Procurer

REPUBLIC OF ESTONIA
INFORMATION SYSTEM AUTHORITY

Cryptographic Algorithms Lifecycle

REPORT

2017

Cryptographic algorithms lifecycle report 2017

Technical document

Version 2.0

May 23, 2018

30 pages

Doc. A-101-9

Project leaders:	Kaur Virunurm (Estonian Information System Authority)
	Mari Seeba (Cybernetica)
Contributing authors:	Arne Ansper, MSc (Cybernetica)
	Ahto Buldas, PhD (Cybernetica)
	Jan Willemson, PhD (Cybernetica)

Estonian Information System Authority, Pärnu maantee 139a, 15169 Tallinn, Estonia.
E-mail: ria@ria.ee, Web: `https://www.ria.ee`, Phone: +372 663 0200.

Cybernetica AS, Mäealuse 2/1, 12618 Tallinn, Estonia.
E-mail: info@cyber.ee, Web: `https://www.cyber.ee`, Phone: +372 639 7991.

Contents

1 Introduction

This report is a follow-up to four previous studies from 2011 to 2016 [3, 4, 5, 7]. The report covers three major topics.

In Chapter 2, we look at the most important cryptographic algorithms that were broken last year (of which, there are fortunately only a few) and give recommendations on the primitives and key lengths to be used. In this area, the major open question concerns the development speed of a general purpose quantum computer, which is very difficult to predict, but directly determines the lifetime of currently used algorithms.

The major cryptographic event in Estonia in 2017 was undoubtedly the discovery of an ID-card vulnerability by a Czech team of researchers. In Chapter 3, we briefly outline the nature of the problem and describe the steps taken to solve it along with the introduction of new elliptic curve cryptography algorithms.

Globally, however, one of the fastest growing cryptographic applications is the blockchain. Chapter 4 of this study provides a systematic overview of blockchains and helps the reader decide whether and which blockchain technology meets their needs. As a complementary contribution, drafting of this overview has resulted in the development of blockchain terminology in the Estonian language.

2 Security levels of cryptographic primitives and protocols

2.1 Assessment of key lengths

Progress in cryptanalysis

In February 2017, a Google-sponsored team of researchers finally found the long-awaited SHA-1 hash collision [26]. Although the collision does not automatically mean that all use scenarios have become insecure, we hereby repeat the recommendation provided in all previous cryptographic algorithm lifecycle studies to deprecate the usage of SHA-1. All hash functions of the SHA-2 and SHA-3 families (considering the desired level of security, see Section 2.2) are suitable as replacements.

During the period covered in this report (2016–2017), no significant breakthroughs have been made in cracking other main cryptographic primitives. Therefore, the list of recommended algorithms and key lengths is largely the same as in the previous report [7].

The primary unknown variable in estimating the life-time of a key length is the rate of development of general purpose quantum computers. In October 2017, Intel announced the release of a 17-qubit superconducting quantum computing chip.[1]

How significant is this step? Google researchers have estimated that a quantum computer with approximately 50 superconducting qubits would suffice to beat the standard computers of today at performing some functions [16].

Proos and Zalka found that about $2n$ qubits were required to factor an n-bit RSA key and $6n$ qubits were required to find a discrete logarithm in an n-bit elliptic curve group [23]. Thus, breaking a 2048-bit RSA requires approximately 4096 qubits and finding a discrete logarithm for curve P-256 requires about 1500 qubits. Proos and Zalka also point out that due to the instability of physical quantum bits, error-correction codes should be added to bits used in the Shor's algorithm, which in practice can cause these figures to be a few times higher.

It has been shown [19] that using Grover's quantum algorithm, a general purpose quantum computer can accelerate the full search of k-bit key spaces of symmetric ciphers (e.g. AES) from 2^k operations to $2^{\frac{k}{2}}$ operations. The number of qubits needed for this is shown in Table 1. This means that upon the emergence of quantum computers with sufficiently long quantum registers, block cipher key lengths must be doubled to achieve the same security level as today. This application also needs to take into account the need for error correction.

[1] https://newsroom.intel.com/news/intel-delivers-17-qubit-superconducting-chip-advanced-packaging-

Table 1. Number of qubits needed to crack AES

AES key length	Number of qubits needed
128	2953
192	4449
256	6681

Aggarwal et al. predicted in 2017 that over the period of 2025–2035, the number of available qubits in a single computer would reach $10,000$ [12]. As their estimates were based on only a very limited set of data points, this prediction leaves room for scepticism.

However, the development, standardisation and implementation of quantum-safe algorithms require a considerable amount of time. Therefore, it is necessary to undertake preparations for transitioning to post-quantum algorithms today. In September 2017, the ITU (*International Telecommunication Union*) issued a statement announcing the planned update of the X.509 standard [11]. The aim of the update is to provide support for several public key algorithms in X.509 certificates. This support will allow the gradual deployment of post-quantum algorithms.

Other standardisation organisations, such as NIST[2] and ETSI[3], are also involved in selecting post-quantum algorithms suitable for implementation. In Estonia, work on selecting base technologies for the new generation of crypto solutions should commence in the coming years.

Electronic identity solutions are particularly critical. The contract to issue ID-cards lasts for five years, and the validity period of the cards is five years as well. The procurement process itself also requires about three years. Altogether, the lifecycle of the ID-cards lasts for approximately 13 years.

In the event that the most optimistic estimate for the arrival of the quantum computer materialises, the life of the next generation ID-card will partially coincide with that period. For this reason, the procurement of the new ID-card (or more generally any eID platform) should recognise the necessity to introduce quantum-safe algorithms. It is necessary to clarify the plans and capabilities of manufacturers and stay up to date with international efforts in the field of standardisation of new cryptographic mechanisms. Much is at stake for Estonia, and the timely collection of information and the planning of activities will significantly reduce future risks and costs.

2.2 Key length recommendations

As mentioned above, aside from the SHA-1 collision, there have been no significant cryptographic breakthroughs in the past year. Therefore, the recommendations given in the previous versions of the report [5, 7] are still largely applicable.

These recommendations are mainly based on the reports of ECRYPT II and ENISA [13, 25], most recently updated in 2012 and 2014. In this study, we rely on the assessments made by NIST (USA National Institute of Standards and Technology) in 2016 [14] and the

[2] https://csrc.nist.gov/Projects/Post-Quantum-Cryptography
[3] https://portal.etsi.org/TBSiteMap/CYBER/CYBERQSCToR.aspx

recommendations of the German Federal Office for Information Security (*Bundesamt für Sicherheit in der Informationstechnik*) of 2017 [8].

