Procurer

REPUBLIC OF ESTONIA
INFORMATION SYSTEM AUTHORITY

CYBERNETICA

Cryptographic Algorithms Lifecycle

REPORT

2016

Cryptographic algorithms lifecycle report 2016

Research report

Version 3.0

June 22, 2016

81 pages

Doc. A-101-3

Project leaders:	Kaur Kullman (Estonian Information System Authority)
	Mari Seeba (Cybernetica)
Contributing authors:	Ahto Buldas, PhD (Cybernetica)
	Kristo Heero, PhD (Cybernetica)
	Peeter Laud, PhD (Cybernetica)
	Riivo Talviste, PhD (Cybernetica)
	Jan Willemson, PhD (Cybernetica)

Estonian Information System Authority, Pärnu maantee 139a, 15169 Tallinn, Estonia.
Email: 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.

Foreword

Estonia has a reputation for being a capable and mature IT country. This is enabled by the very high standard of natural sciences education Estonians had during the Soviet era and, on the other hand, by our geographical and cultural position. Afterall, Estonia is a country on the frontiers of the Western and the Eastern civilisations. This enables us to understand the differences and similarities, threats and opportunities of both cultural hemispheres for our unique set of intercultural wisdom in understanding Eurasia, and good education level enables us to innovate.

Due to Estonia's tiny population (1,3 M[1]) and moderate GDP (€22,5 B[2]), we are not able to offer neither fundamental inventions nor create principally new technologies for the international community. However, Estonia has been extremely successful in creating technological solutions that enable modern e-society, e-governance and digital lifestyle. Among those are

- nationwide mandatory public key infrastructure (PKI) that enables citizens, institutions and companies to digitally sign documents,
- X-Road connectivity system that serves as a primary trust anchor for the messaging system our public and private shareholders' IT systems use for communication, and, last but not least,
- a unique and cryptographically strong Internet voting system that we have used 8 times already for local and national elections, with up to one third of ballots cast electronically over the Internet.

However, as an advanced digital society with a dependency model comparable to the traditional Western approach, Estonia with its IT-dependent systems is inherently vulnerable to various threats. These include malicious attacks, administrative mistakes and political turmoil, but also general human stupidity.

As many of our government and private sector services have been optimised for the e-society, most of such institutions (e.g. banks and postal service providers) no longer have enough offices to serve citizens in the physical world alone. These institutions usually only have a few offices in larger towns.

Because of our tiny economy, it is not economically feasible to serve clients in the traditional way, face to face – it would be too expensive for the clients, i.e. the Estonian taxpayers. In the contemporary world, it means that Estonia has no plan B, would the risks on our

[1]`https://www.stat.ee/277564`
[2]`https://www.stat.ee/277536`

e-society materialise. If we fail to keep our critically important IT infrastructure running, our e-lifestyle will collapse, bringing down the whole Estonian society as well.

The Dutch DigiNotar case in 2011 demonstrated the hard choices a country faces if a PKI supporting its government's IT systems is compromised. To sustain the society, the Dutch government had to rely on potentially compromised systems for the contingency of its operations. The DigiNotar case served as a wake up call for the Estonian politicians who finally understood the need for our country to plan ahead against similar threats. Estonia needs to be able to continue its operations in case the trust in Estonia's PKI fails, some cryptographic primitive is suddenly broken or the trust in the cryptographic primitives used for our digital signatures is undermined, rendering the systems supporting our e-society unreliable and/or untrustworthy.

Therefore, it was decided in 2011 to assemble a scientific task force to analyse the problems and risks that reliance on cryptography is posing on the sustainable functioning of our society. One has to take into account that while Estonia has a well-established cryptographic school of thought, the country most likely is not (yet) able to work out and standardise its own cryptographic primitives. Thus, we rely on some known cryptographic algorithms standardised by some other countries.

Establishment of this scientific task force was delegated to Estonian Information System Authority[3] (*Riigi Infosüsteemi Amet* or RIA in Estonian). Among other duties, RIA coordinates the development and administration of the state's information systems, organises activities related to information security and handles the security incidents that have occurred in Estonian computer networks. RIA also advises the public services and critical infrastructure providers on how to manage their information systems. For that, RIA participates in development of the corresponding requirements, including legal framework and the baseline protection system ISKE (based on BSI's *IT-Grundschutz*).

Therefore, maintaining the state of cryptography that our country depends on was a natural addition to the portfolio of responsibilities of RIA. After the first report of this task force, it was clear that we had to keep on updating the reports, as this field keeps on developing.

The report you, dear reader, are reading is the fourth in this series. It is an update, not a rewrite of previous reports. Nonetheless, we in RIA understand that the problems we have discovered on our path are generic and valuable for the international audience, especially for the countries that are following Estonia's example on the quest for a secure and sustainable digital society.

Anto Veldre
Estonian Information System Authority

[3] https://www.ria.ee/en/about-estonian-information-system-authority.html

Contents

1 Introduction

This study is a natural continuation of three previous studies conducted in 2011, 2013 and 2015 [9, 12, 18]. Whereas the two first iterations aimed at giving a self-contained overview of the status all major cryptographic primitives and protocols in practical use, the report published in 2015 only contained updates on some specific topics. The current version follows this approach and only covers certain topics that are of interest from the viewpoint of information systems used in Estonia in 2016.

However, both the procurer (Estonian Information System Authority) and the authors (Cybernetica's research team) feel that many of the topics are actually quite universal and hence subject of a wider international interest. This is the reason why the 2016 report is the first one in its sequence to be written in English. Even though we do not aim to be self-contained and may occasionally make back references to the older Estonian versions, most of the study should be comprehensible without being fluent in Estonian language.

The report is covering both the state of cryptographic primitives and their application in higher-level protocols. In general, cryptographic primitives are rather well understood and studied objects. Compared to the frequency of protocol-level vulnerabilities, primitives are broken rather rarely. However, since many of the primitives are implemented in very low-level layers of communication infrastructure (often even in hardware), such a break may have far-reaching consequences.

Luckily, cryptographic primitives are typically not broken overnight. First, some structural weaknesses are discovered and it takes time to develop exploits based on those findings. Even then, a considerable amount of computing power may be needed to exploit some weakness, which may initially not be affordable to all interested parties.

This is why developments in cryptanalysis need to be constantly monitored, and the current report is one step in the global effort. We will not do a full round through all the major primitives, but rather present updates to the previous reports [9, 12, 18]. A very thorough treatment of the matter can also be found in ECRYPT II report [39].

In Estonia, the previous reports have already made tangible impact, with the last example being our national Certification Authority (CA) AS Sertifitseerimiskeskus stopping the use of SHA-1 and SHA-224 in time-stamping requests in the beginning of 2016 [30]. We hope that this report will contribute in improving the security of our digital environment worldwide.

This report covers the following topics. Chapter 2 describes the most prominent weaknesses found in 2015 in the TLS/SSL standards and some of its implementations. Chapter 3 reviews the state of one of the most widely used primitives, the RSA public key cryptosystem. We study the difficulty and monetary investment required to break RSA using various proposed methods, including quantum computers and specialised hardware.

The 2015 version of the report [18] contained a thorough introduction to elliptic curve cryptography. This time we will only present a short discussion concerning the controversy created by the August 2015 NSA announcement (in Chapter 4) and an updated version of the table summarising implementations of various curves in different libraries (in Appendix B).

Another class of primitives that have got a lot of attention in recent years are hash functions. Chapter 5 reviews the best currently known attacks against existing constructions (including SHA-1) and Chapter 6 describes the new SHA-3 standard that was finally accepted in 2015 after years of development.

The rest of the report deals with various application-level protocols and solutions. We start from an overview of secure computation outsourcing mechanisms (see Chapter 7), being motivated by the general drive towards cloud computing.

Chapter 8 describes the major candidates for asymmetric primitives in the post-quantum era. Chapter 9 studies radio frequency authentication tokens and Chapter 10 presents Mobile-ID protocol analysis. Finally, Chapter 11 draws some conclusions and gives recommendations to improve security of the systems to be deployed.

Acknowledgements

This report builds upon efforts of many people. Over the years, more than 20 researchers and developers from Cybernetica have been involved by either directly contributing various chapters or giving their feedback after proof-reading. Still, most of the results described in this report are not created by Cybernetica's team, but originate from many groups and many countries. Our thanks goes to all the researchers who have made creating this report possible.

To improve quality of the material to be presented, we have conducted several interviews with specialists in fields other than cryptography. We would like to thank Hardmeier, G4S and Johannes Heinsoo for their time and contribution to this report.

And last but not least – Estonian Information System Authority deserves the credit for initiating the whole process back in 2011, by giving the authors constant input and feedback, and helping to carry the results into practice. Thank you!

2 TLS attacks in 2015

Transport Layer Security (TLS) (latest version 1.2 [84], but ver. 1.3 is forthcoming [151]) and its predecessor, the *Secure Sockets Layer (SSL)* are presently perhaps the most widely used protocols for authentication and key agreement over the Internet between two principals, as well as for securing their traffic afterwards. For the purposes of security analysis, TLS consists of the *handshake protocol* for entity authentication and key agreement, and the *record protocol* for maintaining a secure channel. The two protocols are intertwined — while the record protocol uses the keys derived in the handshake protocol, the handshake protocol itself runs inside the channel provided by the record protocol.

In the handshake protocol, the client and the server first send each other the messages specifying which protocol versions and *cipher suites* they support, followed by the messages containing their certificates and the values from which the *master secret* for keys used in the record protocol can be derived. The parties will then switch to the agreed-upon keys and ciphers (by exchanging `ChangeCipherSpec`-messages) in the record protocol. Finally, they send to each other the authenticated hash of all messages exchanged in the handshake protocol.

A cipher suite specifies the public-key primitives used in the handshake protocol, as well as the symmetric cipher and message authentication code used in the record protocol. Depending on the used public-key primitives, the handshake protocol may have a slightly varying structure. Available cipher suites number in hundreds[4].

The TLS protocol specifies an optimisation for faster start-up of the secure channel. It is possible to save the state of a previous connection between a particular client and server, and resume its usage afterwards. Such resumption requires less public-key operations to be performed than a full handshake protocol. Alternatively, it is possible to keep the secure channel up if it is known that it will be used again in the future.

As the TLS protocol has many different versions and options, it is unlikely that all of them simultaneously can be accounted for in a security analysis of that protocol. In previous versions of this report [12], we have attempted to give an overview of existing security analyses of TLS, we will not repeat it here. In this report, we will give an overview of attacks against the TLS/SSL protocol and its implementations, and discuss possible remedies.

