

Cryptography in Blockchain

Nikita Storublevtcev(✉)

St. Petersburg State University, Saint Petersburg, Russia
100.rub@mail.ru

Abstract. There is a vast multitude of different hashing and cryptography algorithms out there, and more are developed every year. Blockchain technology, and crypto-currencies specifically, raised many concerns in different institutions. Users want to be sure that their assets are adequately protected by the best available algorithms, but some institutions around the world are opposed to that, or enforce their own standards. Ideally, every platform would allow its users to choose any algorithm they want, but in most situations this is, unfortunately, not the case. In this paper we present a review of the most popular blockchain platforms and the options they provide, and compare their cryptographic strength.

Keywords: Cryptography · Algorithms · Hashing · Blockchain · Digital signature

1 Introduction

1.1 Importance of Cryptography

Every year, cryptography becomes more and more a part of the daily life for many people around the world. The rise of crypto-currencies brought wider general awareness of cryptography to the masses, and showcased its practical usefulness outside military applications. Increased interest brought more researchers and investors to the field, and previously niche problems and concerns became more widely-discussed. Today most governments understand that crypto-platforms can change many aspects of our daily lives and are forced to take a stance, be it for or against them.

Nowadays many Internet connections are encrypted in some capacity. Be it web-sites using HTTPS, SSH connections, or VPN services, most users make use of some sort of cryptographic protection every day, and blockchain platforms use the same tools and algorithms. While privacy and data security is important for individual user, it is much more so for corporate entities with much more to lose in case of a security breach.

But crypto-currencies, while being the most widely known, are far from the only use-case for blockchain technology and cryptography in general. Blockchain is a relatively new technology, but it already has found many uses in various fields. Since the rise of Bitcoin, which showed the potential of blockchain,

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11620, pp. 495–508, 2019.
https://doi.org/10.1007/978-3-030-24296-1_39

several platforms emerged, offering to provide easy-to-use and well-developed frameworks for creating and using blockchain-based enterprise-level services. The most popular and successful among them are Ethereum, Hyperledger Fabric, R3 Corda, Ripple and Quorum [1]. Among them, Ethereum and Hyperledger Fabric are the biggest general-purpose platforms, have the widest cross-industry reach and are designed to support almost any use-case imaginable. Ethereum is known for its smart-contract functionality, but it is primarily a crypto-currency platform, while Hyperledger Fabric does not have an in-built currency, and is designed for creation of any number of private networks. R3 Corda and Ripple are designed and used for financial sector operations, which require faster transaction processing, and Quorum is a modification of Ethereum, also aimed at financial services. In this paper we will review cryptographic tools used by each platform and compare them in terms of reliability and security.

1.2 Importance of Choice

A user of a blockchain platform may wonder - why do I need to bother with cryptographic algorithms other than provided by the developers? This is, unfortunately often, a question born of necessity. It is no secret that different algorithms have different cryptographic properties, strength and weaknesses, but they also differ greatly in speed and resources required to execute them. This, coupled with other aspects of the blockchain platform, can make it unsuitable for some types of services. For example, one of the biggest complains about Bitcoin platform is the long transaction confirmation time (an average of 10–20 min, on some occasions rising all the way to 2 h [2]), making it unfit for quick everyday purchases. Many solutions were proposed and created to combat this issue, one being the Ripple platform, which only takes a couple seconds per transaction. All prominent blockchain platforms are general-purpose services, to greater or lesser extent, offering application developers their infrastructure and technology. But each type of service demands different things from the platform, so to accommodate all of them it has to offer deep customization of its infrastructure.

There are also unfortunate cases when the choice of any particular algorithm is forced upon develops by third parties. This is usually the case in projects developed for or with government departments. For example, Russian Government requires some personal information to be encrypted exclusively by approved cipher algorithms, all of which were developed in-house relatively recently and are not supported out-of-the-box by any blockchain systems. By providing the ability to customize encryption algorithms, blockchain platforms can ease the pain of developers in similar situations.

