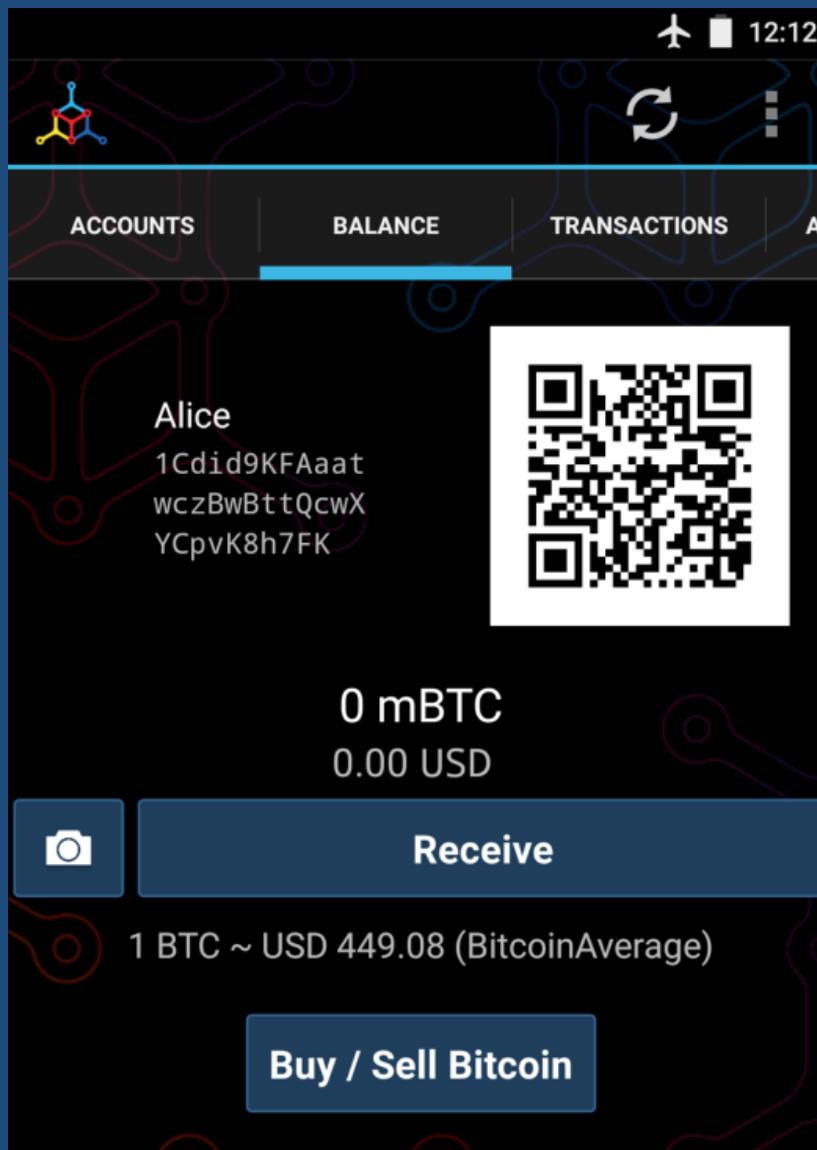


Hacker would need to own 51% of all the cryptocurrency on the network, which is practically impossible and therefore, making 51% attacks impossible.

The **proof of work (POW)** is a **common consensus algorithm** used by the most popular cryptocurrency networks like bitcoin and lite-coin. It requires a participant node to prove that the work done and submitted by them qualifies them to receive the right to add new transactions to the blockchain.

02 Bitcoin | Ethereum Blockchain Technology – DEMO

with iOS and Android Mycellium



Transaction

View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fb8a57286c345c2f2

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)

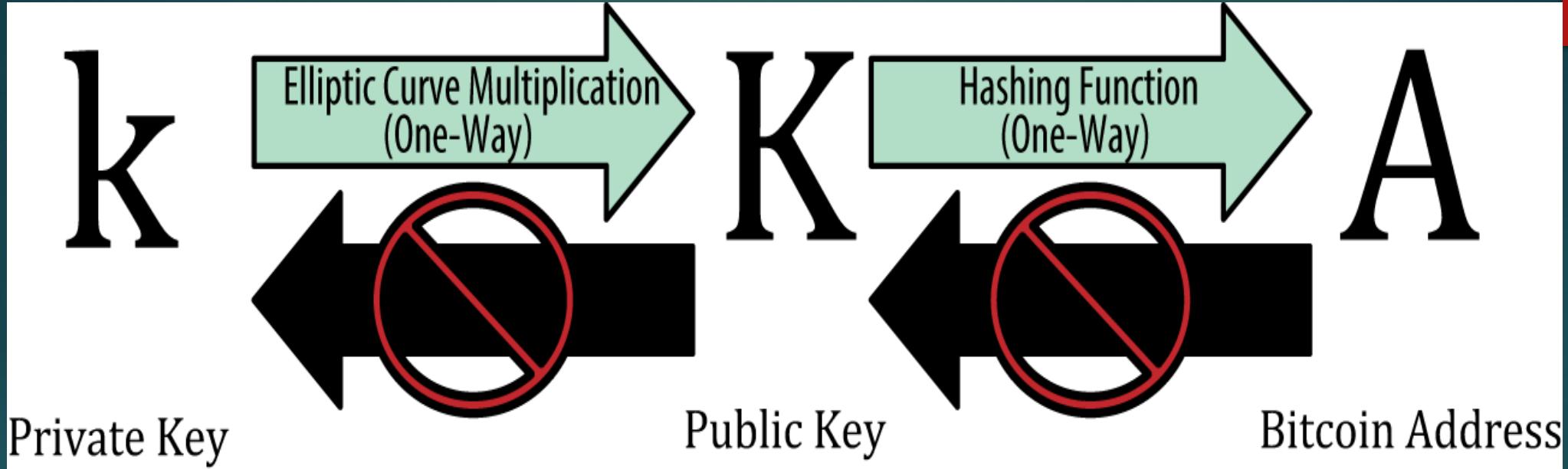
→ 1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA
- (Unspent) 0.015 BTC

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK -
(Unspent) 0.0845 BTC

97 Confirmations 0.0995 BTC

Summary		Inputs and Outputs	
Size	258 (bytes)	Total Input	0.1 BTC
Received Time	2013-12-27 23:03:05	Total Output	0.0995 BTC
Included In Blocks	277316 (2013-12-27 23:11:54 +9 minutes)	Fees	0.0005 BTC
		Estimated BTC Transacted	0.015 BTC

02 Bitcoin / Ethereum Blockchain Technology - Keys



k = private key is simply a number, picked at random. (e.g. `bitcoin-cli getnewaddress`)

```
$ bitcoin-cli getnewaddress  
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy
```

```
$ bitcoin-cli dumpprivkey  
1J7mdg5rbQyUHENYdx39WVWK7fsLpEoXZy  
KxFc1jmwwCoACiCAWZ3eXa96mBM6tb3TYzGmf6YwgdGWZgawrtJ
```

02 Bitcoin / Ethereum Blockchain Technology - Keys

K = public key is calculated from the private key using elliptic curve multiplication, which is irreversible: $K = k * G$, where k is the private key, G is a constant point called the generator point, and K is the resulting public key.

Bitcoin/Ethereum uses a specific elliptic curve and set of mathematical constants, as defined in a standard called **secp256k1**, established by the National Institute of Standards and Technology (NIST). The **secp256k1** curve is defined by the following function, which produces an elliptic curve:

$$Y^2 \bmod P = (X^3 + 7) \bmod P$$

$$Y = \text{SQRT}(8) \Rightarrow Y = \pm 2.82$$

$$(X=1, Y = \pm 2.82) \text{ or } (X=3, Y = \pm 5.83)$$

$$y^2 = (x^3 + 7) \text{ over } (\mathbb{F}_p)$$

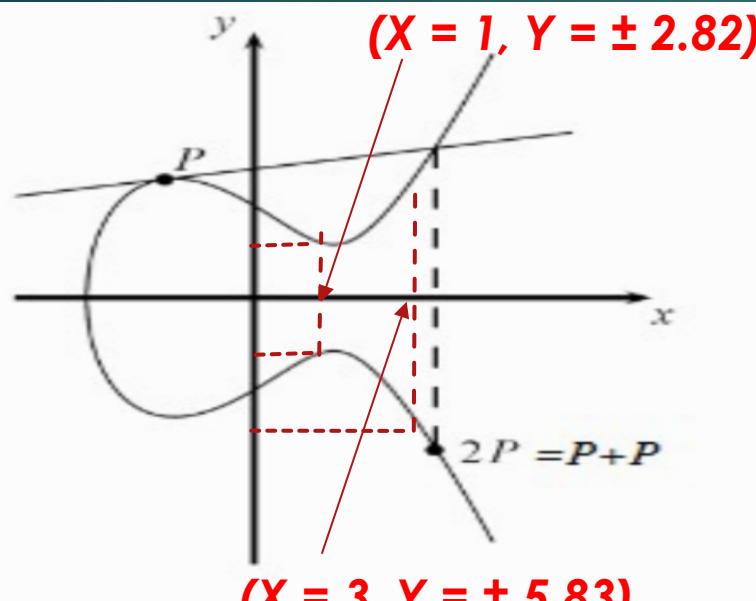
or

$$y^2 \bmod p = (x^3 + 7) \bmod p$$

The $\bmod p$ (modulo prime number p) indicates that this curve is over a finite field of prime order p , also written as \mathbb{F}_P , where $p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$, a very large prime number.

02 Bitcoin / Ethereum Blockchain Technology - Addresses & Keys

Most bitcoin/ETH implementations use the OpenSSL cryptographic library to do the elliptic curve math. For example, to derive the public key, the function EC_POINT_mul() is used.



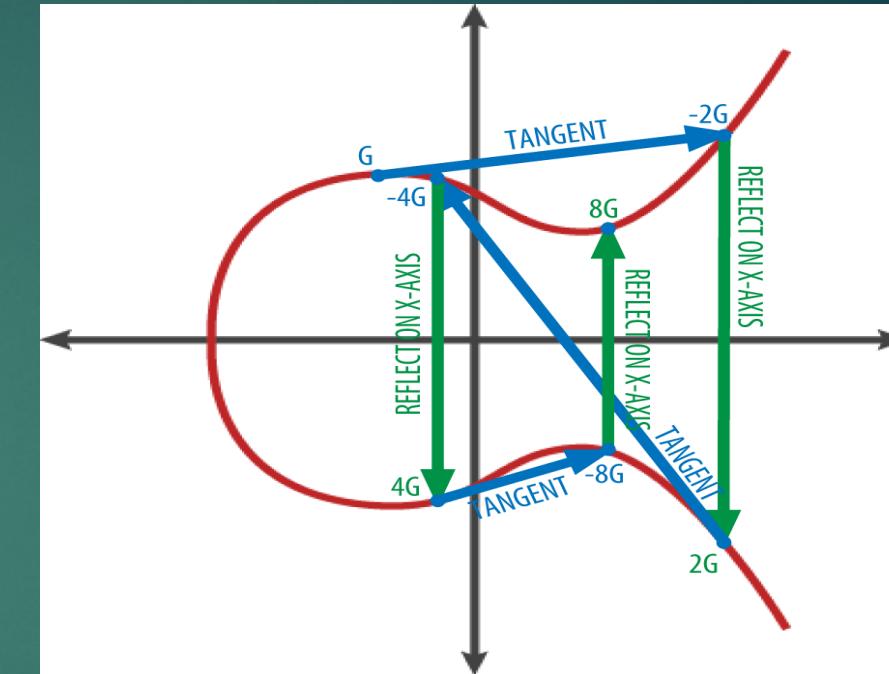
Graph representing point doubling over real numbers

In the case of point doubling, the equation becomes:

$$x_3 = s^2 - x_1 - x_2 \bmod p$$

$$y_3 = s(x_1 - x_3) - y_1 \bmod p$$

$$S = \frac{3x_1^2 + a}{2y_1}$$



The private key in this picture is 8, e.g. privKey = 8

G (compressed) = 02 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9
59F2815B 16F81798

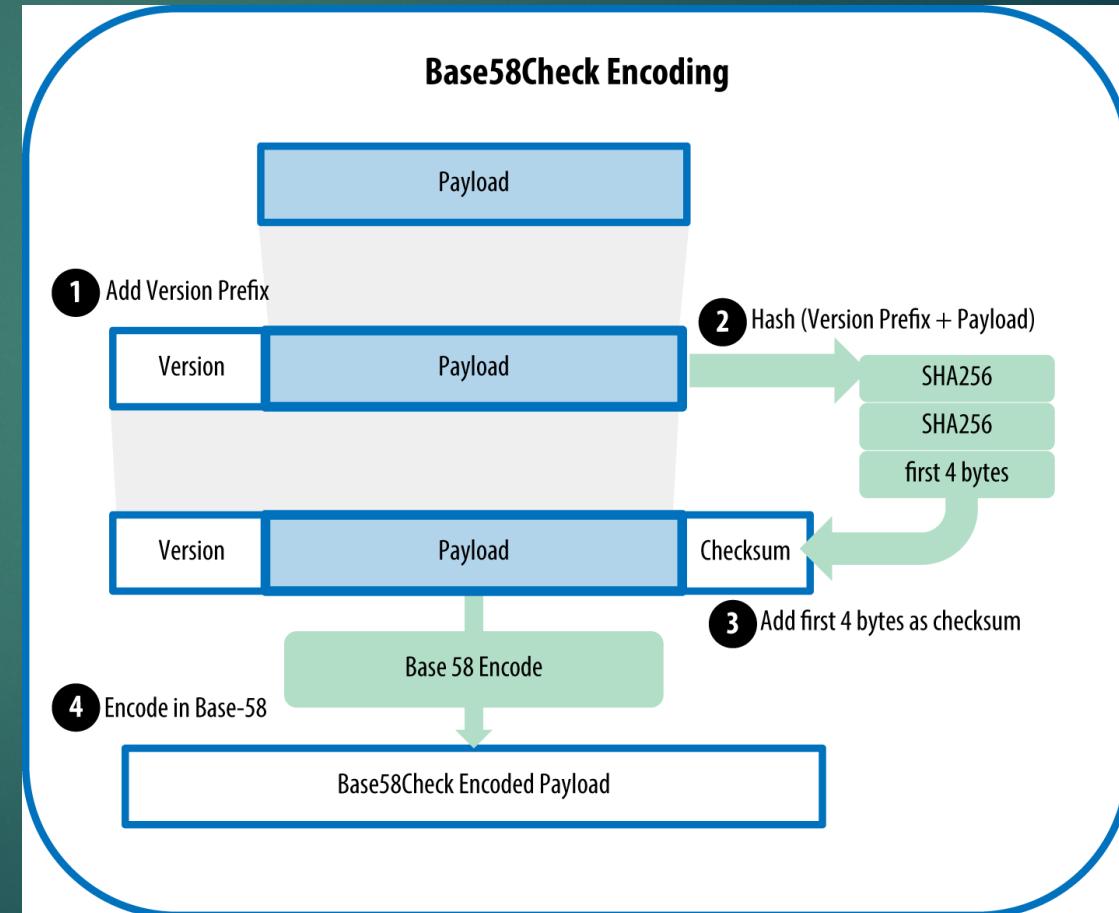
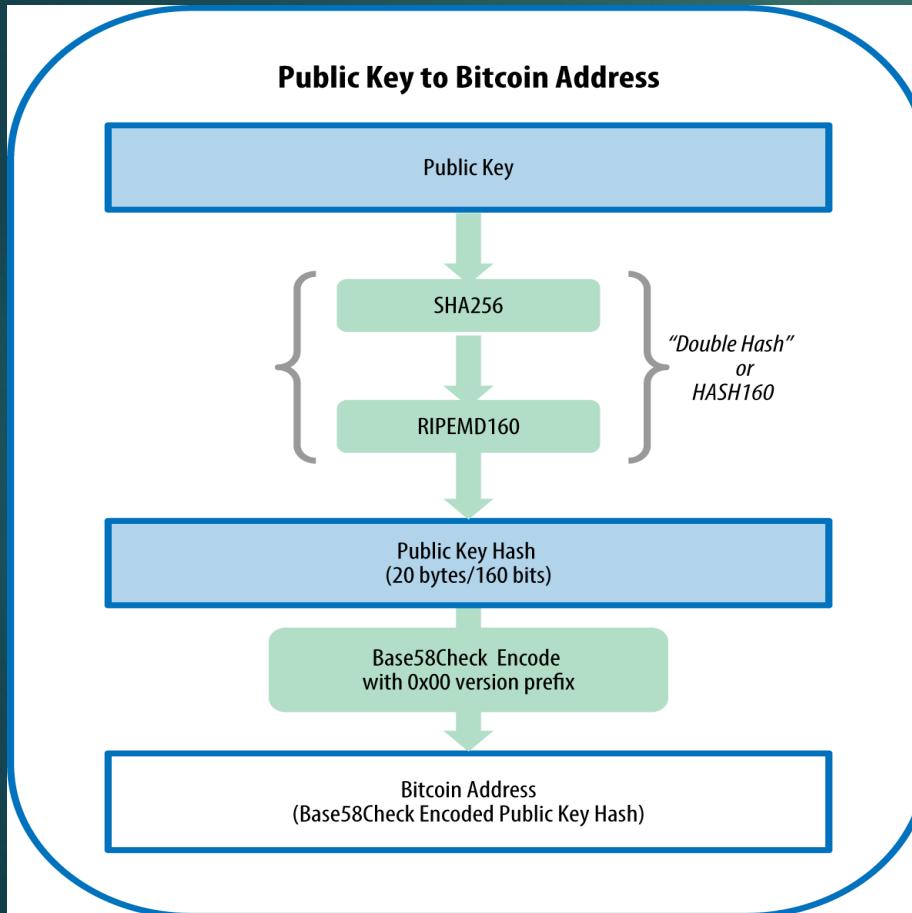
G (uncompressed) = 04 79BE667E F9DCBBAC 55A06295 CE870B07 029BFCDB 2DCE28D9
59F2815B 16F81798 483ADA77 26A3C465 5DA4FBFC 0E1108A8 FD17B448 A6855419
9C47D08F FB10D4B8

02 Bitcoin / Ethereum Blockchain Technology - Addresses & Keys

Starting with the public key K , we compute the SHA256 hash and then compute the RIPEMD160 hash of the result, producing a 160-bit (20-byte) number:

$$A = \text{RIPEMD160}(\text{SHA256}(K))$$

where K is the public key and A is the resulting bitcoin address.



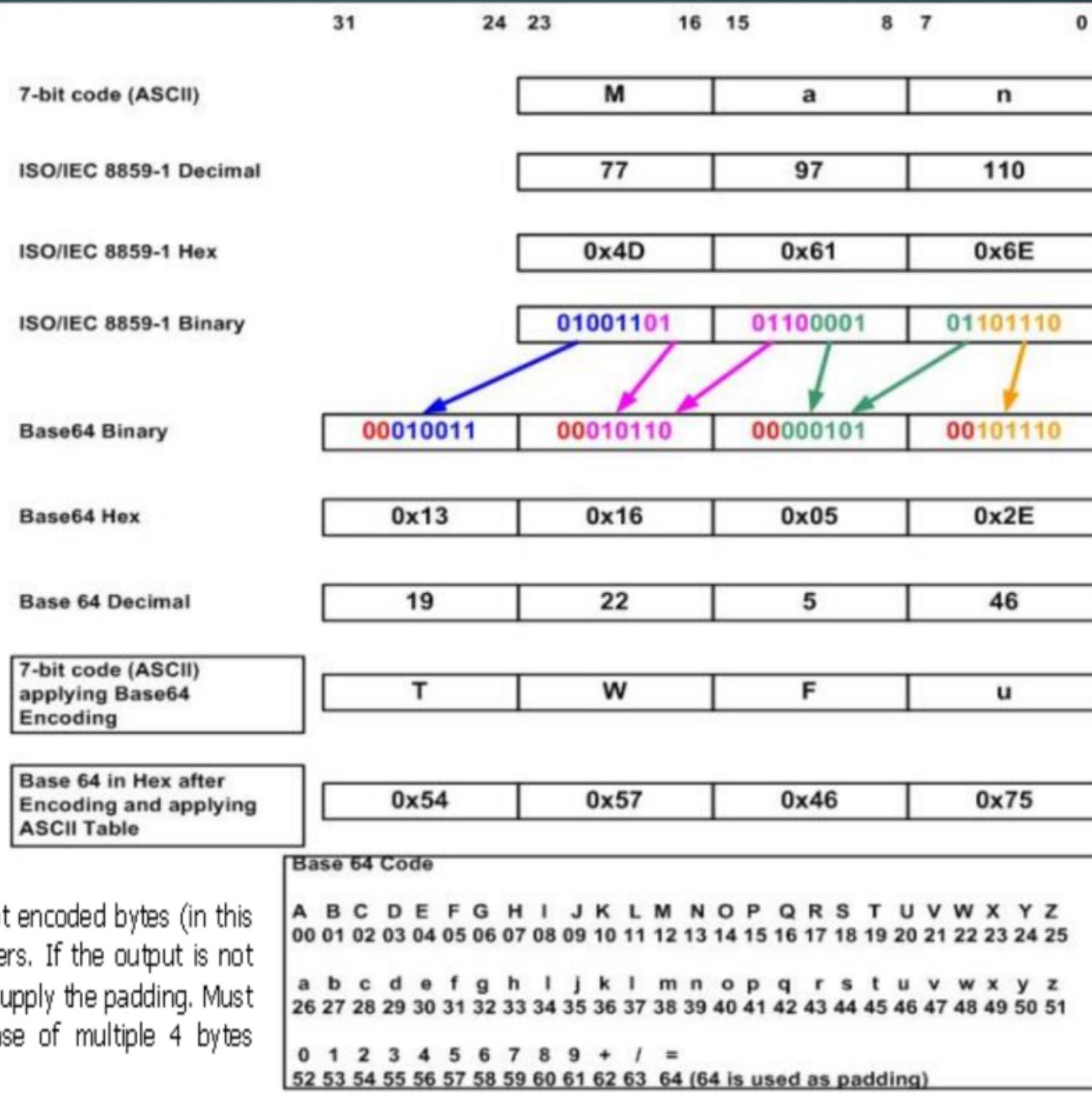
Public key to bitcoin address: conversion of a public key into a bitcoin address.

02 Bitcoin / Ethereum Blockchain Technology - Addresses & Keys

Base64 encoding is used in practice usually for transport over the network and heterogeneous environments binary code such as pictures or executable code. The techniques is very simple: to transform each 3 bytes values into 4 bytes value in order to avoid to obtain values greater then 127 per byte.

For instance, if the scope is to encode the word "Man" into Base64 encoding then it is encoded as "TWFu". Encoded in ASCII (in ISO 8859-1, one value per byte), M, a, n are stored as the bytes 77 (0x4D), 97 (0x61), 110 (0x6E), which are 01001101, 01100001, 01101110 in base 2.

As this example illustrates, the encoding converts 3 not encoded bytes (in this case, ASCII characters) into 4 encoded ASCII characters. If the output is not multiple of 4 bytes then the '=' sign is put in order to supply the padding. Must be considered the padding with = (64 value) in case of multiple 4 bytes number.



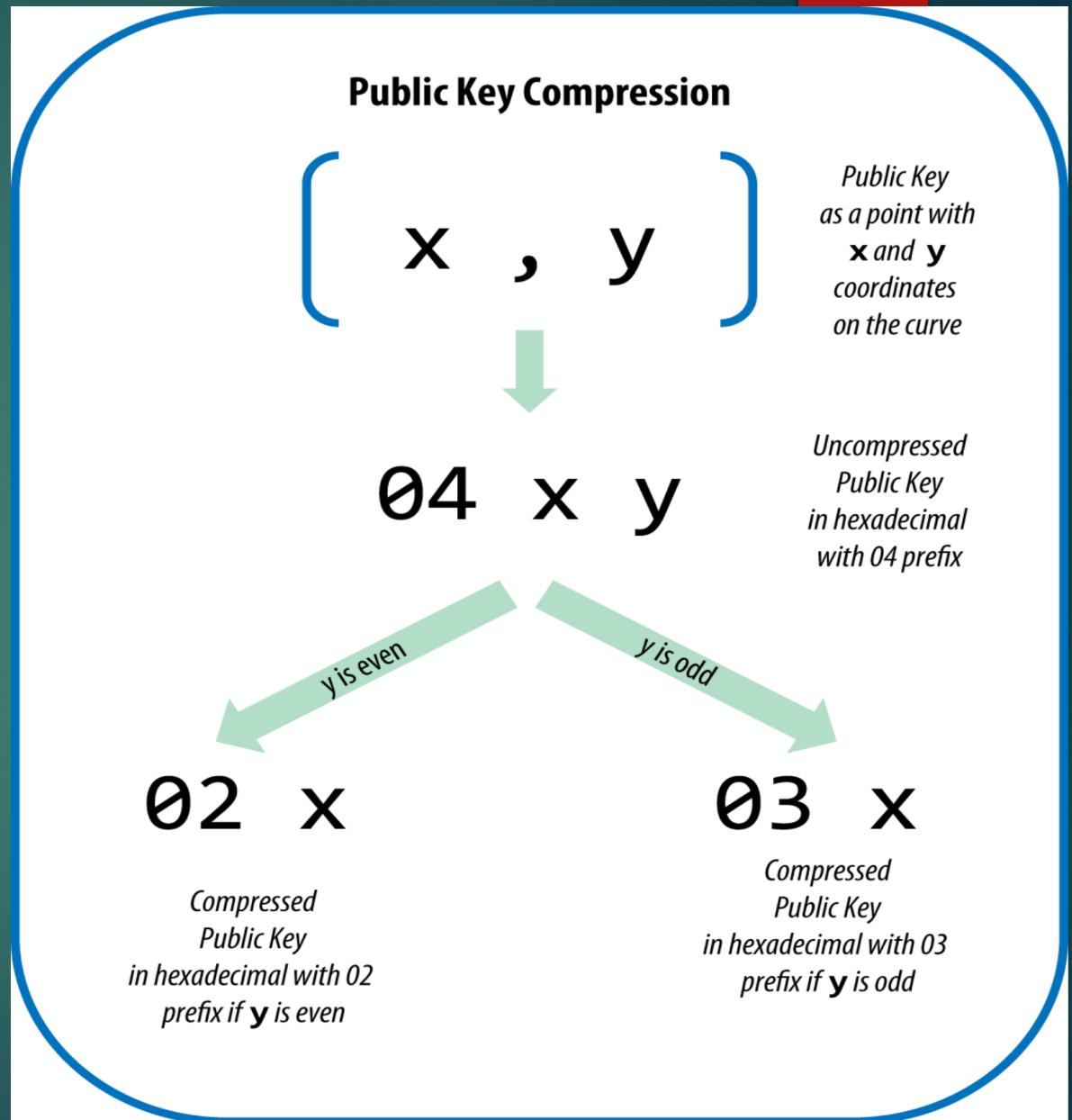
Base64
to
Base58
Analogy

02 Bitcoin / Ethereum Blockchain Technology - Addresses & Keys

A public key is a point **(x,y)** on an elliptic curve. Because the curve expresses a mathematical function, a point on the curve represents a solution to the equation and, therefore, if we know the x coordinate we can calculate the y coordinate by solving the equation:

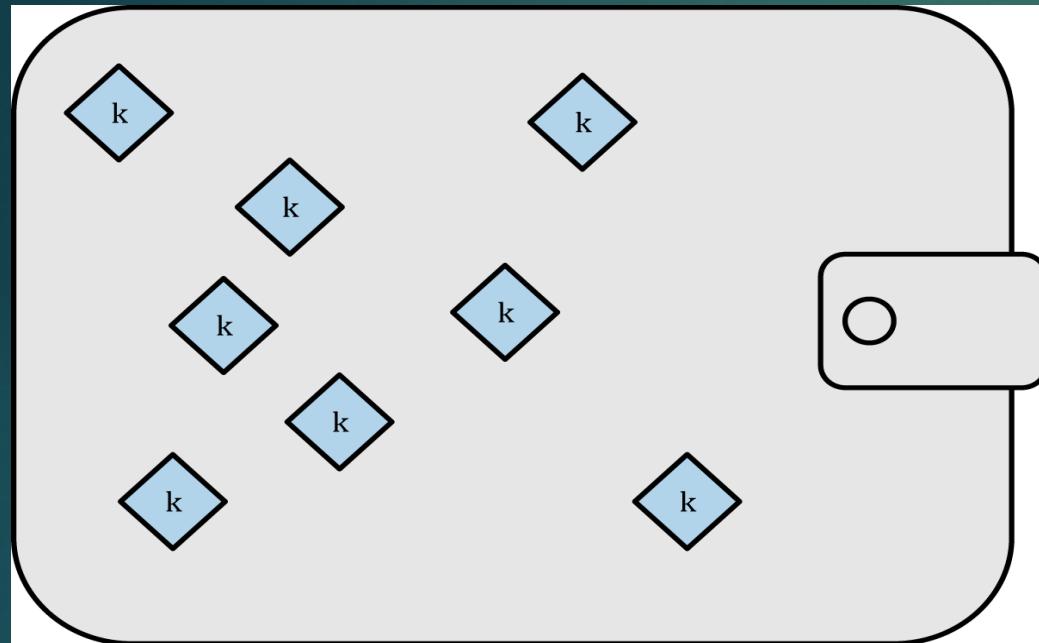
$$y^2 \bmod p = (x^3 + 7) \bmod p.$$

That allows us to store only the x coordinate of the public key point, omitting the y coordinate and reducing the size of the key and the space required to store it by 256 bits. An almost 50% reduction in size in every transaction adds up to a lot of data saved over time!