2.1 Nature of attacks

The attacks against TLS have a variety of causes, genuine errors in protocol logic, leading to keys becoming known to the adversary or to a mismatch between the views of the client

[4]`https://www.iana.org/assignments/tls-parameters/tls-parameters.xhtml`

and the server, being just one of them. Such errors were more prevalent in older versions of TLS/SSL, particularly in the first versions of SSL, the support of which may still be lingering around in some applications. The *downgrade* attacks that first try to make the client and server agree on an old version of the protocol, are another significant attack class.

The legacy manifests in the TLS protocol set also in some other ways. Up to the year 2000, the export of cryptographic software and hardware was restricted by the United States, with exceptions granted only to products that used encryption with at most 40-bit key strength. This lead to the introduction of cipher suites with short keys. The lifecycle of such security-crippled cryptographic algorithms was not thoroughly considered and nowadays, these cipher suites may still be supported by some servers. By interfering with the handshake protocol, it may be possible to convince a client and a server to agree on such a cipher suite, leading to serious attacks.

Somewhat later, evidence against the security of certain symmetric primitives started mounting, these primitives include the DES block cipher, the RC4 stream cipher, and the hash functions MD5 and SHA-1. Support for cipher suites containing these primitives may still be available. Again, certain attacks against TLS attempt to trick the clients and servers to use such weak ciphers.

A networked application together with an implementation of secure channels consists of several different layers, with the implementations of cryptographic primitives being on the lowest layer. On top of it, there is the layer of cryptographic constructions that turn a block cipher into a symmetric encryption scheme or a trapdoor one-way permutation into an asymmetric encryption scheme. The next higher layer is the protocol layer that determines which messages are sent and when, how they are protected using cryptography and which checks are made. This layer presents the secure channel abstraction to the application above it. Some potential weaknesses of this layer have been mentioned in previous paragraphs. Other layers have also had their share of issues. It has long been known that the cryptographic constructions in TLS are somewhat different from the provably secure constructions [47], attacks based on these differences have appeared in recent years. At the other side of the protocol layer, the details of the secure channel abstraction may be non-trivial and lead to insecure usage by the application. In particular, there exist attacks based on the non-hiding of the *length* of messages inserted into the channel.

The implementation of all layers is also a non-trivial matter. There have been different kinds of implementation errors which have lead to attacks. Basic programming errors, e.g. buffer overflows are perhaps the simplest kind, but they have provided the basis of some of the most well-known attacks. Another basis of attacks is the leakage of the internal protocol state due to sloppy programming; an attacker can use this to deduce the secrets used in the record protocol. Such leaks are mostly due to timing; as the logic of all layers is complex, it is surprisingly difficult to close all timing channels or at least sufficiently reduce their width. Yet another set of implementation errors is the absence of certain checks demanded by the protocol logic. In cryptographic protocols, it may be necessary to stop the execution if the format of the received message does not precisely match the specification, even though the message itself may still be understandable. The confusion on the necessity of certain checks is sometimes exacerbated by the lack of clarity in documents specifying the protocol.

In the following, we present known attacks against TLS/SSL, grouped by the targeted weaknesses. Some attacks make use of several weaknesses, we have grouped them under the "main" weakness which seems to enable them.

2.2 Attacks against weaknesses in protocol logic

During the lifetime of the secure channel provided by TLS, maintained by running the record protocol, a *renegotiation* may be triggered by the parties. In this case, the handshake protocol is run again, and the keys of the channel are updated. The new handshake protocol may use different parameters than the original one; this is e.g. used to request the client to identify itself more strongly if it wants to access a more sensitive resource.

The *renegotiation attack* [153], discovered in 2009, allowed also the identity of the client to change during renegotation. In this attack, the attacker holds up the client's initial connection to the server, establishes a secure channel with the server itself, sends the server some data and triggers a renegotiation. The attacker releases client's connection, and lets the renegotiation proceed between the client and the server, acting as a man in the middle. At the end of the renegotiation, the attacker can no longer read or inject messages on the secure channel. Nevertheless, the messages that the attacker previously sent to the server are now associated with the client and the server thinks they came from the client. The renegotiation attack was thwarted by an extension to TLS [152], which is supported in OpenSSL since version 0.9.8m.

Many things are allowed to change during a subsequent handshake, and these changes together with identifying TLS sessions during resumption have enabled the *triple handshake attack* [52]. In this attack, the client establishes a session with the attacker, while the attacker establishes a different session with the server. In both servers, the client won't be identified at first. The attacker chooses certain parameters of sessions so, that they can be mixed up after interrupting and resuming them. During resumption, the attacker connects the client and the server to each other and causes the client to be identified. At this point, the client still thinks it is talking to the attacker, but the server thinks it is talking to the client. More importantly, anything that the attacker sent to the server before interrupting and resuming the connections, the server also attributes to the client. After one more handshake, the client obtains a changed certificate from the server, but this is allowed by TLS specification. At the time of publication, the attack was applicable against web browsers. It was mitigated by changing how the interrupted TLS sessions were identified [51], removing the possibility to mix up different sessions.

Often, a protocol error stems from the fact that cryptographically protected messages contain too little information about their meaning, allowing the attacker to create confusion in the recipient of such messages. In [131], an attack is described where the values sent by the server to the client for computing the master secret, and intended to be interpreted as the description of a certain elliptic curve and two points on it (for doing Diffie-Hellman key exchange over an elliptic curve), are modified by the attacker, such that the client thinks they are parameters for ordinary Diffie-Hellman key exchange. If the attacker can compute the necessary discrete logarithms (which is not so unlikely, due to the apparent randomness of the parameters, including also the description of the group), then it can itself complete the handshake with the client and set up a session where the client still thinks it is connected to the server. The attack is probably not very significant, because the probability that the necessary discrete logarithms can be computed, is still low (to the order of 2^{-40}).

If the attacker learns a client's secret key, then it can impersonate this client to any server. Can it also impersonate servers to this client? In [54], the notion of Key Compromise Impersonation (KCI) attacks was introduced. It turns out that TLS is not secure against KCI attacks [110], i.e. servers can be impersonated to a compromised client. This can become an issue if the installation of client's certificates and secret keys is handled sloppily, e.g. if the source of the secret key and the certificate may be malicious. The attack requires both the client and the server to have fixed (EC)DH key pairs.

2.3 Attacks against insecure cryptographic constructions

Among the various cipher suites defined for TLS, there are those based on RSA encryption. The "textbook RSA" is semantically insecure, hence it is always used by encapsulating the plaintext into a format specified in the standard PKCS #1 [1]. This standard has several versions, and the format specified in PKCS #1 v2.0 [2] has provable security guarantees.

However, TLS makes use of the padding specified in v1.5 of this standard, and this version of the standard has ubiquitous support in various hardware and software applications. This is unfortunate, because the famous Bleichenbacher's attack [56] allows a decryption server supporting PKCS #1 v1.5 to be used as a decryption oracle, by making a large number of queries to it with messages related to the message that the attacker wants to decrypt.

There exist countermeasures against this attack [82], but these can in turn be exploited [117], such that we still have a decryption oracle. In 2012, the complexity of the original attack was reduced by a couple orders of magnitude [42]. Still, the number messages is high and requires the connection to be kept alive for many hours after receiving the challenge from the server in the `ServerHello`-message, but without completing a handshake. The TLS protocol itself does not contain the means to enforce a timely response to the challenge. The implementations of TLS or higher-level protocols using it may contain mechanisms to time out the connection, but these mechanisms have been designed in *ad-hoc* manner and may be easily circumventable [145]. E.g. for the implementation of the HTTPS protocol in the Apache web server, it is actually possible to keep the connection alive for many hours, such that the Bleichenbacher's attack can be completed [145]. Therefore we have given a suggestion [18] to not use RSA encryption based cipher suites in applications where the several hours long time for performing a TLS handshake might pass unnoticed. In interactive applications (e.g. serving the web to browsers), this should not be an issue.

A secure block cipher can be turned to a secure symmetric encryption scheme and to a secure channel in a variety of provably secure ways [121]. Unfortunately, the IV-chained CBC mode of block ciphers specified in a number of cipher suites is not one of these ways and theoretical weaknesses against it have long been known [40, 41]. The weaknesses have enabled the BEAST attack [88], if the attacker can control the network traffic between the client and the server and can also have some control over the messages that the client sends (e.g. by loading malicious JavaScript to the client's browser). The BEAST attack resulted in dropping the support for block ciphers in IV-chained CBC mode since TLS version 1.1 [83].

In SSL v3.0, the contents of the padding in the protocol records is unspecified, and the padding itself is not authenticated. This is a weakness that is exploited in the POODLE attack [136]. Under the same attacker model as for BEAST, the attacker can check the equality of certain plaintext bytes in the messages the client sends out, eventually leading to the leakage of client's secrets. There is no good mitigation of this attack besides not using SSL v3.0.

If both the client and the server support SSL v3.0 (due to the need to talk to some parties that do not support any higher versions of the protocol), then even if they both also support recent versions of TLS, the attacker may interfere with the initial messages of the handshake protocol, causing them to fall back to SSL v3.0. There is an extension to TLS [135] that mitigates such downgrade attacks, but the proper way is still to drop the support for SSL v3.0.

In all versions of TLS, the contents of the padding have been fully specified, rendering the POODLE attack inoperative. However, some implementations of TLS have turned out to not strictly check whether the padding of incoming messages corresponds to TLS standard [123]. These implementations are thus vulnerable to POODLE.

2.4 Attacks against short keys

Certain cipher suites of TLS make use of short, so-called *export strength* keys, in particular 512-bit RSA or Diffie-Hellman keys. These suites are deprecated since the release of TLS v1.1 [83], but support is still present in deployed clients and servers. They are a critical component of the FREAK attack [50], where bugs in client code make it accept a 512-bit RSA key from the server that still supports it and thinks the client was requesting an export strength cipher suite due to the adversary tampering with the first messages of the handshake protocol. We describe these bugs in more detail below, as they are really an implementation error.

Factoring a 512-bit RSA modulus takes hours [50], but as the lifetime of an RSA key is longer than that (due to the search for 256-bit primes during the generation of the key being a relatively difficult operation), the attacker can find the factors ahead of mounting the FREAK attack. Computing discrete logarithms modulo a 512-bit prime is somewhat more complex. Furthermore, in a Diffie-Hellman key exchange (with ephemeral keys, as is the usual deployment), new elements of the group are generated in each protocol session.

The Logjam attack [32] is a practical attack against export-strength Diffie-Hellman key exchanges, where the discrete logarithm of a fresh group element has to be found. The input parameters to the discrete logarithm problem are the size of the group p (a 512-bit prime number), the base element g in this group, and the element h to be logarithmed. The output is a number y, such that $g^y = h \pmod{p}$. The attack depends on the following two facts:

- The search of a 512-bit safe prime number is a difficult operation. Hence the servers use the same p for a long time. Even more, there are certain standard primes used by many servers [143].
- The state-of-the-art algorithms for discrete logarithms only use p (and perhaps g, which is fixed together with p) for the large majority of their computations. The value h is only needed at the very end.