1.3 Basics of Blockchain

Before we begin comparing algorithms, we need to understand where they are used and why. In the core of all blockchain technologies lies the eponymous Blockchain, which is a set of ordered blocks of data. Each block contains a number of verified transactions and usually contains a pointer to the previous

block in the chain. This, together with the fact that copies of each block are stored on multiple nodes in the network, ensures that no one user can modify the contents of any block without other users detecting it. In most cases, to have any chance to successfully make a malicious transaction, one user has to control more than 50% of the network. This creates a trusted environment where the architecture of the system itself plays the role of a trusted medium.

When a transaction is performed, before it is written into the block, it is validated by other nodes and users of the platform. The exact validation mechanism differs for each platform, and they all require different amount of time and computational resources. If and when a transaction is considered valid, it is written to the current block. A block has a set size, and when it is filled with transactions, it is also verified and is considered written into the blockchain.

The second important part for the reliable system are so-called Smart Contracts, which are essentially small programs that can interface with the blockchain platform, detect and perform certain actions, like transfer of funds, shares and other assets. This allows users to avoid middlemen and costs/risks associated with them.

1.4 Hashes and Keys

There are many different cryptographic algorithms used in different blockchain platforms, but most of them fall into one of two categories - hashing algorithms and asymmetric algorithms. Before we begin comparing them, we need to define what they are, when they are used, and what issues they have.

Hashing Algorithms (or Hash Functions) are designed to generate a "hash value" (or just "hash") from a piece of data. Hash is a bit sequence, frequently presented in character string form. Its length is usually fixed for any given function. The original data can usually be of any size, it will be mapped to a hash of specific length. Changing even one bit in the original data will result in an entirely different hash value for it. This property is used, for example, to verify the integrity of any piece of transferred data. The sender sends the hash value of the message together with it, and receiver can calculate the hash value of the received message and compare it to the one provided by the sender. Any inconsistencies will mean corruption of the message during transfer.

Unfortunately, it is impossible to ideally map a set of values of variable length to a set of values of fixed length, as the first set has higher magnitude. This means that there inevitably will occur situations where multiple different input data packages are mapped to the same hash value. This is called a "collision". Developers do everything they can to make their algorithms stronger, but there is no way to create an ideal function, which would have to fully satisfy the following criteria:

– *Pre-image Resistance.* Knowing a hash value, it should be impossible (or too difficult to be worth) to find a message that produces the same hash value [3].

- *Second Pre-image Resistance.* Knowing a message, it should be impossible (or too difficult to be worth) to find a different message that produces the same hash value [3].
- *Collision Resistance.* It should be impossible (or too difficult to be worth) to find two different messages that produce the same hash value [3].

Collision Resistance implies Second Pre-image Resistance, but does not necessary mean Pre-image Resistance. A hash function which is only Second Pre-image Resistant is considered too weak for practical applications. Together, these criteria mean that an attacker cannot modify the message, without also changing the hash. Thus, if two strings have the same hash, one can be very confident that they are identical. Second Pre-image Resistance prevents an attacker from creating a document with the same hash as a document the attacker cannot control. Collision resistance prevents an attacker from creating two distinct documents with the same hash [3].

In blockchains hashing is used extensively and can be considered essential for their operation. For example, the strength of links between blocks is usually ensured by writing the hash of the previous block into the new one. This way if anyone attempts to change anything in a block, they would also need to change all blocks after it, and do this on a large number of nodes at the same time.

Asymmetric Algorithms (or Public-Key Cryptography) use a pair of keys to create a secure communication channel between two peers. Each peer has its own set of keys. One key is chosen to be a Public Key, and is sent openly to the other peer, another is a Private Key and is kept secret. There are two common uses for Public-Key Cryptography are:

- *Public Key Encryption*, is used when a peer wants to send a secret message to another peer. They use target peer Public Key to encrypt the message and send it through the (potentially compromised) channel. Upon receiving the message, target peer can use their own Private Key to decrypt the message. The message is impossible (or too difficult to be worth) to decrypt without possessing the recipient Private Key.
- *Digital Signatures* are very similar to Public Key Encryption, but the message is also signed with the senders Private Key, and can be verified by anyone who possesses sender's Public Key. This helps ensure that the source of the message is likely to the sender, and that the message was not tampered with.