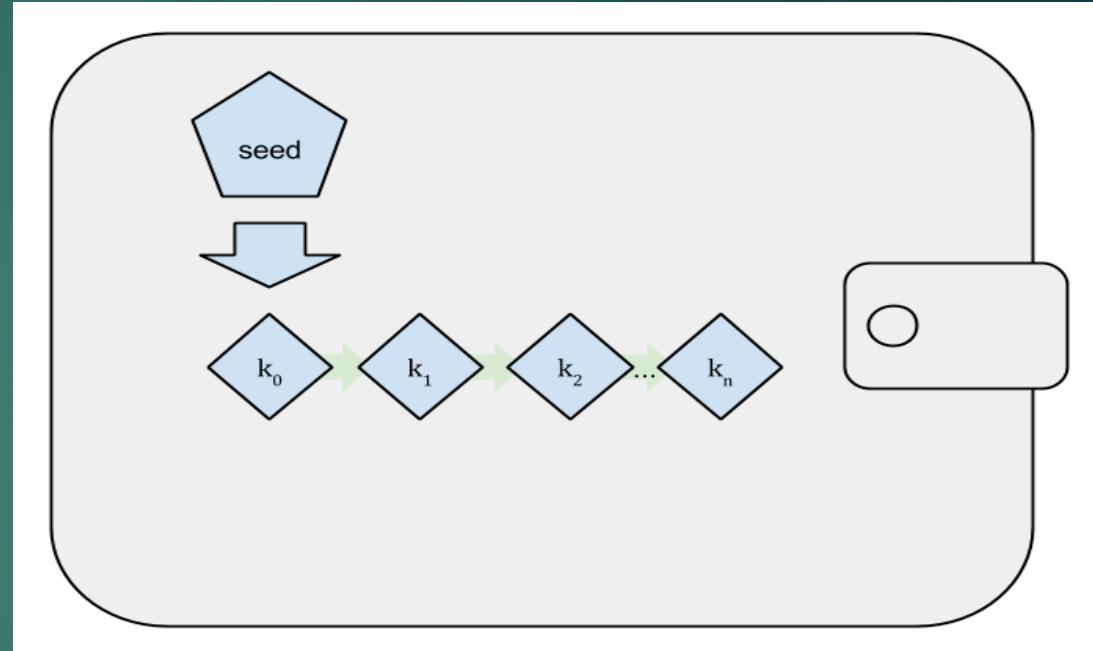
02 Bitcoin / Ethereum Blockchain Technology - Wallets

A common misconception about bitcoin is that bitcoin wallets contain bitcoin. In fact, the wallet contains only keys. The “coins” are recorded in the blockchain on the bitcoin network. Users control the coins on the network by signing transactions with the keys in their wallets. In a sense, a bitcoin wallet is a keychain.



The first type is a nondeterministic wallet, where each key is independently generated from a random number. The keys are not related to each other. This type of wallet is also known as a JBOK wallet from the phrase “Just a Bunch Of Keys.”

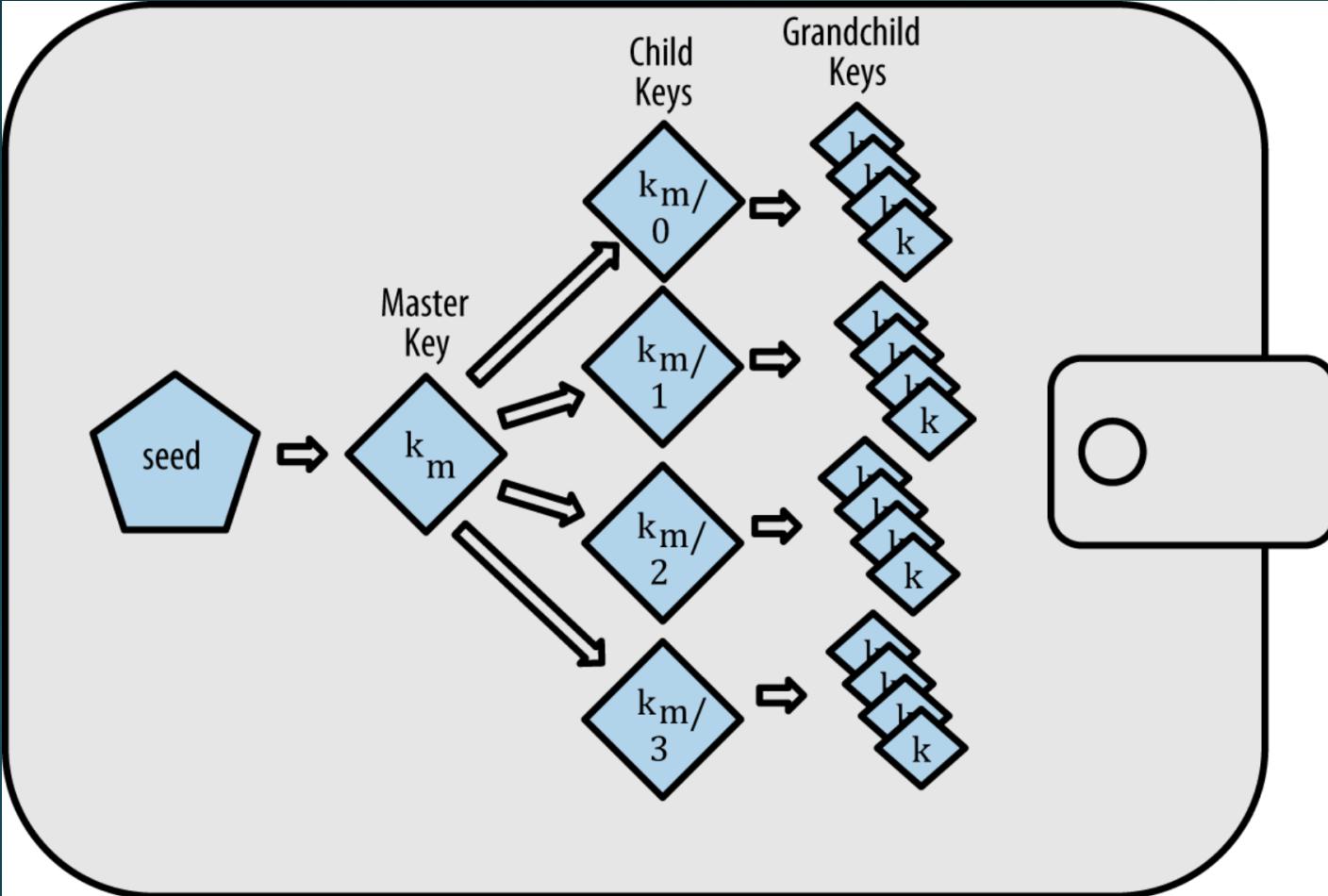
Type-0 nondeterministic (random) wallet: a collection of randomly generated keys



The second type of wallet is a deterministic wallet, where all the keys are derived from a single master key, known as the seed. All the keys in this type of wallet are related to each other and can be generated again if one has the original seed.

Type-1 deterministic (seeded) wallet: a deterministic sequence of keys derived from a seed

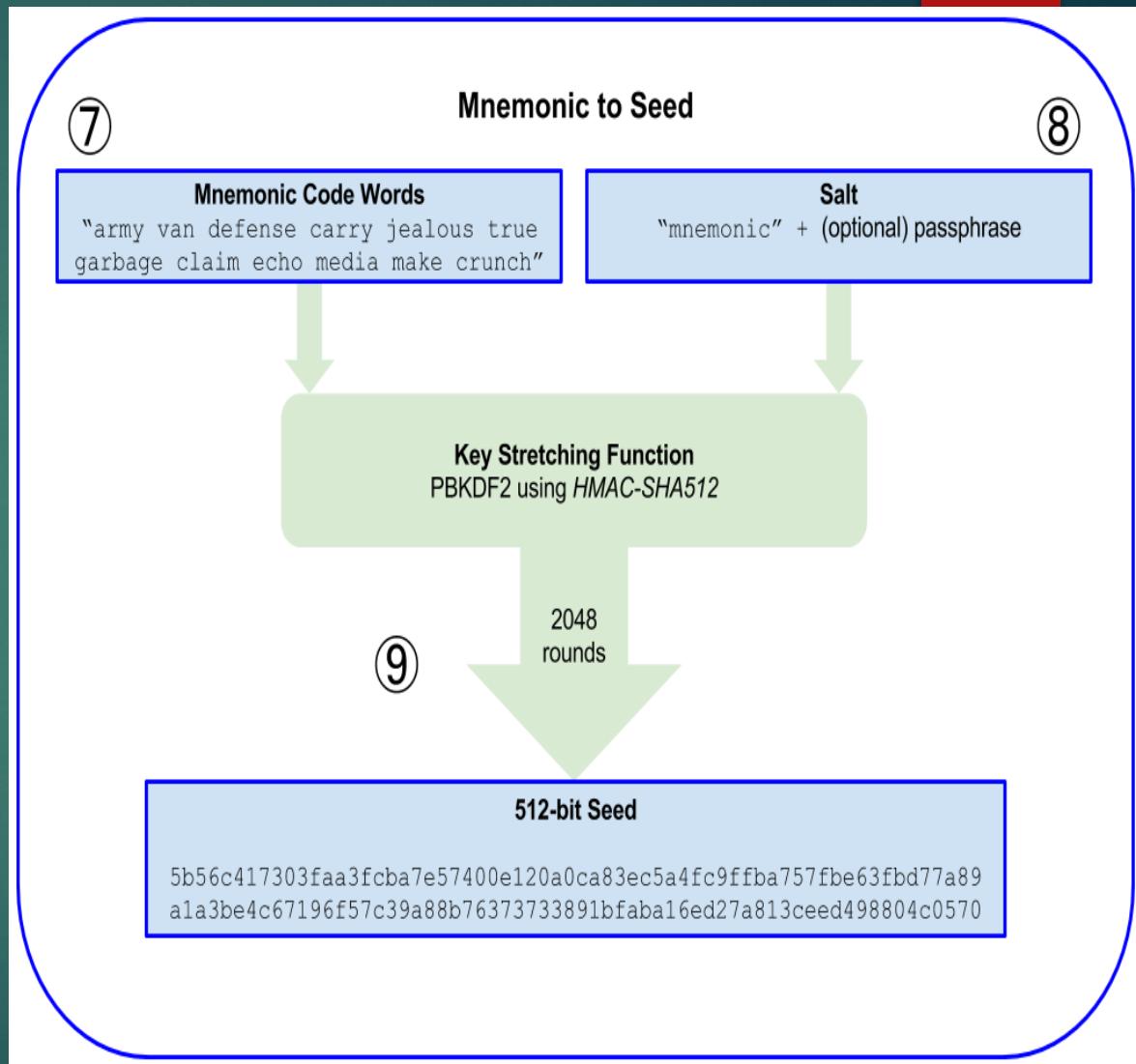
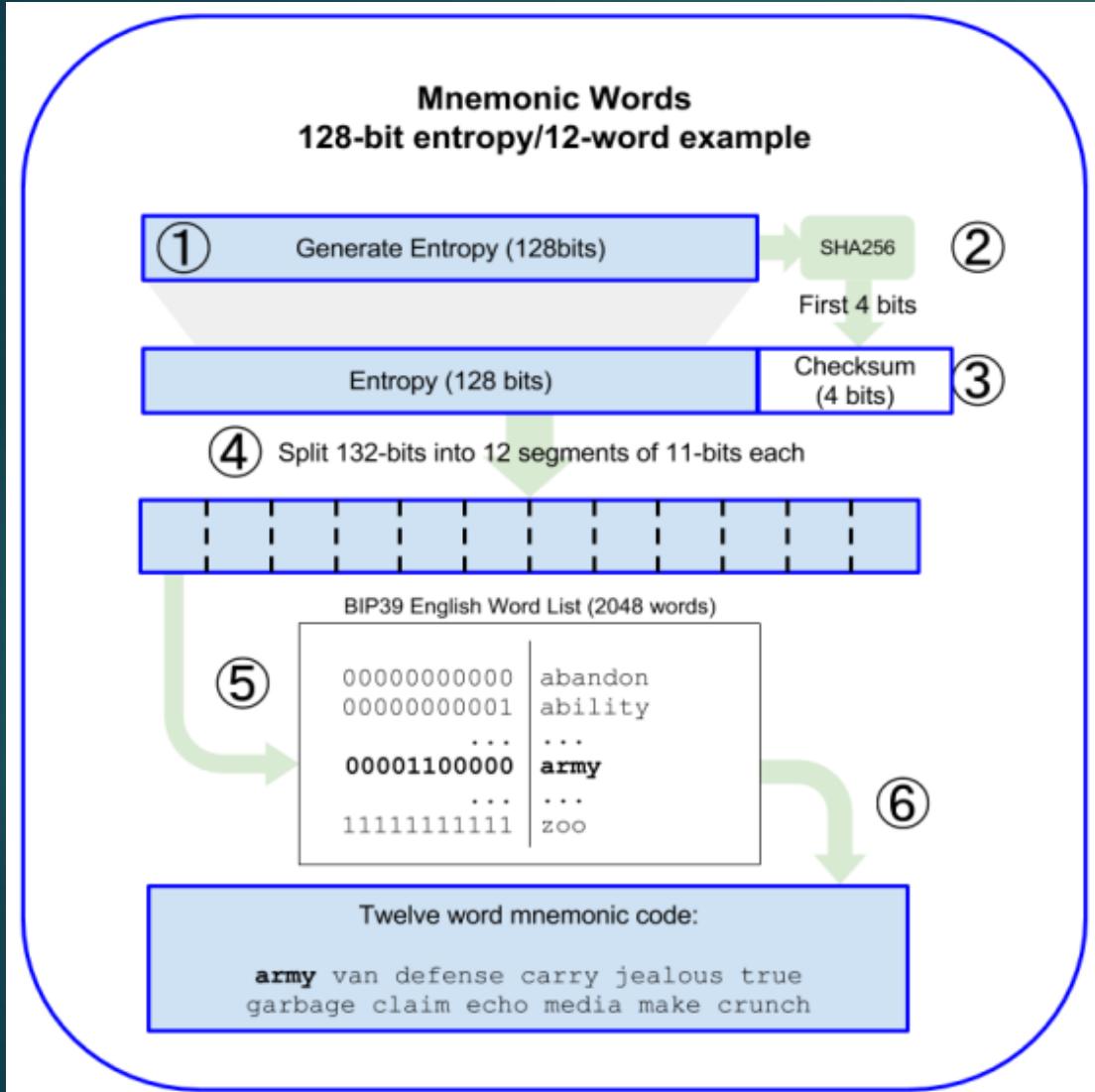
02 Bitcoin / Ethereum Blockchain Technology - Wallets



Type-2 HD (*Hierarchical Deterministic*) Wallet: a tree of keys generated from a single seed
HD Wallets (Bitcoin Improvement Proposal: BIP-32/BIP-44)

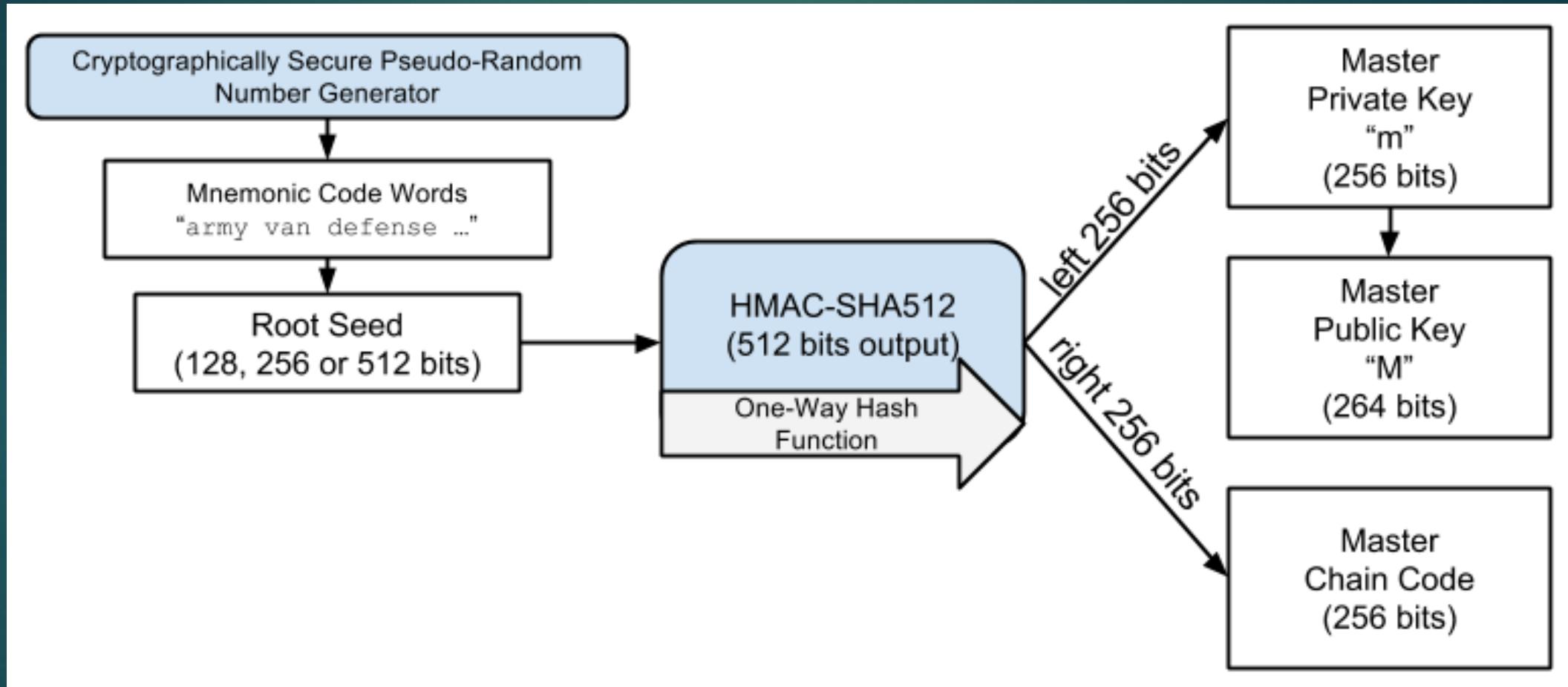
02 Bitcoin / Ethereum Blockchain Technology - HD Wallets

Generating entropy and encoding as mnemonic words



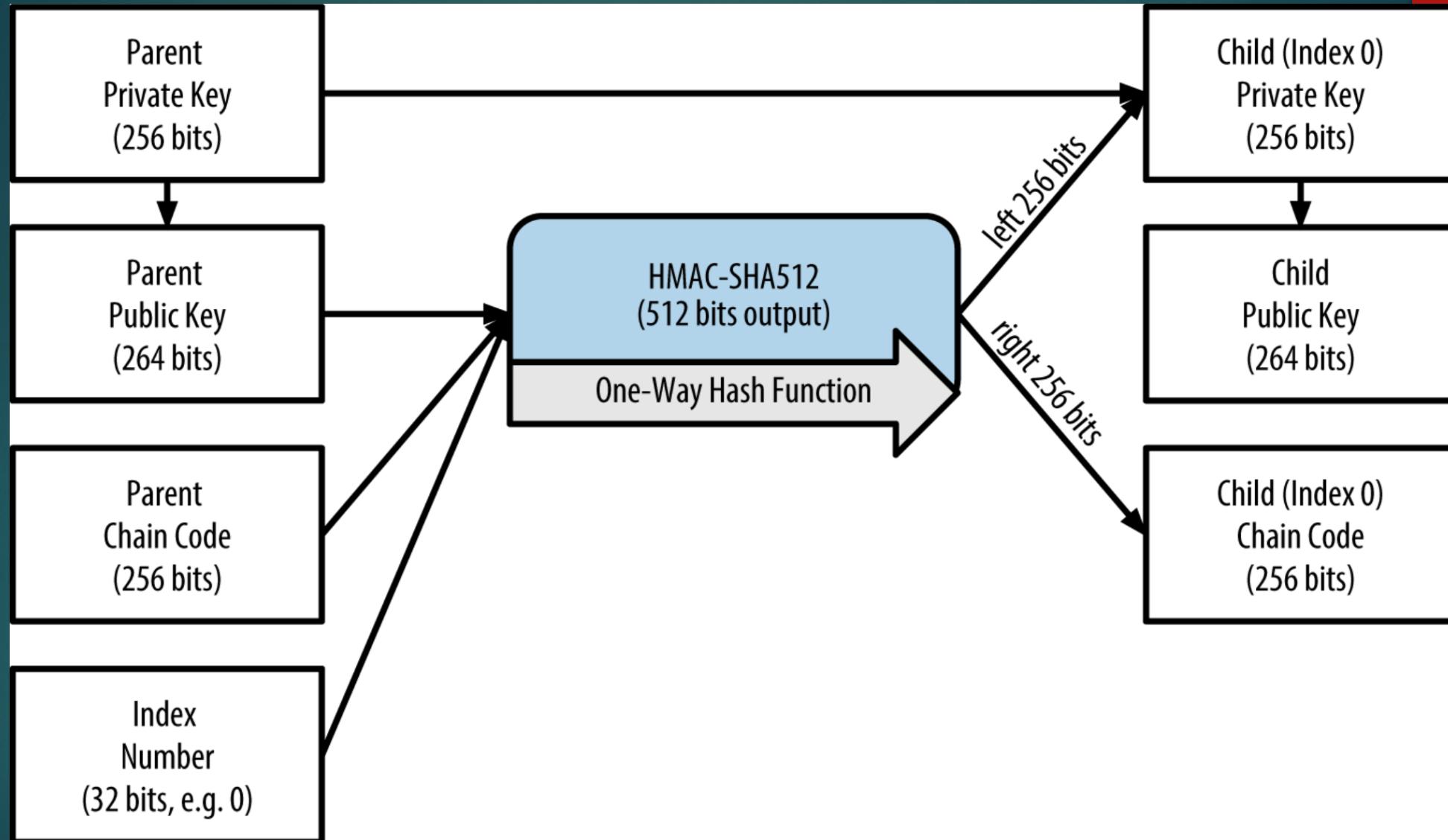
02. Bitcoin / Ethereum Blockchain Technology - HD Wallets

Creating an HD Wallet from the Seed HD wallets are created from a single root seed, which is a 128-, 256-, or 512-bit random number. Most commonly, this seed is generated from a mnemonic:



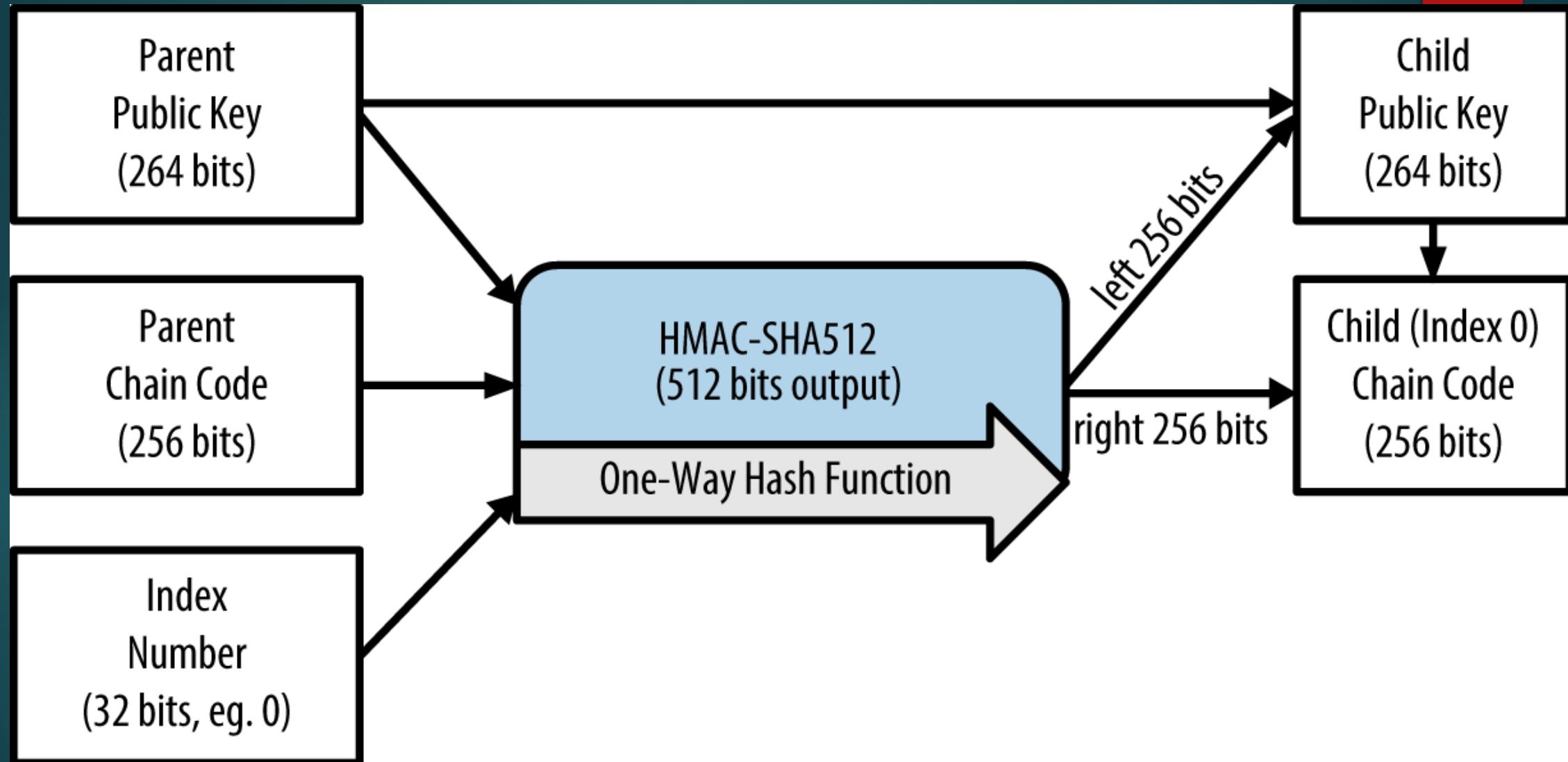
Creating master keys and chain code from a root seed