Hence, knowing the 512-bit prime the server is going to use, we can do most of the discrete logarithm computation ahead of time, such that after learning the group element in the key exchange session, we can find its discrete logarithm sufficiently fast. The authors of Logjam report that their precomputations for a fixed p took them slightly over a week (and consumed over 10 core-years of computing power). Afterwards, the computation of a discrete logarithm in this group requires on average 10 core-minutes, taking around 70 seconds to compute. They state that their computations are not optimized.

If the server supports export strength Diffie-Hellman key exchange, then forcing the connection down to such cipher suite is again just a matter of tampering with the first handshake messages. After the server has sent the 512-bit group element in authenticated manner, the attacker can take over and complete the handshake with the client. At the end of the handshake protocol, the message authentication code has to be applied to the sequence of messages exchanged during the handshake protocol, but the attacker can do it because it knows the master secret at that time.

The authors of the Logjam attack also speculate about the complexity of computing discrete logarithms in larger groups. They estimate that the precomputation for a single 1024-bit prime requires around 45 million core-years. The entire computation is parallelizable, but a large fraction of it is not embarassingly parallelizable. They try to estimate the cost of the hardware that is capable of performing this precomputation in one year (for a single 1024-bit prime). They guess that the cost is in hundreds of millions of dollars (within an order of magnitude). This is definitely in reach of large countries. After the precomputation, the computation of a single discrete logarithm requires roughly one core-month, and this computation is easily parallelizable.

To mitigate the attack, one has to use larger keys or unique 1024-bit groups. In the latter case, the server has to generate the modulus itself, making sure that it does not have any weaknesses that may simplify the computation of discrete logarithms in the group it defines. This modulus will be used in the handshake protocol sessions the server participates in. The generation of a safe modulus takes significant time (tens of seconds). Another possible mitigation is switching to elliptic curves, as the current discrete logarithm algorithms for elliptic curves do not gain as much from precomputation [95].

DROWN [38] is a different attack that relies both on the support of export strength cipher suites and old versions of TLS/SSL. In this attack scenario, the attacker has captured a TLS session between a client and a server, where the master secret has been generated from the pre-master secret that the client sent to the server encrypted with server's RSA public key. In order to read the session, the attacker needs to decrypt that pre-master secret.

The attack needs the server (or a different one, but it needs to use the same RSA key) to support SSL v2.0 and export strength cipher suites. The attacker connects to the server and chooses SSL v2.0 for the handshake and a cipher suite with export strength (40-bit) symmetric key. The attacker passes modified captured encrypted pre-master secret as its own encrypted pre-master secret. The server's response indicates if this passes as a valid pre-master secret for the chosen cipher suite. To understand server's response, the attacker has to brute-force a 40-bit secret.

For 2048-bit RSA keys, the success probability of the step described in the previous paragraph is such, that the attacker should have around 1000 captured TLS handshakes and make around 20000 connections to the server (incurring the computational cost of around 2^{50} in the process). After finding a valid SSL v2.0 encrypted pre-master secret, the attacker can use techniques similar to Bleichenbacher's attack to decrypt it in around 20000 further connections to the server. The obtained plaintext can be transformed back to the pre-master secret of one of captured TLS handshakes.

Up to version 1.0.2, OpenSSL contained a bug that allowed a much more efficient attack of the same kind, without the support for export strength cipher suites. It allows the recovery of one in 260 TLS session keys using around 17000 connections to the server and negligible computational resources. The number of targeted keys can be further lowered to 100, with the expense of extra 10000 connections to the server.

The obvious mitigation to the DROWN attack is to make sure that SSL v2.0 or export strength cipher suites are not supported.

2.5 Attacks against weak cryptographic primitives

During the lifetime of TLS/SSL, the attacks against several primitives in its cipher suites have significantly improved. Attacks become possible, if the support for such cipher suites linger around for too long.

The RC4 stream cipher has been supported in TLS for a long time. Its weaknesses have also been known for a quite long time: in 2001, the keystream produced by RC4 was shown to be strongly non-random [91], this weakness was used to break *Wired Equivalent Privacy (WEP)*. However, the attack was not applicable to TLS/SSL and in the meantime, the BEAST attack described above increased the popularity of RC4, as this attack applied only to block ciphers.

Over time, more correlations and biases in RC4 keystreams have been discovered, especially in the first ≈ 100 bytes of it [34, 168]. In TLS sessions, if the client is sending any passwords or secret cookies to the server, then these secrets typically occur at the beginning of the session and are encrypted with the bytes of the keystream that are near its beginning. Current knowledge about the biases in RC4 keystream allows the guessing of an RC4-encrypted password in around 2^{26} tries on average, using Bayesian analysis [99]. In February 2015, the cipher suites containing RC4 were prohibited from use in TLS [149].

TLS also makes use of hash functions, both in the record protocol (for computing the message authentication codes) and in the handshake protocol (when computing the signatures). Often, MD5 or SHA-1 are used for these purposes, with the justification that these hash functions have been shown to be weak only for the purposes of collision-resistance, but a weaker property is needed in the TLS protocol. Recently [53], the SLOTH attacks showed that collision resistance is indeed the property that is needed. These *Transcript Collision Attacks* are a form of man-in-the-middle attacks, where the attacker chooses random values in two TLS handshake protocol sessions (acting as server in one, and as client in another session) in a way that makes the hashes of protocol transcripts equal. In the end, the client and the server have seemingly authenticated each other, but actually each of them has shared a key with the attacker. An obvious mitigation to these attacks is to stop using signatures using the MD5 hash function. In the context of the SLOTH attack, SHA-1 is still relatively safe to use, because the sort of collisions needed (chosen-prefix collisions, see Sec. 5.2) are not yet so easy to find for it.

2.6 Attacks against misunderstandings in abstractions

Intuitively, a secure channel can be thought of as an opaque pipe: a message goes in from one end and exits at the other, with the outsiders having no means to observe it during the transit. The actual secure channels provided by cryptographic protocols are a bit more transparent — they leak the lengths of communicated messages. If the message has been compressed before sending, then its length depends on its compressibility, i.e. on its content [114]. If the attacker knows a part of the message, then the length of the compressed message tells him something about the similarity of the other parts of the message to the known part.

The actual attacks using this channel appeared in 2012 and 2013 [89, 45, 106]. They noticed that if the attacker can change a part of the message then the changes in the compressed length tell him whether the changed part became more or less similar to other parts. In the attacks, the attacker has some control over the client's web browser. He makes several slightly different queries towards the server, to which the web browser also appends the secret cookie. Observing the lengths of compressed and encrypted queries, the attacker learns the secret.

The attack can be mitigated if the messages, the length of which the attacker observes, are sent uncompressed. However, the compression in the HTTPS protocol is supported at several levels. TLS record protocol supports compression; switching it off won't much affect the performance, because compression is also supported in HTTP. One may also turn off the HTTP-level compression in queries made by the client, but turning it off in server responses would seriously hamper performance. It is possible to execute the attack by only observing the lengths of responses, if the server reflects back the contents of the queries in the responses, including the secret cookie and the part of the query affected by the attacker.

The mitigations against compression-based attacks are largely application specific. One may try to fuzz the length of HTTP responses, or to disable the compression of responses to cross-site queries [155].

When a client (web browser) makes a query towards an HTTPS-enabled web server, we expect the created secure channel to be between this client and the server serving this HTTPS domain. If the server uses virtual hosting, serving many different domains, then the channels towards different domains may share certificates and/or state. As recently shown [79], in this case a page from one domain may be loaded under the other's origin in the client browser. The countermeasures (proposed in the same paper) include tightening the configurations of web servers, but there is no single simple countermeasure that works equally well across all deployment scenarios.

2.7 Attacks against implementation errors

A few cryptographic libraries implement TLS. Errors in these implementations have been found from time to time, and these can also lead to high-profile attacks. The mitigation against such attacks is to upgrade to a version where the error has been corrected.

Attacks against buffer overflows

Perhaps the most famous attack against TLS, *Heartbleed* [14] resulted from an error in OpenSSL in implementing the heartbeat extension to TLS. The error only existed in OpenSSL versions between 1.0.1 and 1.0.1f. It was not present in other branches of OpenSSL. The description of the error was published in April 7th, 2014, and patched in OpenSSL version 1.0.1g released on the same day.

The TLS heartbeat extension is intended to keep a TLS channel up at times where there is no traffic to send. The party interested in maintaining the channel sends a message to the other party and expects the same message back. The format of the message also includes the length of the byte string to be repeated. A malicious party may construct a message, where the alleged length is larger than the actual length of the byte string that the other party has to send back. The buggy versions of OpenSSL did not verify that the stated length was not excessively large. Instead, they sent back a message with length equal to the alleged length of received message. Such response message contained both the received message, as well as the contents of the memory immediately following it. This area of memory may have contained values that the other party should not have learned, e.g. parts of the private key. A good explanation to the Heartbleed bug is given in [140].

Another buffer overflow error, discovered and patched in November 11th, 2014 [15], was the *Winshock* error [174] in the *Schannel* library used by Microsoft's web browsers and servers. This error was due to using one method for finding the length of the buffer to be created, using another method to find the length of the data copied to this buffer, and not verifying that the second value was not larger than the first [74]. The error could appear when a certificate was decoded from its transport format, and was henceforth triggerable by a malicious communication partner.

Timing attacks

The timing attacks against TLS are related to the possibility of using one of the parties in a TLS session as a padding oracle. If a block cipher is used in the TLS record protocol, then a message authentication code is first added to the outgoing message, it is then padded to a multiple of block length, and the result is encrypted using CBC mode. A padding oracle is created if the recipient of a message, crafted by the attacker, publishes why the decryption and decoding failed — did the padding have the wrong format, or was the message authentication code found to be wrong. A padding oracle allows the attacker to decrypt messages from the record protocol without knowing the session key [169]. The applicability of padding oracles in web hacking was demonstrated in [156].

A typical way for creating a padding oracle is through timing channels: discovering wrong padding takes less time than discovering a bad message authentication code. Since version 0.9.6i and 0.9.7a, OpenSSL contains protections against padding oracle attacks. Unfortunately, modern processor architectures are complex and it is difficult to ensure that the processing time of a packet is independent of the contents of this packet. The Lucky Thirteen attack [35] showed that even a noisy oracle is useful for decryption. Since versions 1.0.1d, 1.0.0k and 0.9.8y, OpenSSL has contained code aimed at thwarting such attacks.