It is also important to enforce the authenticity of any particular key and its link to the user. There are several ways to do it:

- *Public Key Infrastructure (PKI)*, where one or more parties are designated as *Certificate Authorities* and are responsible for key ownership certification.
- *Web of Trust*, where individual users endorse the ownership in a decentralised system.
- *Domain Name System (DNS)* can be used to look up and verify the signed message.

Public-Key Cryptography is extensively used in blockchain platforms, especially in crypto-currency ones, like Bitcoin. It also underpins such widely-used technologies as Secure Shell (SSH) and Transport Layer Security (TLS).

2 Usage in Blockchain Platforms

That that we know what to look for, we can look at each blockchain platform in-depth and determine which algorithms are in use in each part of the infrastructure. This information is surprisingly hard to find in open sources, and some of it may be outdated.

2.1 Ethereum

Ethereum, arguably the most mature platform currently available, launched in 2015 with intent to provide a platform with robust decentralised application scripting tools. It has a built-in crypto-currency called Ether (ETH) which can be used for both direct transactions and as "fuel" for applications, to compensate mining nodes for performed computations. Like Bitcoin it has "wallets" (or accounts) which consist of a public-private key pair, but Ethereum generates them using Elliptic Curve Digital Signature Algorithm (ECDSA) and secp256k1 [4] constants to define the curve. Message signing process requires both public and private keys and is analogous to DSA [5], but uses KECCAK-256 as a hash function. It is note-worthy that it is referred to as SHA-3 on multiple occasions in code and documentation, but does not follow the FIPS-202 based standard, and generates different hash values. The same hash function is used throughout the platform, and developers are aware of the confusion it causes are are working on fixing it [6].

For inter-node communications Ethereum uses RLPx (not an acronym) transport protocol [7]. It is a TCP-based protocol using Elliptic Curve Integrated Encryption Scheme (ECIES) in its handshake. It also includes the following in its cryptosystem:

- The elliptic curve secp256k1 with generator G.
- NIST SP 800-56 Concatenation Key Derivation Function.
- Hash-based Message Authentication Code (HMAC) using the SHA-256 hash function.
- AES-128 encryption function in CTR mode.

According to publicly available information, Ethereum does not currently support any customization of its cryptosystem, and any tweaking would require changes to the codebase. There are projects looking to find a work-around or fix to this problem [8], but it will not be an easy task.

2.2 Hyperledger Fabric

In contrast to Ethereum, Hyperledger Fabric doesn't have the main blockchain, but instead focuses on providing developers with a framework to create their own blockchains. It does not have an in-built crypto-currency, instead focusing more on the Distributed Ledger side of the blockchain technology, and sports fee-free transactions. Fabric supports both public and private blockchains and has extensive tools for identity and permission management.

Hyperledger is designed to be highly modular and supports BlockChain Cryptographic Service Provider (BCCSP) modules, which can theoretically implement any algorithm. Unfortunately, at this time this feature is not fully-implemented yet as the choice of SHA2-256 hash function is hard-coded in several places [9] and is the only working option in the configuration file. Some evidence also suggests use of SHA3 family of functions as default in early development [10].

User accounts are handled by Certificate Authority (CA) servers, which can be configured to generate X.509 certificates and ECDSA keys [11]. ECDSA has the following options:

- prime256v1 constants with SHA256 hash.
- secp384r1 constants with SHA384 hash.
- secp521r1 constants with SHA512 hash.

We were unable to find information on what exact hash function is used in X.509 certificates created by Fabric, but current standard requirements forbid issuance of certificates using MD2, MD5 and SHA-1 functions.

The platform also allows for encryption (full or partial) of the smart-contract (locally referred to as Chaincode) data. It is handled by the BCCSP wrapper, which allows to perform the same cryptographic operations [12].