Table 2 summarises their recommendations for the main cryptographic algorithm families. The security level of an algorithm is determined in bits, where the b-bit security level of an algorithm stands for the most efficient known form of attack that uses computing resources equal to encrypting approximately 2^b blocks with a block cipher.

In Table 2, the column 'DSA, DH' refers to the DSA signature algorithm and Diffie-Hellman key exchange over a quotient ring, with the recommended length of the public and secret key denoted as L and N, respectively. Column 'RSA' provides the recommended length for the RSA modulus, while column 'ECC' displays recommended key lengths for elliptic-curve cryptography algorithms. Columns 'Block ciphers' and 'SHA-2, SHA-3' are the recommended key length for symmetric-key algorithms and the output length of hash functions, respectively. AES continues to be the main recommended symmetric-key algorithm.

Table 2. Recommended key lengths for cryptographic algorithms

Security level	DSA, DH	RSA	ECC	Block ciphers	SHA-2, SHA-3
128	$L = 3072$, $N = 256$	3072	256 . . . 383	128	256
192	$L = 7680$, $N = 384$	7680	384 . . . 511	192	384
256	$L = 15360$, $N = 512$	15360	512+	256	512

The NIST report [14] finds that all key lengths in Table 2 are suitable for both short and medium term use (up to 2030 and beyond). Algorithms providing only 112-bit security level (e.g. SHA2-224 and 3TDEA, as well as 2048-bit RSA) are suitable for short term use only and in legacy systems. The BSI 2017 report [8] clarifies the latter recommendation, calling for the deprecation of algorithms with 128-bit security level by 2022.

We stress that these estimates do not take into account possible quantum computer attacks. Thus, for fear of the impending emergence of quantum computers, National Security Agency (NSA) of US recommends against using elliptic curve keys with lengths under 384 bits, or AES with lengths under 256 bits [6]. This recommendation was also one of the reasons why the elliptic curve P-384 was chosen for the new cryptographic algorithms of the Estonian ID-card.

2.3 TLS

TLS (Transport Layer Security) is a protocol suite used to secure the majority of Internet traffic. Currently (in early 2018), the latest version of the standard is still 1.2, although work on the newer version, 1.3, should finish shortly.

Compared to version 1.2, 1.3 brings several significant updates, including:

- Exclusion of a number of weak ciphers and operating modes (e.g. SHA-1, RC4, DES,3DES, AES-CBC, MD5, etc.),
- Symmetric algorithms work only in the authenticated mode,
- handshake requires less messages and is, therefore, faster,
- the addition of new elliptic-curve cryptosystems (e.g. X25519).

Although support for TLS 1.3 already exists in the source code of some browsers and libraries, it has not been adopted for default use, as other network devices and services have not caught up in its implementation [4]. It is probable that after its formal adoption, the full transition to the new standard will take some additional time. For the time being, we recommend using TLS version 1.2 cipher kits with the strongest possible cryptographic primitives to ensure compatibility. Suitable examples are:

- TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384,
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,
- TLS_DHE_RSA_WITH_AES_256_GCM_SHA384.

In any case, the use of known weak primitives (e.g. DES, MD5, SHA-1) should be prohibited.

[4] https://blog.cloudflare.com/why-tls-1-3-isnt-in-browsers-yet/

3 Vulnerability in the Estonian ID-Card

3.1 Problem definition

In late 2017, Nemec, Svenda et al. published their research on the generation of RSA keys on several widespread hardware platforms [21]. The research also revealed this vulnerability on the Infineon chip, used in the Estonian ID-cards. The following is a brief overview on the cryptographic essence of the vulnerability.

Setting up an RSA key pair starts from generating two secret prime numbers p and q of roughly equal length, and multiplying them to get the public modulus $N = pq$. E.g. if the modulus N is required to have 2048 bits, both p and q need to be approximately 1024 bits long. The problem stems from the fact that generating such primes is a non-trivial task, especially for a limited platform like a smart card chip.

To speed up the prime number generation, smart card manufacturers implement various optimisations. It is the optimisation mechanism chosen for the Infineon chip that was actually attacked by Nemec, Svenda *et al.*.

By observing the distribution of remainders after division by small primes of the moduli generated by the chip, the researchers were able to determine that the primes produced during the RSA key generation follow the pattern

$$p = k \cdot M + (65537^a \bmod M) \ , \tag{1}$$

where M is the product on n first prime numbers for some n that guarantees M to be of roughly the same length as the required p and q. So for example, in order to generate 1024-bit primes, M would be the product of 126 first primes from 2 to 701 (so M itself is a 961-bit number).

A naïve approach to factoring N would be going through possible values of a and using an algorithm due to Don Coppersmith [18] to find k. This algorithm would need to go through $\mathrm{ord}_M(65537) \approx 2^{254}$ candidates of a.

However, the Coppersmith's algorithm allows quite a lot of flexibility. Namely, it only requires $\frac{1}{4}$ of the bits of the modulus to be known to succeed. Hence for factoring, M can be replaced by its divisor M' that would have more than $\frac{\log_2 N}{4}$ bits, but for which the search space of $\mathrm{ord}_{M'}(65537)$ elements would be manageable. Note that for such an M', the primes p would still be expressible in the form

$$p = k' \cdot M' + (65537^{a'} \bmod M') \ .$$

The explicit value for a suitable M' is not given in the article [21], but a method of searching for one is presented.

The authors [21] claim that using their value for M', the search space for a' in case of 2048-bit modulus would decrease to about 2^{34}. As a result, a 2048-bit modulus generated by Infineon chips can be factored in an expected time of about 140 CPU years (which would cost about USD1000 in electricity). Later, Dan Bernstein presented an optimised version of the attack which would be even $5 \ldots 25\%$ more efficient[5].

3.2 Solutions and lessons learned

The Estonian ID-card is used nation-wide for both governmental and private sector services. Several critical processes rely on the operability of the digital identity infrastructure, while some of the systems support the ID-card exclusively (in Estonia, mobile-ID is also available for authentication and digital signatures in many, but not all services). "Shutting down" all ID-cards would have had a severe impact on the entire country, including the economic impact to the businesses, and probably resulted in the deployment of substitute measures with lesser security standards while making several services more costly and time-consuming.