02. Bitcoin / Ethereum Blockchain Technology - HD Wallets



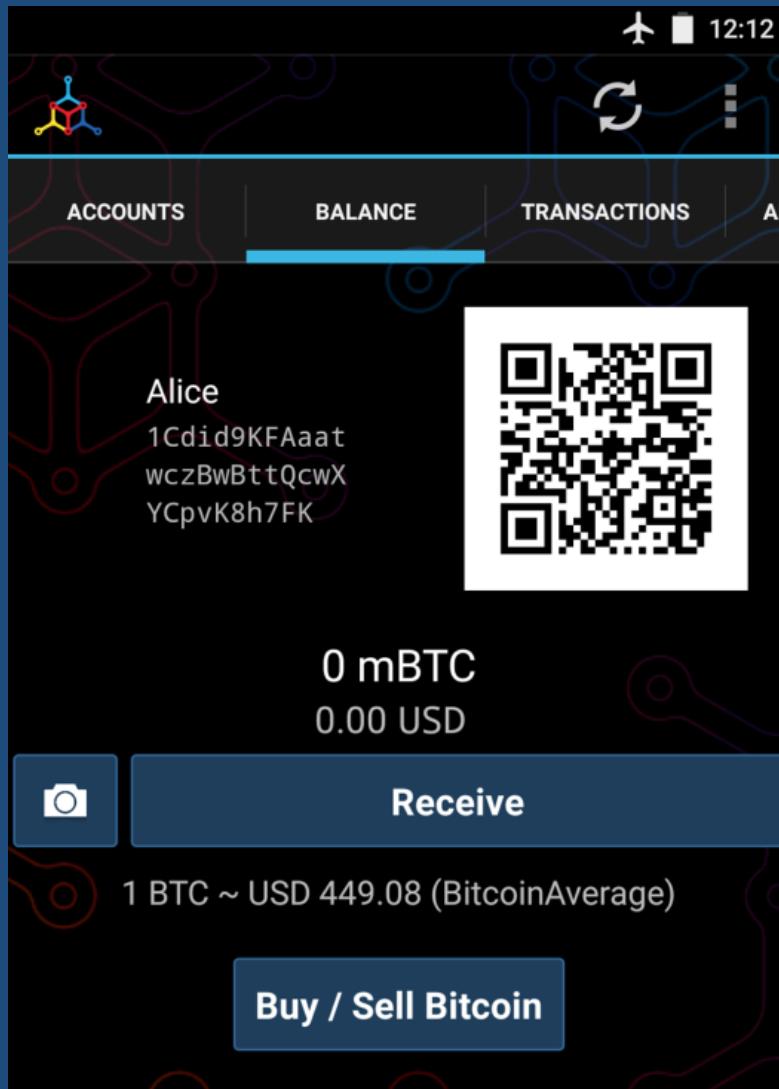
Extending a parent private key to create a child private key

02. Bitcoin / Ethereum Blockchain Technology - HD Wallets



Extending a parent public key to create a child public key

2.1 Bitcoin / Ethereum Blockchain Technology - Transactions



Transaction View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fb8a57286c345c2f2

1CdId9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)

1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA
- (Unspent) 0.015 BTC

1CdId9KFAaatwczBwBttQcwXYCpvK8h7FK -
(Unspent) 0.0845 BTC

97 Confirmations 0.0995 BTC

Summary

Size	258 (bytes)
Received Time	2013-12-27 23:03:05
Included In Blocks	277316 (2013-12-27 23:11:54 +9 minutes)

Inputs and Outputs

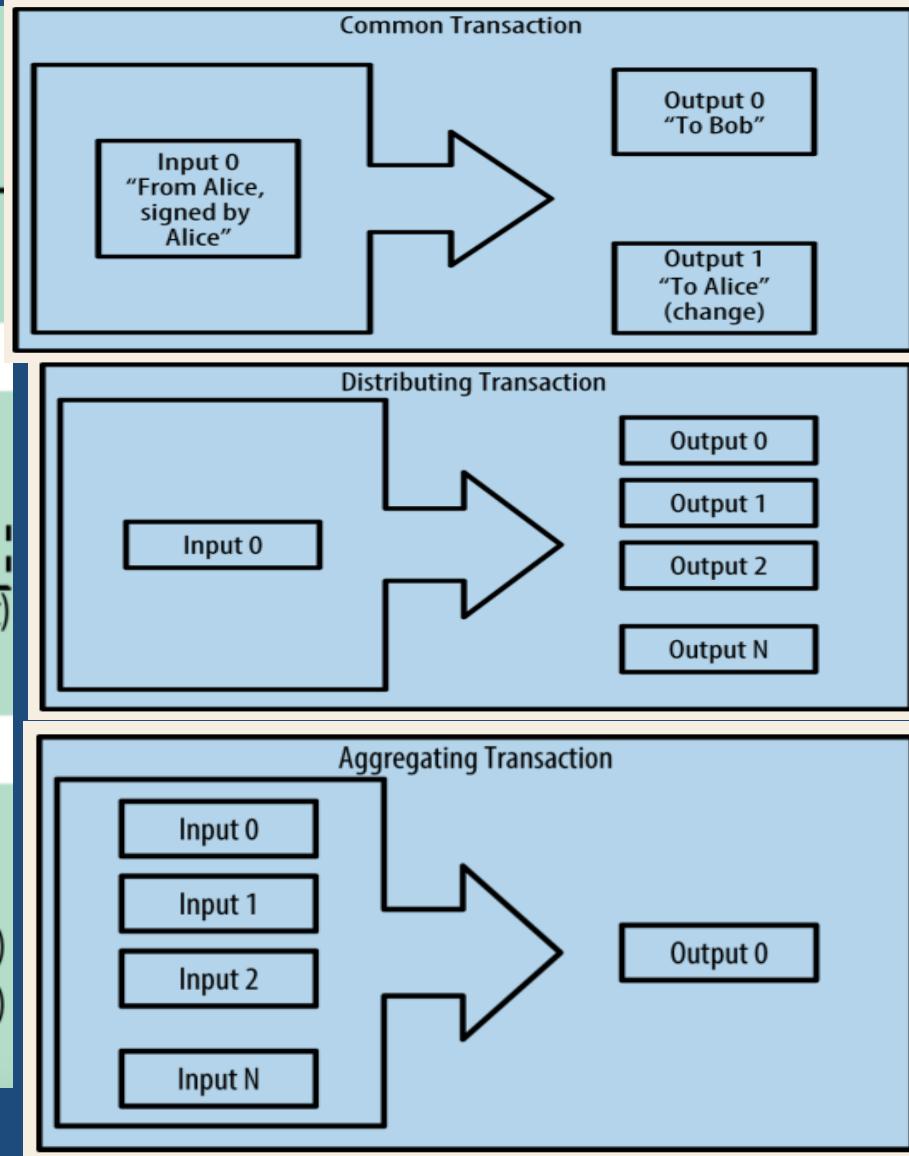
Total Input	0.1 BTC
Total Output	0.0995 BTC
Fees	0.0005 BTC
Estimated BTC Transacted	0.015 BTC

2.1 Bitcoin / Ethereum Blockchain Technology - Transactions

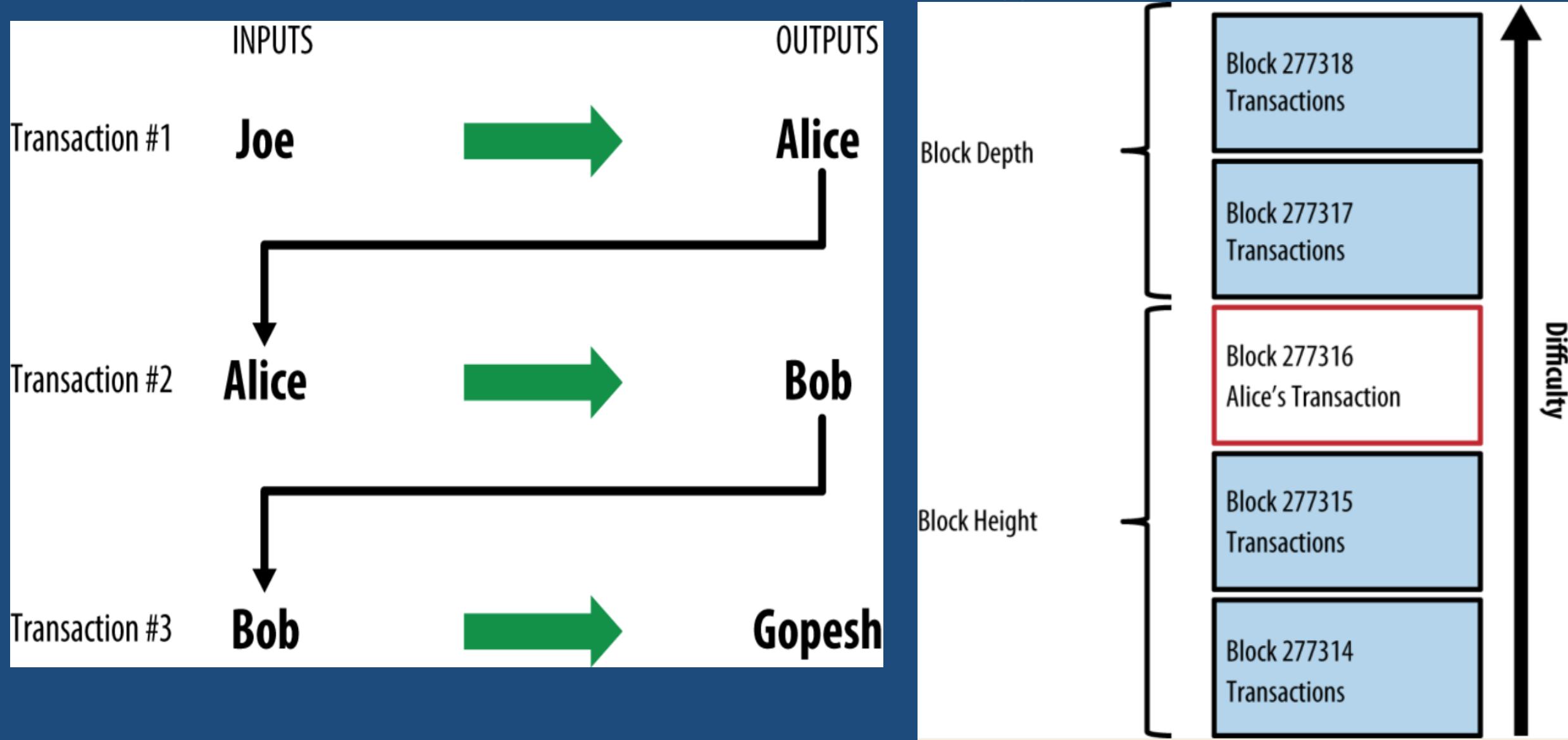
Transaction 7957a35fe64f80d234d76d83a2a8f1a0d8149a41d81de548f0a65a8a999f6f18			
INPUTS From		OUTPUTS To	
From (previous transactions Joe has received):	Joe	Output #0 Alice's Address	0.1000 BTC (spent)
		Transaction Fees:	0.0000 BTC

Transaction 0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fb8a57286c345c2f2			
INPUTS From		OUTPUTS To	
From (previous transactions Alice has received):	Alice	Output #0 Bob's Address	0.0150 BTC (spent)
		Output #1 Alice's Address (change)	0.0845 BTC (unspent)
		Transaction Fees:	0.0005 BTC

Transaction 2bbac8bb3a57a2363407ac8c16a67015ed2e88a4388af58cf90299e0744d3de4			
INPUTS From		OUTPUTS To	
From (previous transactions Bob has received):	Bob	Output #0 Gopesh's Address	0.0100 BTC (unspent)
		Output #1 Bob's Address (change)	0.0045 BTC (unspent)
		Transaction Fees:	0.0005 BTC

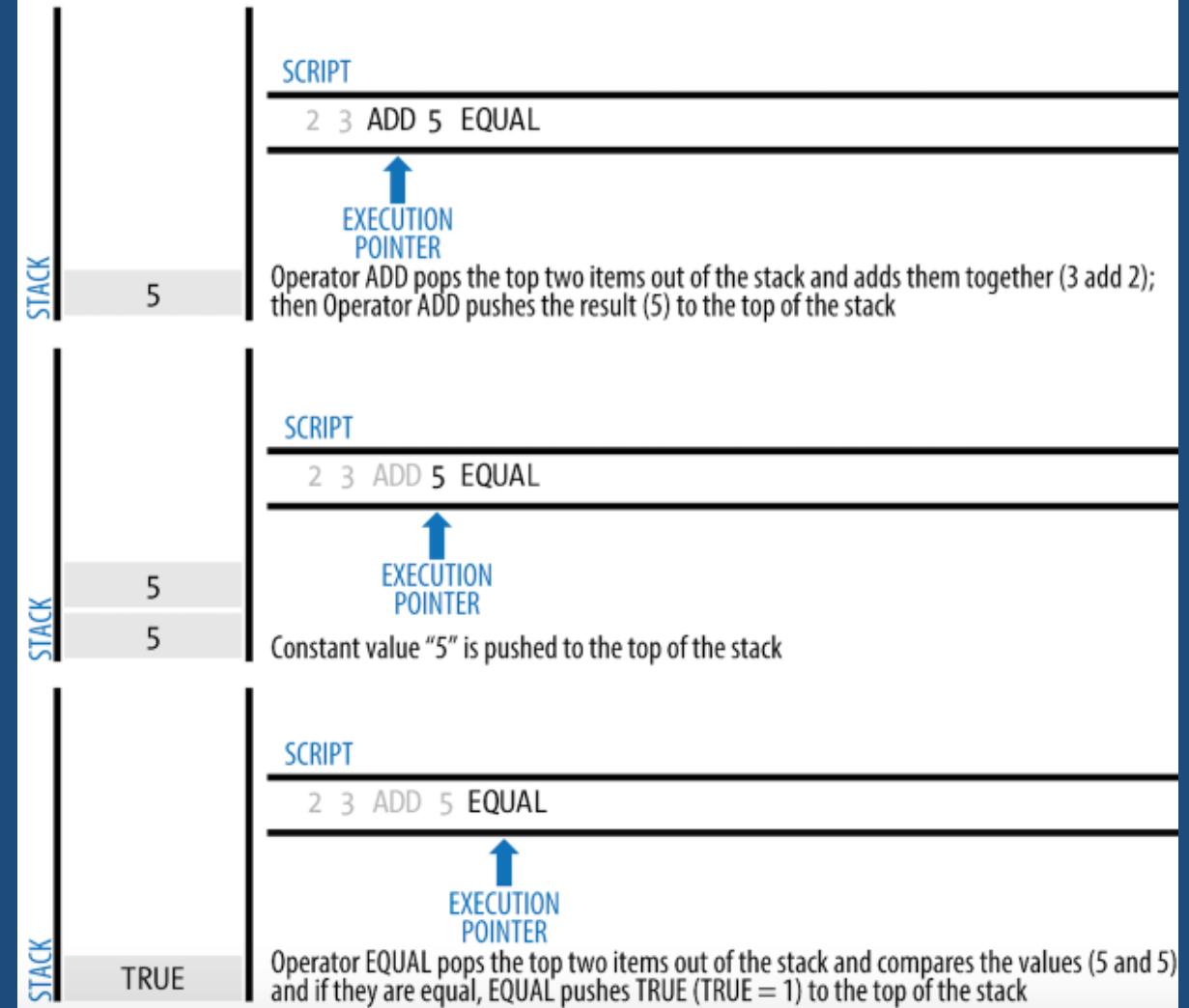
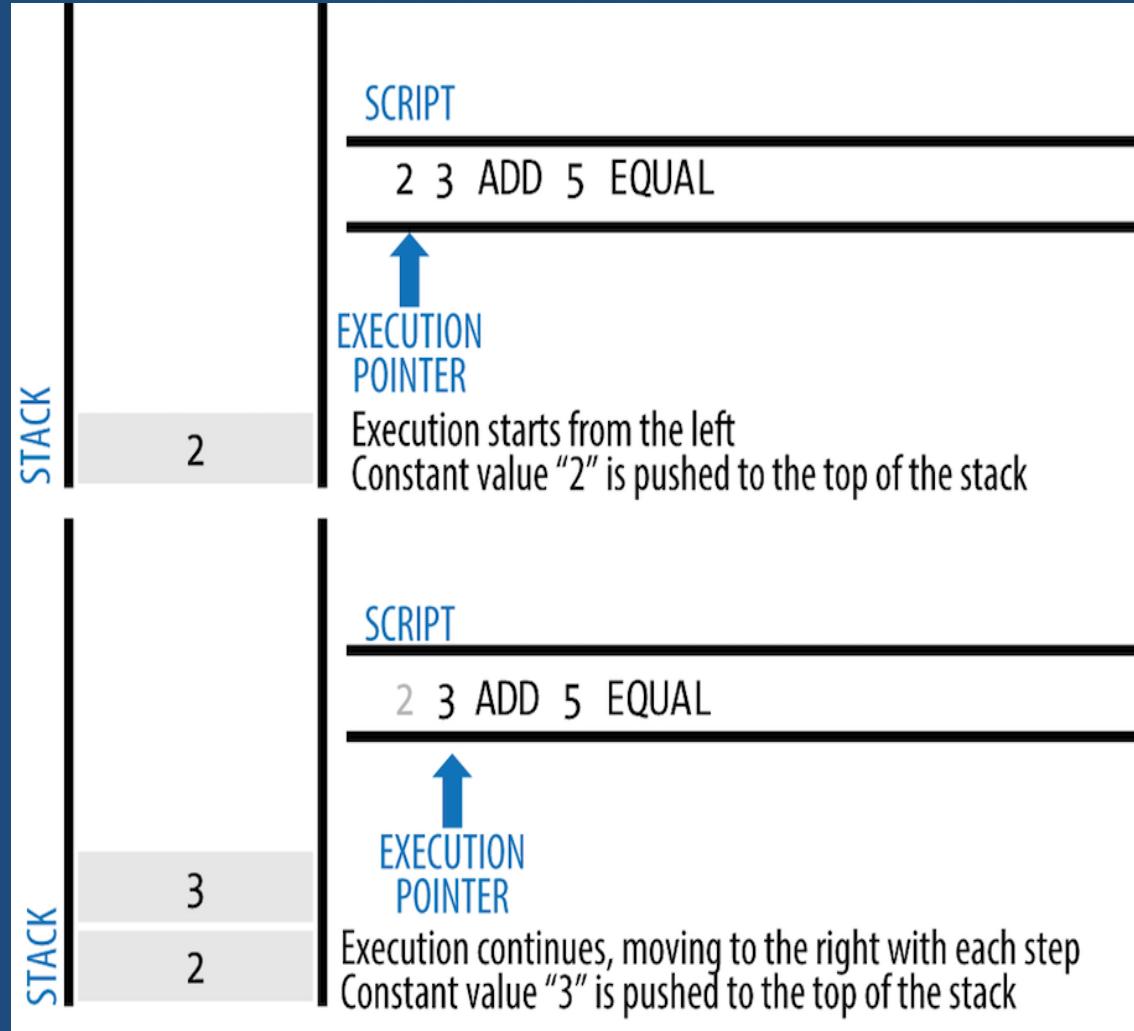


2.1 Bitcoin / Ethereum Blockchain Technology - Transactions



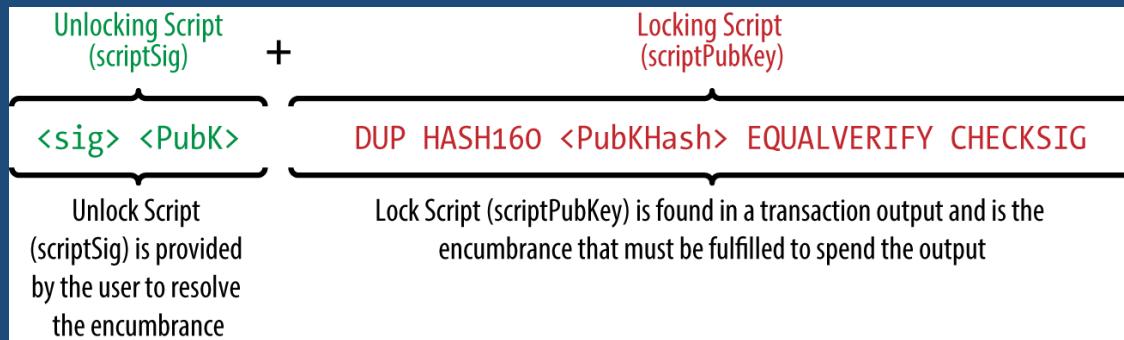
2.1 Bitcoin / Ethereum Blockchain Technology - Transactions

Bitcoin's script validation doing simple math



2.1 Bitcoin / Ethereum Blockchain Technology - Transactions

Evaluating a script for a P2PKH (Pay-to-Public-Key-Hash) transaction



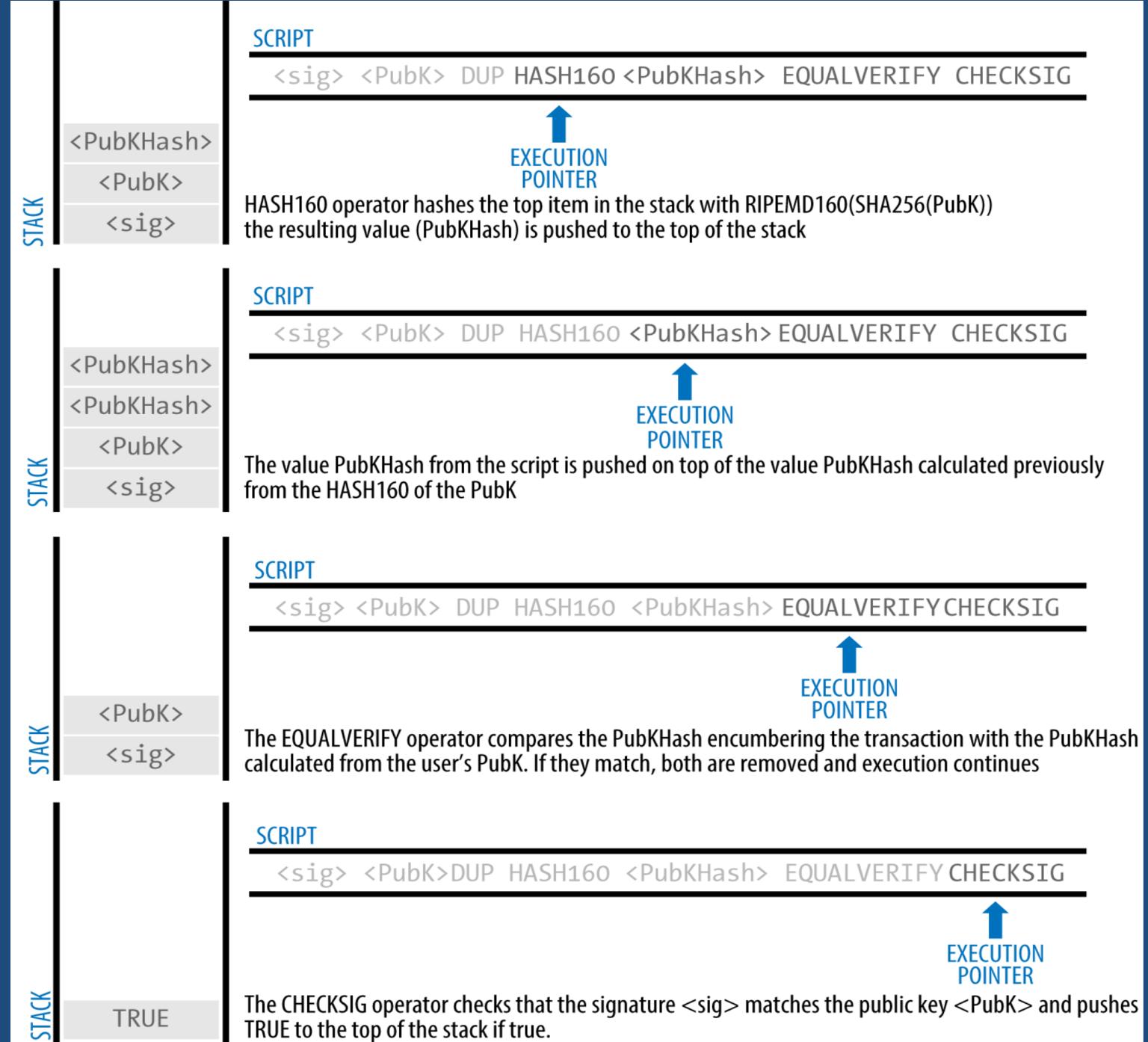
0100000001186f9f998a5aa6f048e51dd8419a14d8a
0f1a8a2836dd734d2804fe65fa35779000000008
b483045022100884d142d86652a3f47ba4746ec
719bbfb040a570b1deccbb6498c75c4ae24cb02
204b9f039ff08df09cbe9f6addac960298cad530a
863ea8f53982c09db8f6e381301410484ecc0d46
f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8
416ab9fe423cc5412336376789d172787ec3457e
ee41c04f4938de5cc17b4a10fa336a8d752adffff
ffff0260e31600000000001976a914ab68025513
c3dbd2f7b92a94e0581f5d50f654e788acd0ef80000
00000001976a9147f9b1a7fb68d60c536c2fd8aea5
3a8f3cc025a888ac00000 000



2.1 Bitcoin / Ethereum Blockchain Technology - Transactions

Pay-to-Public-Key-Hash (P2PKH)

The vast majority of transactions processed on the bitcoin network spend outputs locked with a Pay-to-Public-Key-Hash or “P2PKH” script. These outputs contain a locking script that locks the output to a public key hash, more commonly known as a bitcoin address. An output locked by a P2PKH script can be unlocked (spent) by presenting a public key and a digital signature created by the corresponding private key



2.1 Bitcoin / Ethereum Blockchain Technology – Transactions/Signature

Signature = implementation of the ECDSA algorithm; the “message” being signed is the transaction, or more accurately a hash of a specific subset of the data in the transaction. The signing key is the user’s private key. The result is the Bitcoin/Ethereum signature:

$$Sig = F_{sig}(F_{hash}(m), dA)$$

where:

- dA is the signing private key
- m is the transaction (or parts of it)
- F_{hash} is the hashing function
- F_{sig} is the signing algorithm
- Sig is the resulting signature

In Ethereum’s implementation of ECDSA, the “message” being signed is the transaction, or more accurately, the Keccak256 hash of the RLP-encoded data from the transaction. The signing key is the EOA’s private key. The result is the signature:

$$Sig = F_{sig}(F_{keccak256}(m), k)$$

where:

- k is the signing private key
- m is the RLP-encoded transaction
- $F_{keccak256}$ is the Keccak256 hash function
- F_{sig} is the signing algorithm
- Sig is the resulting signature

2.1 Bitcoin / Ethereum Blockchain Technology – Transactions/Signature

The function F_{sig} produces a signature Sig that is composed of two values, commonly referred to as R and S :

$$Sig = (R, S)$$

Now that the two values R and S have been calculated, they are serialized into a byte-stream using an international standard encoding scheme called the *Distinguished Encoding Rules*, or *DER*.