Communication between nodes is secured using Transport Layer Security (TLS) protocol. TLS supports a wide range of algorithms, but only some variations of Advanced Encryption Standard (AES), ChaCha20, Rivest–Shamir–Adleman (RSA) and ECDSA are currently considered to be secure.

2.3 R3 Corda

R3 Corda is an interesting platform, in many ways different to others. It was originally developed as closed-source and only went open source in November 2018. It does not actually use blockchain or distributed ledger technology, but rather what is being described as a "shared ledger", and it does not have an built-in currency. By default, information about transactions is only shared with those who are parties to that transaction. This provides increased privacy to users, but eliminates the concept of the block chain itself from the system. Instead Corda links "States" containing references to contract code written in computer code, a hash of a legal document backing the state written in legalese, and a snapshot of various data. User accounts are represented by Identities, which are attested to by X.509 certificate signed by a well known identity.

Corda has been designed to be cryptographically agile, but still restricts user's choice to a set of pre-selected signature schemes [13]:

- Pure EdDSA using the ed25519 curve and SHA-512.
- ECDSA using the NIST P-256 curve (secp256r1) and SHA-256.
- ECDSA using the Koblitz k1 curve (secp256k1) and SHA-256.
- RSA (3072bit) PKCS#1 and SHA-256.
- SPHINCS-256 and SHA-512 (experimental).

There also exists a proposed Blockchained Post-Quantum Signatures (BPQS) algorithm, which has not yet been implemented [14].

Corda uses Advanced Message Queuing Protocol (AMQP) 1.0 over TLS between nodes which is currently implemented using Apache Artemis [15].

2.4 Ripple

Ripple was initially founded in 2012 under a name OpenCoin, and it aims to connect different banks and payment systems through it's crypto-currency called Ripples (or XRP). User account is represented by a Master key pair, and supports the ability to authorize a secondary, Regular key pair, to sign future transactions, while keeping the Master key pair offline. If the regular keys are compromised, they can be replaced by new ones. Master keys cannot be replaced, only disabled.

Instead of one singular ledger, Ripple has a series of ledger versions. At any given time, a node instance has a "current" open ledger, some number of closed ledgers that have not yet been approved by consensus, and any number of historical ledgers that have been validated by consensus. Only validated ledgers are certain to be correct and immutable. Each ledger version consists of the following parts:

- Header, containing the Ledger Index, hashes of its other contents, and other metadata.
- Transaction tree, with transactions that were applied to the previous ledger to make this one.
- State tree, with all the ledger objects that contain the settings, balances, and objects in the ledger as of this version.

Ripple uses SHA256 and SHA512 for all its hashing operations and supports the following cryptographic signing algorithms [16]:

- ECDSA using the elliptic curve secp256k1.
- EdDSA using the elliptic curve Ed25519.

Servers in Ripple communicate to each other using the XRP Ledger peer protocol (RTXP). To establish a connection, a peer makes an outgoing TLS-encrypted connection to a remote peer, then sends a HTTP request with no message body.

2.5 Quorum

Quorum is an enterprise-focused version of Ethereum, and therefore uses much of the same algorithms. It provides level2 service on top of Ethereum which enables it to perform private transactions, and speeds them up using a different consensus algorithm. Public transactions are executed as on standard Ethereum, but private transactions use a different system. Before propagating such transaction, its payload is replaced by a hash. After receiving the transaction, only authorised parties will be able to replace the hash with the actual payload. Quorum uses KECCAK and SHA3 hash functions, ECDSA with secp256k1 constants and Elliptic Curve Integrated Encryption Scheme (ECIES) with AES ciphers.

Each peer in Quorum consists of several semi-independent modules, each communicating using several technologies. It includes Pretty Good Privacy (PGP), mutually-authenticated TLS (in some cases over HTTPS) and NaCl (Networking and Cryptography library).