Replacing all the cards physically would have taken a long time, given the necessary steps for creating an entirely new card: choosing the chip, programming and testing the application, acquiring the necessary certification and procuring the new cards equipped with the new chips. Only after these steps would the actual replacement procedure take place, limited by the low amount of available personnel in the Police and Border Guard service points. According to our estimates, the process would have taken at least a year, if not more, to complete.

Solution

The alternative was to create a solution that would bypass the vulnerability by updating the existing cards. There is a requirement that keys must be generated on-card and never leave the card. This is required in order to be able to use the ID-card to give legally binding digital signatures (see ID-card Certification Policy [10], Section 6.1.1). The vulnerability that we had to bypass was found in the on-chip RSA key generation procedure. We had to abandon the implementation of RSA in this particular chip altogether. Thankfully, the ID-card chip also supported elliptic curve cryptography, which has no known security threats. The solution was to update the cards to use elliptic curve cryptography instead of RSA. We analysed the possibilities to continue with RSA by using alternative key lengths, but ruled them out for several considerations (e.g. there was no capability of generating 3k keys within the chip). Moreover, the decisions had to be made before the article on the vulnerability was published when all the details of the attack were not yet known, and there was uncertainty regarding the security of longer keys. In that situation, migrating to ECC was the most viable decision.

NIST curve P-384 was chosen from the elliptic curves supported by the ID-card chip. Our main criterion for the choice was the level of support by operating systems and applications as well as recommendations by the NSA (see Section 2.2). The uptake of non-NIST elliptic curves with superior security properties by standardisation organisations, operating system and application developers has been slow and support is lacking. It must be noted that it is

[5] `http://blog.cr.yp.to/20171105-infineon4.txt`

highly recommended to require the use of new cryptographic primitives by chip manufacturers. We might be facing an incident in the future that requires replacing the algorithms currently in use, and it is best to be prepared.

Since mobile-ID had migrated to elliptic curve cryptography some years ago, most of the ecosystem (including digital signature applications) was prepared for the change. Even the changes to the profiles of the identity document certificates were minimal — certificate owner public key value was updated to accept elliptic curve P-384 (thanks to Mobile-ID, P-256 had already been enabled) [9]. An adjustment was also made to the certificate extensions, which disables the use of elliptic curve authentication certificates directly for encryption purposes.

The system for remote updates existed as well, as it had been in use before for replacing incorrectly coded certificates. The main weak point actually lay in the distribution — the system was not prepared to handle the updates for such a high volume of cards simultaneously.

Apart from the card application itself, ID-card drivers for different operating systems needed changing, web applications of the service providers needed updating and a new solution for the encryption and decryption of files had to be developed. During the process, several interesting RSA-specific ID-card use cases were noted that needed updating by service providers.

Operating systems and browsers generally supported the elliptic curves — there were a few issues noted with using elliptic curves on hardware crypto devices.

Devising the concept for the solution itself was done rather quickly, mainly due to the lack of alternatives. Most of the time was spent on the development and testing of the ID-card base software and card application, retuning the remote update system and updating the service provider systems to support the elliptic curves.

Practical consequences

The two core functions of the Estonian ID-card are authentication and digital signatures. Legally binding digital signatures in Estonia have always been time-stamped, retaining the authenticity of digitally signed documents even throughout this situation, as it is possible to provide evidence that the signature was given before the information about the vulnerability became available. If disputes over the validity of a particular digital signature arise in the future, they should be addressed separately. The timestamp may prove decisive in resolving these disputes. This is exemplary of how relevant the regulatory steps of devising a digital signature framework are.

The ID-card's current authentication function is no longer being affected – if service providers follow the procedure of validating the certificates, invalid and/or expired certificates affected by the vulnerability are prevented from accessing the services.

Should legitimate doubts arise in the future about the abuse of the authentication function in the 2014–2017 period due to this particular weakness, these cases should be addressed separately.

The main remaining problem regards encrypted documents sent over the years to ID-card holders. The confidentiality of these documents is still at stake. Due to the nature of

the vulnerability, an attacker may decrypt CDOC files in his possession without requiring access to the ID-card. At present, doing so would still require a significant investment, but over time, attacks will probably become cheaper. This example clearly illustrates that more effort should be put into designing and creating widely used encryption applications, and that risks, albeit unlikely, should be addressed nonetheless.

In the future, the infrastructure dependency on one digital identity platform must be decreased, the use of several alternatives must be encouraged and promoted. In addition, the update and replacement capacity, both remote and physical, should be increased. We also recommend the government to procure the readiness to act fast in *force majeure* situations from the eID providers. While deciding on the new eID platforms, the need to replace cryptographic primitives must be taken into account – particularly the possibility of the need to replace algorithms with those that are not even in existence yet.

Encryption for confidentiality purposes requires careful consideration. The emergence of sufficiently powerful quantum computers opens up the risk that data could be decrypted by parties storing communications or documents. Therefore, materials with a confidentiality term longer than 10–15 years (see Section 2.1) should not be transmitted using today's tools (e.g. encrypted for decryption with the current ID-card).

Hybrid encryption schemes for this purpose exist today, but their implementation requires solving integration problems. Solutions to these problems should take into account the possibility that the primitives used will be broken. Confidentiality mechanisms should be designed in such a way that if any of the algorithms used gets compromised, it would not jeopardise the confidentiality of the encrypted information in its entirety.

3.3 New cryptographic protocols introduced for the ID-card

As explained previously, in ID-cards, RSA had to be replaced with an elliptic curve cryptosystem. An interested reader will find a general introduction to elliptic curve mathematics in the report of 2015 [5]. The following is a more detailed description of two main protocols – Elliptic Curve Digital Signature Algorithm ECDSA and Integrated Encryption Scheme ECIES.

ECDSA

The setup of Elliptic Curve Digital Signature Algorithm (ECDSA) starts with the selection of parameters. Public parameters are defined by the elliptic curve P-384 being used (see standard FIPS 186-4 [1]). They contain

- a large prime number p and a finite field $GF(p)$ defined by it,
- the constant term b of the equation of the curve

$$y^2 = x^3 - 3x + b \bmod p \ ,$$

- the base point G represented by its coordinates (G_x, G_y), and
- the order n of the resulting group.

The key pair is generated by picking a coefficient $d \in [1, n-1]$, and calculating the public point $Q = dG$, where dG denotes the multiplication of point G by integer d (see [5]). The secret component of the key pair is the integer d and the public component is Q.