- `0x30`—indicating the start of a DER sequence
- `0x45`—the length of the sequence (69 bytes)
- `0x02`—an integer value follows
- `0x21`—the length of the integer (33 bytes)
- R —
`00884d142d86652a3f47ba4746ec719bbfb040a570b1
deccbb6498c75c4ae24cb`
- `0x02`—another integer follows
- `0x20`—the length of the integer (32 bytes)
- S —
`4b9f039ff08df09fbe9f6addac960298cad530a863ea8
f53982c09db8f6e3813`
- A suffix (`0x01`) indicating the type of hash used (`SIGHASH_ALL`)

2.1 Bitcoin / Ethereum Blockchain Technology – Transactions/Block

- The blockchain data structure is an ordered, back-linked list of blocks of transactions.
- The blockchain can be stored as a flat file, or in a simple database.
- A block is a container data structure that aggregates transactions for inclusion in the public ledger, the blockchain.
- The block header is 80 bytes, whereas the average transaction is at least 400 bytes and the average block contains more than 1900 transactions.
- A complete block, with all transactions, is therefore 10,000 times larger than the block header.

2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain

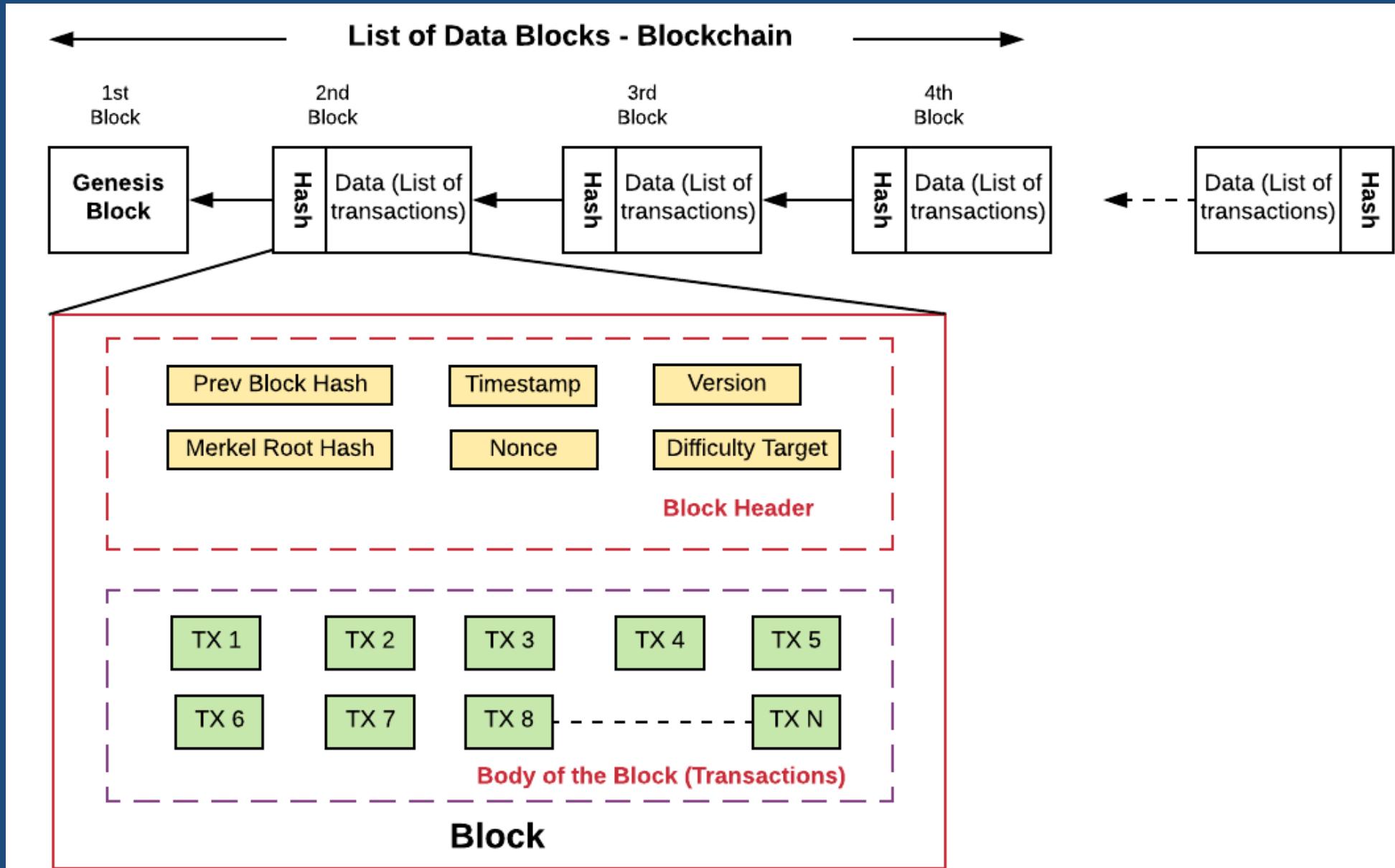
A. The block structure

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header
1–9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

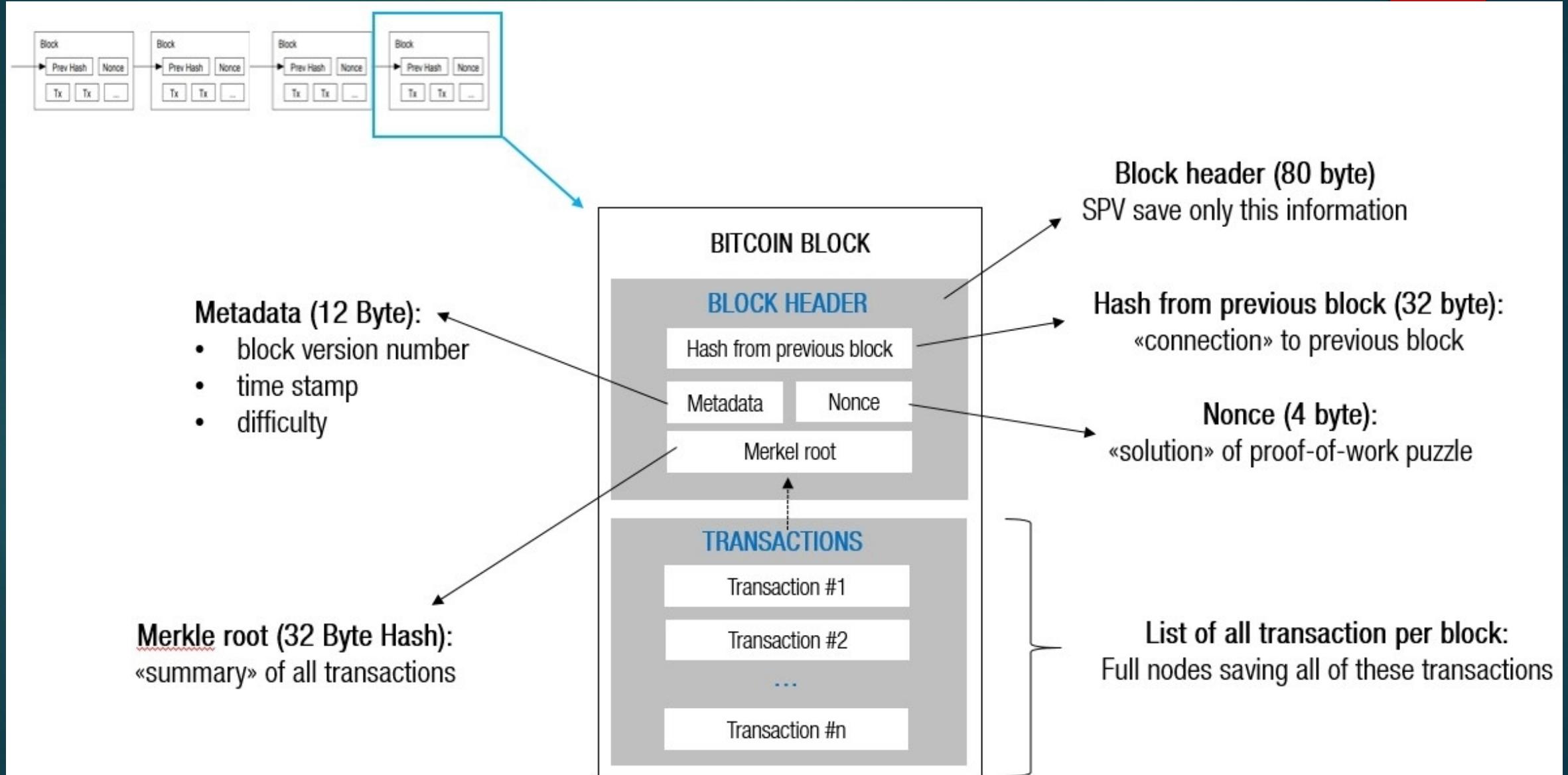
B. The Block Header structure

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the merkle tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Difficulty Target	The Proof-of-Work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the Proof-of-Work algorithm

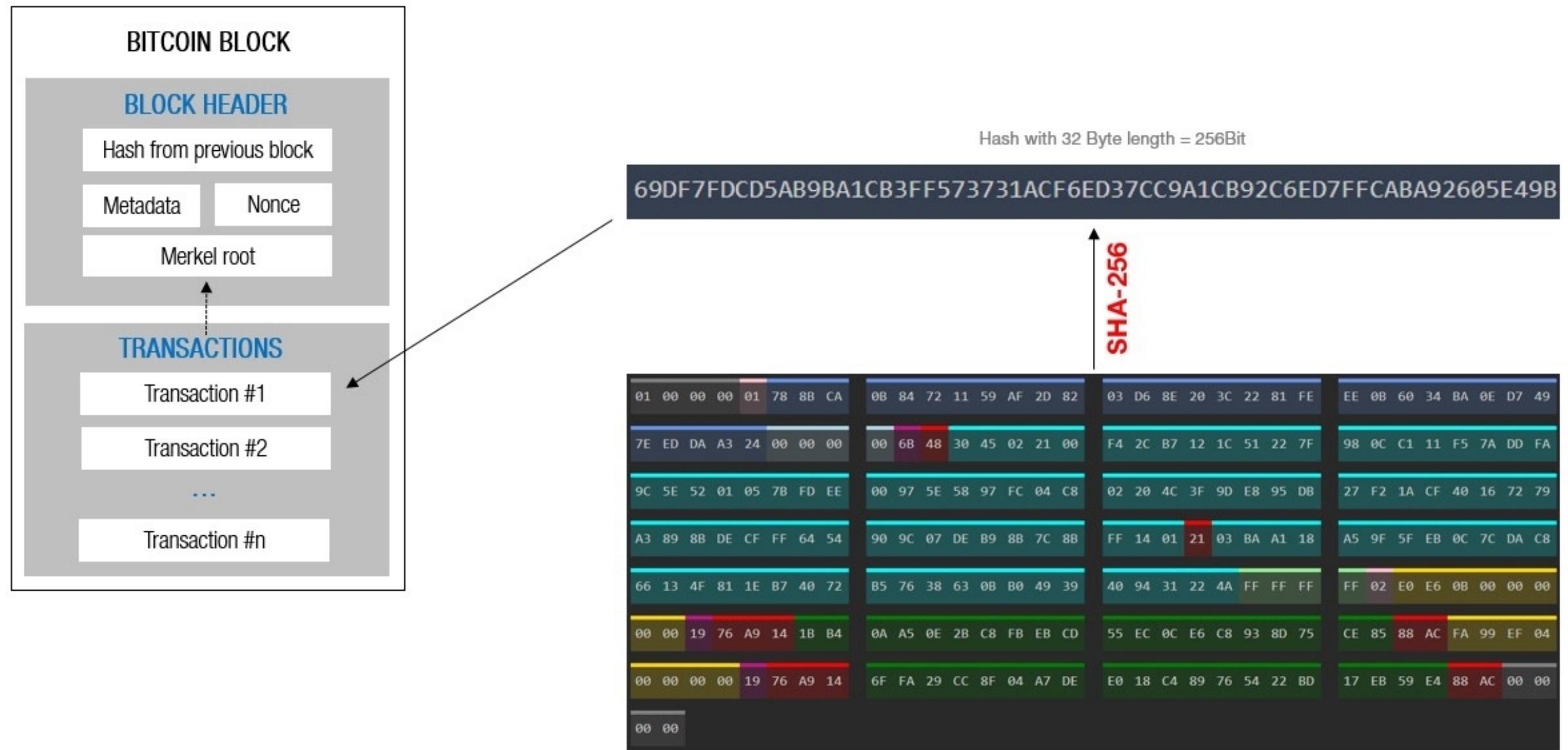
2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain



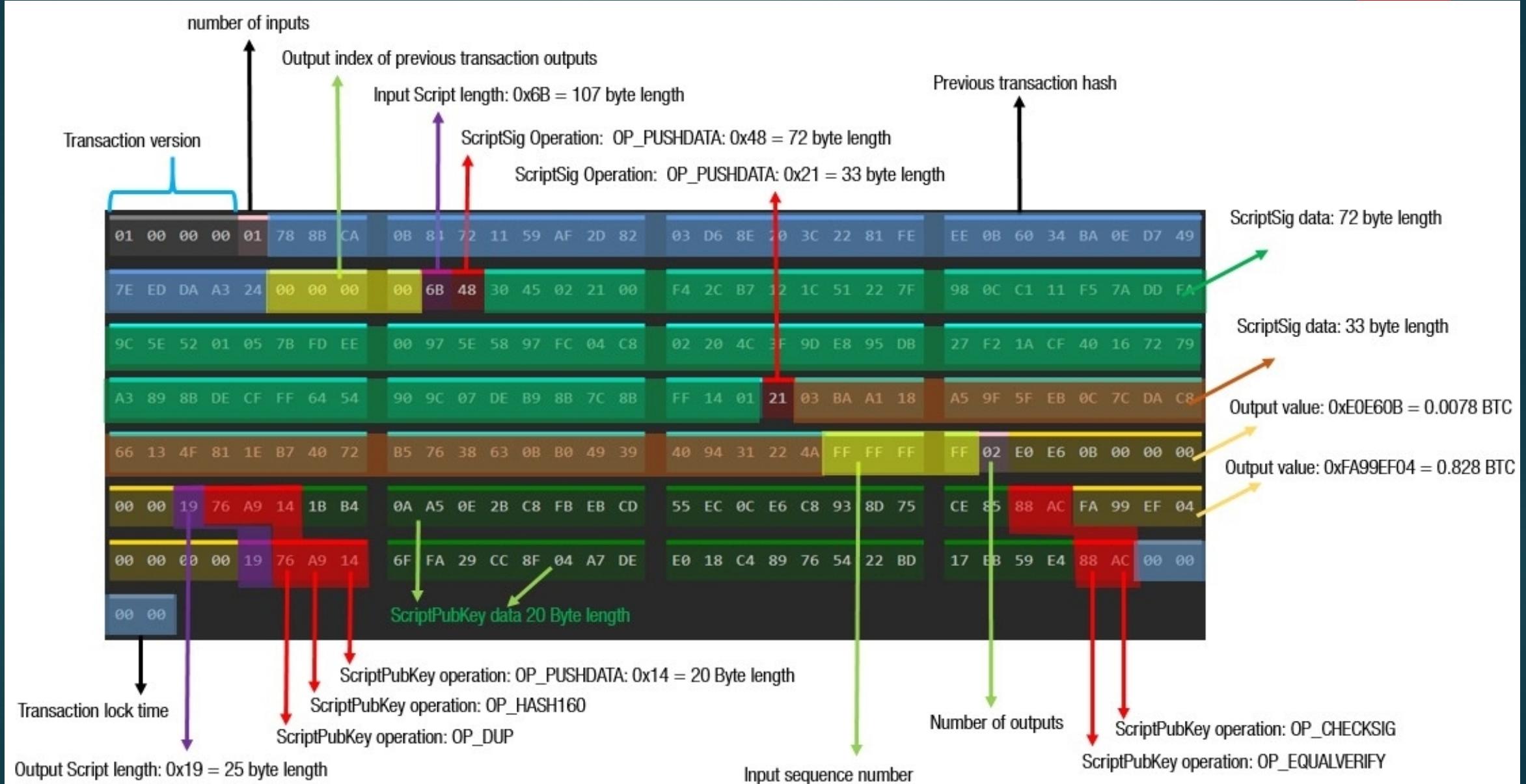
2.1 Bitcoin Block from the chain



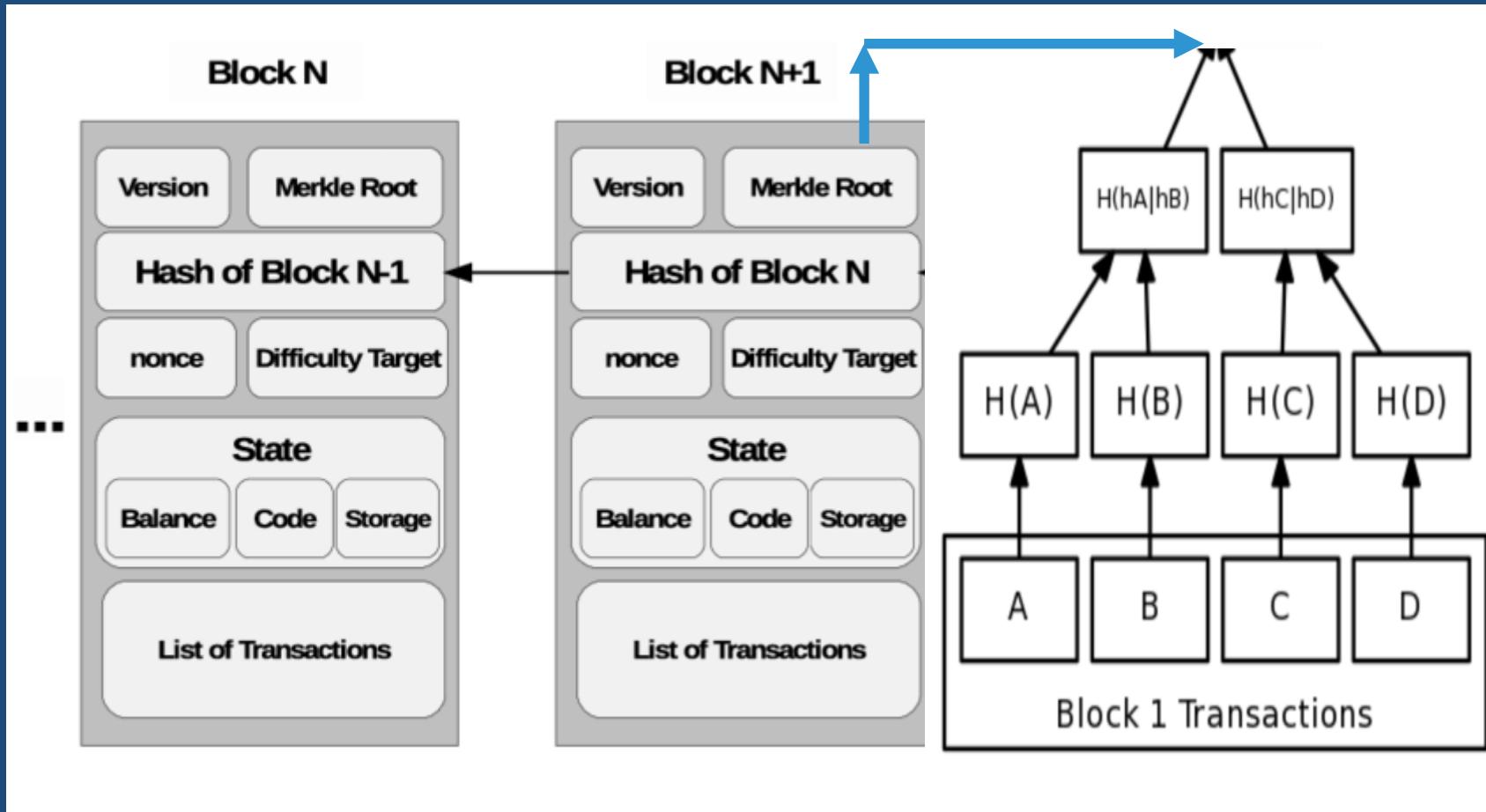
2.1 Bitcoin - Method from a raw Bitcoin HEX transaction into a block



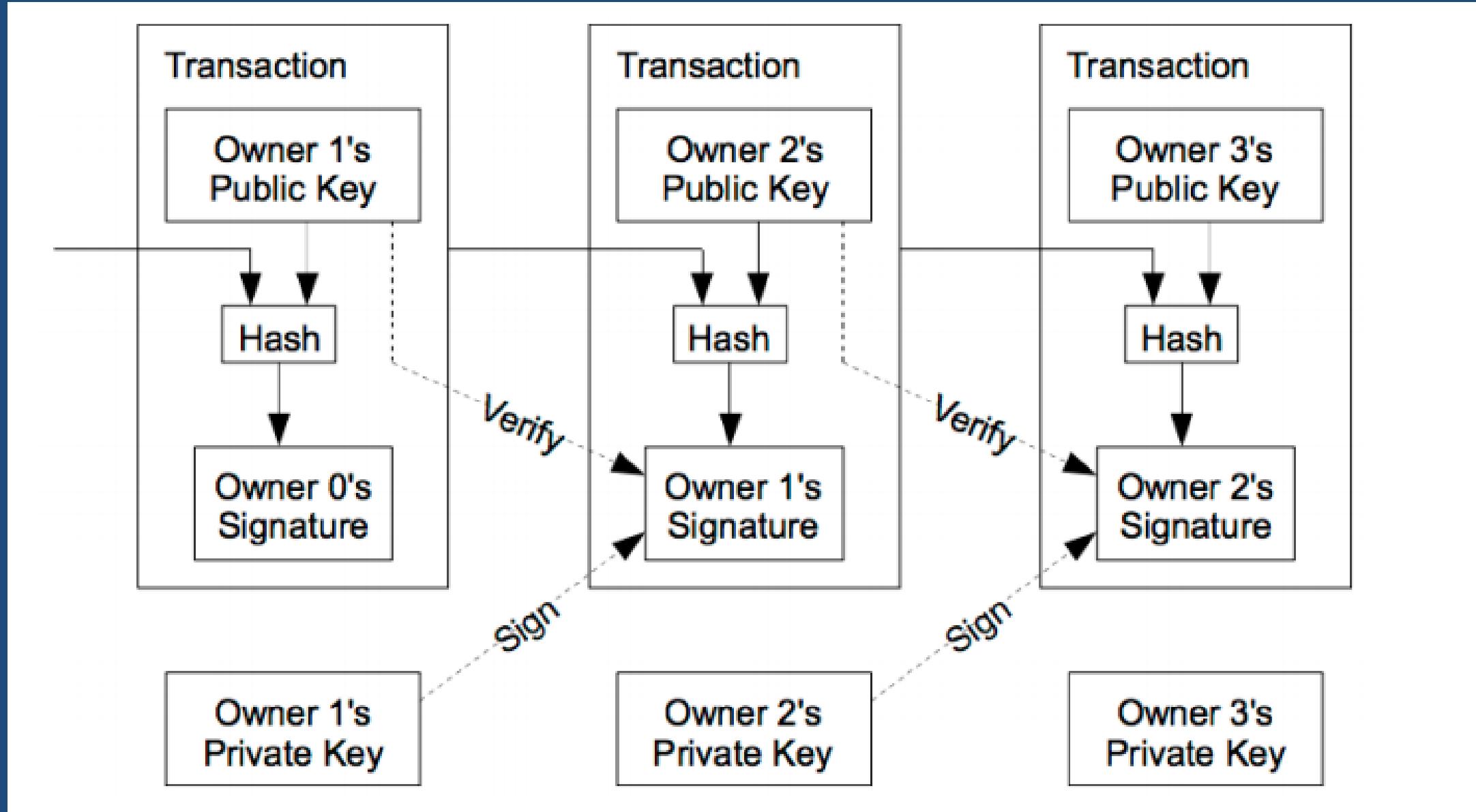
2.1 Detailed analysis of a raw Bitcoin transaction