A related class of timing attacks are the *cache attacks*, where the attacker code runs on the same processor as the victim code, but in a different process. The goal of the attacker is to determine a secret used by the victim code. The attacker and victim processes may interact to a certain extent, because they share the processor cache. The attacker tries to bring the cache into a state where the timing of victim's next steps depends on the value of the secret. The attacker will then try to measure the time it takes for the victim to execute.

A recent example of such attack is the Cachebleed attack [178]. The authors introduced cache-bank conflicts and measured the timing variations caused by them. In around 16000 executions, they managed to retrieve the private exponent from the implementation of modular exponentation in OpenSSL 1.0.2f which already contains state-of-the-art protection against timing attacks. The authors also proposed a mitigation to the attack — disable hyperthreading, or only allow it between processes within the same protection domain.

Attacks against missing checks

When implementing complex protocol logic, which is perhaps also presented in somewhat confusing manner, the implementors may overlook the need for some security checks. There exists a variety of attacks aimed against such oversights. We have already mentioned the POODLE attack which may work on TLS implementations due to them not checking that the padding of messages has correct value.

An interesting shortcoming was presented in [50], where the complexity of state machines for supporting TLS handshake was stressed. Depending on the chosen cipher suite and on other options of the protocol, different messages are expected to be sent and received by the parties. For key exchange based on ephemeral Diffie-Hellman, as well as on export strength RSA, the server is expected to send a `ServerKeyExchange` message, this message should not be sent or accepted for other methods of key exchange. Also, the messages are expected to come in a certain order, skipping them should not be allowed. It turned out that in OpenSSL, messages could be sent out of order. In particular, even if strong RSA-based key exchange was selected, the client still accepted the `ServerKeyExchange`-message, using the key in this message instead of the one in server's certificate. This bug lead to the FREAK attack, explained above. Another fault, discovered earlier [52], consisted in both the client and the server accepting a `ChangeCipherSpec` too early, before a new key was successfully agreed upon. In this case, the actual key material could have been generated from a master secret that the attacker was capable of guessing.

In the version of OpenSSL library below 1.0.1i, the server chooses TLS v1.0 as the protocol version, if the first `ClientHello` message from the client is too fragmented. This allows an attacker to perform downgrade attacks. Since version 1.0.1i, a fragmented `ClientHello` message leads to the closure of connection instead [13].

Failures in correctly checking the certificate of the other party are really a class of its own, although they can also be grouped under "missing checks". In the 2015 version of this report, we tried to give an overview of such failures, at least in the most used pieces of software, but we believe now that this is a futile attempt. The code for validating TLS certificates in non-browser software has been called "the most dangerous code in the world" [105].

3 RSA public key cryptosystem

In this Chapter, we will present descriptions and estimates for various attacks against RSA cryptosystem. The reader should however keep in mind that, to the best of our knowledge, none of the major attacks have been fully implemented in practice. This means that all the given resource estimates are imprecise and may be off by a few orders of magnitude.

3.1 Breaking RSA with a classical computer

In recent years, no major cryptanalytic breakthroughs against RSA have occurred. This means that the security level estimates given in the 2012 ECRYPT II report on cryptographic algorithms and key sizes [39] still hold (see also Table 1).

Table 1. ECRYPT II table for security levels of asymmetric encryption

RSA/DLOG keylength	Security level
512	50
768	62
1024	73
1536	89
2048	103

So for example, in order to break RSA-1024 or RSA-2048, one needs to perform respectively about 2^{73} or 2^{103} operations on a (classical) computer. Based on this, we can give a rough estimate of the monetary investment needed to perform such computations.

In April 2016, a year of Amazon's cloud computation package m3.medium costs \$353[5]. For this price, one gets access to a single-core Intel Xeon E5-2670 v2 (Ivy Bridge) processor with maximum performance of 3.3 GHz. This gives a yearly performance of

$$3,3 \cdot 10^9 \cdot 60 \cdot 60 \cdot 24 \cdot 365 \approx 2^{56,53}$$

elementary operations. This in turn implies that the investment needed to break RSA-1024 and RSA-2048 with a classical computer is respectively

$$\frac{2^{73}}{2^{56,53}} \cdot \$353 \approx \$32,000,000 \quad \text{and} \quad \frac{2^{103}}{2^{56,53}} \cdot \$353 \approx \$3.44 \cdot 10^{16} .$$

[5]`http://aws.amazon.com/ec2/pricing/`, last visited April 21st, 2016

3.2 Breaking RSA with dedicated hardware

In 2005, Franke *et al.* proposed a special-purpose hardware device called SHARK that implemented General Number Field Sieving algorithm for factoring large integers [93]. They estimate that in year 2005 prices, hardware for an ASIC farm that would be capable of breaking RSA-1024 in one year would cost about $160 million. Power consumption per one factoring would add $60 million more.

In 2007, de Meulenaer *et al.* improved the factoring step within the number field sieve algorithm using elliptic curve methods [78]. Their estimate is that the production cost of the resulting hardware would be about $25.8 million.

The method used by de Meulenaer *et al.* relies on the approach developed by Gaj *et al.* [94]. In 2010, Zimmermann *et al.* claimed to have improved the price-performance ratio of this approach by factor 37 [179].

All in all, the rough estimate for the investment required to factor one RSA-1024 modulus is in the order of magnitude $1 million.

3.3 Breaking RSA with a quantum computer

It was already predicted by Richard Feynman in 1980s that a quantum computer may outperform classical ones in certain applications. The real breakthrough was made by Peter Shor who presented an algorithm for finding hidden subgroup orders of large groups in 1994 [159]. This in turn gives rise to algorithms for integer factoring and finding discrete logarithms.

Since majority of the present-day asymmetric cryptography relies on the assumption of either the hardness of factoring or computing discrete logarithms, practical implementation of such a computer would have enormous implications on the cryptographic applications. Among others, using RSA, DSA, ElGamal and elliptic curve cryptosystems would become insecure.

To give an idea how factoring using Shor's algorithm works, consider a toy example of $N = 15$ [137]. One starts by picking a random number $a \in [2, N-1]$, say, $a = 7$. First, we compute $\gcd(a, N)$ which can be done fast on a classical computer using Euclidean algorithm (here $\gcd$ stands for the Greatest Common Divisor of the two input numbers).

If $\gcd(a, N) > 1$ then a factor of N has already been found and the algorithm may stop. Otherwise we know that $a \in \mathbb{Z}_N^*$, where $\mathbb{Z}_N^*$ denotes the multiplicative group of residues modulo N.

We will be interested in the *order* of a in group $\mathbb{Z}_N^*$, i.e. the smallest positive integer r such that $a^r \equiv 1 \mod N$. Alternatively, r is the order of group $\langle a \rangle$, the subgroup of $\mathbb{Z}_N^*$ *generated* by a. For $N = 15$ and $a = 7$ we would get

$$\langle a \rangle = \{7^0 \equiv 1, 7^1 \equiv 7, 7^2 \equiv 49 \equiv 4, 7^3 \equiv 343 = 13, 7^4 \equiv 2401 \equiv 1, \ldots\} = \{1, 4, 7, 13\},$$

hence $r = 4$.

There is no fast classical algorithm for finding r, but Shor's algorithm allows to achieve exactly this on a quantum computer. Due to considerable technical complexity, we leave the quantum part for Appendix A. An interested reader can also find further details in G. Eric Moorhouse's excellent presentation [138].

If r turns out to be odd, we run the algorithm again, otherwise we can find (classically) the value of $a^{r/2}$; essentially a square root of 1 modulo N. If N is a product of two primes (as in the case of RSA), there will be four possible values of this square root. Two of them (1 and -1) are not interesting and if we hit them, we run the whole algorithm again with a new choice of a. But for the two others we know that $\gcd(a^{r/2} \pm 1, N)$ are non-trivial factors of N.

In case of $N = 15$, $a = 7$ and $r = 4$, we would get

$$\gcd(7^2 + 1, 15) = \gcd(50, 15) = 5 \quad \text{and} \quad \gcd(7^2 - 1, 15) = \gcd(48, 15) = 3\ .$$

A straightforward implementation of Shor's algorithm requires a quantum computer of $3\lceil \log_2(N) \rceil$ qubits [137]. So for example factoring 15 would require 12 qubits and factoring a 1024-bit RSA modulus would require 3072 qubits. The latter number is currently wildly out of reach for the available technology.

In 2016, Monz *et al.* proposed an improved version of a quantum computer that is able to factor an n-bit number using only $n + 1$ qubits (instead of $3n$ required by classical Shor algorithm) [137]. However, this improvement in the number of qubits comes for the price of three times larger running time of the algorithm. As qubits have very short life times, this poses a great engineering challenge in practice.

The claims of Monz *et al.* are also disputed by Cao and Liu [71].

There are also other possible implementations of Shor's algorithm. Interpreting the recent work by John Martins *et al.* [43], Jeffrey Morris [139] has estimated the effort required to break a 2048-bit RSA modulus as given in Table 2.

Table 2. Effort required to break RSA-2048

Parameter	Classical computer	Quantum computer
Time to run	10 years	24 hours
Size of hardware	Server farm covering $\frac{1}{4}$ of North America	100K logical qubits and 200M physical qubits, in less than a small room
Cost	$\$10^{18}$	$\$10^{11}$

For comparison, the GDP of USA was approximately $\$1,82 \cdot 10^{13}$ in 2015 [28]. Note also that the cost estimate for breaking RSA-2048 obtained earlier in this Section was $\$3.44 \cdot 10^{16}$, so the truth probably lies somewhere in between.

3.4 Breaking RSA with a D-Wave computer

D-Wave Systems, Inc. is a Canadian company devoted to developing a special kind of quantum computing device also generally known as D-Wave.

First it needs to be clearly emphasised that D-Wave is not a fully fledged quantum computer in the sense stated by Richard Feynman. Rather, one can say that D-Wave is to a quantum computer what an analogue computer is to a classical digital one.

D-Wave computer is designed to solve very specific kind of optimisation problems, namely finding global minima for objective functions. It does so by applying a method called quantum annealing which is similar to simulated annealing used on classical computers. Thanks to a special quantum phenomenon known as quantum tunnelling, quantum annealing can theoretically outperform the classical version of the algorithm.

Several prototypes of the D-Wave computer have been built and demonstrated. However, the demonstrations have not been convincing in that they actually can achieve the quantum speedup. [72].

If the quantum speedup would turn out to be real, it would, in principle, be possible to create a device that breaks RSA. For that, one first needs to select a suitable NP-complete optimisation problem and build a D-Wave computer to solve it. (See the textbook [37] for further details on NP-completeness.) Since integer factoring problem is in the NP complexity class, one would have a polynomial reduction that would turn an instance of the factoring problem to an instance of the optimisation problem. Solving the latter one would give rise to factoring the original RSA modulus.