The generation of a digital signature is described in standard SEC1 [2]. The following steps are required to sign a message M.

1. Choose a random integer $k \in [1, n-1]$ and find $kG = R = (x_R, y_R)$.
2. Calculate $r = x_R \bmod n$. If $r = 0$, go back to step 1 and pick a new k.
3. Calculate the hash $e = H(M)$ of M.
4. Calculate $s = k^{-1}(e + rd) \bmod n$. If $s = 0$, go back to step 1.
5. The signature is the pair $S = (r, s)$.

The following steps are required to verify the signature (r, s) of message M.

1. Verify that r and s are integers within the range $[1, n-1]$. If not, an error message is returned.
2. Calculate the hash $e = H(M)$ of M.
3. Calculate
$$u_1 = es^{-1} \bmod n \quad \text{ja} \quad u_2 = rs^{-1} \bmod n \, .$$
4. Calculate
$$R = (x_R, y_R) = u_1 G + u_2 Q \, .$$
If R is the neutral element of the group, an error message is issued.
5. Verify that
$$r = x_R \bmod n \, .$$
If the last equality holds, the signature verification is successful; if it does not, an error message is returned.

The signing and authentication functionality of the ID-card are both based on the ECDSA protocol. For the authentication of the other party, the authenticator generates a new random number (nonce), whereupon the other party signs it using the authentication key. If the signature verification is successful, the other party is considered to be authenticated.

ECIES

Unlike in the case of RSA cryptosystem, the elliptic curve implementation in the ID-card does not offer direct support for the decryption operation. Therefore, the encryption and decryption functionality must be built on top of the key exchange layer. The resulting design is called the Elliptic Curve Integrated Encryption Scheme (ECIES), and it is the cryptographic basis of the new CDOC format of the Estonian ID-card. This scheme is described in more detail in standard SEC1 [2].

In essence, the ECIES scheme consists of two parts. First, a Diffie-Hellman key exchange is performed, where static curve point $Q = dG$ is the recipient's portion of the key, which originates from the public key certificate of the recipient. The sender chooses a dynamic (new for each encryption) secret value $c \in [1, n-1]$ and calculates the shared secret

$$cQ = cdG \, .$$

Using the deterministic Key Derivation Function (KDF), a symmetric key K is derived, and is then used for encrypting the message.

Along with the encrypted message, the recipient also receives the curve point cG (recall that the base point G is public, but coefficient c is a sender's secret). The recipient uses the secret value d and calculates

$$d(cG) = cdG \ ,$$

from which, using KDF, they can derive the secret key required for decryption.

The second part of the ECIES protocol is decryption. In principle, the key K could be used directly to encrypt the message, but in the case of the Estonian ID-card, a scenario where one file is encrypted for multiple recipients must also be supported. A sensible way to achieve this is to encrypt the file once with a unique universal transport key and then encrypt the transport key for each individual recipient B_i using the generated key K_i.

The SEC1 standard [2] also prescribes the use of message authentication code for encryption. In the case of the new CDOC format of the Estonian ID-card, it has been replaced with the AES-GCM authenticated encryption mode.

4 Blockchains

On the global scale, one of the fastest growing cryptographic applications is the blockchain. This chapter provides a systematic overview of blockchains and helps the reader decide whether (and which) blockchain technology meets their needs. As an additional contribution, the drafting of this overview has resulted in the development of blockchain terminology in the Estonian language (Table 3).

4.1 Terminology

Blockchain is a data structure composed of successive data blocks. Each new block is either created after a fixed time period or as a result of some other event, such as successful mining.

The content of the blocks as a data structure represents a specifically encoded ledger which records events that may have informal, commercial or legal significance. The blockchain format is known to its operators and users.

The integrity of the blockchain is ensured by iterative hashing. By applying a hash function to block data and the hash of the previous block, a hash is calculated for each block. The integrity of the hashes is ensured through the use of:

- **digital signatures**, i.e. digital signatures of persons authorised in some way (with public keys),
- **hashed time-stamping** that uses a publication mechanism, or
- **non-interactive time-stamping** in the form of special formatting rules for blocks that make the creation of valid blocks a computationally complicated task.

The blockchain is usually stored and managed in the form of a distributed ledger, with multiple parties keeping a copy of the ledger. This implies the use of a handshake protocol between the components.

One of the most popular applications of blockchains is cryptocurrency, where the ledger displays user account balances and inter-user payments in currencies defined within the ledger itself and not necessarily traditional currencies, such as the dollar, euro, etc. Nevertheless, cryptocurrency may be traded on the stock exchange and exchanged for traditional money.

The most widely recognised cryptocurrency system is Bitcoin, which uses bitcoins and satoshis as currency.

Blockchain systems can be

- **centralised**, where the ledger is managed as a centralised service by one legal entity. For example the Guardtime system.
- **permissioned ledgers**, in which the provision of services is distributed between a number of fixed actors acting on a contractual or other legal basis. For example, Ripple, which enables interbank transactions, or Sovrin, which is managed by financial institutions and is seeking to build a global decentralised identity system.
- **permissionless ledgers**, where service providers are not fixed and in principle, anyone can start operating the service. For example, Bitcoin, Ethereum, and others.

Most permissionless blockchain systems include an independent cryptocurrency. The reason is that in the absence of an inter-operator contract, there are usually no other incentives to guarantee voluntary management of the blockchain.

4.2 Historical background

Modern blockchain technologies are based on a number of well-known cryptography-related concepts

Hash functions appeared in the 1970s in relation to the works of Merkle [20], Rabin [24], Yuval [27], and others

Hashed timestamping appeared in the 1990s in relation to the works of Haber, Bayer, and Stornetta [15].

Consensus protocols investigated already as early as the 1970s. The term Byzantine agreement appeared at the beginning of the 1980s in relation to the works of Pease, Shostak, Lamport, and others [22].

Electronic cash appeared already in the beginning of the 1980s in relation to the works of Chaum and others [17].

4.3 Layers of the blockchain technology

The blockchain can be divided into the following layers

- **Network layer**. A software/hardware protocol that governs the integration of new components, the process of connecting new components to the network, sending messages to other components, and receiving messages from other components.
- **Propagation layer**. A protocol for block transfer between network components and for adding new blocks to the ledger.
- **Semantic layer**. A specification describing the format of the block as a part of the ledger and the relationship between neighbouring blocks as well as algorithms that determine the format and verify relationships.
- **Application layer**. The code that produces the desired functionality which is presented in a programming language determined by the rules of the semantic layer, e.g. *Solidity* in the *Ethereum* blockchain system.