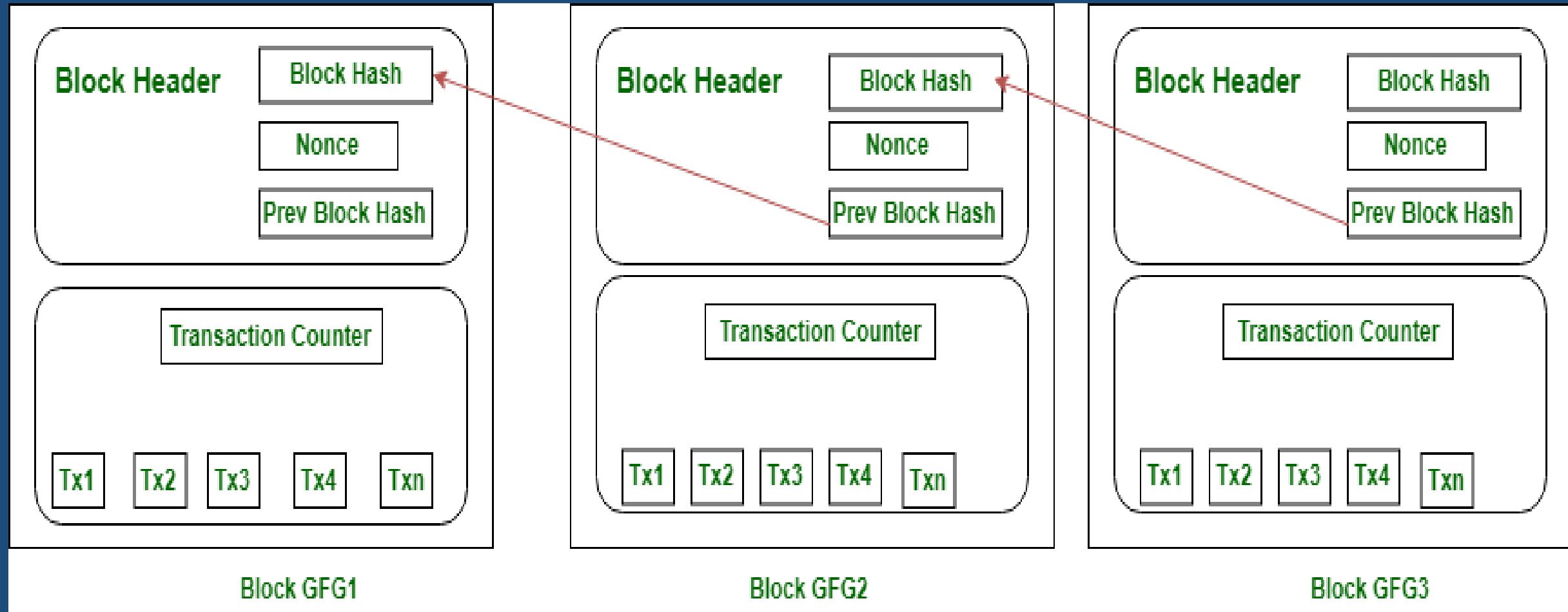
2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain



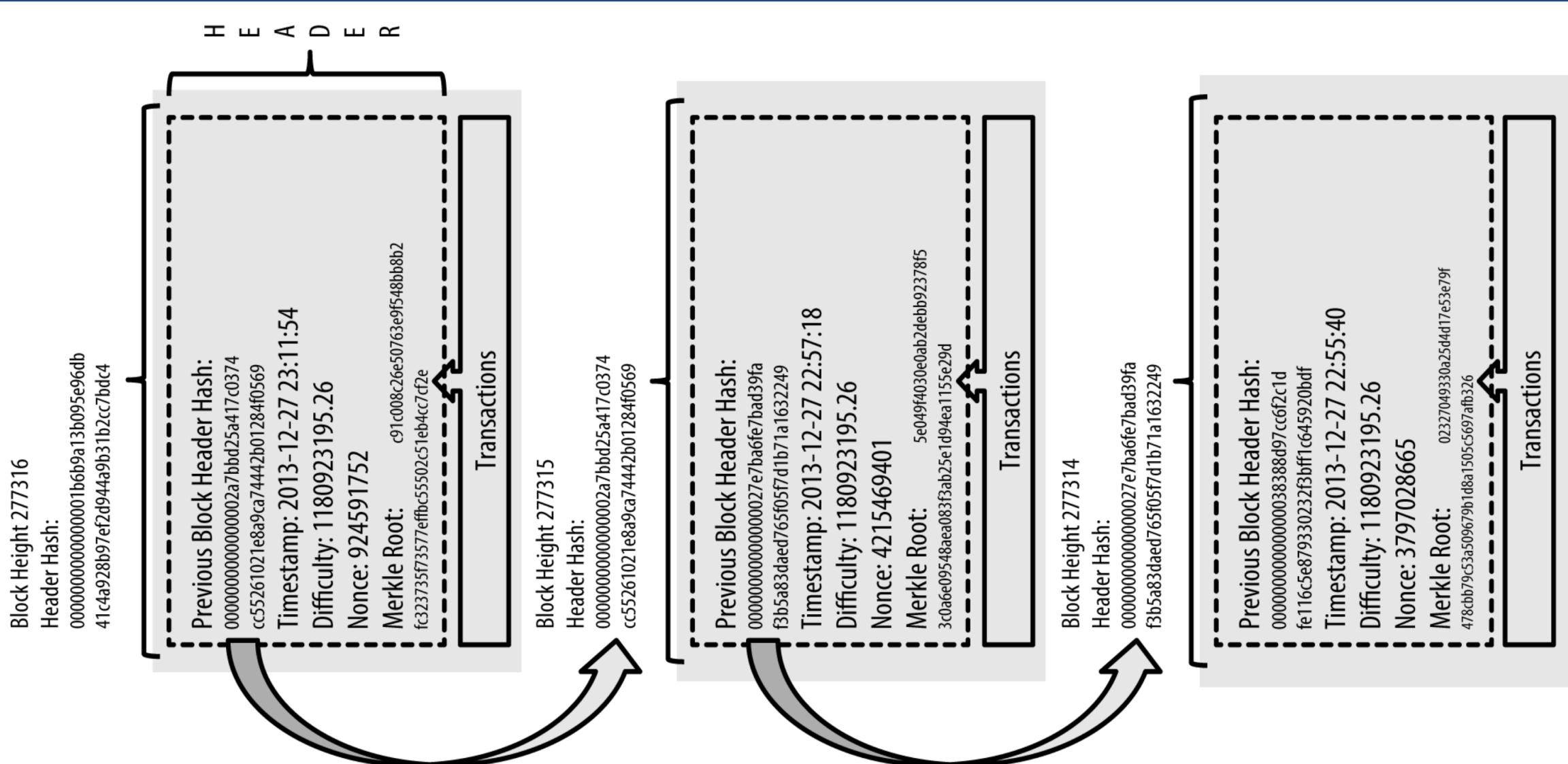
2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain



2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain

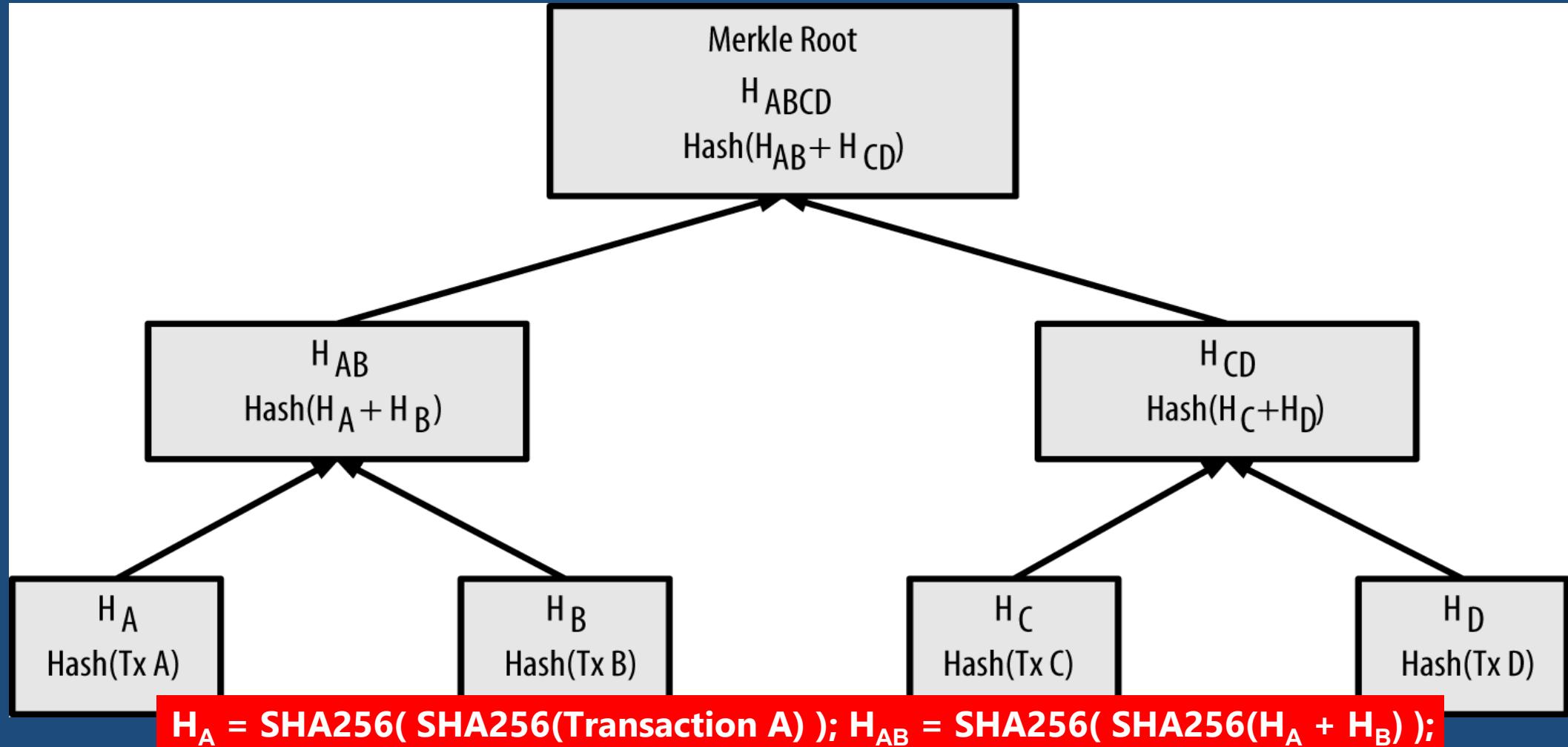


2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain

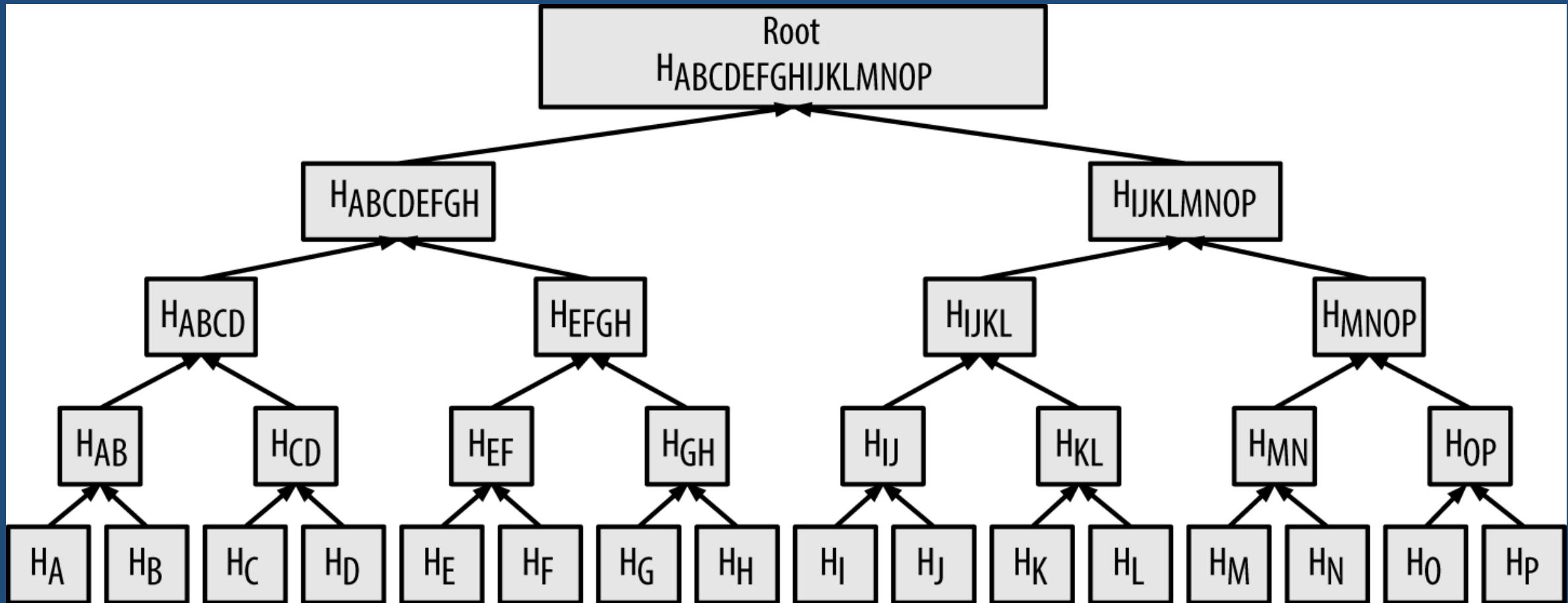


2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain – Transactions Hashes in Merkle Trees

Merkle trees are used in bitcoin to summarize all the transactions in a block, producing an overall digital fingerprint of the entire set of transactions, providing a very efficient process to verify whether a transaction is included in a block. The Merkle Tree is constructed bottom-up. If we have 4 transactions, A, B, C, and D, which form the leaves of the Merkle Tree:

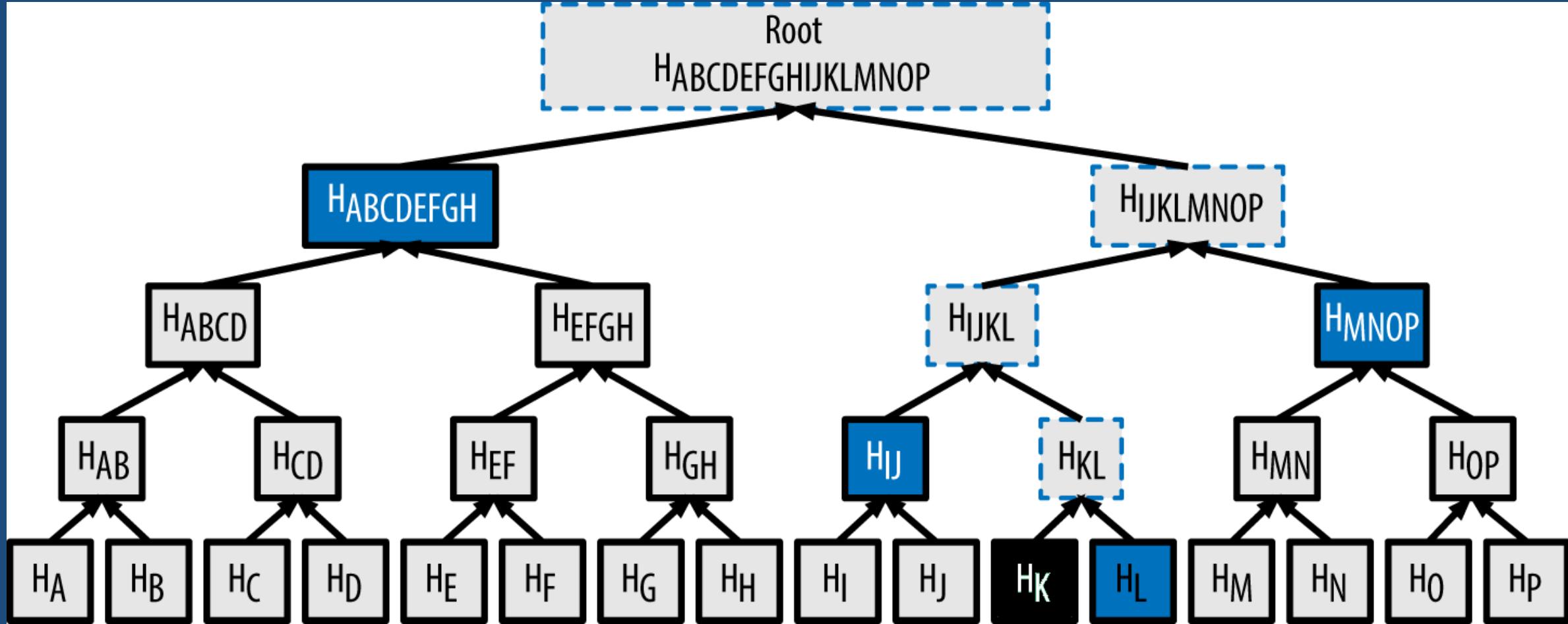


2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain – Transactions Hashes in Merkle Trees



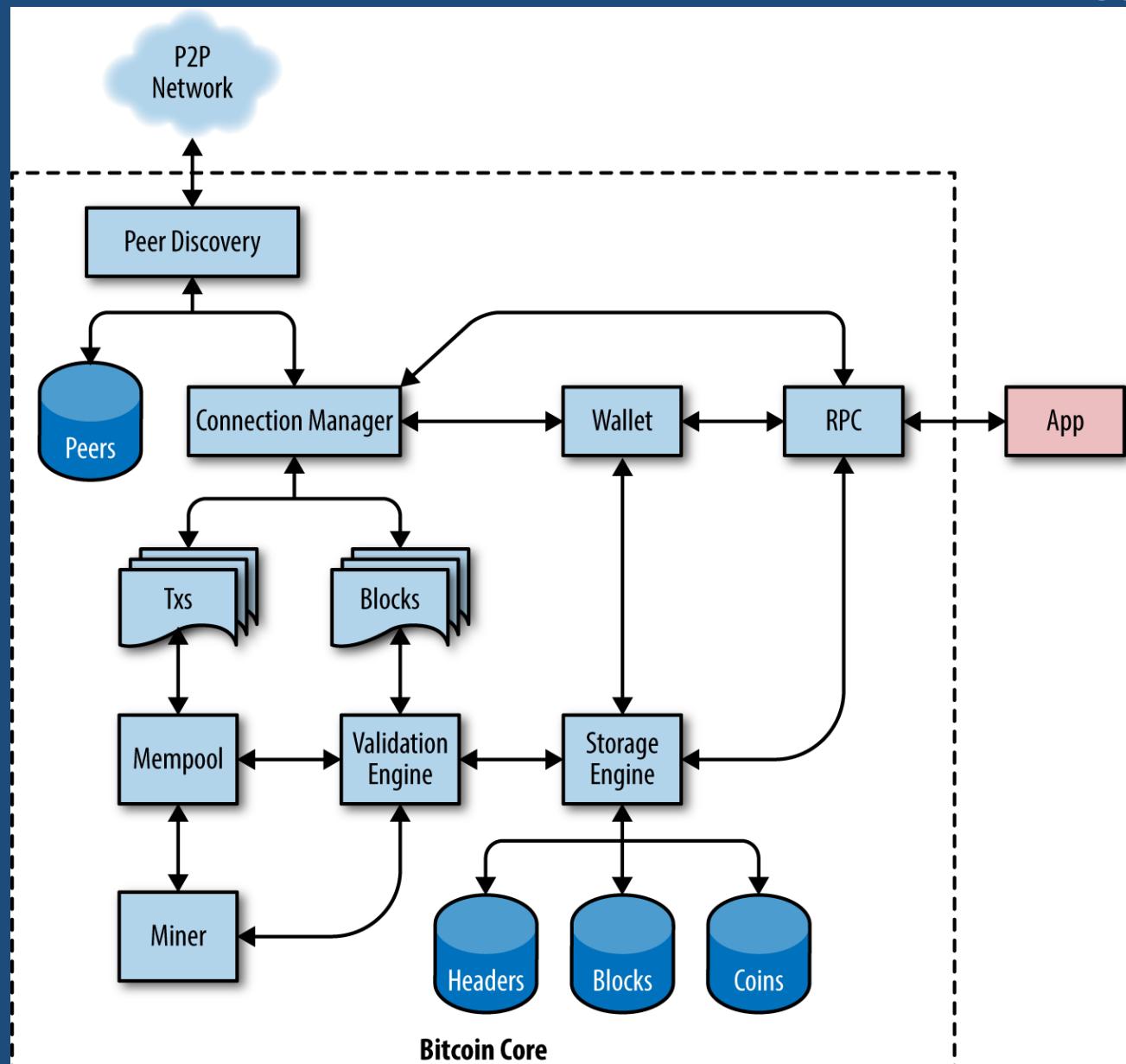
This tree is built from hashes for 16 transactions. Although the root looks bigger than the leaf nodes in the diagram, it is the exact same size, just 32 bytes.

2.1 Bitcoin / Ethereum Blockchain Technology – Blockchain – Transactions Hashes in Merkle Trees



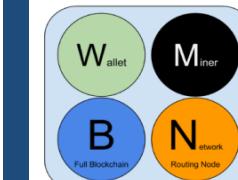
A node can prove that a transaction K is included in the block by producing a Merkle path that is only four 32-byte hashes long (128 bytes total). The path consists of the four hashes H_L , H_{IJ} , H_{MNOP} , and $H_{ABCDEFGH}$. With those 4 hashes provided as an authentication path, any node can prove that H_K (with a black background at the bottom of the diagram) is included in the Merkle root by computing 4 additional pairwise hashes H_{KL} , H_{IJKL} , $H_{IJKLMNOP}$, and the Merkle tree root (outlined in a dashed line in the diagram).

2.1 Bitcoin / Ethereum Blockchain Technology – Reference Implementation



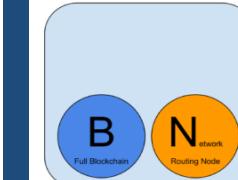
Reference Client (Bitcoin Core)

Contains a Wallet, Miner, full Blockchain database, and Network routing node on the bitcoin P2P network.



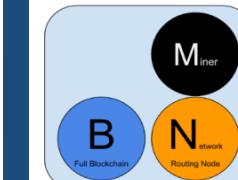
Full Block Chain Node

Contains a full Blockchain database, and Network routing node on the bitcoin P2P network.



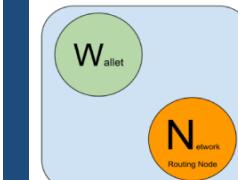
Solo Miner

Contains a mining function with a full copy of the blockchain and a bitcoin P2P network routing node.



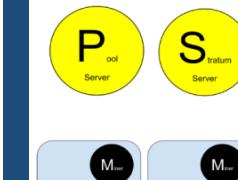
Lightweight (SPV) wallet

Contains a Wallet and a Network node on the bitcoin P2P protocol, without a blockchain.



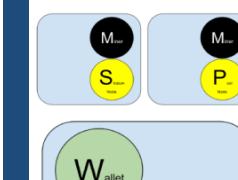
Pool Protocol Servers

Gateway routers connecting the bitcoin P2P network to nodes running other protocols such as pool mining nodes or Stratum nodes.



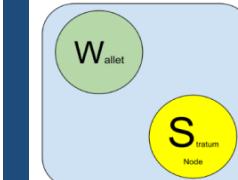
Mining Nodes

Contain a mining function, without a blockchain, with the Stratum protocol node (S) or other pool (P) mining protocol node.



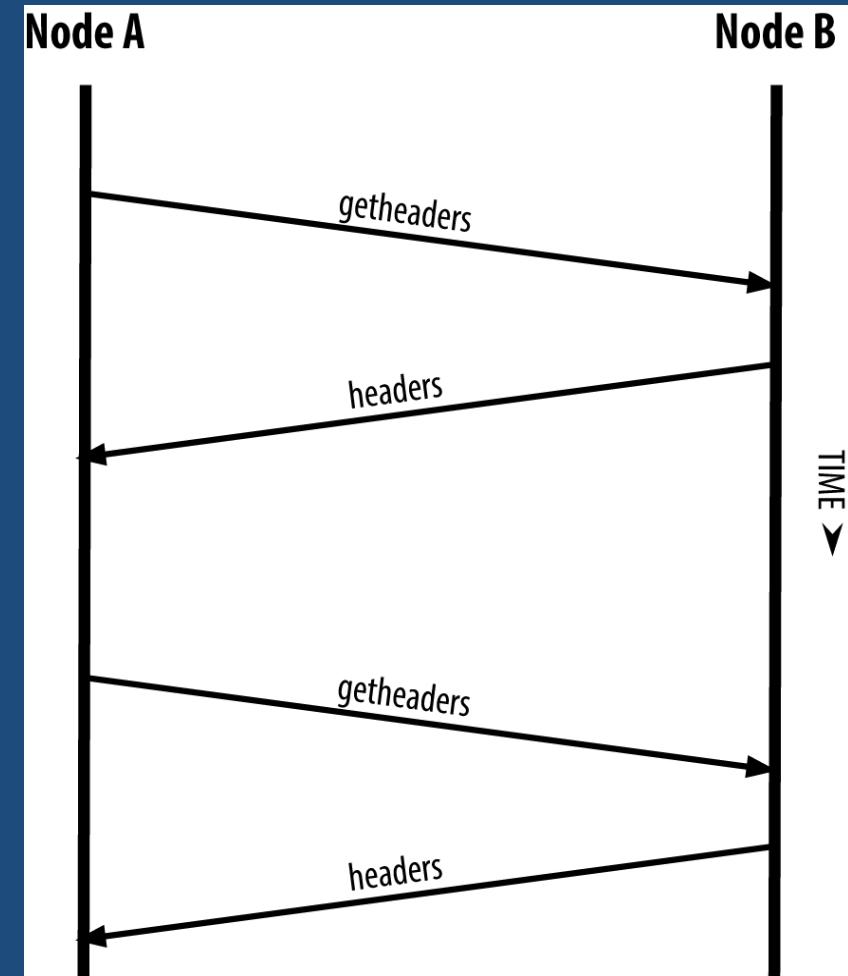
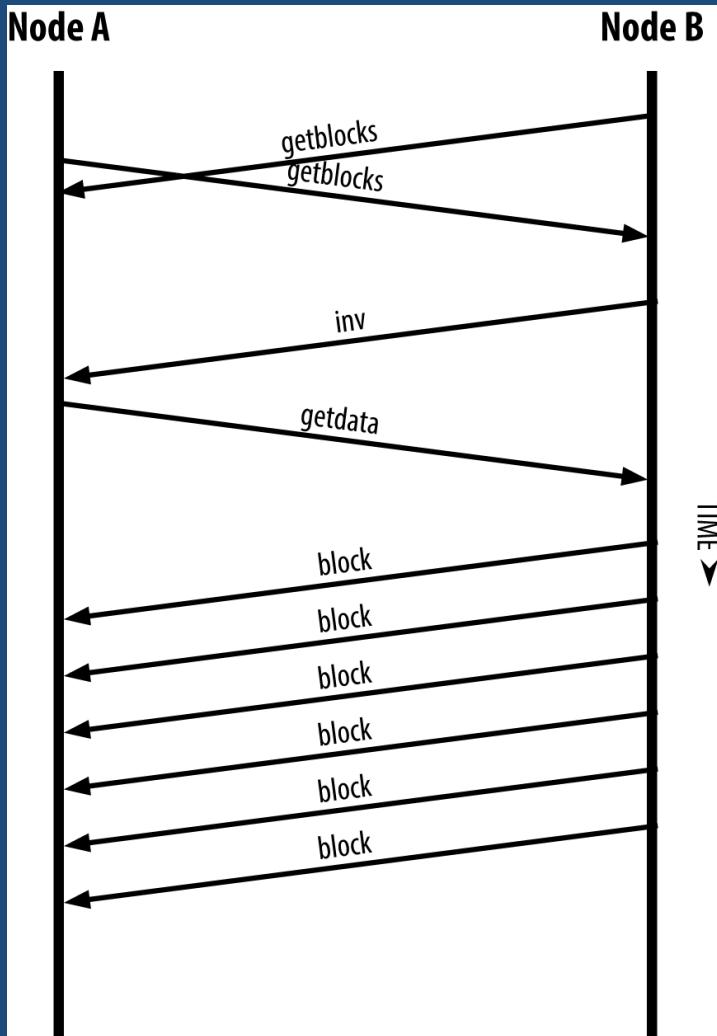
Lightweight (SPV) Stratum wallet

Contains a Wallet and a Network node on the Stratum protocol, without a blockchain.