3 Cryptanalysis

Now that we have a more-or-less complete list of algorithms, let us look into their known strengths and weaknesses. One way to describe the strength of a cryptographic primitive is in terms of a "security level" or "security claim". It can be measured in "bits", with N-bit security level meaning that attacker would have to process at least N^2 operations to break it, or in terms time and memory required. There are other models that better represent true costs for the attacker, but security level offers a convenient way to compare different algorithms and is especially useful in hybrid cryptosystems. Public-key algorithm security is highly dependent on the security of the associated hash function but we will look at them separately from hash functions and ciphers.

3.1 Hash Functions

Cryptographic hash functions can be divided into two categories - "provably secure" ones, based on mathematical problems and proven to be secure, and "ad hoc" functions, where the bits of the message are mixed to produce the hash. Provable functions, while secure, are not unbreakable, and usually too inefficient for practical purposes. Ad hoc functions, on other hand, are much faster, but have no hard guarantees of security, and are subject of rigorous cryptanalysis from researchers around the world. Experts seek vulnerabilities which would allow an attacker to break the function faster than its security claim would indicate.

If such an attack is found, it does not automatically mean that the function no longer secure. An attack can only break a reduced version of the hash or have other issues. We will label such attack as "partial". If an attack does indeed break all rounds of a function, but has not been successfully performed in practice, despite having a substantial theoretical base, we will call it a "theoretical" attack. In an attack was successfully demonstrated in practice it is labeled as a

"practical" attack. The summary of currently known attacks on aforementioned hash functions is presented in Table 1. We also include a Streebog hash function in the comparison as its use is required by the Russian Government.

Table 1. Hash function resistance.

Function	Security claim (collision, preimage)	Best collision attack	Best preimage attack
MD2	2^{64}, 2^{128}	practical, $2^{63.3}$ time, 2^{52} memory [17]	practical, 2^{73} time, 2^{73} memory [18]
MD5	2^{64}, 2^{128}	practical, 2^{18} time [19]	theoretical, $2^{123.4}$ time [20]
SHA-1	2^{80}, 2^{80}	practical, $2^{63.1}$ time [21]	partial, 45 of 80 rounds [22]
SHA-256	2^{128}, 2^{256}	partial, 31 of 64 rounds, $2^{65.5}$ time [23]	partial, 43 of 64 rounds, $2^{254.9}$ time, 2^{6} memory [24]
SHA-512	2^{256}, 2^{512}	partial, 24 of 80 rounds $2^{32.5}$ time [25]	partial, 46 of 80 rounds $2^{511.5}$ time, 2^{6} memory [24]
SHA-3	Up to 2^{512}	partial, 6 of 24 rounds, 2^{50} time [26]	none
RIPEMD-160	2^{80}, 2^{160}	partial, 48 of 80 rounds 2^{51} time [27]	partial, 31 of 80 rounds [28]
BLAKE2s	2^{128}, 2^{256}	partial, 7.5 rounds, 2^{184} time [29]	partial, 6.75 rounds, $2^{253.8}$ time [30]
Streebog	2^{256}, 2^{512}	partial, 9.5 rounds of 12, 2^{176} time, 2^{128} memory [31]	theoretical, 2^{266} time, 2^{259} memory [32]

3.2 Public-Key Algorithms

Public-key cryptography (or asymmetric cryptography) generally makes use of algorithms based on mathematical problems to produce one-way functions. This is what allows users to freely distribute their public keys without compromising security. However, different algorithms are based on different mathematical problems and have different security levels. For practical use, it is important to understand ratios of security-to-size each algorithm provides in regards to their public keys. The summary is presented in Table 2. We also include GOST R 34.10-2012 digital signature standard, required by Russian government in the comparison.

Table 2. Public key algorithm security comparison.

Algorithm	Security level (bits)	Public key size (bits)	Largest broken key (bits)
RSA PKCS#1	128	3072	768
DSA	128	3072	530
ECDSA (secp256k1)	128	256	114
ECDSA (secp256r1)	128	256	114
ECDSA (secp384r1)	192	384	114
ECDSA (secp521r1)	256	521	114
EdDSA (ed25519)	128	256	unknown
ECIES	128	256–283	unknown
ElGamal	128	3072	768
SPHINCS-256	128	1024	unknown
GOST R 34.10-2012	256	512	114

3.3 Ciphers

Comparing cipher security is harder, because it depends heavily on the specific attack used and if the cipher is vulnerable to it in the first place. Thankfully, only a few cipher families are widely-used, so we can take a detailed look into each of them.