4.4 Ledger

The ledger is a constantly updated electronic document that records events of informational, commercial or legal significance. The ledger format depends on its field of use. Organisationally the ledger includes:

- **Users**: Persons or entities who/which add entries to the ledger as appropriate. Users are usually identified in the ledger by their public key, which is mostly associated with traditional digital signatures, such as RSA, DSA, ECDSA, etc. The public key or its hash is generally treated as an account number.
- **Operators**: Persons who ensure the validity, maintenance, and operation of the ledger. Operators only accept and add items to the ledger meeting the format. Operators may have to personally add data to the ledger if it is required by the formatting rules.

Technically, operators are viewed as automatically acting entities. In legal terms, operators can be either legal or physical persons, and their participation in the management of a ledger may not be subject to regulation by any law. For example, Bitcoin system operators operate on a voluntary basis and in most cases anonymously.

In blockchain technology, ledgers are created block by block in the following steps:

1. Users send entries to the operator.
2. The operator verifies that incoming entries are in valid format and stores the valid entries.
3. The operator selects a number of stored entries (for example, all of them) and forms them into a block and adds it to the ledger.

Based on their activities, users can be divided into two types

1. **Passive users**, who only observe the contents of the ledger and may make business and administrative decision on that basis.
2. **Active users**, who, in addition to observing, also store personal entries into the ledger.

It is important to note that although the ledger maintenance activities of the operators may not be subject to regulation by any law, the contents of the ledger may have legal significance if the users (e.g. the state of Estonia) have so defined.

For example, if the ledger manages inter-user payments and account balances, the records may include items such as payer, payee, paid amount, etc. Ledger format may require that:

- the transaction item must be accompanied by a digital signature of the payer (for example with a traditional signing algorithm, such as RSA, ECDSA, etc.),
- the paid amount may not exceed the amount available on the payer's account,
- etc.

If account balances are defined as entries and a payment with valid formatting is being added to the ledger, the operator must also modify the entries representing the respective balances.

Logically, the format can be represented as the condition $\mathcal{P}$, such that $\mathcal{P}(L)$ is true if the content of the ledger L conforms to the format. Verification of the condition $\mathcal{P}$ can be a complex task, especially if users can create complicated conditions, for example, when the format allows for so-called smart contracts (See Section 4.5).

Verification of condition $\mathcal{P}$ may well be a complex task, but the validity of $\mathcal{P}(L)$ depends only on the contents of ledger L (i.e., its bit representation) and not on a single external factor, such as relationships between users as legal entities, etc.

4.5 Smart contracts

A smart contract is an entry in the ledger that represents a contract between users. The smart contract consists of a predicate term $\mathcal{P}_i$ and programmatic instructions $\mathcal{A}_i$ for modifying the contents of ledger L.

For example, users A and B can bet if the hash of a block created after period t is an odd or an even number, i.e., if the hash comes up as even, then A pays S units to B, and if it comes up odd, then B pays S units to A.

To achieve this, a contract description is stored in the ledger and both parties attach their digital signatures, which are also stored in the ledger, to the contract. The task of the operator is to monitor the smart contract and to update the balances of users A and B when time t has elapsed and the hash of the next block has been calculated. It is assumed that the ledger blocks contain the time of their creation as entries, otherwise the validity of ledger L would depend on an external parameter and would not be definable as the predicate $\mathcal{P}$, which depends solely on the bit representation of L.

For example, if we would like to create a sports results tote-board based on blockchain technology, then the ledger should also store the sport results. Naturally, the validity predicate cannot include result verification logic, but may require that the stored results be signed and verifiable using a fixed set of 'trustworthy' public keys.

Smart contracts may contain complex calculations. The blockchain system of Ethereum, for example, allows rules to be written in a full-fledged (so-called Turing complete) programming language. Infinite loops are avoided by limiting the number of executable instructions, defining a corresponding upper limit in the smart contract. The smart contract format may require charging a fee based on the number of instructions that the signers of the smart contract will pay to the operator of the smart contract.

4.6 Distributed ledger

If there are multiple ledger operators, then the ledger is called a distributed ledger. In such a case, entries sent by users are received, directly or indirectly, by all operators managing the ledger independently.

Purpose of distributedness

Distributedness may be required for two main reasons:

- **Trust**. A solution with a single operator is not deemed sufficiently reliable considering the potential for corruption.

- **Reliability**. A solution with a single operator is not deemed sufficiently reliable considering the possibility that an operator may become unavailable due to network or other issues.

Permissioned and permissionless ledgers

If the number of operators and their identities are known (e.g. identified by public keys), then the ledger is considered a permissioned ledger. It is assumed that in order to become an operator, other operators must accept the new operator, i.e., authorise it.

If the number of operators and their identities are not known, and, in principle, anyone can become an operator, then that ledger is considered a permissionless ledger.

4.7 Consensus protocols

Network problems may cause a situation where all operators do not receive all entries, and thus the received entries may vary across operators. To ensure consistency across all copies of the ledger, a consensus protocol between the operators is required.

Let $(B_1, B_2, \ldots, B_t)$ be the contents of the ledger at moment t, represented in blocks. Then, strategies for reaching consensus can be divided into two classes:

- **Agreement protocols**, in which the next block B_{t+1} is agreed upon between the operators only if all the preceding blocks $B_1, B_2, \ldots, B_t$ are agreed upon and consistent across the board.
- **Nakamoto consensus**, which uses a uniform chain comparison criterion: if there are two chains $(B_1, B_2, \ldots, B_m)$ and $(B'_1, B'_2, \ldots, B'_{m'})$, then one of them is "stronger" and the operators always prefer the strongest chain.

Agreement protocols

Operators $P_1, P_2, \ldots, P_n$ participating in the agreement protocol may have different versions of block B_{t+1}, represented as $B^1_{t+1}, B^2_{t+1}, \ldots, B^n_{t+1}$, respectively. The agreement protocol must be designed in such a way that a sufficiently large number of properly functioning operators involved in that protocol reach a consensus on the same version of B_{t+1}.

Agreement protocols are usually time- and message-intensive. If the number of operators n is high, then the use of the protocol by all n operators becomes impractical or even impracticable.

Therefore, some blockchain technologies (e.g. Algorand) use lottery based subcoalition $P_{i_1}, P_{i_2}, \ldots, P_{i_{n'}}$, where $n' \ll n$.