However, this construction relies on several strong assumptions.

1. D-Wave computer outperforms a classical computer in an asymptotically significant manner.
2. The function to optimise obtained after the polynomial reduction can be efficiently implemented on a D-Wave computer.
3. The D-Wave optimisation device really computes global optima of the required functions on a sufficiently large fraction of input instances.

None of these assumptions is yet reliably verified, hence we can consider breaking RSA with a D-Wave computer to be inconceivable at this point.

In fact, it is conjectured by leading complexity theorists that quantum computers (of any kind) can not efficiently solve NP-hard problems [92]. At the same time, NP-completeness of integer factoring is not proved either.

3.5 Transition to 3072-bit RSA

2048-bit RSA is no longer recommended for new implementations requiring long-term (over 10 years) security [39, 160, 18]. However, support for the next logical candidate, 4096-bit RSA, may be missing or too inefficient on some more limited platforms. Hence, an intermediate transition step to RSA3072 may be considered.

Even though being less used (and hence also less tested) than RSA2048 and RSA4096, the deployment of 3072-bit RSA is not expected to cause major problems. In principle, every installation utilizing RSA4096 should also be capable of handling RSA3072. Still, making sure that the transition will not cause any problems is responsibility of the implementer.

4 Elliptic curve cryptography

The 2015 version of the report [18] contained a thorough introduction to elliptic curve cryptography (ECC) together with the required mathematical background. Also, the 2013 report [12] covered patent issues of ECC. Even though the previous reports were in Estonian, we are not going to translate the corresponding parts here. (This may happen in the future versions of the report, though.)

Instead, this report only covers the most important updates on the topic. An updated table of ECC support in various libraries can be found in Appendix B, and the next Section covers the controversial announcement made by NSA concerning phasing ECC out in favour of future, not-yet-existing cryptographic primitives.

4.1 NSA Suite B controversy

In August 2015, the U.S. National Security Agency (NSA) released a public announcement declaring faster-than-previously-planned transition to post-quantum cryptography [17]. Among its implications is rather an unexpected change in recommendations concerning elliptic curve cryptography. Namely, the use of the curve P-256 is no more recommended, and only P-384 has remained.

The explanations given by the NSA are rather short-worded and leave a lot of room for speculations. The theories range from speculating that NSA knows of some non-public breakthrough, to trying to confuse Russians and Chinese. The main versions have been analysed by Neal Koblitz and Alfred J. Menezes in their report [119]. The report does not make any conclusions concerning which version is more likely to be correct. However, it is interesting to note that Koblitz and Menezes argue that the so-called NIST curves (including P-256 and P-384) are likely to be free from the backdoors planted during their generation in 1990s.

From US government point of view, the situation was somewhat clarified by the FAQ released by the Information Assurance Directorate in January 2016 [21]. It refers to the long required life cycle (20-30 years) of national security systems as the main reason, together with the belief that a sufficiently powerful quantum computer will be developed during that time.

However, assuming that creating sufficiently large quantum computers would become a reality in near future, both the curves P-256 and P-384 would be broken with comparable amount of effort. On the other hand, the FAQ conveniently ignores the question why NSA believes that the curve P-384 is considerably more secure than P-256. It seems likely that the FAQ only presents part of the story.

5 Known attacks against hash functions

Hash functions are used in many important security-critical applications like digital signatures, timestamps, message authentication codes and authentication protocols. Attacks against hash functions may thereby have a large influence on the overall security of electronic services. Several hash functions (like MD5, SHA-1) that are still in use in some applications today have been successfully attacked in terms of collisions. As replacing a hash function in widely used applications tends to be very costly, it is extremely important to know what exactly are the practical consequences of the collision attacks. Finding a collision for a hash function does not necessarily mean that the hash function is insecure in every possible application. Some applications may just need *pre-image resistance* of *second pre-image resistance*, not the full *collision-resistance*.

5.1 Types of attacks and the notions of hash function security

Several different security properties of hash functions could be assumed based on applications. Often in formal security proofs, hash functions are assumed to be random functions (the so-called *random oracle* model). Having such an assumption in mind, we may define an attack against a hash function as any method capable of finding input-output pairs for the hash function in a way that would be infeasible in the random oracle model.

The following three types of attacks against hash function $h()$ are the most common:

- *Pre-image attack*: given an output y, find an input M for which $h(M) = y$.
- *Second pre-image attack*: given an input M, find a different input M' so that $h(M') = h(M)$.
- *Collision attack*: find two different inputs M and M' such that $H(M) = H(M')$.

When claiming something about the *security* of a hash function, we always have to refer to a particular attack. We say that a hash function is S-secure against a certain attack, if every possible attack uses at least the time comparable to computing the hash $h(M)$ for S inputs M. Note that no hash function with n-bit output can be more than:

- 2^n-secure pre-image or second preimage resistant, because using the simple trial and error method, a pre-image of any output y can be found using about 2^n hash computations on average.
- $2^{\frac{n}{2}}$-secure collision-resistant, because using the Birthday attack, a collision can be found by computing the hash-outputs for about $2^{\frac{n}{2}}$ randomly chosen inputs.

For example, SHA-1 (with 160-bit output) cannot be more than 2^{160}-secure preimage-resistant, nor 2^{80}-secure collision-resistant. MD5 (with 128-bit output) is no more than

Table 3. Known attacks against hash functions.

Hash function	Collision attack	(2nd) Pre-image attack
MD2	$2^{63.3}$	2^{72}
MD4	3 operations	2^{102}
MD5	2^{18}	$2^{123.4}$
SHA-0	$2^{33.6}$	2^{160}
SHA-1	$2^{60.3}$	2^{160}
RIPEMD-160	2^{80}	2^{160}
SHA2-256	2^{128}	2^{256}
SHA2-512	2^{256}	2^{512}

2^{128}-secure preimage resistant, nor 2^{64}-secure collision-resistant. In practice, more efficient *collision attacks* have been found to both of these hash functions. Table 3 summarises the status of the most well known hash functions.

5.2 Collision attacks against Merkle-Damgård hash functions

Most of the known practical hash functions have the so-called Merkle-Damgård structure, which means that the input M is partitioned into blocks $M_1, \ldots, M_\ell$ of fixed size and then processed iteratively, taking $h_0 = \mathrm{IV}$ (where IV is a fixed initial value) and

$$h_i = F(h_{i-1}, M_i)$$

for all $i = 1 \ldots \ell$. The hash $H(M)$ is defined to be the last-computed value h_ℓ. The function F is called the *compression function*. For example, MD2, MD4, MD5, SHA-1 and SHA-2 are all Merkle-Damgård hash functions.

The most common types of collision attacks are the following.

Collision attack Find $M \neq M'$ so that $H(M) = H(M')$. The first collision attack against MD5 was launched by Wang, Feng, Lai, and Yu in 2004 [171]. No known successful collision attacks against SHA-1 have been launched. The first theoretical attack (with 2^{69} hash steps) was described by Wang, Yin and Yu in 2005 [173]. The best known attack (with $2^{60.3}$ hash steps) was presented by Stevens in 2013 [162]. Grechnikov presented an attack against the reduced version (74 from 80 rounds) of SHA-1 with 2^{35} hash steps [90]. Note that the general collision attack is not necessarily dangerous in practice, as M and M' are not necessarily meaningful messages.

Chosen-prefix collision attack Given two inputs M_1 and M_2, find continuations M_1' and M_2', so that $M_1 \| M_1' \neq M_2 \| M_2'$ but $H(M_1 \| M_1') = H(M_2 \| M_2')$. The first chosen-prefix collision attack (with 2^{39} hash steps) against MD5 was launched by Stevens, Lenstra and de Weger in 2007 [164, 165]. No known chosen-prefix collision attacks have been launched against SHA-1. The first theoretical attack (with 2^{74} hash steps) was published by Stevens in 2013 [162]. Chosen-prefix collision attacks are indeed dangerous in practice. For example, a chosen-prefix collision attack against MD5 was successfully used to forge certificates of Certification Authorities [166].

The target of the next attacks is the compression function F of the hash function and hence, these attacks are not attacks against the full hash function.

***Freestart collision attack* (or *pseudo-collision attack*)** Find pairs $(V, M) \neq (V', M')$ so that $F(V, M) = F(V', M')$. This is equivalent to finding any collision for the compression function. The first successful free-start collision attack against MD5 was launched and published by de Boer and Bosselaers in 1993 [81]. The first successful free-start collision attack (with $2^{57.5}$ hash steps) against SHA-1 was launched and published by Stevens, Karpman, and Peyrin in 2015 [163].

Semi-freestart collision attack Find $M \neq M'$ and V so that $F(V, M) = F(V, M')$. The first successful semi-freestart collision attack against MD5 was launched and published by Dobbertin in 1996 [86]. No semi-freestart collision attacks against SHA-1 have been launched.

Near-collision attack Given V and V', find $M \neq M'$ so that $F(V', M')$ and $F(V, M)$ differ by a small number of bits or are "close" in another previously defined way. Near-collision attacks have no direct practical implications, but they are used as tools in attacks against the full hash function. For example, near-collisions were used in the the chosen-prefix collision attack by Stevens in 2013 [162].

Collision attacks against the compression function F of a Merkle-Damgård hash function H do not necessarily mean a collision attack against the full hash function H, because the prefix V is not necessarily the right (standard) initial value IV. If $V \neq \mathrm{IV}$, then in order to extend a collision (with pairs (V, M) and (V, M')) to a collision of the full hash function, one has to find a prefix W such that $H(W) = V$, because then indeed $H(W\|M) = H(W\|M')$, because due to the Merkle-Damgård structure of H:

$$\begin{aligned} H(W\|M) &= F(F(\mathrm{IV}, W), M) = F(H(W), M) = F(V, M) = F(V, M') = F(F(\mathrm{IV}, W), M') \\ &= H(W\|M')\ . \end{aligned}$$

If, however, an efficient attack is found that for a *randomly chosen* V finds (with reasonable probability) $M \neq M'$ so that $F(V, M) = F(V, M')$, then it would also be efficient to find collisions for the full hash function. Indeed, one can launch this attack for $V = F(\mathrm{IV}, W)$ where W is a uniformly random prefix to find M and M' so that $F(V, M) = F(V, M')$ and then $H(W\|M) = H(W\|M')$ as explained above.