AES-128 or Advanced Encryption Standard is a subset of of the Rijndael block cipher family, each with a block size of 128 bits and key lengths of 128, 192 and 256 bits. It is the first (and only) publicly accessible cipher approved by the National Security Agency (NSA) of United States for top secret information. AES is based on a design principle known as a substitution–permutation network, and has in-built support in many hardware systems and devices [33].

Until 2009, the only known successful attacks against AES were side-channel attacks on some specific implementations [33]. Such attacks seek to exploit weaknesses in incorrect or partial algorithm implementation, which usually significantly lowers its security level. In that year, a related-key attack was discovered which makes use of AES's key schedule. This attack requires several different known keys and some knowledge of their mathematical relationship. Its complexity was soon brought down to $2^{99.5}$, however such attacks only work in improperly-designed protocols. Subsequent versions of this attack were published later, but none works against a full 14-round version of AES-256.

In November 2009, the first known-key distinguishing attack was published, but it only worked against 8-round version of AES-128, while full version uses 10 rounds. Only in 2011 the first key-recovery attacks on full AES were discovered [33]. However, their workloads are infeasible on current hardware, with current records being 2^{126} for AES-128, $2^{189.9}$ for AES-192 and $2^{254.3}$ for AES-256 [34].

At present, there are no practical attacks on full and correct AES implementations themselves, but there are multiple side-channel attacks, which target specific hardware. One of the fastest allows to recover the complete 128-bit AES key in just 6–7 blocks of plaintext/ciphertext [35].

ChaCha20 is a family of ciphers closely related to Salsa20. Both ciphers use pseudorandom function based on ARX (add-rotate-xor) operations. Supported key lengths are 128-bit and 256-bit, with 8, 12 and 20-round versions. The only currently known attack can only break 256-bit ChaCha6 with complexity 2^{139} and ChaCha7 with complexity 2^{248}. 128-bit ChaCha6 within 2^{107}, but fails to break 128-bit ChaCha7 [36].

CAST5 or (CAST-128) is a symmetric-key block cipher, approved for Government of Canada use by the Communications Security Establishment, with a 64-bit block size and a key size of between 40 and 128 bits (in 8-bit increments) [37]. Best currently known attack on it is a theoretical one on 6 of 16 rounds ($2^{88.51}$ time, $2^{53.96}$ memory) [38].

3DES - Triple DES (TDES) or Triple Data Encryption Algorithm (TDEA or Triple DEA), is a symmetric-key block cipher, which applies DES three times to each block [39]. In general, Triple DES with three independent keys has a key length of 168 bits (three 56-bit DES keys), but due to the meet-in-the-middle attack, the effective security it provides is only 112 bits. Practical Sweet32 attack on 3DES-based cipher-suites in TLS required $2^{36.6}$ blocks for a full attack, but it is possible to get a collision after around 2^{20} blocks which takes 25 min. OpenSSL does not include 3DES by default since version 1.1.0, and considers it a "weak cipher".

IDEA or International Data Encryption Algorithm is a symmetric-key block cipher intended as a replacement for the Data Encryption Standard (DES) [40]. In 2012, full 8.5-round IDEA was broken using a narrow-bicliques attack (($2^{126.1}$ time), however this attack does not threaten the security of IDEA in practice. The very simple key schedule makes IDEA subject to a class of weak keys containing a large number of 0 bits produce weak encryption [40].

Twofish is a symmetric key block cipher with a block size of 128 bits and key sizes up to 256 bits [41]. Twofish's distinctive features are the use of precomputed key-dependent S-boxes, and a relatively complex key schedule. Best attack is an Impossible differential attack on 6 of 16 rounds (2^{256} time).