The main drawback of blockchain technologies based on agreement protocols is that the agreement protocol must necessarily succeed before the creation of new blocks. If, for some reason, it gets stuck, then it disrupts the entire ledger management.

The Nakamoto consensus

The main purpose of the Nakamoto consensus is to create blockchain $(B_1, B_2, \ldots, B_t, \ldots)$, and to arrive, as fast as possible, at a situation where, for the greatest possible $t' \leq t$, the chains of properly functioning operators have the identical initial segments $(B_1, \ldots, B_{t'})$.

The operator who managed to create block B_{t+1} sends it to other operators, who try to attach it to the "previous" known blocks $B_t, B'_t, \ldots$ to chain them. The correct chain will be chosen based on the chain comparison criterion.

It is important to understand that although the new block B_{t+1} refers unambiguously (based on hash values) to some earlier block B_t, there may not be an existing consensus yet about the previous block B_t, and it is therefore possible that later on, the chain $B_0, \ldots, B_{t-1}, B'_t, B'_{t+1}$, where blocks B'_t and B'_{t+1} may not match blocks B_t and B_{t+1}, is deemed the valid chain.

It may happen that block B_t contains some entry r, but later, the "winning" blocks B'_t and B'_{t+1} do not contain the entry r. Therefore, a situation may arise where some recently added entry r is removed from the ledger. However, it can be generally assumed that the "older" the entry r, the smaller the likelihood of it being removed.

Keeping that in mind, operators typically keep a separate account for incoming entries. If any entry r is removed from the ledger, the operator will re-enter it into the ledger as soon as possible.

As the Nakamoto consensus is based solely on chain comparison and does not assume that the entire preceding chain $(B_1, \ldots, B_t)$ is synchronised, the consensus protocols using the Nakamoto consensus are much more reliable during poor network conditions than agreement protocols. At the same time, the possibility to remove freshly added items and to wait for the ledger to "stabilise" should be considered.

Bitcoin block strength criterion

In the Bitcoin system, the strength s_i of block B_i is determined by the highest number of zero bits of the SHA-256 hash value in block expressed as a positive 256-bit integer i.e., blocks with the lowest hash values are always preferred. The strength s of the whole chain $(B_1, \ldots, B_m)$ is the sum

$$s = 2^{s_1} + 2^{s_2} + \ldots + 2^{s_m} \ .$$

In addition, the Bitcoin system has an agreed minimum strength that changes over time and is adjusted in such a way that enables the creation of one minimum strength block in about every ten minutes by utilising the full combined computational resource of all operators. As the likelihood for the creation of two such blocks is significantly smaller, multiple versions of B_{t+1} will not usually form.

Mining and proof of work

Finding a block with a specified strength is called mining. For example, in the Bitcoin system, mining is defined as the shuffling of the 64-bit parameter v in block B so that the hash value $h(B_v)$ would be as small as possible.

Finding and presenting a valid parameter v can also be called proof of work, because if the highest k bits of the hash value $h(B_v)$ are zero, it indirectly proves the expenditure of computational resource equivalent to 2^{k-1} hash computations.

Proof of work helps protect the integrity of the ledger by making it difficult to modify "old" entries, as changing block $B_{t'}$, where $t' \ll t$ at time t, would also require changing all

subsequent blocks $B_{t'+1}, \ldots, B_t$ equivalent to the expenditure of computational resource of

$$\frac{1}{2}(2^{s_{t'}} + 2^{s_{t'+1}} + \ldots + 2^{s_t})$$

hashings.

Economic aspects

Operators of permissionless ledgers must have some incentive to manage the system (e.g. mining). The only known way to motivate them is to have system-based cryptocurrency. The operator responsible for the creation of a block will receive a certain amount of cryptocurrency. This logic is built into the format of the ledger. The operators are offered a secondary incentive in the form of an option to request a fee from participants for the creation of block entries.

Mining is costly in terms of resource, making the cost of creating a single block very high. For example, Bitcoin has a built-in mechanism that regularly reduces the reward for block-creation. If there is a temporary dip in the exchange rate of the cryptocurrency, a situation may arise where mining is no longer economically feasible.

Proof of stake

Proof of stake is an alternative to proof of work that helps increase operators' economic incentives.

Proof of stake requires that operators have a sufficient amount of personal cryptocurrency in the system and therefore have enough economic incentive to maintain the system. The rules for creating a block and determining its difficulty are created in such a way that operators who have more cryptocurrency gain priority in block creation. For example, the right to create a block may be decided by lottery, where the likelihood of winning is proportional to the amount of cryptocurrency possessed.

4.8 Comparison of blockchain integrity mechanisms

The following is an analysis of the key features of essential integrity mechanisms of blockchain technology, including guarantees that these mechanisms provide against unauthorised modification of the ledger. Let us consider an abstract scenario where blockchain $(B_1, B_2, \ldots, B_{t'}, B_{t'+1}, \ldots, B_t)$ is replaced by blockchain $(B_1, B_2, \ldots, B_{t'}, B'_{t'+1}, \ldots, B'_t)$.

Hashed timestamping

In trusted timestamping solutions, the regular publication of hash values of blockchain blocks is carried out in a way that is independent of the management of the ledger itself, for example in the form of publication in a newspaper. If, for example, the hash of block B_t has been published at some point $T \geq t$ in time, then, after that, the modification of chain $(B_1, B_2, \ldots, B_t)$ is possible only if:

a) the published hash is modified, or

b) a successful collision attack is performed on the hash function used.

Proof of work

As the creation of every subsequent block requires a relatively high amount of computational resource C, the resource required for the creation of the new chain is $C \cdot (t - t')$. In the case of a sufficiently large difference $t - t'$, the calculation may be considered infeasible for the attacker.

The assumption of infeasibility is weaker than that of trusted timestamping, as block creation should not be overly complex for the combined operators (otherwise it would not be possible to create a blockchain at all), yet at the same time it should not be too simple for the attacker. Therefore, it must be assumed that the attacker's resources are significantly smaller compared to the combined resources of the operators.

Digital signature and proof of stake

If block integrity is protected with digital signatures, then the creation of a new blockchain is possible by:

a) influencing or controlling a significant portion of the operators, or

b) successfully counterfeiting the digital signature schemes used.

It is assumed that operators who own a sufficient amount of cryptocurrency associated with a blockchain lack economic incentive to undermine the integrity of the ledger. Therefore, the integrity mechanisms of blockchains that utilise proof of stake are largely based on social sciences.