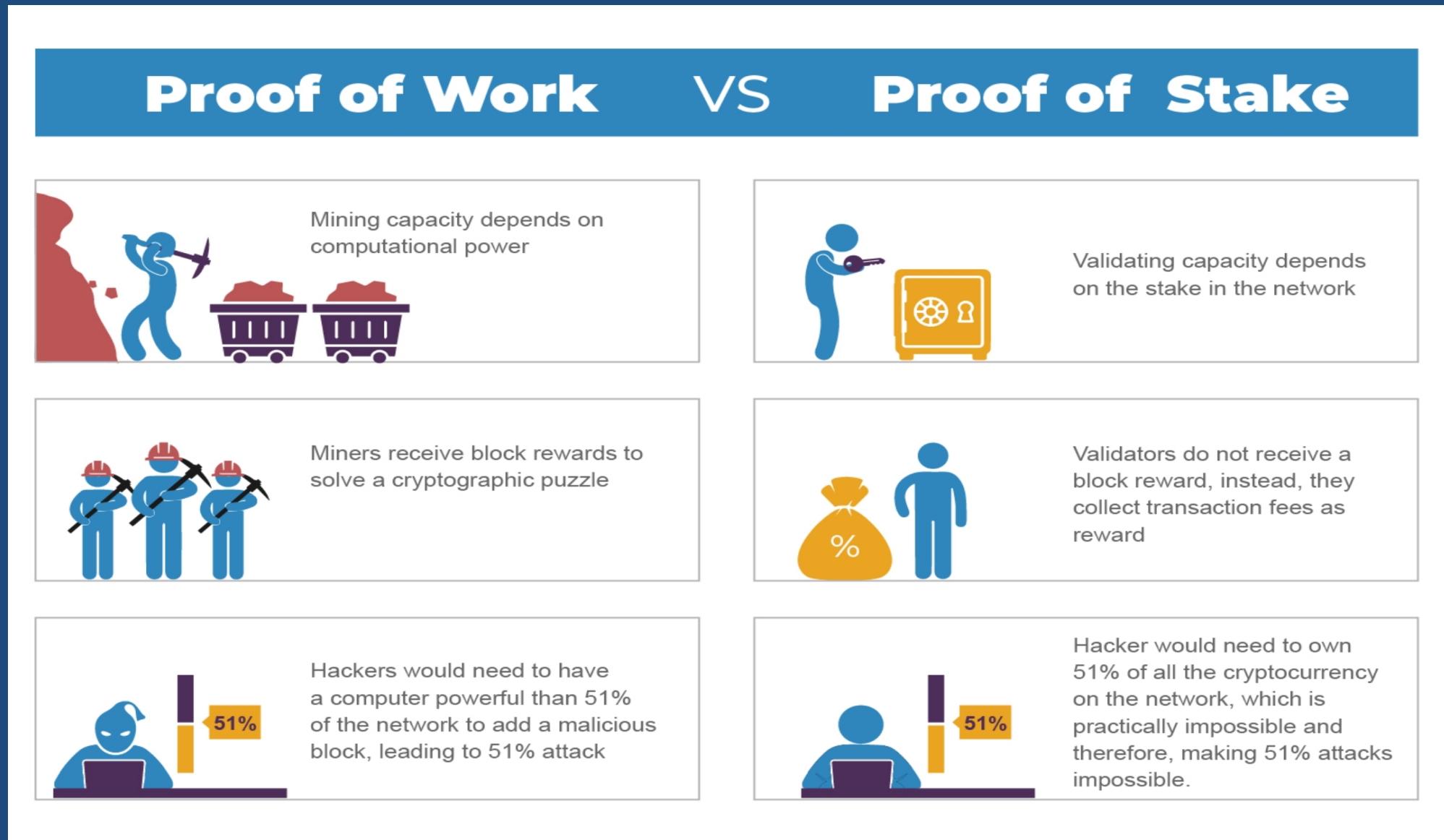
2.1 Bitcoin / Ethereum Blockchain Technology – P2P Network

Node synchronizing the blockchain by retrieving blocks from a peer



SPV node synchronizing the block headers

2.1 Consensus Algo - Bitcoin / Ethereum Blockchain Technology – P2P Network



The proof of work (POW) is a common **consensus algorithm** used by the most popular **cryptocurrency** networks like **bitcoin** and **litecoin**. It requires a participant node to prove that the work done and submitted by them qualifies them to receive the right to add new transactions to the **blockchain**.

2.1 Bitcoin Standards

 luke-jr Merge pull request #740 from skddc/patch-1	...	Latest commit 954df0d on 14 Dec 2018
 bip-0001	Fix formatting	5 years ago
 bip-0002	Add obsolete status to process image	2 years ago
 bip-0008	Amend BIP8 by height	2 years ago
 bip-0009	BIP 9: Misplaced table cells typo	11 months ago
 bip-0016	fix bip-0016 link 404	4 months ago
 bip-0032	Fix formatting	5 years ago
 bip-0039	Fix two errors in the BIP 39 French wordlist	a year ago
 bip-0042	Include image for BIP42	5 years ago
 bip-0047	BIP-0047: Reusable payment codes	4 years ago
 bip-0068	Improve title, add encoding diagram and small fixup	3 years ago
 bip-0069	add EOF newlines per @luke-jr	3 years ago
 bip-0070	Put BIP 75 in the right place in README, and clean up formatting a bit	3 years ago
 bip-0073	Fix formatting	5 years ago
 bip-0075	- Update identifier comment in paymentrequest.proto	2 years ago
 bip-0098	BIP-0098: Fast Merkle Trees	a year ago
 bip-0114	BIP114: MAST proposal v2	3 years ago
 bip-0122	Assign BIP 122	3 years ago
 bip-0135	Assign BIP 135: Generalized version bits voting	2 years ago
 bip-0144	BIP141: commitment clarification. BIP144: new diagram	3 years ago
 bip-0152	header message can also get replied by getdata (CMPCT_BLOCK)	3 years ago
 bip-0156	renamed files and updated README.mediawiki index	5 months ago
 bip-0158	BIP 158: Add more cases to test vectors.	5 months ago
 bip-0174	BIP 174 workflow graphics	7 months ago
 scripts	scripts/buildtable: Support License-Code header	a year ago



Ethereum is often described as “the world computer.”

From a more practical perspective, Ethereum is:

- an open source,
- globally decentralized computing infrastructure that executes programs called ***smart contracts***.
- it uses a blockchain to synchronize and store the system's state changes, along with a cryptocurrency called ***Ether*** to meter and constrain execution resource costs.

The Ethereum platform enables developers to build powerful decentralized applications with built-in economic functions. While providing high availability, auditability, transparency, and neutrality, it also reduces or eliminates censorship and reduces certain counterparty risks.

#Ethereum



- **Similar:**

Ethereum shares many common elements with other open blockchains – e.g. Bitcoin:

- a **(P2P)** peer-to-peer network connecting participants;
- a **Byzantine fault-tolerant consensus algorithm** for synchronization of state updates (a **Proof-of-Work** blockchain);
- the use of cryptographic primitives such as digital signatures and hashes, and a digital currency (**Ether**).

- **Not similar:**

Unlike Bitcoin, which has a very limited scripting language, Ethereum is designed to be:

- a general-purpose programmable blockchain that runs a virtual machine (EMV) capable of executing code of arbitrary and unbounded complexity.
- Where Bitcoin's Script language is, intentionally, constrained to simple true/false evaluation of spending conditions,
 - **Ethereum's language is Turing complete, meaning that Ethereum can straightforwardly function as a general-purpose computer.**

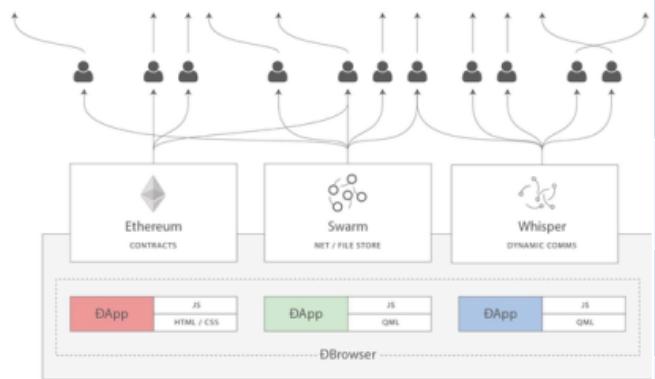
#Ethereum

Components of a Blockchain: The components of an open, public blockchain are (usually):

- A **peer-to-peer (P2P)** network connecting participants and propagating transactions and blocks of verified transactions, based on a standardized “gossip” protocol
- **Messages**, in the form of transactions, representing state transitions
- A **set of consensus rules**, governing what constitutes a transaction and what makes for a valid state transition
- A **state machine** that processes transactions according to the consensus rules
- A **chain of cryptographically secured blocks** that acts as a journal of all the verified and accepted state transitions
- A **consensus algorithm** that decentralizes control over the blockchain, by forcing participants to cooperate in the enforcement of the consensus rules
- A **game-theoretically sound incentivization scheme (e.g., proof-of-work costs plus block rewards)** to economically secure the state machine in an open environment
- One or more **open source software implementations** of the above (“clients”) - All or most of these components are usually combined in a single software client.

* For example, in Bitcoin, the reference implementation is developed by the *Bitcoin Core* open source project and implemented as the *bitcoind* client. In Ethereum, rather than a reference implementation there is a *reference specification*, a mathematical description of the system in the Yellow Paper

#Ethereum



Blockchain Components	Ethereum
P2P Network	Ethereum runs on the <i>Ethereum main network</i> , which is addressable on TCP port 30303, and runs a protocol called <i>DΞVp2p</i> . The dynamic communication between the peers is implemented with <i>Whisper</i> . Whisper is a part of the Ethereum P2P protocol suite that allows for messaging between users via the same network that the blockchain runs on.
Consensus rules	Ethereum's consensus rules are defined in the reference specification, the Yellow Paper - https://ethereum.github.io/yellowpaper/paper.pdf
Transactions	Ethereum transactions are network messages that include (among other things) a sender, recipient, value, and data payload.
State machine	Ethereum state transitions are processed by the Ethereum Virtual Machine (EVM) , a stack-based virtual machine that executes <i>bytecode</i> (machine-language instructions). EVM programs , called " smart contracts ," are written in high-level languages (e.g., Solidity or Vyper) and compiled to bytecode for execution on the EVM.
Data structures	Ethereum's state is stored locally on each node as a <i>database</i> (usually Google's LevelDB), which contains the transactions and system state in a serialized hashed data structure called a <i>Merkle Patricia Tree</i> . Swarm or IPFS may be used to store the Solidity smart contracts files.
Consensus algorithm	Ethereum uses Bitcoin's consensus model, Nakamoto Consensus, which uses sequential single-signature blocks, weighted in importance by PoW to determine the longest chain and therefore the current state. However, there are plans to move to a PoS weighted voting system, codenamed Casper , in the near future.
Economic security	Ethereum currently uses a PoW algorithm called Ethash, but this will eventually be dropped with the move to PoS at some point in the future.
Clients	Ethereum has several interoperable implementations of the client software, the most prominent of which are Go-Ethereum (Geth) and Parity .

#Ethereum

Further Reading- The following references provide additional information on the technologies mentioned here:

- **Framework and consensus rules:**
 - The Ethereum Yellow Paper: <https://ethereum.github.io/yellowpaper/paper.pdf>
 - The Beige Paper, a rewrite of the Yellow Paper for a broader audience in less formal language: <https://github.com/chronaeon/beigepaper>
- **P2P Network:**
 - DΞVp2p network protocol: <http://bit.ly/2quAlTE>
- **State Machine:**
 - Ethereum Virtual Machine list of resources: <http://bit.ly/2PmtjiS>
- **Data structures:**
 - LevelDB database (used most often to store the local copy of the blockchain): <http://leveldb.org>
 - Merkle Patricia trees: <https://github.com/ethereum/wiki/wiki/Patricia-Tree>
- **Consensus Algorithms:**
 - Ethash PoW algorithm: <https://github.com/ethereum/wiki/wiki/Ethash>
 - Casper PoS v1 Implementation Guide: <http://bit.ly/2DyPr3l>
- **Clients:**
 - Go-Ethereum (Geth) client: <https://geth.ethereum.org/>
 - Parity Ethereum client: <https://parity.io/>

#Ethereum

From General-Purpose Blockchains to Decentralized Applications (DApps / DApps)

Ethereum started as a way to make a general-purpose blockchain that could be programmed for a variety of uses. But very quickly, Ethereum's vision expanded to become a platform for programming DApps. DApps represent a broader perspective than smart contracts. A DApp is, at the very least, a smart contract and a web user interface.

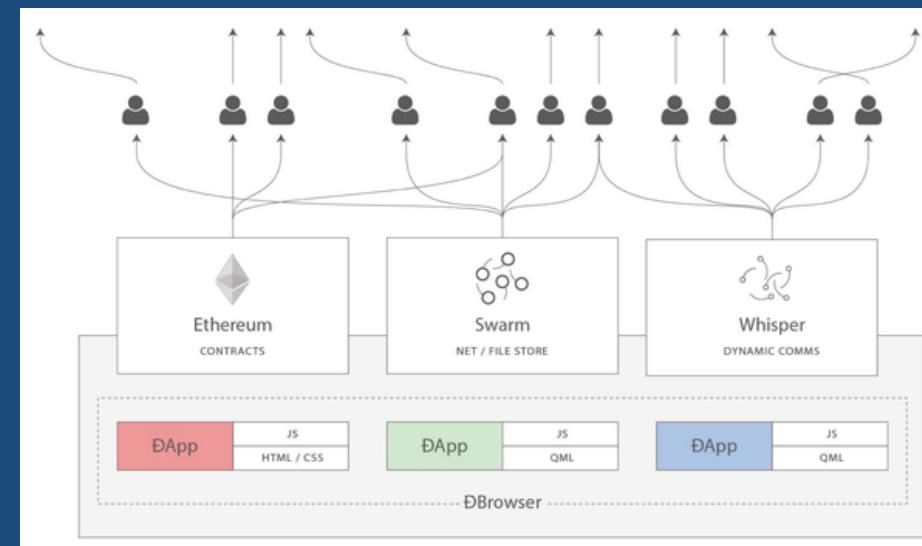
More broadly, a DApp is a web application that is built on top of open, decentralized, peer-to-peer infrastructure services.

A DApp is composed of at least:

- Smart contracts on a blockchain
- A web frontend user interface

In addition, many DApps include other decentralized components, such as:

- A decentralized (P2P) storage protocol and platform – Swarm
- A decentralized (P2P) messaging protocol and platform – Whisper



#Ethereum – P2P Networks

P2P Name	Description
Main Ethereum Network	The main public Ethereum blockchain. Real ETH, real value, and real consequences.
Ropsten Test Network	Ethereum public test blockchain and network. ETH on this network has no value.
Kovan Test Network	Ethereum public test blockchain and network using the Aura consensus protocol with proof of authority (federated signing). ETH on this network has no value. The Kovan test network is supported by Parity only. Other Ethereum clients use the Clique consensus protocol, which was proposed later, for proof of authority-based verification.
Rinkeby Test Network	Ethereum public test blockchain and network, using the Clique consensus protocol with proof of authority (federated signing). ETH on this network has no value.
localhost 8545	Connects to a node running on the same computer as the browser. The node can be part of any public blockchain (main or testnet), or a private testnet.
Custom RPC	Allows you to connect MetaMask to any node with a Geth-compatible Remote Procedure Call (RPC) interface. The node can be part of any public or private blockchain.

#Ethereum

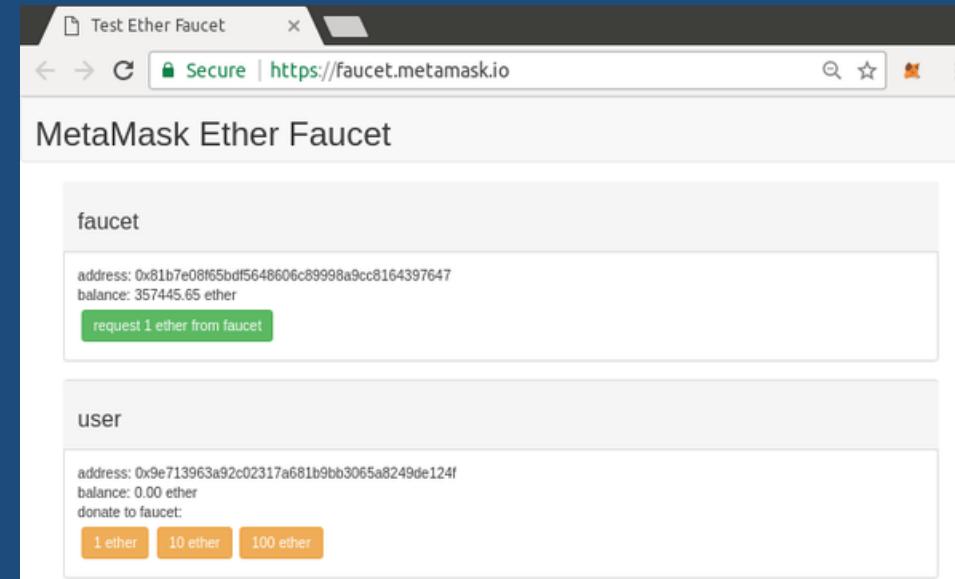
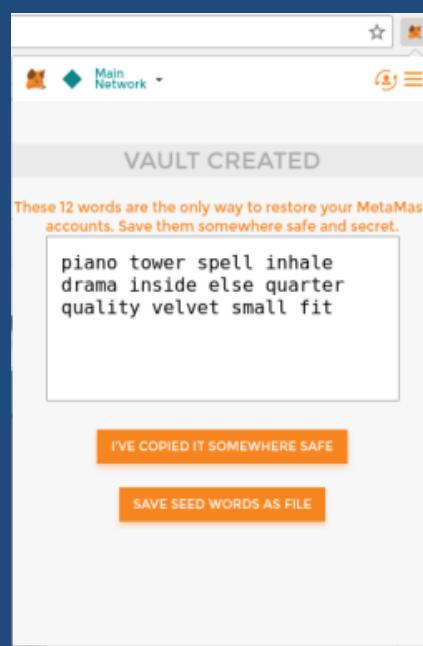
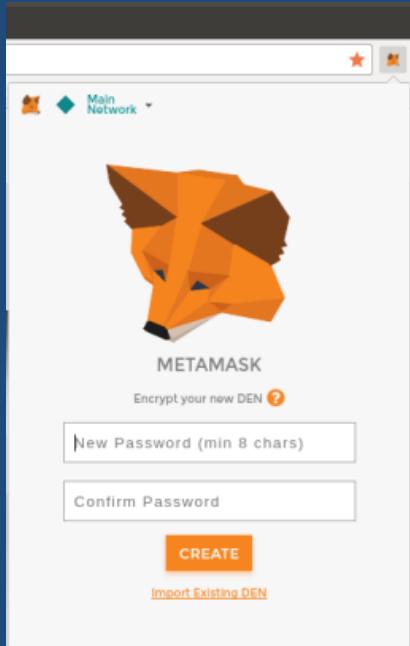
Ether denominations and unit names

	Ethereum Wallets
MetaMask https://metamask.io/	<p>It is a browser extension wallet that runs in your browser (Chrome, Firefox, Opera, or Brave Browser). It is easy to use and convenient for testing, as it is able to connect to a variety of Ethereum nodes and test blockchains.</p> <p>MetaMask is a web-based wallet.</p>
Jaxx	<p>It is a multiplatform and multicurrency wallet that runs on a variety of operating systems, including Android, iOS, Windows, macOS, and Linux. It is often a good choice for new users as it is designed for simplicity and ease of use. Jaxx is either a mobile or a desktop wallet, depending on where you install it.</p>

Value (in wei)	Exponent	Common name	SI name
1	1	wei	Wei
1,000	10^3	Babbage	Kilowei or femtoether
1,000,000	10^6	Lovelace	Megawei or picoether
1,000,000,000	10^9	Shannon	Gigawei or nanoether
1,000,000,000,000	10^{12}	Szabo	Microether or micro
1,000,000,000,000,000	10^{15}	Finney	Milliether or milli
1,000,000,000,000,000,000	10^{18}	Ether	Ether
1,000,000,000,000,000,000,000	10^{21}	Grand	Kiloether
1,000,000,000,000,000,000,000,000	10^{24}		Megaether

#Ethereum – DEMO with MetaMask and Remix IDE in Mozilla Firefox

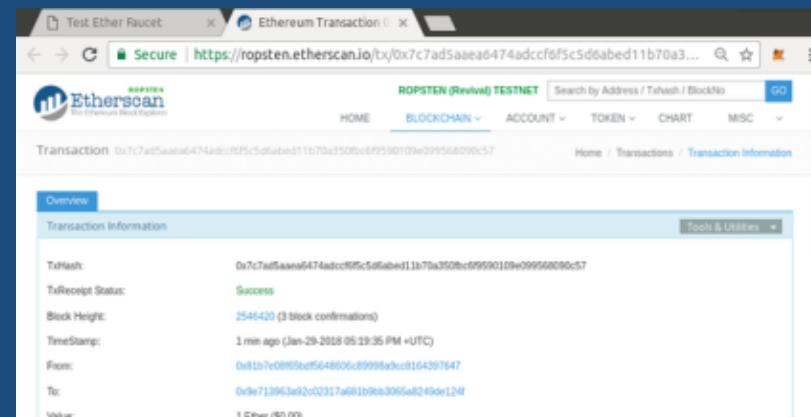
D1. MetaMask Password and Mnemonic setup D2. MetaMask buy Ether from Ropsten Test Network's Faucet



There are two types of accounts:

- **Externally owned account (EOAs):** an account controlled by a private key, and if you own the private key associated with the EOA you have the ability to send ether and messages from it.
- **Contract:** an account that has its own code, and is controlled by code.

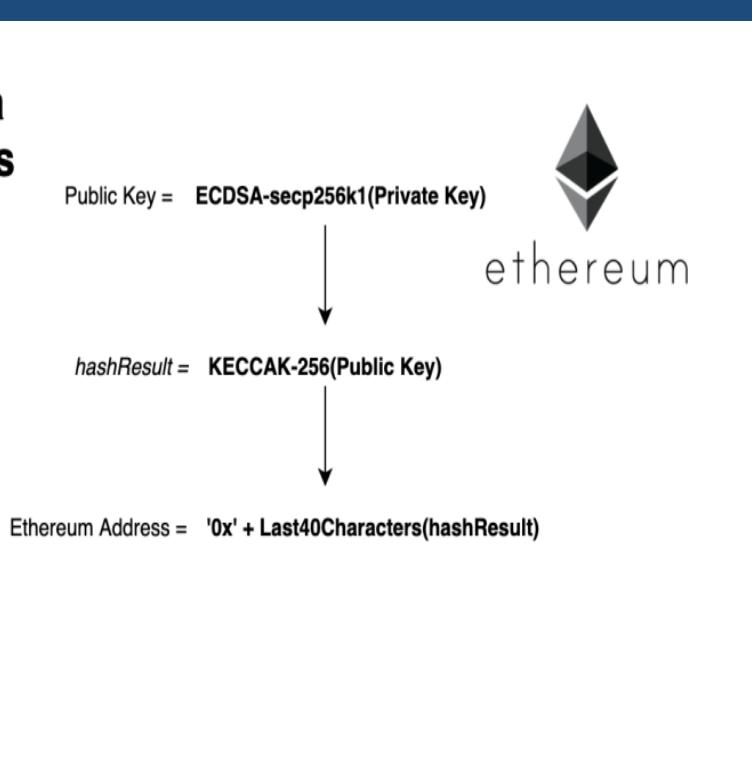
D3. Transaction in RopstenEtherscan.io



#Ethereum – Wallet Address Generation is (a little bit) different than Bitcoin

How to generate a Ethereum Address

Ethereum addresses can be created using open-source libraries based on algorithmic hash functions. More information about Ethereum address generation at <http://mycrypto.tools>



1 - Generate Private Key using open source libraries such as **Ethereumj** or **SHA256** hash function with a randomly generated number.

0: Private Key:

09e910621c2e988e9f7f6ffcd7024f54ec1461fa6e86a4b545e9e1fe21c28866

2. - Generate Public key using ECDSA – secp256k1 – algorithm applied to Private Key. The public key is a point on the Elliptic Curve Algorithm. It has x and y coordinates which are used to create Ethereum address.

2.1: Public Key:

**048e66b3e549818ea2cb354fb70749f6c8de8fa484f7530fc447d5fe80a1c424e4
f5ae648d648c980ae7095d1efad87161d83886ca4b6c498ac22a93da5099014a**

-Apply keccak256 – Ethereum hash function – using x and y coordinates to create Ethereum address.