Blowfish is a symmetric-key block cipher, which provides a good encryption rate in software and no effective cryptanalysis of it has been found to date [42]. Its use of a 64-bit block size makes it vulnerable to birthday attacks, particularly in contexts like HTTPS, and the SWEET32 attack demonstrated how to perform plaintext recovery against it.

Camellia is a symmetric key block cipher developed by Mitsubishi Electric and NTT of Japan [43]. It has a block size of 128 bits and key sizes of 128, 192 and 256 bits. There are no known successful attacks that weaken the cipher considerably.

Kuznyechik ("grasshopper") is a symmetric block cipher with a block size of 128 bits and key length of 256 bits, defined in the National Standard of the Russian Federation GOST R 34.12-2015 and also in RFC 7801. Kuznyechik is based on a substitution-permutation network, though the key schedule employs a Feistel network. Currently there are only two known attacks on the cipher. One is a meet-in-the-middle attack on the 5-round version, allowing to recover the key with 2^{140} time and 2^{153} memory [44]. The second is a side-channel attack [45].

3.4 Cryptosystems

Most blockchain platform make use of third-party cryptographic solutions, which provide multi-level protection and include several cryptographic primitives. It is almost impossible to judge the security of the overall system, especially considering all the possible combinations.

PGP - Pretty Good Privacy is an encryption program that provides cryptographic privacy and authentication for data communication. It can be used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications [46].

Symmetric cryptography is performed using AES, CAST5, 3DES, IDEA, Twofish, Blowfish or Camellia with a session key, which is encrypted using the public key provided by RSA or Elgamal (key length 1024–4096 bits). Signing is done using RSA or DSA, combined with MD5, SHA-1, RIPEMD-160, SHA-2 hash functions.

NaCl - abbreviation of "Networking and Cryptography library" is a public domain library for network communication, encryption, decryption and signatures.

Public-key cryptography uses EdDSA (Ed25519) for signatures, while private-key is done with Salsa20 and AES. Hashing is performed using SHA-2 or BLAKE2.

4 Conclusion

In this paper we presented an overview of cryptographic algorithms used in prominent enterprise-oriented blockchain platforms and compared their security level. As expected, only secure algorithms see any wide use and all platforms provide an acceptable security, but show very low agility in terms of range of supported algorithms, frequently only using a single one for all parts of the system.

This choice of a particular algorithm may be dictated by parameters other than security. For example, Ethereum developers justify their choice of hash function by its inability to be optimized for Application-specific Integrated Circuits, and therefore requirement to use relatively-expensive GPU/CPU mining farms. This is expected to prevent consolidation of mining resources in the hands of any specific group. This can also be viewed from the users perspective - if the only available algorithm does not have the required qualities, user has no choice but to look for another platform. This lack of choice can create problems on the part of the user, especially if they are under pressure from a third party.

Additionally, the security claim of any algorithm can be viewed as proportional to the length of time before it is eventually broken. This is especially concerning considering the developments in quantum computing. Very few blockchain platforms provide any sort of post-quantum secure algorithm, and without ability to choose, the user's security is at mercy of developers. If the currently used algorithm becomes compromised, it would require developer action to change it.