4.9 Criteria for the necessity of blockchain

The following are criteria that help decide whether inter-user data exchange requires:

a) peer-to-peer **distributed data exchange** without a central database,

b) **centralised service**,

c) **permissioned ledger**, or

d) **permissionless ledger**.

General data exchange task

Users $U_1, U_2, \ldots, U_n$ send each other messages. Each user U_i has a personal ledger L_i, which serves as a basis for some important decisions. Decisions can be made by operators and users (both passive and active). For example, government agencies grant social benefits to citizens on the basis of digitally signed applications that have been received.

In terms of formal logic, $L = (L_1, L_2, \ldots, L_n)$, can be viewed as an aggregate ledger that reflects all information about data exchange between parties.

Questionnaire

A questionnaire-schema representing the requirements of the blockchain is presented in Figure 1 and comprises of the following questions:

1. *Can each user U_i make his/her own decision solely on the basis of the personal ledger L_i?* If so, then the system does not require the use of a centralised service and distributed data exchange is sufficient.

2. *Is a legal/physical person a sufficiently reliable central service provider?* The reliability criterion may depend on the context, the application, or the legal dispute resolution framework. It may also depend on a country's laws, and therefore, the applicable laws of the country in question. If the answer is yes, then a centralised service is required.

3. *Can the service be entrusted to a fixed group of legal/natural persons (e.g. several public authorities, banks, etc.)?* In the context of a country, this question essentially aims at whether the country as such can be trusted. If the answer is yes, then a permissioned ledger is required.

A permissionless ledger is required only to resolve tasks where neither a country nor any consortium of private individuals nor a consortium of public authorities or private parties are considered to be sufficiently reliable, or there is a request for protection against major powers that might affect any consortium.

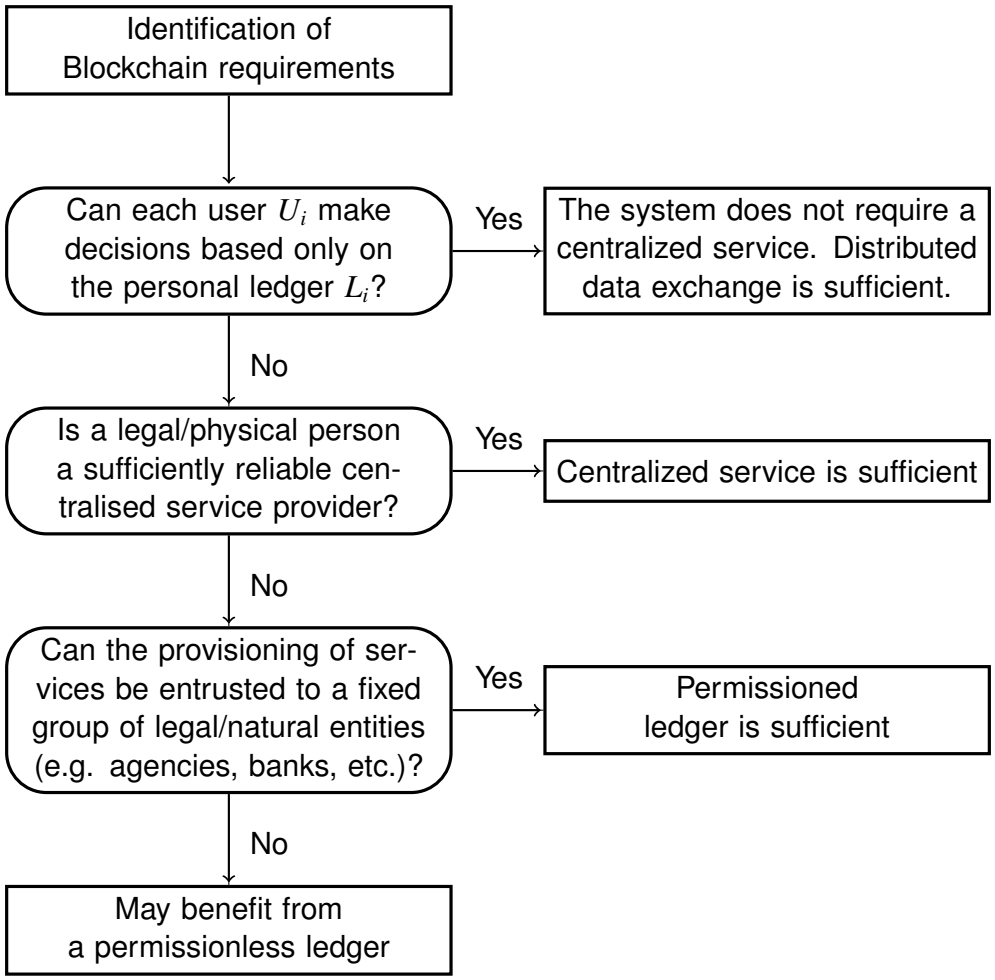

Figure 1. Scheme to decide necessity of the blockchain.

Some example systems and the analysis of their blockchain requirements

In this section, we take a look at some of the simplified systems and analyse them briefly in the context of whether they need blockchain technology. The systems under focus are just examples illustrating the short questionnaire.

Inter-institutional data exchange. Public authorities exchange data necessary for the performance of their duties under the provisions applicable.

Distributed data exchange is sufficient here, as each institution can make decisions based on an individual ledger.

Social benefits system. Users (e.g. residents of Estonia) send applications to institutions to receive aid with the condition that institution A can only accept an aid application if other institutions will not.

Here, centralised service can be of benefit, as decision making requires data from other institutions as well. It might also benefit from a permissioned ledger, as it would help alleviate the risk of corruption arising from trusting a single institution.

Electronic cash system with the central bank. The central bank issues digital coins to users who use them to buy products from merchants. It is important that the same coin is not used multiple times.

Here, it is sufficient if the central bank manages the centralised service, which, in addition to issuing coins, accounts for their use to avoid dual use.

Electronic currency system without the central bank. As in the previous system, but the central bank is not trusted, i.e., coins circulate freely between users. It must be guaranteed that the user U_i can pay user U_j using coin m, where m has, at some point t in time, been in the possession of user U_i, and meanwhile (i.e. from t to present), coin m belonging to user U_i, has not already been used for payment.

Checks on double spending depends on user data and is therefore a necessary part of the ledger. Whether a permissionless ledger is required depends on whether a coalition of banks can be trusted or not.