2.2: Ethereum Address: 0x00B54E93EE2EBA3086A55F4249873E291D1AB06C
See address on Ethereum

Blockchain **0x00B54E93EE2EBA3086A55F4249873E291D1AB06C**

Dev Overview – it is modified as implementation but it is of the ETH creator:

<https://eth.wiki/en/howto/ethereum-development-tutorial>

Dev Resources:

<https://ethereum.org/en/developers/docs/programming-languages/>

JAVA:

<https://ethereum.org/en/developers/docs/programming-languages/java/>

<https://github.com/web3j/web3j>

<https://kauri.io/#communities/Java%20Ethereum/connecting-to-an-ethereum-client-with-java-eclips/>

Node.js – JavaScript:

<https://codeforgeek.com/how-to-generate-ethereum-private-key-and-address/>

2. Ethereum Blockchain Block – ETH Transaction Schematic

From web3 console, execute the following (depending on the client, might need to use 'eth.getRawTransaction' instead)

```
eth.getTransaction('0x14a298c1eea89f42285948b7d51eeac2876ca7406c9784b9b90dd3591d156d64').raw
```

This should give

```
"0xf86b80850ba43b7400825208947917bc33eea648809c285607579c9919fb864f8f8703baf82d03a0008025a0067940651530790861714b2e8fd8b080361d1ada048189000c07a66848afde46a069b041db7c29dbcc6becf42017ca7ac086b12bd53ec8ee494596f790fb6a0a69"
```

which is 109 bytes. If we parse the data:

f86b length

80 nonce (0: this is the minimum an account can have)

85 0ba43b7400 gas price

82 5208 gas limit (this is fixed for simple payments)

94 7917bc33eea648809c285607579c9919fb864f8f (address, always 20 bytes)

87 03baf82d03a000 (value, in theory this can be shrunken to zero)

80 (data, already zero length)

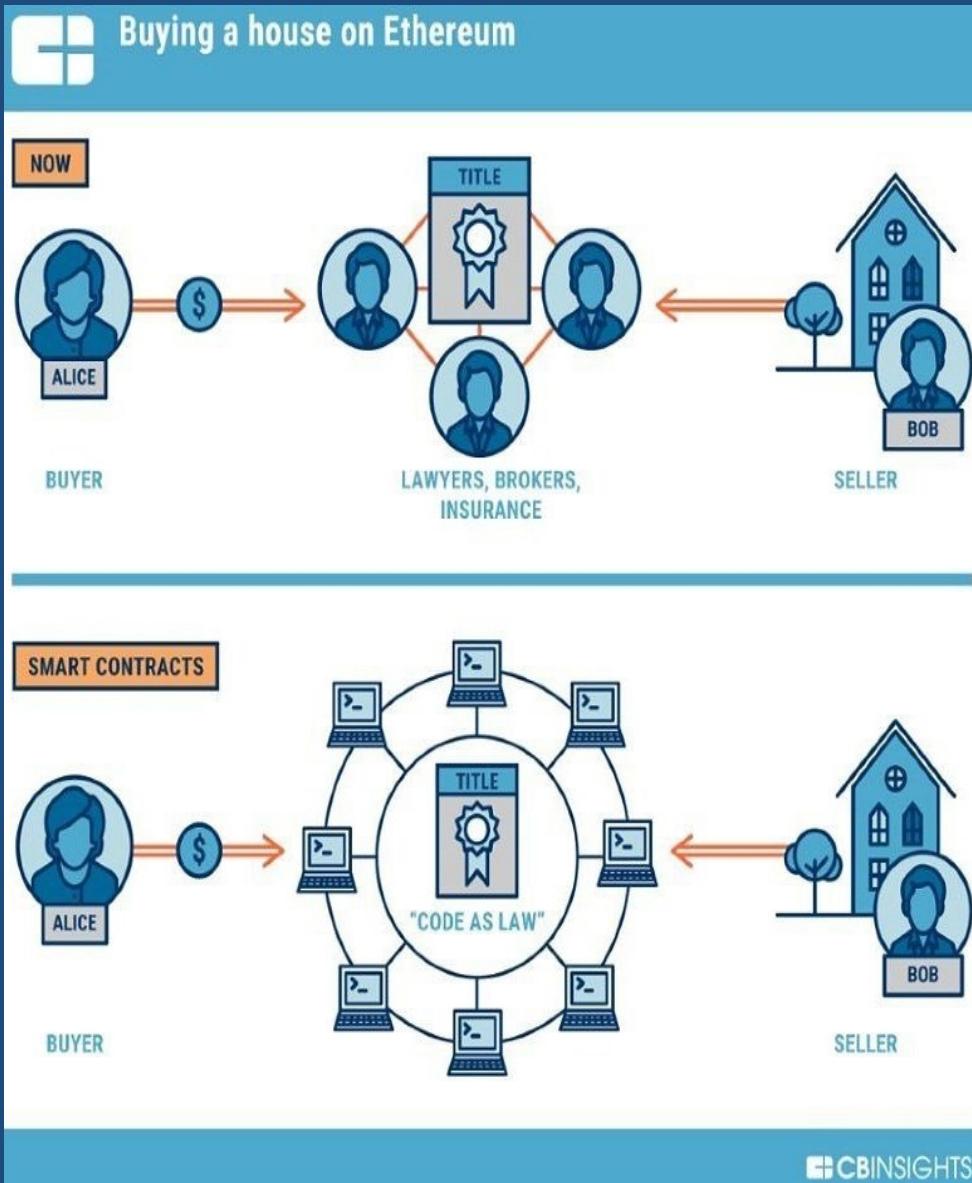
25 (V, one byte)

a0 067940651530790861714b2e8fd8b080361d1ada048189000c07a66848afde46 (R)

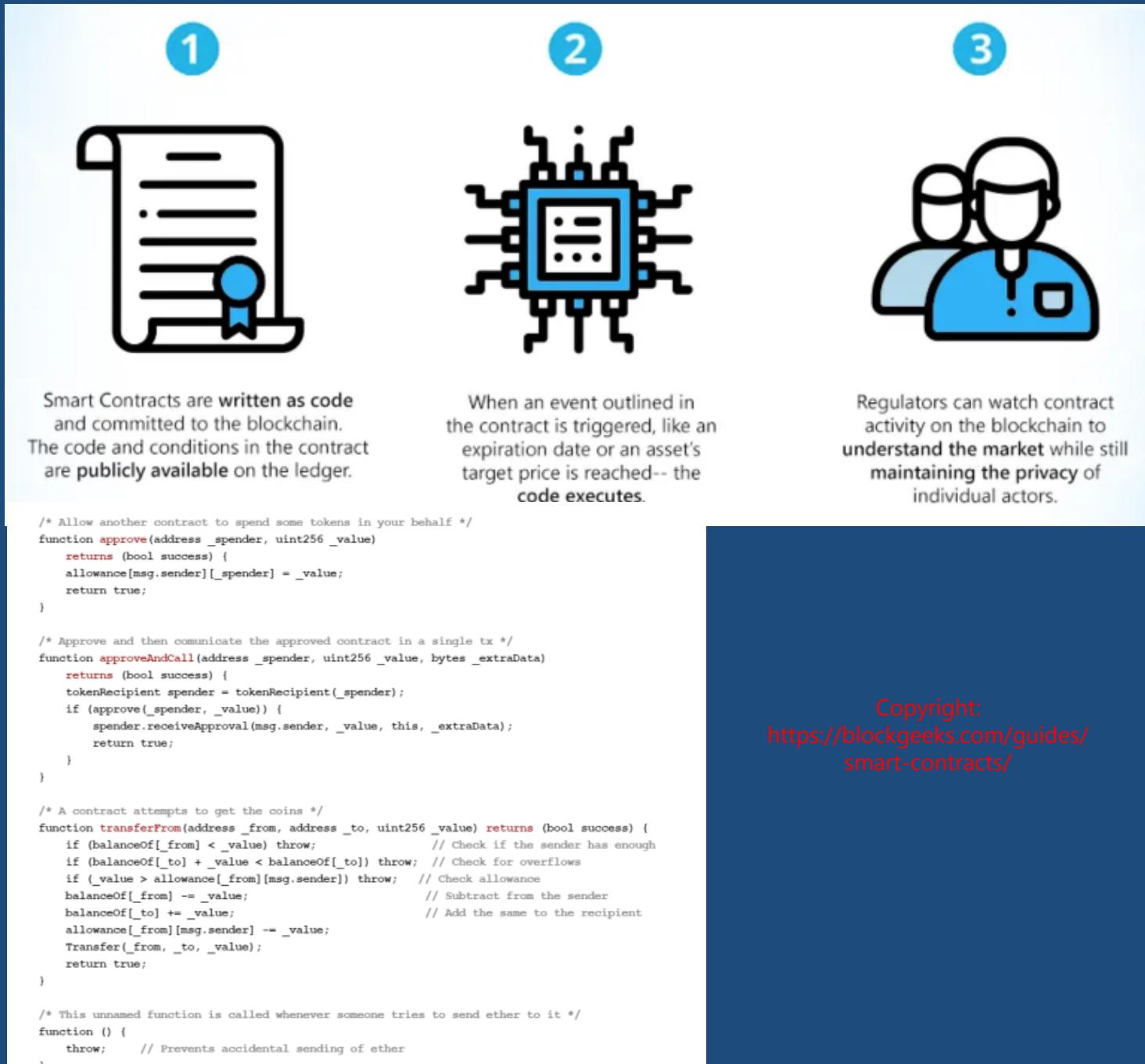
a0 69b041db7c29dbcc6becf42017ca7ac086b12bd53ec8ee494596f790fb6a0a69 (S)

From this sample, there isn't much more to be shortened. A smaller value can reduce the value field to one byte. Lower gas price may shrink one or two bytes further. Thus it seems a realistic "minimum" transaction size is should be greater than 100 bytes (109 - 7 -2).

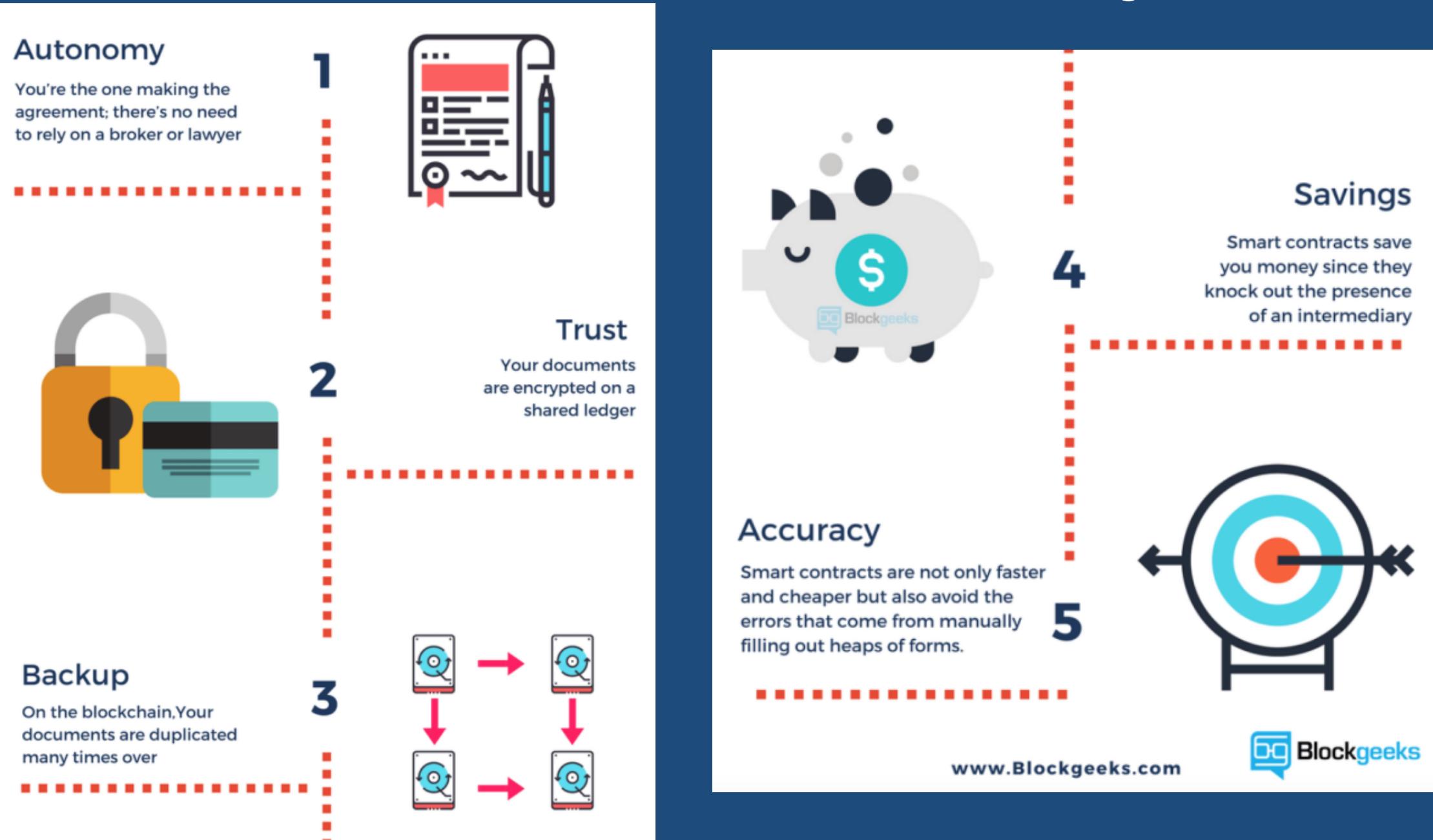
#Ethereum Blockchain Wiki Dev Tutorial – How Smart Contract is working?



Copyright: <https://www.investopedia.com/news/ethereum-smart-contracts-vulnerable-hacks-4-million-ether-risk/>



#Ethereum Blockchain Wiki Dev Tutorial – How Smart Contract is working?

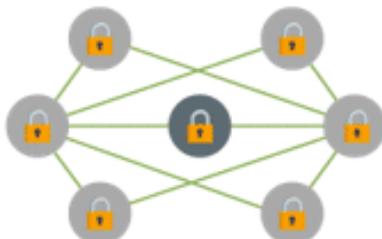


#Ethereum Blockchain Wiki Dev Tutorial – How Smart Contract is working?

Physical Contracts



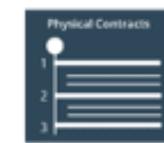
Alice +
Bob
Blockchain/permissioned ledger,
programming & encryption



Transacting parties
Individuals or Institutions

Smart Contracts

Lower operational
Overheads & costs leading
To economical financial
products

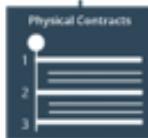


Smart Contracts
A Software program
on the distributed
Ledger, allowing an
immutable & Verifiable
records of all Contracts &
Transactions



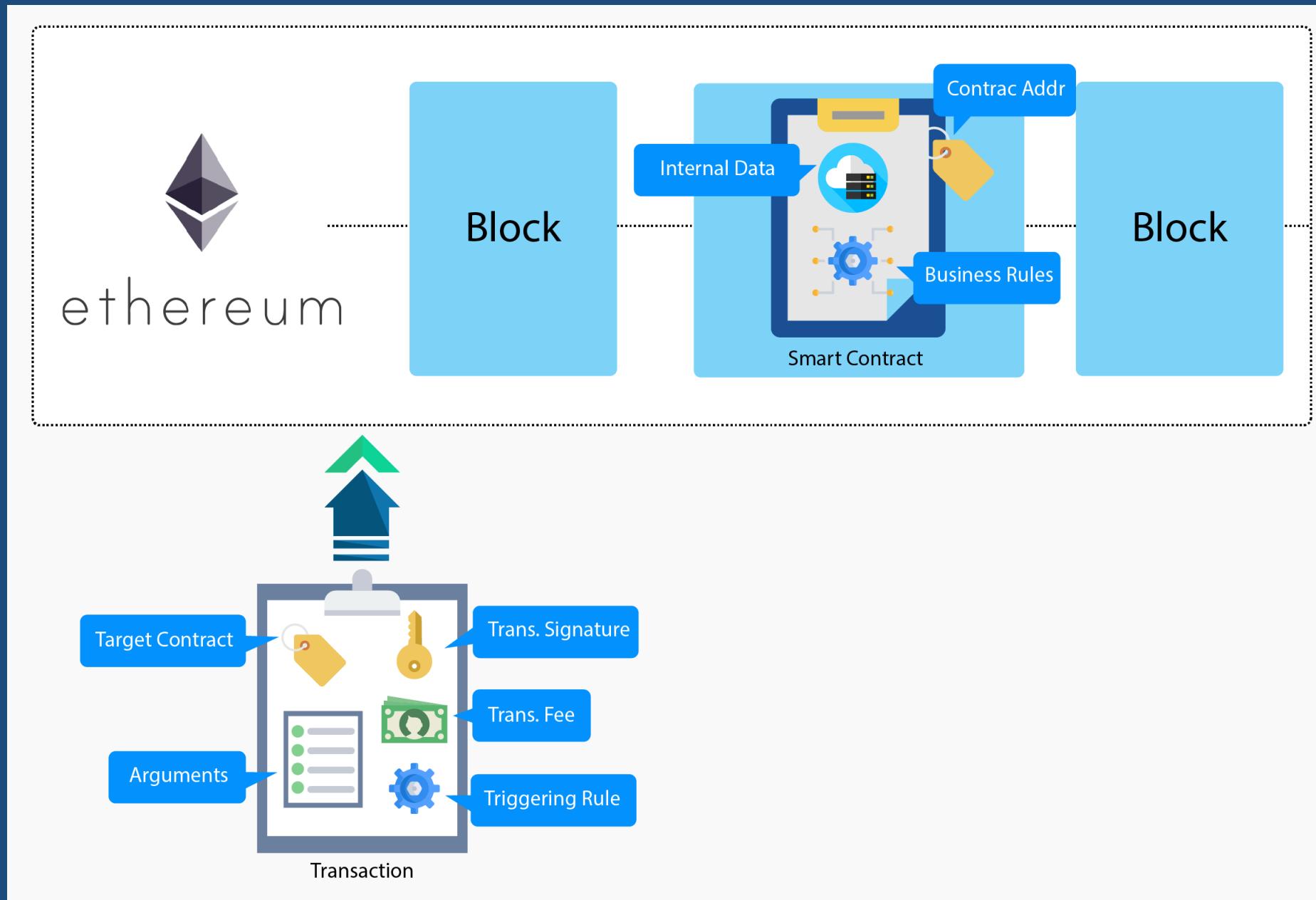
Banks, Insurers,
Capital Markets
Act as custodians of assets,
validators & authorities of all
contracts & transactions

Reduced administration
& service costs Owing
to automation & ease
of compliance & reporting



Central authorities that keep a tab on the system with a
wide ranging read-access to blockchain

#Ethereum Blockchain Wiki Dev Tutorial – How Smart Contract is working?



#Ethereum – DEMO with MetaMask and Remix IDE in Mozilla Firefox

D4. Develop, Deploy and Run Smart Contract in Remix IDE @ same address for all MetaMask Wallets

The screenshot shows the Mozilla Firefox browser window with the Remix IDE and the MetaMask extension loaded.

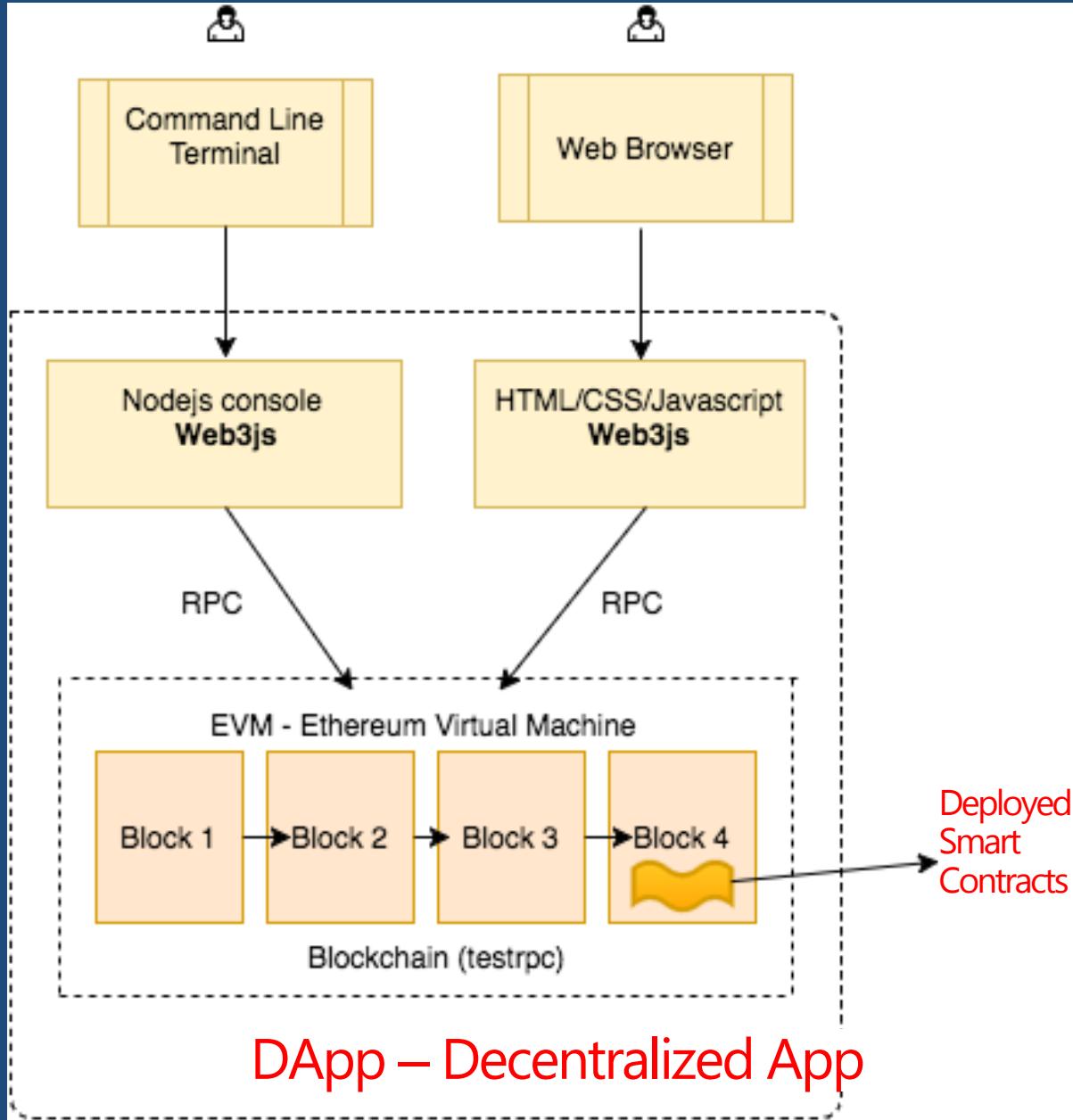
Remix IDE Area:

- File Explorers:** Shows a tree view of project files including browser, contracts (with artifacts, 1_Storage.sol, 2_Owner.sol, 3_Ballot.sol, Faucet.sol), scripts (deploy_web3.js, deploy_ether.js, withdraw_faucet.js), tests, and README.txt. **Faucet.sol** is selected.
- Code Editor:** Displays the Solidity code for the `Faucet` contract. The code handles ether withdrawal requests.
- Assembly View:** Shows the low-level assembly code generated from the Solidity code.

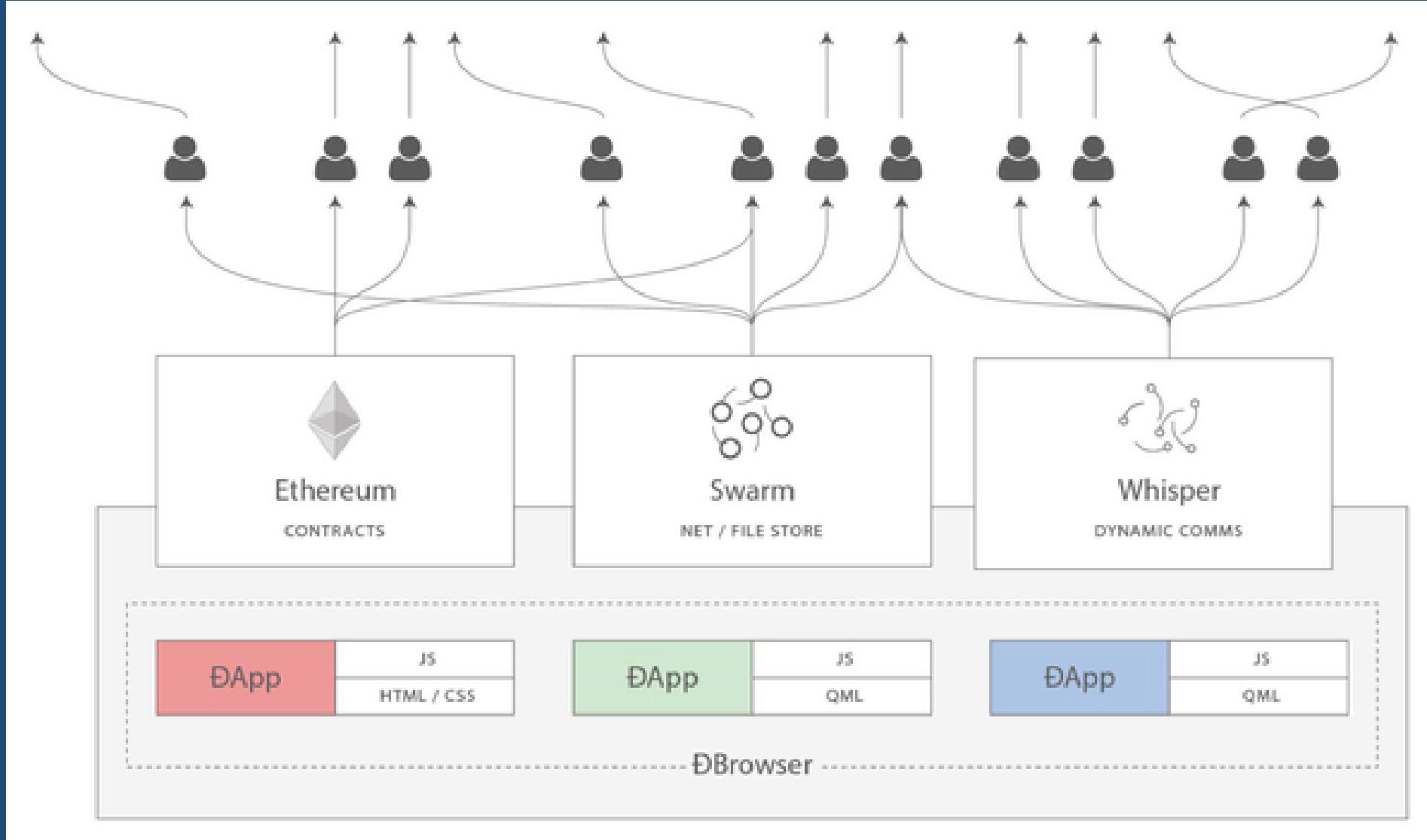
MetaMask Extension Area:

- Account:** Shows a fox icon and "Connected" status. Network is set to "Ropsten Test Network".
- Actions:** Includes "Expand view", "Account details", "View on Etherscan", and "Connected sites". Buttons for "Buy", "Send", and "Swap" are also present.
- Assets:** Shows a balance of 2.5 ETH.
- Activity:** Lists recent transactions:
 - Give Right To Vote:** -0 ETH (Feb 19)
 - Receive:** 1 ETH (Feb 19, From: 0x81b7...7647)

2. Ethereum Blockchain Peer2Peer Network Architecture & DApp



2. Ethereum Blockchain Peer2Peer Network Architecture & DApp



Web3: A suite of Dapp - Decentralized APP-lication components for the next evolution of the web

#Ethereum



	Bitcoin (BTC)	Ether (ETH)
What is it?	A currency	A token
Inventor	Satoshi Nakamoto	Vitalik Buterin, Joseph Lubin, Gavin Wood, etc.
Went alive	January 2009	July 2015
Supply Style	Deflationary (a finite # of bitcoin will be made)	Inflationary (much like fiat currency, where more tokens can be made over time)
Supply Cap	21 million in total	18 million every year
Smallest Unit	1 Satoshi = 0.00000001 BTC	1 Wei = 0.0000000000000001 ETH
New token issuance time	Every 10 minutes approximately	Every 10 to 20 seconds
Amount of new token at issuance	12.5 at the moment. Half at every 210,000 blocks	5 per every new block
Utility	Used for purchasing goods and services, as well as storing value (much like how we currently use gold).	Used for making dApps (decentralized apps) on the Ethereum blockchain.
Price	Around \$18,000 - \$55,000 (in 2020 - 2021)	Around \$610 - \$2,000 (in 2020 - 2021)
Purpose	Bitcoin is a new currency created to compete against the gold standard and fiat currencies	Ethereum is a token capable of facilitating Smart Contracts (For example a lawyer's contract, an exchange of ownership of property, and voting)

03. Blockchain Use-cases in Fin-Tech

141

- ▶ Introduction
- ▶ **Use case 1** – Reductions of Fraud
- ▶ **Use case 2** – KYC – Know Your Customer
- ▶ **Use case 3** – Trading platforms
- ▶ **Use case 4** – Payments
- ▶ **Use case 5** – Simplifying Cross border payments
- ▶ **Use case 6** – The rise of smart assets
- ▶ **Use case 7** – Loyalty & Rewards Program
- ▶ XRP Ripple as SWIFT replacement

Introduction



- ▶ A blockchain is a data structure that makes it possible to create a digital ledger of transactions and share it amongst a distributed network of computers.
- ▶ It uses cryptography to allow each participant on the network to manipulate the ledger in a secure way without the need for a central authority.