5.3 Practical chosen-prefix collision attack against certificates

In 2009, Stevens, Sotirov, Appelbaum, Lenstra, Molnar, Osvik and de Weger [166] showed that chosen-prefix collision attacks against MD5 can be used to forge real-life web-server certificates. The idea of the attack is to generate two different X.509 certificates with the same MD5 hash which is used by a Certification Authority to sign certificates. One certificate (the *rogue certificate*) is a so-called CA-certificate that has authority to sign new certificates (for example to the google.com website) and which many web browsers would blindly accept and trust. The second certificate, the data of which is presented to the Certification Authority, is an ordinary user certificate without the right to issue new certificates. By signing the second certificate, the Certification Authority also implicitly signs the rogue certificate, because their hashes match. The rogue certificate can be used to sign new certificates that might later be used with several e-business related theft schemes. For example, the attacker is able to simulate arbitrary internet service (like gmail.com or swedbank.ee), steal passwords, read arbitrary e-mails, etc.

Considering the progress in several attacks against SHA-1 such as Wang *et al.* [173, 172] and Stevens [162], and that many Certification Authorities still use SHA-1, it is natural to study the practical consequences of the rogue certificate attack against different types of Certification Authorities.

Note that the reliability of root certificates, which are self-signed, does on depend on the security of the hash function that is used to sign those certificates. Self-signed certificates are just software containers for public keys and do not provide any security or reliability. The integrity and authenticity of self-signed certificates has to be achieved with non-cryptographic (physical, organisational, etc.) means.

5.4 The cost of chosen-prefix collision attacks against SHA-1

Considering the estimations based on the techniques used in collision-attack against SHA-1 in 2015 [163], the cost of a collision attack against SHA-1 is between 75,000 and 120,000 US dollars when renting Amazon EC2 cloud over a few months, which is about 100 times lower than estimated in the previous report [12].

As the chosen-prefix collision attacks against SHA-1 are about 2^{17} times more costly than the ordinary collision attack, the estimated cost of the chosen-prefix collision attacks is between 9.8 to 15.7 billion US dollars, which means that these attacks are already feasible for large organisations and companies.

5.5 Attacks against existing certificates

Chosen-prefix collision attacks are the only known practically feasible attacks with practical security implications. Their application against Certification Authorities is only possible during the process of applying for a new certificate, and cannot be used against the existing certificates. In order to apply for certificates, the applicants must identify themselves to Certification Authorities. This means that if a rogue certificate is discovered, the attacker's identity can be determined using the logs of Certification Authorities.

At the first glance, much more dangerous attacks are those targeted towards the existing certificates, because such attacks can be launched without communicating with Certification Authorities and without risks of being identified as an attacker. For example, an attacker may try, based on an existing certificate X, to find a new rogue certificate $X' \neq X$ so that their SHA-1 hashes match. For example, if X is the certificate of an OCSP server of the Estonian ID-card infrastructure, then the attacker would be able (under certain conditions) to forge digital signatures of any Estonian eID user.

Note, however, that such an attack is not a collision attack but an attack against the second pre-image resistance, because X is fixed before the attack. No practical second pre-image attacks against SHA-1 are known. Moreover, even against MD5, such attacks would take about 2^{123} hash steps, which is about 2^{46} ($\approx 10^{14}$) times more than the chosen-prefix collision attack against SHA-1.

Table 4 summarises the practical consequences of attacks against hash functions to the most important electronic services. As we see, second pre-image attacks have the most serious practical consequences, but such attacks are by far not feasible today, even against MD4 and MD5 that are totally broken in terms of collisions. General collision attacks can, in principle, be used to forge digital signatures, but just on meaningless random-looking documents.

Table 4. Practical consequences of attacks against hash functions.

	2nd preimage attack	Chosen-prefix collision attack	Collision attack
Attacker stays anonymous	yes	no	no
Impersonation	yes	yes, with rogue certificates	no
Forged signatures	yes	yes	yes, but for meaningless documents
Repeatable forgeries	yes, with rogue certificates	yes, with rogue certificates	no
Forged time stamps	yes	yes, with rogue certificates and only for RFC3161 timestamps	no
Forged archival documents	yes, if not timestamped securely (with a new, more secure hash function) before the hash function becomes insecure	no	no
Practical feasibility against MD5	Infeasible, requires 2^{123} hash steps	Feasible for everyone, requires 2^{39} hash steps	Simple, a few seconds in ordinary computer
Practical feasibility against SHA-1	Infeasible, requires 2^{160} hash steps	Feasible for large organisations	Feasible for large organisations

Only the chosen-prefix collision attacks are capable of creating forged signatures on meaningful documents and launching some other practical attacks. The rogue certificate attacks have central importance here, especially those not requiring direct identification of the applicant of the certificate. For example, web-server certificates are suitable targets of such attacks. Hence, the most important practical consequence of chosen-prefix collision attacks and their estimated cost is that SHA-1 must not be used for signing web-server certificates.

6 The SHA-3 hash function

The SHA-3 cryptographic hash function standard was released by NIST in August 2015. The SHA-3 standard is based on the Keccak hash function that was designed by Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche.

During 2006–2012, NIST organised a hash function competition in order to choose a new hashing algorithm for the upcoming hash standard, SHA-3. The reason was not to replace SHA-2 as no practical attacks were known against SHA-2. However, successful attacks against MD5 and the earlier version of SHA-1 showed that more hash functions are needed, possibly with the structure that deviates from the standard Merkle-Damgård design. Indeed, Keccak is based on the so-called *sponge* design that is somewhat different from all the previous hash functions. In October 2012, Keccak was selected as the winner of the competition and after some adjustments concerning

- increased number of rounds,
- simplified padding scheme from a more complex scheme to the $10 * 1$ pattern,
- increased hashing rate (to the maximum rate that the security calculation allowed).

In 2014, NIST published a draft version of FIPS 202 "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", and the final version of FIPS 202 was approved in August 5, 2015 [20].

6.1 Sponge construction

A sponge construction has the following components:

- b-bit state memory S, that is divided into two parts: (1) R of size r (called the *bitrate*), and (2) C of size c (called the *capacity*),
- state transformation function $f: S \to S$,
- padding function P that appends bits to the input string in order to adjust the input length to a multiple of r.

The sponge construction is initialized as follows:

1. The state S is initialized to 0^b;
2. The input p is padded, and divided into r-bit blocks by using the function P.

Then, for all blocks (starting from the first block), the following operations are performed:

1. R is XOR-ed with the r-bit block;

2. S is replaced by $f(S)$;
3. Next block is taken.

This process resembles absorbing water in a sponge. If there are no blocks left, then the output is produced with the following process that resembles squeezing water out of the sponge.

1. Output the r-bit R portion of the state memory S;
2. If more output bits are needed, S is replaced by $f(S)$ and the process is repeated.

If the output length is not a multiple of r, it is truncated.

The collision-resistance and preimage-resistance of a sponge-type hash function depends of the exact construction of f, as well as the parameters r and s.

Figure 1 illustrates this construction.

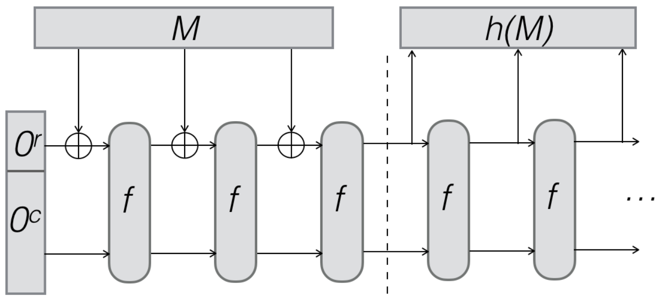


Figure 1. The sponge construction: padded input message M is converted to the output hash $h(M)$.

6.2 Padding

Given the bit-rate r and the message length m in bits, the padding algorithm returns a padding string P such that $m + |P|$ is a positive multiple of r. The padding string is of the form

$$P = 1||0^j||1 \ ,$$

where $j = (-m - 2) \bmod r$ and || denotes the concatenation operation.

6.3 State transformation function

In SHA-3, the state S is logically divided into a 5×5 array of 64-bit words with total length of $b = 1600$ bits. The state transformation function is defined for any power-of-two word size, $w = 2^\ell$ bits. In all instances of SHA-3, $\ell = 6$ and $w = 64$. The state can also be considered as a $5 \times 5 \times w$ array of bits. Let $A[i][j][k]$ be the bit $(5i + j) \cdot w + k$ of the input. Note that all the index operations are performed $\bmod 5$ for the first two indices, and $\bmod w$ for the third index.

The state transformation function iterates $12 + 2\ell = 24$ times the following five sub-rounds, denoted in the standard by Greek letters:

θ: Find the parity bit of each 5-bit column, and XOR that into two nearby columns in a regular pattern:

$$A[i][j][k] \leftarrow A[i][j][k] \oplus parity(A[0...4][j-1][k]) \oplus parity(A[0...4][j+1][k-1]) .$$

ρ: Rotate each of the 25 words bitwise by a triangular number $0, 1, 3, 6, 10, 15, \ldots$, i.e. $A[0][0]$ is not rotated, and for all $0 \leq t < 24$:

$$A[i][j][k] \leftarrow A[i][j][k - (t+1)(t+2)/2] \ ,$$

where $\begin{bmatrix} i \\ j \end{bmatrix} = \begin{bmatrix} 3 & 2 \\ 1 & 0 \end{bmatrix}^t \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, where computations are performed $\bmod 5$.

π: Permute the 25 words in a fixed pattern: $A[j][2i + 3j] \leftarrow A[i][j]$.

χ: Bitwise combine along rows, using $a \leftarrow a \oplus (\neg b \& c)$, i.e.

$$A[i][j][k] \leftarrow A[i][j][k] \oplus \neg A[i][j+1][k] \& A[i][j+2][k] \ .$$

ι: XOR a round constant into one word of the state, i.e. in round n, for $0 \leq m \leq \ell$, the bit $A[0][0][2m-1]$ is XOR-ed with bit $rc[m + 7n]$ of an LFSR sequence of degree 8 defined by

$$rc[t] = (x^t \bmod x^8 + x^6 + x^5 + x^4 + 1) \bmod x \ ,$$

where the computations with the coefficients of the polynomials are always $\bmod 2$. The round constants are represented in Table 5.

Table 5. Round constants for SHA-3 in hexadecimal.

Round	Constant	Round	Constant
0	0000000000000001	1	0000000000008082
2	800000000000808A	3	8000000080008000
4	000000000000808B	5	0000000080000001
6	8000000080008081	7	8000000000008009
8	000000000000008A	9	0000000000000088
10	0000000080008009	11	000000008000000A
12	000000008000808B	13	800000000000008B
14	8000000000008089	15	8000000000008003
16	8000000000008002	17	8000000000000080
18	000000000000800A	19	800000008000000A
20	8000000080008081	21	8000000000008080
22	0000000080000001	23	8000000080008008

6.4 Instances of SHA-3

There are four instances of SHA-3 hash functions, named SHA3-224, SHA3-256, SHA3-384, and SHA3-512. The last number indicates the output size in bits. This means that all instances of SHA2 can easily be replaced with the corresponding instances of SHA3 (and vice versa). For all these four instances, an additional padding of the message (before the regular padding) is applied: a two-bit suffix `01` is concatenated to the message.