References

1. Fersht, P.: The top 5 enterprise blockchain platforms you need to know about. https://www.horsesforsources.com/top-5-blockchain-platforms_031618. Accessed 20 Mar 2019
2. Average Confirmation Time. https://www.blockchain.com/ru/charts/avg-confirmation-time
3. Cryptographic hash function. https://en.wikipedia.org/wiki/Cryptographic_hash_function
4. SEC 2: Recommended Elliptic Curve Domain Parameters. http://www.secg.org/sec2-v2.pdf
5. Digital Signature Algorithm Patent. https://patents.google.com/patent/US5231668
6. Issue about SHA-3/KECCAK confusion. https://github.com/ethereum/EIPs/issues/59
7. The RLPx Transport Protocol. https://github.com/ethereum/devp2p/blob/master/rlpx.md
8. Multihashing in Ethereum 2.0. https://ethresear.ch/t/multihashing-in-ethereum-2-0/4745/8
9. Peer node configuration file, line 275–277. https://github.com/hyperledger/fabric/blob/release-1.4/sampleconfig/core.yaml
10. "Use SHA-2 as the default hash in all Hyperledger fabric components". https://jira.hyperledger.org/browse/FAB-887
11. Fabric CA User's Guide. https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/users-guide.html
12. Chaincode for Developers. https://hyperledger-fabric.readthedocs.io/en/release-1.4/chaincode4ade.html
13. Cipher suites supported by Corda. https://docs.corda.net/cipher-suites.html
14. Blockchained Post-Quantum Signatures. https://eprint.iacr.org/2018/658.pdf
15. Networking and messaging. https://docs.corda.net/messaging.html

16. Ripple Cryptographic Keys. https://developers.ripple.com/cryptographic-keys.html
17. Cryptanalysis of MD2. https://link.springer.com/article/10.1007
18. An improved preimage attack on MD2. https://eprint.iacr.org/2008/089.pdf
19. Fast Collision Attack on MD5. https://eprint.iacr.org/2013/170
20. Finding Preimages in Full MD5 Faster Than Exhaustive Search. https://link.springer.com/chapter/10.1007
21. The first collision for full SHA-1. https://shattered.it/static/shattered.pdf
22. Preimages for Reduced SHA-0 and SHA-1. https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=36848
23. Improving Local Collisions: New Attacks on Reduced SHA-256. https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=69018
24. Preimages for Step-Reduced SHA-2. https://link.springer.com/chapter/10.1007
25. New Collision Attacks against Up to 24-Step SHA-2. https://link.springer.com/chapter/10.1007
26. Song, L., Liao, G., Guo, J.: Non-full sbox linearization: applications to collision attacks on round-reduced Keccak. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10402, pp. 428–451. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63715-0_15
27. On the Collision Resistance of RIPEMD-160. https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=17675
28. Preimage Attacks on Step-Reduced RIPEMD-128 and RIPEMD-160. https://link.springer.com/chapter/10.1007
29. The Boomerang Attacks on BLAKE and BLAKE2. https://eprint.iacr.org/2014/1012.pdf
30. Higher-Order Differential Meet-in-The-MiddlePreimage Attacks on SHA-1 and BLAKE. https://eprint.iacr.org/2015/515.pdf
31. Cryptanalysis of GOST R Hash Function. https://eprint.iacr.org/2013/584
32. The Usage of Counter Revisited: Second-Preimage Attack on New Russian Standardized Hash Function. https://eprint.iacr.org/2014/675
33. Advanced Encryption Standard. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard
34. Improving the Biclique Cryptanalysis of AES. https://link.springer.com/chapter/10.1007
35. Highly Efficient Algorithms for AES Key Retrieval in Cache Access Attacks. https://ieeexplore.ieee.org/document/7467359
36. New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba. https://eprint.iacr.org/2007/472.pdf
37. CAST-128. https://en.wikipedia.org/wiki/CAST-128
38. New Linear Cryptanalytic Results of Reduced-Round of CAST-128 and CAST-256. https://link.springer.com/chapter/10.1007
39. Triple DES https://en.wikipedia.org/wiki/Triple_DES
40. International Data Encryption Algorithm. https://en.wikipedia.org/wiki/International_Data_Encryption_Algorithm
41. Twofish. https://en.wikipedia.org/wiki/Twofish
42. Blowfish. https://en.wikipedia.org/wiki/Blowfish_(cipher)
43. Camellia. https://en.wikipedia.org/wiki/Camellia_(cipher)
44. A Meet in the Middle Attack on ReducedRound Kuznyechik. https://eprint.iacr.org/2015/096.pdf
45. Fault Analysis of Kuznyechik. https://eprint.iacr.org/2015/347.pdf
46. Pretty Good Privacy. https://en.wikipedia.org/wiki/Pretty_Good_Privacy