Introduction

- ▶ Blockchain technology is emerging as the way to let companies make and verify financial transactions on a network instantaneously without a central authority
- ▶ In banking there are many existing and developing use cases to implement blockchain.



Use case 1 – Reductions of Fraud



- ▶ Traditionally, bank ledgers have been created within a centralized database. Hackers and cyber-criminals are well aware of evolving digital technology and have been able to bypass these security systems to commit data breaches and fraud.
- ▶ In contrast, as the blockchain is decentralized it is less prone to this type of fraud. By using blockchain there would not only be real-time execution of payments but also complete transparency which would enable real-time fraud analysis and prevention.

Use case 1 – Reductions of Fraud

- ▶ Blockchain helps to eliminate the possible fraud on all transactions .

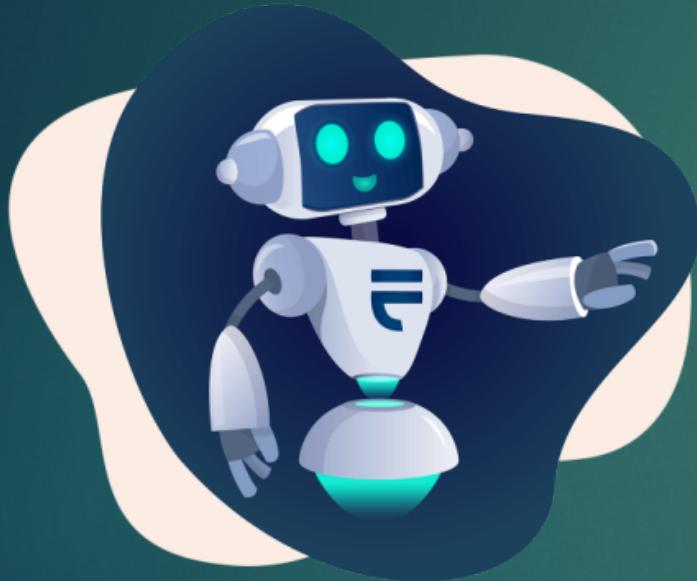


Use case 1 – Reductions of Fraud

On 30 December 2015 Nasdaq announced that it had made its first ever share trade using blockchain technology. Nasdaq used its proprietary Linq platform (developed in collaboration with Chain.com and global design firm IDEO) to sell shares.

In August 2016, nearly 120,000 units of digital currency Bitcoin worth about US \$72 million was stolen from the exchange platform Bitfinex in Hong Kong. The Bitcoin was stolen from users' segregated wallets and amounted to about 0.75% of all Bitcoin in circulation at that time. Since the hack, Bitfinex has taken steps to reimburse accountholders with "BFX tokens" which are cryptographic tokens on the Omni blockchain that can be exchanged for \$1 beneficial interests in iFinex (Bitfinex's parent company).

Use case 1 – Reductions of Fraud



Everledger



Use case 2 – KYC

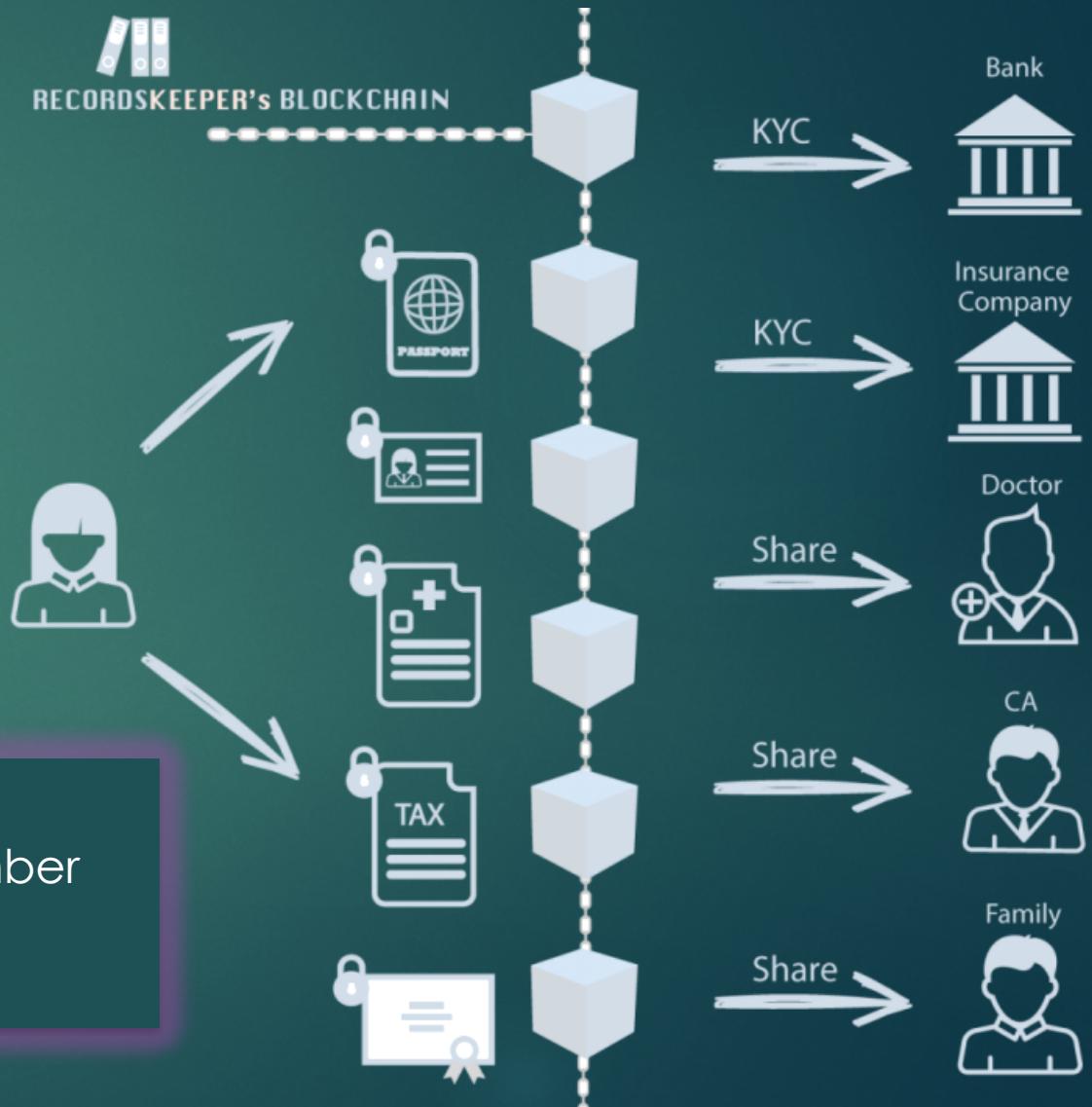
- ▶ Know Your Customer (“KYC”) requests currently can cause delay to banking transactions, typically taking 30 to 50 days to complete to a satisfactory level.



Use case 2 – KYC

- ▶ On a blockchain system the information about a customer can be used by other banks and other accredited organizations without the need to ask the customer to start the KYC process all over again.

SWIFT has established a KYC Registry with 1,125 member banks sharing KYC documentation



Use case 2 – KYC



Bank Chain



SELFKEY



SECURE KEY

Use case 3 – Trading platforms

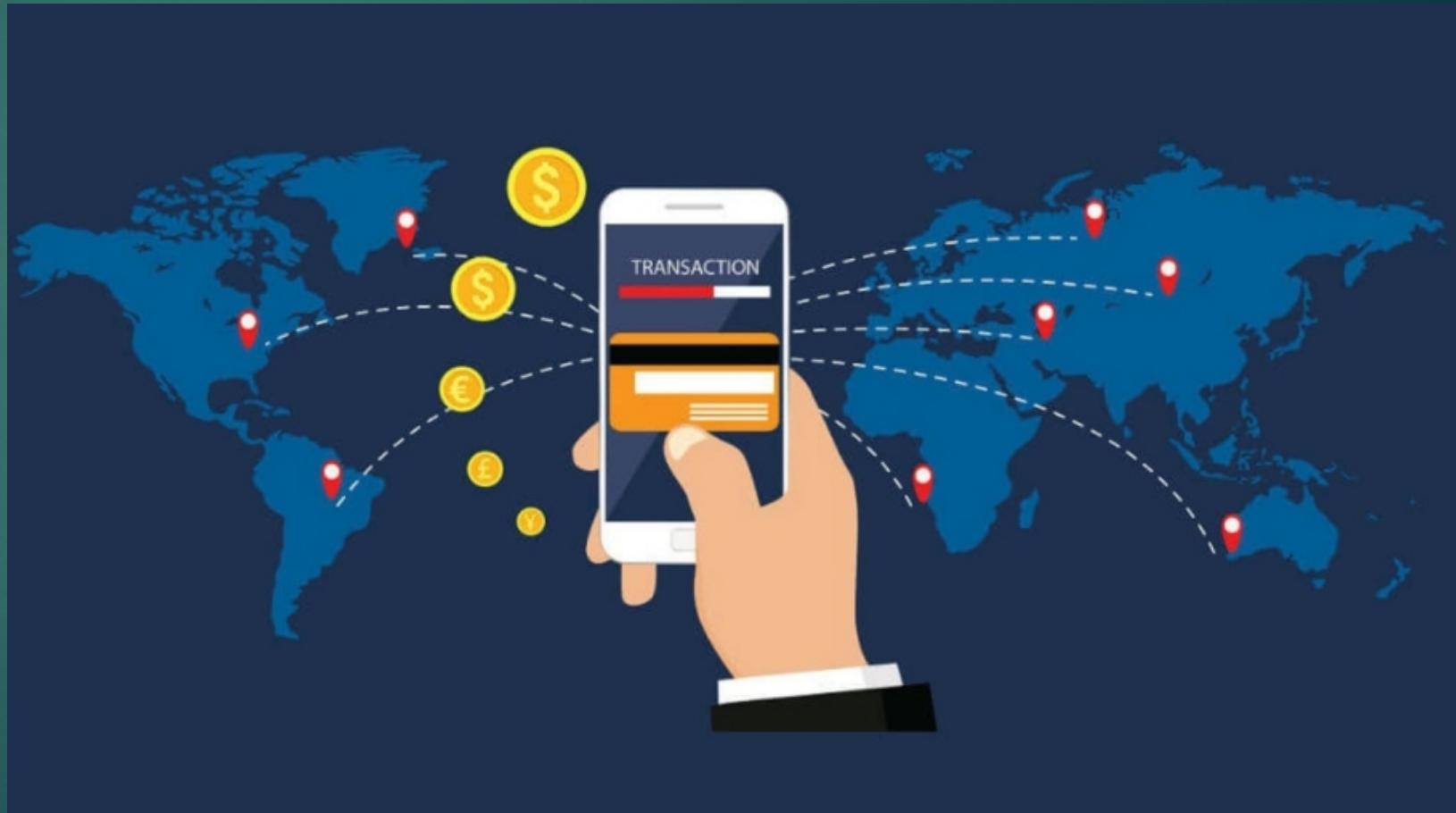
- ▶ The blockchain technology offers a potential new medium to exchange assets without centralized trusts or intermediaries – and without the risk of double spending.
- ▶ Blockchain can eliminate the threat or the risk of fraud in all areas of banking, and this could equally apply to a trading platform.
- ▶ The blockchain gives the benefit of distributed and verifiable trust that was not present before.

Use case 3 – Trading platforms



Use case 4 – Payments

- ▶ The main use case of blockchain for banking is that of payments.



Use case 4 – Payments



Dash
Digital Cash

VVERGE

Use case 5 – Cross border payments

- ▶ Mastercard and R3 announced in 2019 a strategic partnership to develop and pilot a blockchain-enabled cross-border payments solution that will initially focus on connecting global faster payments infrastructures.



Use case 5 – Cross border payments



stellar

Open Source, Distributed Payments Infrastructure

Use case 6 – The rise of smart assets



HYPERLEDGER



DIGIX



BANKEX

Use case 7 – Loyalty & Rewards Program

loyyal
& blockpoint



XRP Ripple as SWIFT replacement

- ▶ Ripple is a “real-time gross settlement system” (RTGS), currency exchange and remittance network.



XRP Ripple as SWIFT replacement

- ▶ Official Monetary and Financial Institutions Forum (OMFIF) presents different reports on blockchain technology and the advantages of Ripple products as an alternative to SWIFT.
- ▶ These reports present blockchain technology as a solution for multiple sectors because can bring improvements in six key points: security, speed, transparency, traceability, cost, and risk management.



<https://www.omfif.org/wp-content/uploads/2020/05/The-role-of-blockchain-in-banking.pdf>

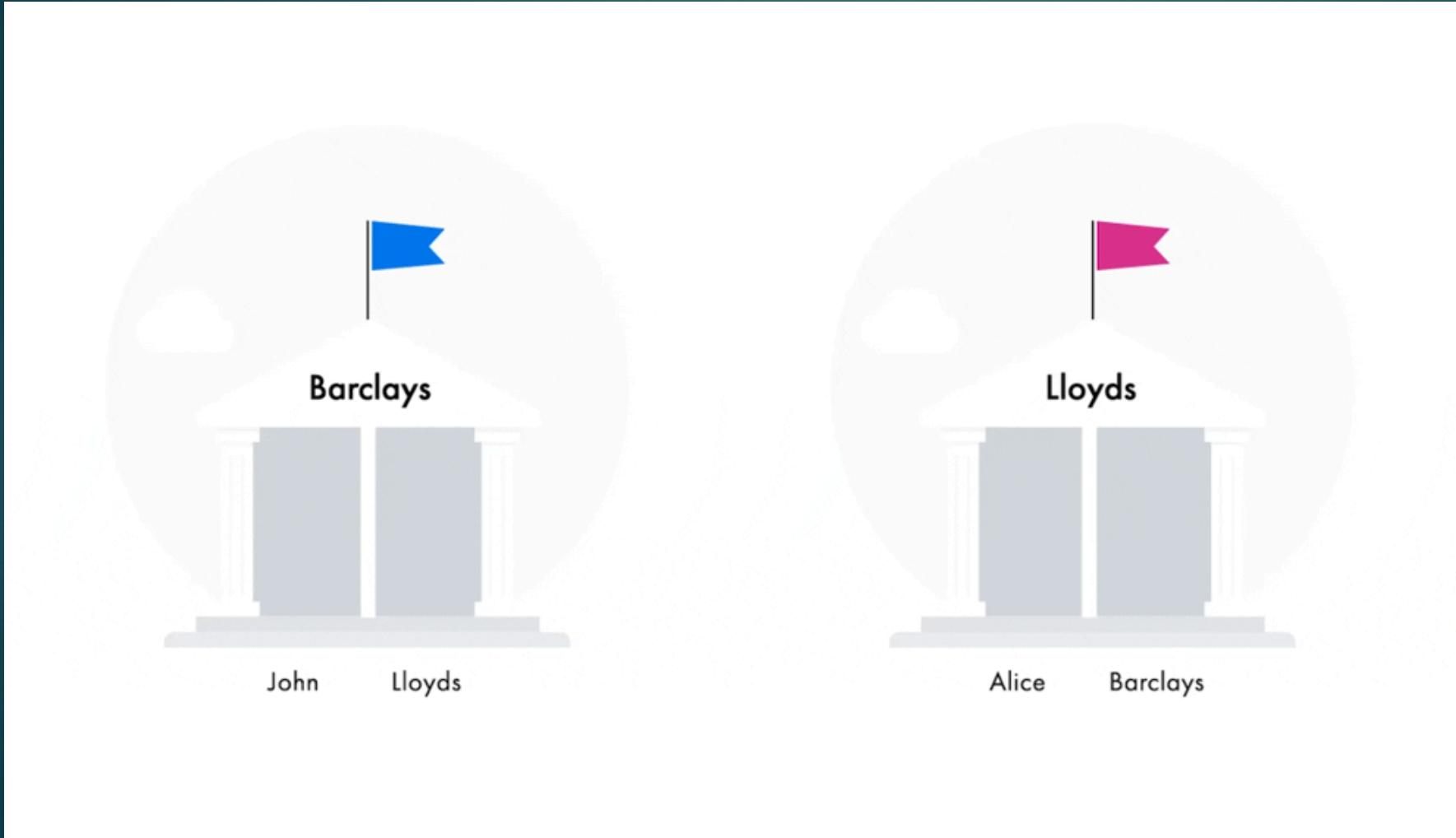
XRP Ripple as SWIFT replacement

- ▶ Ripple comes up in 2012 with the name OpenCoin. Their objective has been to provide secure payments via a global network.
- ▶ In 2013 their system started to be used in banks for payment systems.
- ▶ In 2014 Ripple helped to create a new cryptocurrency: XRP. This cryptocurrency helps to make transfer money more easier.
- ▶ In February 2020, Ripple made a partnership with AZIMO to make transfer in European money.

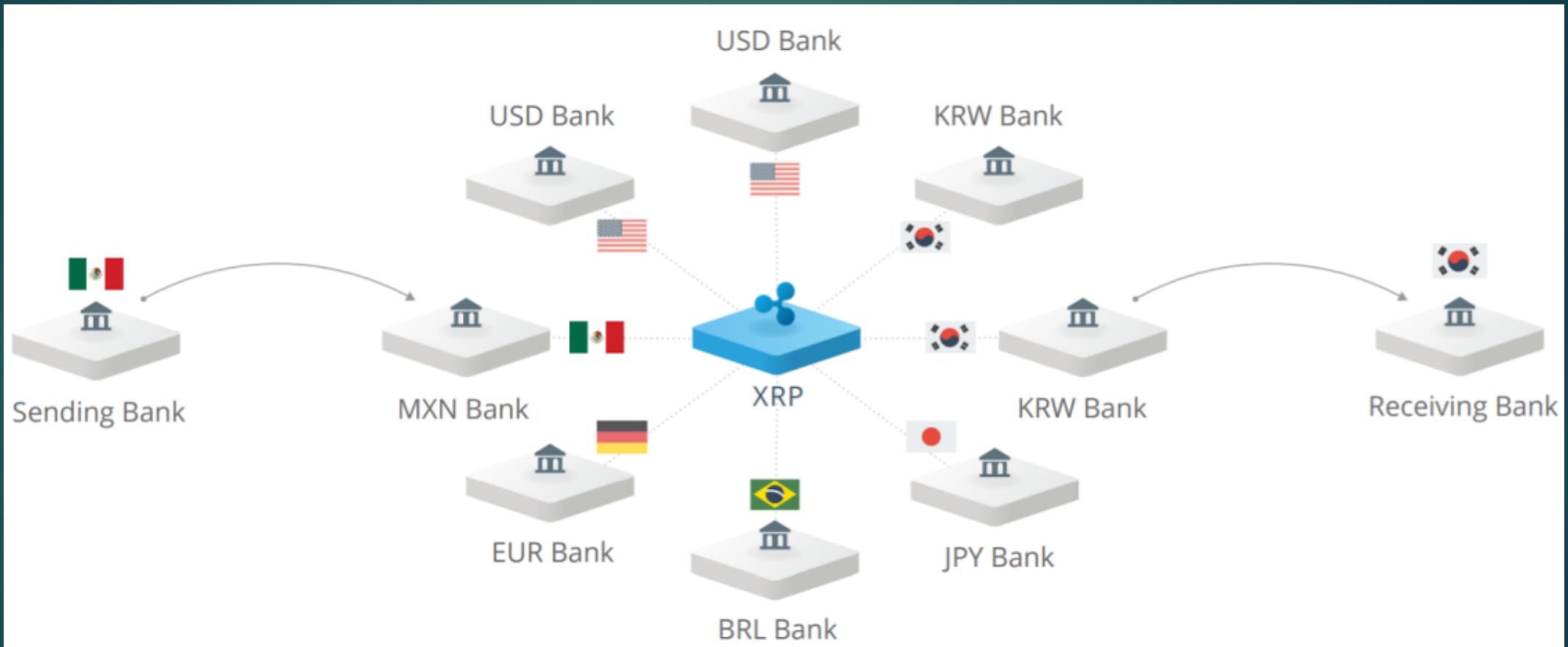
XRP Ripple as SWIFT replacement

- ▶ Swift was founded in 1973 and has more than 10,000 bank members.
- ▶ Swift is used to send payment orders. The actual transfer of funds is typically made from a pre-funded nostro account that a bank has with a local correspondent bank.
- ▶ Ripple is attempting to change this way of conducting cross-border payments, by using XRP as a ‘bridge’ between fiat currencies.

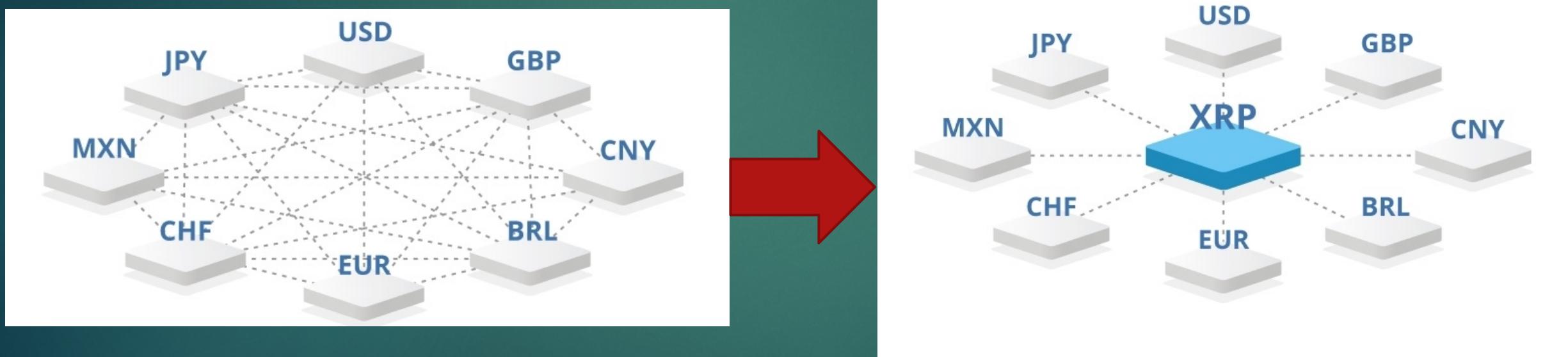
XRP Ripple as SWIFT replacement



XRP Ripple as SWIFT replacement



XRP Ripple as SWIFT replacement



XRP Ripple as SWIFT replacement

- ▶ The main battleground between Ripple and SWIFT is the competition for banks and institutions to use their services to transfer money. This competition has been ongoing for years now. Ripple only became a real competitor after the development of XRP.
- ▶ SWIFT has made improvements to their messaging system that allows tracking of payments, putting greater emphasis and responsibility on the banks to create a faster payment network.



The screenshot shows a web browser displaying a press release from the U.S. Securities and Exchange Commission (SEC). The URL in the address bar is <https://www.sec.gov/news/press-release/2020-338>. The page features the SEC logo and the text "U.S. SECURITIES AND EXCHANGE COMMISSION". A sidebar on the left lists "Newsroom" categories: Press Releases (selected), Public Statements, Speeches, Testimony, Spotlight Topics, Media Kit, Press Contacts, Events, Webcasts, What's New, and Media Gallery. The main content area is titled "Press Release" and discusses the SEC charging Ripple and two executives with conducting a \$1.3 billion unregistered securities offering. It includes a "FOR IMMEDIATE RELEASE" section and the date "2020-338". The text of the press release states that the SEC has filed an action against Ripple Labs Inc. and two of its executives, alleging they raised over \$1.3 billion through an unregistered, ongoing digital asset securities offering. The complaint alleges that Ripple raised funds beginning in 2013 by selling XRP to investors in the U.S. and worldwide. Ripple allegedly distributed billions of XRP in exchange for non-cash consideration like labor and market-making services. The complaint also states that Larsen and Garlinghouse effected personal unregistered sales of XRP totaling approximately \$600 million without registering them.

SEC Charges Ripple and Two Executives with Conducting \$1.3 Billion Unregistered Securities Offering

FOR IMMEDIATE RELEASE
2020-338

Washington D.C., Dec. 22, 2020 — The Securities and Exchange Commission announced today that it has filed an action against Ripple Labs Inc. and two of its executives, who are also significant security holders, alleging that they raised over \$1.3 billion through an unregistered, ongoing digital asset securities offering.

According to the SEC's complaint, Ripple; Christian Larsen, the company's co-founder, executive chairman of its board, and former CEO; and Bradley Garlinghouse, the company's current CEO, raised capital to finance the company's business. The complaint alleges that Ripple raised funds, beginning in 2013, through the sale of digital assets known as XRP in an unregistered securities offering to investors in the U.S. and worldwide. Ripple also allegedly distributed billions of XRP in exchange for non-cash consideration, such as labor and market-making services. According to the complaint, in addition to structuring and promoting the XRP sales used to finance the company's business, Larsen and Garlinghouse also effected personal unregistered sales of XRP totaling approximately \$600 million. The complaint alleges that the defendants failed to register their offers and sales of XRP or satisfy any exemption from registration, in violation of the registration provisions of the federal securities laws.

XRP Ripple as SWIFT replacement

LET'S SEND THE MONEY

moderated by **Elizabeth Schulze**, *CNBC International*

Brad **Garlinghouse**
Gottfried **Leibbrandt**

#PFF19

www.parisfintechforum.com

An event produced and organized by **Altéir**

<https://www.youtube.com/watch?v=SRoCGr4cess>

Make Blockchain Technology working for Fin-Techs



Visit us at www.fintech-ho2020.eu

or contact us at info@fintech-ho2020.eu

or follow us on [LinkedIn](#)

The development of the materials has also been supported by the project **A FINancial supervision and TECHnology compliance training programme**.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825215 (Topic: ICT-35-2018 Type of action: CSA)

All material presented here reflects only the authors' view. The European Commission is not responsible for any use that may be made of the information it contains.

Q & A



www: ism.ase.ro | acs.ase.ro | dice.ase.ro

**Scan the Tag
to get the web
Mobile Address**