There are two additional instances SHAKE128 and SHAKE256 of SHA-3 instances that can be used as extendable-output functions (XOFs), i.e. hash functions in which the output can be extended to any given length. The suffixes “128” and “256” indicate the security strengths that these two functions can generally support, in contrast to the suffixes for the hash functions, which indicate the digest lengths. For these two instances, a different additional padding of the message (before the regular padding) is applied: a four-bit suffix `1111` is concatenated to the message. This is to distinguish between the input sequences of the SHA3 and SHAKE families.

Table 6 summarises the instances of SHA-3 family of hash functions.

Table 6. The instances of SHA-3 and their technical and security parameters.

Instance	Output size	Capacity	Block-size	Collision resistance security bits	Preimage resistance security bits
SHA3-224	224	448	1152	112	224
SHA3-256	256	512	1088	128	256
SHA3-384	384	768	832	192	384
SHA3-512	512	1024	576	256	1024
SHAKE128	arbitrary d	256	1344	$\min(\frac{d}{2}, 128)$	$\min(d, 128)$
SHAKE256	arbitrary d	512	1088	$\min(\frac{d}{2}, 256)$	$\min(d, 256)$

7 Secure computation outsourcing

Recent developments in cloud-based environments have made computation outsourcing more accessible and flexible than ever before. However, vanilla cloud computing environments are not suitable if the data to be processed has privacy restrictions.

The core of the problem lies within increasing the *trusted computing base*. In a classical scenario, where data processing is performed in the analyst's computer, the computing environment can, in principle, be controlled. The analyst may be given a background check, the computer may be cut off the Internet, its firmware intensely scrutinized, etc. However, in case of computation outsourcing (e.g. to a cloud provider), many hard-to-control components are added ranging from the actual cloud hardware used to provide services, to the cloud provider's personnel.

While private companies could accept the risks of trusting a third party, say a public cloud provider by utilizing insurance to mitigate their risks of possible damages caused by the loss or exposure of their customers' private information, governments may not be able to do so – the risk of inadvertently exposing private or government entities and citizens' data cannot be adequately measured, nor compensated financially when the risk has realized. Therefore cost-effectiveness calculations on using public cloud resources for non-public information may be heavily misleading or incorrect, however tempting these calculations may seem in an Excel sheet.

Standard state of the art encryption (e.g. AES) protects data privacy while in transit or at storage. However, when one wants to use that data for analysis, it must be first decrypted. This means that when outsourcing computations, the data processors must still be trusted to have access to the data or its decryption key.

Secure multi-party computation (SMC) provides a way to relax this trust requirement as it enables to compute functions on data without the computation party having access to the (whole) input. More formally, in SMC, we have multiple parties that want to compute a given function on their collective inputs. These parties can compute the function so that each of them gets their (part of the) output. At the same time, SMC guarantees that none of the parties learn anything more than their own input and their part of the output.

That said, it is important to understand that secure multi-party computation technology is not a drop-in replacement for applications and services, where plaintext access to individual data records is necessary, e.g. e-mail and file sharing services or classical information systems. The technology is most suitable in scenarios where only computation end results or aggregate values need to be published, e.g. statistical analysis on combined data sets, benchmark surveys with inputs from mutually distrusting parties and outsourcing resource-intensive computations. More example scenarios are given in Sections 7.5 and 7.6.

There are three basic methods to do secure multi-party computation: two-party approach based on garbled circuits, computing on encrypted data with fully homomorphic encryption and multi-party computation based on secret sharing schemes. Each of those methods has different use cases and performance, which we will briefly introduce.

7.1 Garbled circuits

Garbled circuits (GC) is a two-party secure computation approach introduced by Yao in 1980s [176, 177] and later detailed by many others [107, 44, 141]. In GC, there are two computation parties, both having their own input and the function they want to compute is given as a Boolean circuit composed of AND, OR, and XOR gates like in an electronic circuit. Inputs and the functionality are hidden by "garbling" the circuit – replacing all bit values on wires with randomly generated strings.

One computation party garbles the circuit and the other one evaluates it, while also communicating with the first party. By the design of garbled circuits, the party evaluating the circuit receives the computation output. However, depending on the application requirements, it may be encrypted with only the first party having the decryption key.

This description suggests that the computations are CPU-bound with only a few communication rounds. The communication load is also asymmetrical, as the evaluator starts its computations only after the first party has finished. Nevertheless, there are designs, where the garbled circuit is streamed to the evaluator to reduce this asymmetry and the memory footprint at the first party [128].

A more in-depth descriptions of the garbled circuit model along with security proofs are given in [127, 46]. Some practical implementations based on garbled circuits include Fairplay [129, 48], TASTY [109], FastGC [111] and ABY [80].

7.2 Fully homomorphic encryption

Fully homomorphic encryption (FHE) is a client-server model, where the server performs computations on encrypted data without access to the private key held by the client. FHE takes advantage of the homomorphic properties of the underlying public key encryption scheme. This means that it is possible to modify the value under encryption in a desired way without decrypting it first.

Practical cryptosystems allow only either homomorphic additions, multiplications or evaluating low-degree polynomials (e.g. many additions with a single multiplication). The reason for the latter restriction is that each homomorphic operation introduces a small amount of noise into the ciphertext. As this noise is cumulative, the allowed number of homomorphic operations are bounded on a given ciphertext as eventually the ciphertext becomes impossible to decrypt correctly. In 2009, Gentry proposed to solve this restriction with *bootstrapping* – homomorphically re-encrypting the ciphertext under encryption yielding a fresh noise-free ciphertext [100]. FHE has received a lot of attention and alternative constructions from the research community [64, 102, 101, 103, 104, 161]. Some of the practical implementations include HElib [108], hcrypt [148], Stanford's FHE library [175] and FHEW [87].

Unfortunately, both homomorphic operations and bootstrapping are costly operations.

Moreover, in FHE, the ciphertext blow-up from homomorphic operations is significant and makes the scheme impractical for many applications. Nevertheless, FHE is used as a sub-operation for non time-critical pre-computations for other SMC systems, e.g. SPDZ [77].

7.3 Secure computation based on secret sharing

Secret sharing [158, 55] is a concept, where a secret value is split into random parts called "shares" and these shares are distributed among several (usually three or more) independent parties. Depending on the secret sharing scheme used, all or at least some threshold of the total number of shares are needed to reconstruct the original secret value. Any set of shares less than this threshold gives no information about the secret value. By itself, secret sharing provides an information-theoretically secure means of storing sensitive data, i.e. it is secure even against computationally unbounded adversary[6].

There are cryptographic protocols that allow to compute on such data by taking secret-shared values as input and producing secret-shared output values. The latter allows to chain individual elementary function protocols to build algorithms and privacy-preserving applications.

In the secret sharing based model it is important to notice that the secret-shared input data does not have to come from the parties participating in the computation. This allows several input parties to collaboratively analyse their inputs or link whole data sets by outsourcing the computation to a set of non-colluding computation parties.

There are several practical implementations of secure multi-party computation based on secret sharing. Examples include VIFF [76], SEPIA [70], Sharemind [61, 58], ShareMonad [125], FRESCO [75], SPDZ [77] and ABY [80]. As detailed in Section 7.6, the secret sharing based SMC technology has also been used in real-world large-scale applications [63, 62, 59].

7.4 Security considerations

All three described SMC methods require mutually authenticated communication channels between participating parties. In addition, secret sharing based SMC protocols require communication channel encryption. The latter is also useful in garbled circuit based approach to hide the computable functionality (circuit structure) from third parties.

Different SMC protocols are secure in different adversarial models. The two most common such models are active and passive security. In the passive model, the protocol remains secure if the computation parties are only allowed to analyse the shares available to them in order to deduce some extra information. However, in the passive model the computation parties are required to follow the agreed upon protocol and not change the values provided to them. In contrast, protocols that are secure in the active model allow arbitrary deviations.

While clearly superior in security, actively secure SMC protocols require computation verification that introduces a significant performance penalty. In many practical scenarios, where the non-collusion of computation parties can be guaranteed, e.g. by finding suitably motivated parties, the passive model provides a reasonable security-performance trade-off. For

[6]However, in practical implementations, pseudo-random number generators are used to generate random shares. Moreover, encrypted communication channels (e.g. TLS) are only computationally secure.

example, all three aforementioned practical secret sharing based SMC applications worked in the passive model.

7.5 Applications and performance

The three described SMC methods are all applicable in different use cases. Fully homomorphic encryption is best suited for a single entity to outsource computations on its confidential data to an untrusted server, e.g. cloud. Garbled circuits approach can be used by two mutually distrusting parties to compute a common function on their inputs. Secret sharing based secure computing is best suited in scenarios, where individual secret values or data sets are combined and aggregated from several input parties.

As the described methods are fundamentally different, it is almost impossible to compare their performance on application level. However, there are individual algorithms that are accepted by the research community for benchmarking. For example, Figure 2, published in the secure computation maturity overview by Archer, Bogdanov, Pinkas and Pullonen [36], shows AES-128 block cipher performance for different implementations of SMC. As seen, implementations based on linear secret sharing (LSS) give the lowest amortised running time.