Table 3. Terms related to the blockchain and their Estonian equivalents

Term	**English equivalent**	**p.**
arvestusraamat	ledger	17, 19
bittmünt	bitcoin	17
hajusraamat	distributed ledger	17, 20
kaeve	mining	22
krüptoraha	cryptocurrency	17
lepingumonitor	smart contract	20, 20
leppeprotokoll	agreement protocol	21, 21
loaline hajusraamat	permissioned ledger	18, 21
loatu hajusraamat	permissionless ledger	18, 21
Nakamoto konsensus	Nakamoto consensus	21, 21
panusetõendus	proof of stake	23
plokiahel	block-chain	17
töötõendus	proof of work	22

Bibliography

[1] Digital Signature Standard (DSS). FIPS PUB 186-4. http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf.

[2] SEC 1: Elliptic Curve Cryptography, 2009. http://www.secg.org/sec1-v2.pdf.

[3] Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms). http://www.riso.ee/sites/default/files/elfinder/article_files/kryptoalgoritmide_elutsykli_uuring.pdf, 2011. Cybernetica report no. A-60-1 (in Estonian).

[4] Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms). https://www.ria.ee/public/PKI/kruptograafiliste_algoritmide_elutsukli_uuring_II.pdf, 2013. Cybernetica report no. A-77-5 (in Estonian).

[5] Krüptograafiliste algoritmide kasutusvaldkondade ja elutsükli uuring (Report on the life cycle of cryptographic algorithms). https://www.ria.ee/public/RIA/Kruptograafiliste_algoritmide_uuring_2015.pdf, 2015. Cybernetica report no. A-101-1 (in Estonian).

[6] CNSA Suite and Quantum Computing FAQ, January 2016. https://www.iad.gov/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm.

[7] Cryptographic algorithms lifecycle report 2016). https://www.ria.ee/public/RIA/Cryptographic_Algorithms_Lifecycle_Report_2016.pdf, 2016. Cybernetica report no. A-101-3.

[8] Kryptographische Verfahren: Empfehlungen und Schlüssellängen, 2017. BSI – Technische Richtlinie, https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/TechnischeRichtlinien/TR02102/BSI-TR-02102.pdf.

[9] Sertifikaadi, CRL-i ja OCSP profiil Eesti Vabariigi isikut tõendavatel dokumentidel). https://www.sk.ee/upload/files/SK-CPR-ESTEID-ET_v8_1_20171104.pdf, 2017. Versioon 8.1.

[10] SK ID Solutions AS – Certificate Policy for ID card). https://sk.ee/upload/files/SK-CP-ID%20CARD-EN-v7_0-20171101.pdf, 2017.

[11] ITU-T Study Group 17. LS on ITU-T SG17 work on quantum-safe PKI. https://www.ietf.org/lib/dt/documents/LIAISON/liaison-2017-09-13-itu-t-sg-17-ipsecme-lamps-ls-on-itu-t-sg17-work-on-quantum-pdf, 2017.

[12] Divesh Aggarwal, Gavin K Brennen, Troy Lee, Miklos Santha, and Marco Tomamichel. Quantum attacks on Bitcoin, and how to protect against them, 2017. arXiv preprint arXiv:1710.10377, https://arxiv.org/pdf/1710.10377.pdf.

[13] Steve Babbage, Dario Catalano, Carlos Cid, Benne de Weger, Orr Dunkelman, Christian Gehrmann, Louis Granboulan, Tim Güneysu, Jens Hermans, Tanja Lange, Arjen Lenstra, Chris Mitchell, Mats Näslund, Phong Nguyen, Christof Paar, Kenny Paterson, Jan Pelzl, Thomas Pornin, Bart Preneel, Christian Rechberger, Vincent Rijmen, Matt Robshaw, Andy Rupp, Martin Schläffer, Nigel Smart, Serge Vaudenay, Fré Vercauteren, and Michael Ward. ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012). Technical report, European Network of Excellence in Cryptology II, September 2012. http://www.ecrypt.eu.org/ecrypt2/documents/D.SPA.20.pdf.

[14] Elaine Barker. Recommendation for Key Management. Part 1: General, 2016. NIST Special Publication 800-57 Part 1, Revision 4, http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf.

[15] Dave Bayer, Stuart Haber, and W. Scott Stornetta. *Improving the Efficiency and Reliability of Digital Time-Stamping*, pages 329–334. Springer New York, New York, NY, 1993.

[16] Sergio Boixo, Sergei V Isakov, Vadim N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices, 2016. arXiv preprint arXiv:1608.00263, https://arxiv.org/abs/1608.00263.

[17] David Chaum. *Blind Signatures for Untraceable Payments*, pages 199–203. Springer US, Boston, MA, 1983.

[18] Don Coppersmith. Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known. In *Advances in Cryptology – EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 178–189. Springer, 1996.

[19] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's Algorithm to AES: Quantum Resource Estimates. In *Post-Quantum Cryptography*, volume 9606 of *LNCS*, pages 29–43. Springer, 2016.

[20] Ralph Charles Merkle. *Secrecy, Authentication, and Public Key Systems.* PhD thesis, Stanford, CA, USA, 1979. AAI8001972.

[21] Matus Nemec, Marek Sys, Petr Svenda, Dusan Klinec, and Vashek Matyas. The Return of Coppersmith's Attack: Practical Factorization of Widely Used RSA Moduli. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pages 1631–1648. ACM, 2017.

[22] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, April 1980.

[23] John Proos and Christof Zalka. Shor's discrete logarithm quantum algorithm for elliptic curves, 2003. arXiv preprint quant-ph/0301141, https://arxiv.org/abs/quant-ph/0301141.

[24] Michael Rabin. *Digitalized signatures and public-key functions as intractable as factorization*. MIT Laboratory for Computer Science, 1979.

[25] Nigel P. Smart, Vincent Rijmen, Benedikt Gierlichs, Kenneth G. Paterson, Martijn Stam, Bogdan Warinschi, and Gaven Watson. Algorithms, key size and parameters report – 2014. Technical report, ENISA, 2014. http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014/.

[26] Marc Stevens, Elie Bursztein, Pierre Karpman, Ange Albertini, and Yarik Markov. The first collision for full SHA-1, 2017. https://shattered.io/static/shattered.pdf.

[27] Gideon Yuval. How to Swindle Rabin. *Cryptologia*, 3(3):187–191, 1979.