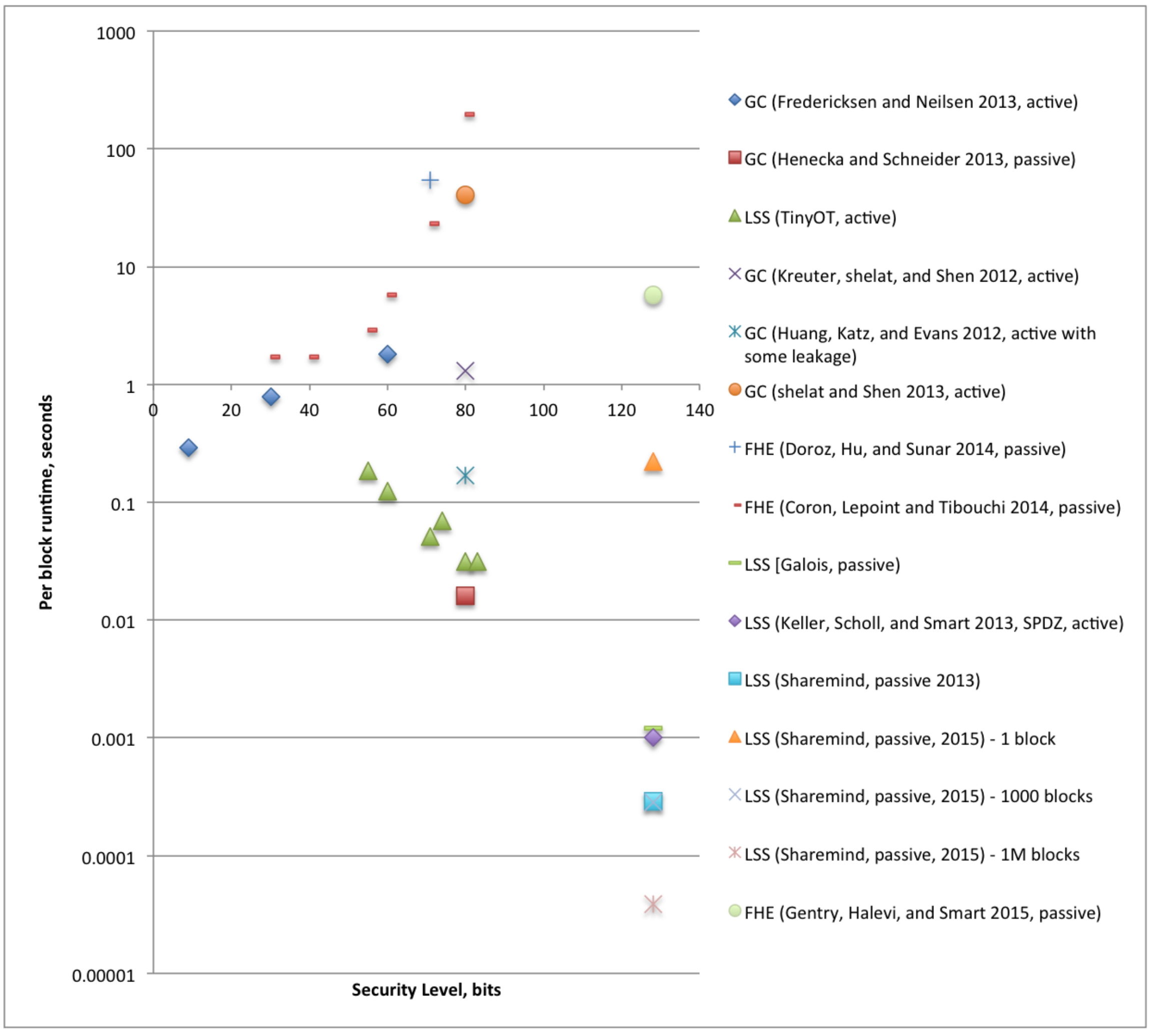


Figure 2. Comparison of SMC paradigms using the AES-128 block cipher [36].

7.6 Practical deployments

Next, we will give a brief overview of three practical SMC applications that worked on real-life data. All of these applications used secure multi-party protocols based on secret sharing and worked in the passive security model.

The first practical SMC application was deployed in Denmark in 2008, where it was used to implement a double auction in order to redistribute sugar beet growing contracts [63]. In double auction, every seller and every buyer makes a bid on how much goods he is willing to sell or buy for each of price from a predetermined set. Based on these bids, the trusted auctioneer computes the most optimal *market clearing price*, where total demand equals total supply.

In this deployment the role of the trusted auctioneer was distributed among the three secure multi-party computation parties – the sugar beet growing farmers association DKS, the sugar producer Danisco and the technology provider SIMAP project. Using SMC to compute the market clearing price guaranteed that the auctioneer learns no extra information about individual bids that could be possibly abused. The users used a Java applet obtained from a web page to enter their bids for all 4,000 pre-configured price points. In this deployment the secret-shared bids were not distributed directly between the computation nodes as they were not online in this phase. Instead, the applet encrypted each of the three shares independently with the public key of the corresponding computation party and stored all encrypted shares in a central proxy server.

In the second phase of the auction, the representatives of the computation parties gathered together, downloaded their encrypted shares from the central database, decrypted them with the corresponding private key in their possession and started the secure multi-party computation process. The necessary multi-party computation operations were fairly simple – the market clearing price was found by binary search using about twelve oblivious comparisons. The computation itself was carried out over local area network and lasted about 30 minutes, the bulk of which was spent on decrypting shares.

The next practical SMC application working on real data was deployed in Estonia in 2011 [62]. It was deployed for the Estonian Association of Information Technology and Telecommunications (officially abbr. as ITL) members to compare their financial indicators with each other and thus get an up-to-date overview of the local ICT sector. The application used the Sharemind SMC implementation with three computation parties hosted by ITL members Zone Media, Microlink and Cybernetica. To make data input more convenient for the participants, it was provided as a web-based form integrated into the member area of the ITL web page. The input financial indicators were secret shared using a JavaScript library in the end user's web browser and distributed directly among the three computation parties over secure HTTPS connections.

The analysis step itself was simple and consisted only of sorting values of each financial indicator independently. This hid the connections between different indicators for the same company and the sorted vectors could be published to the ITL Board for further analysis. Although the number of participants was small, it is the first practical SMC application where the computation is carried over the internet. ITL used this application twice in 2011.

In 2015, a privacy-preserving statistical study using SMC technology was conducted in Estonia in order to find connections between working during university studies and failing to graduate in time [59]. The study used the Sharemind secure multi-party computation framework to securely link together and analyse the education information from the Ministry of Education and Research and the income tax data from the Estonian Tax and Customs Board. The data owners used special data import applications to secret share and upload their data sets to the three computation parties hosted by the Estonian Information System Authority, the Ministry of Finance IT Centre (also the IT department for the Tax and Customs Board) and the technology provider Cybernetica.

In this deployment, the privacy-preserving analysis part was substantial and was carried out by a professional statistician at the Estonian Center for Applied Research. The statistician used the Rmind tool [60] in order to perform necessary data transformations and analysis on the imported data sets. In its functionality and user interface, Rmind is very similar to the popular open source statistical analysis suite R [150], with the exception that all privacy-preserving operations are carried out using SMC protocols by the computation parties. With its input of over 10 million tax records and 600,000 education records, this deployment is the first large-scale registry-based statistical study using secure multi-party computation technology. Moreover, for the first time, the non-collusion of the computation parties in an SMC deployment was enforced by a contract.

8 Post-quantum cryptography

8.1 Quantum computers

Quantum computers are able to perform parallel computations by using the superposition principle from quantum mechanics by which the microscopic processes in nature evolve in parallel via all possible paths until an observation is made after which one concrete path is chosen randomly. For example, an n-bit quantum register, having been suitably prepared, may be simultaneously in all 2^n possible states. Such a hybrid state is called a *superposition*. Every function $f(x)$ may be implemented as a quantum circuit the input of which is a superposition of all possible arguments x and the output is a superposition of all possible values $y = f(x)$ of the function. This resembles the classical parallel computation where 2^n threads run in parallel, but one needs 2^n times more computational resources for making classical parallel computation as fast as a single thread.

In quantum computers, such a parallelism comes almost for free, i.e. 2^n parallel quantum threads use the same amount of computational resources as one single thread. The problem is that these threads are not simultaneously accessible. If one measures (reads) the output of the quantum circuit, we just have a randomly chosen output value $y = f(x)$. Such a parallel computation would give us nothing compared to a classical single-thread computation, because it is equivalent to just computing the output $f(x)$ for a randomly chosen input x.

In classical parallel computing, all threads and their computational results are simultaneously accessible and arbitrary inter-thread communication is possible. In quantum parallelism, the information exchange between the threads is extremely limited. For example, if all threads compute a single bit (a predicate) then it is not known whether it is possible to compute the product of the output bits of all threads (which is 1 if and only if all threads output 1). If this would be possible, then quantum computers would be able to solve $\mathbf{NP}$-complete problems in polynomial time.

One problem that quantum computers can efficiently solve (by using Shor's quantum algorithm, see Appendix A or [159]) is finding a *period* of a function. By a *period* of a function f we mean a positive integer λ such that $f(x + \lambda) = f(x)$ for all x. For example, if for the RSA public key encryption function $E(m) = m^e \bmod n$ we know the period of the (efficiently computable) function $f(x) = a^x \bmod n$, which with high probability (if a has been chosen randomly from the interval $(0...n)$) equals to the so-called Carmichael function $\lambda(n)$, then we can compute the RSA secret exponent $d = \frac{1}{e} \bmod \lambda(n)$. This means that the RSA cryptosystem can be efficiently broken with quantum computers. For similar reasons, also the elliptic curve cryptosystems can be broken.

For some cryptosystems there are no known relations between finding the private key and finding a period of a public function. Therefore, for breaking such cryptosystems, it is unknown whether quantum algorithms would give any advantage over the classical ones.

8.2 What is post-quantum cryptography?

Post-quantum cryptography is a common name for all cryptosystems that are resistant to attacks supported by quantum computers. The security of today's widely used public key cryptosystems mostly reduces to the computational hardness of the factoring or the discrete logarithm problems that both are breakable by quantum algorithms. From the very first years of public key cryptography, a few cryptographic algorithms have been known that are believed to be quantum-safe today. One of them is the Lamport's digital signature scheme. From the year 2005 onwards, cryptographers have been searching for new quantum-resistant algorithms systematically.

Post-quantum cryptography studies four types of cryptographic algorithms:

- lattice-based,
- multi-variate,
- hash-based,
- code-based.

Ordinary block-ciphers (like AES) are also believed to be quantum-safe but for some reasons block-ciphers are not considered to belong to the area of post-quantum cryptography.

8.3 Lattice-based cryptography

Lattices were first studied by Joseph Louis Lagrange ja Carl Friedrich Gauss. In cryptography, lattices have been used, for example, to break the pseudo-random generators that are based on numerical congruences. In 1996, Miklós Ajtai [33] showed first time that lattices can also be used to create new cryptosystems. In 2009, Craig Gentry [100] used lattices to construct the first fully homomorphic cryptosystem.

By a *lattice*[7] we mean a discrete subset of the vectors (points) in the Euclidean space $\mathbb{R}^n$, that is closed under addition and subtraction of vectors. We say that the lattice has dimension n if it is not contained in any proper (lower-dimensional) subspace of $\mathbb{R}^n$. Visually, a lattice is a set of points placed regularly all over the whole space. As an algebraic system, a lattice is a finitely generated free Abelian group.

By a *basis* of a lattice L we mean a set B of vectors such that any point (vector) of L can be expressed as a linear combination (with integer coefficients) of the elements of B. If the dimension of the lattice is at least 2, there are always infinitely many different bases in the lattice. In cryptography, the plaintext, the ciphertext and the key must be of finite size and therefore, the lattices in cryptography are not the subsets of $\mathbb{R}^n$ but K^n, where K is a finite field.

The security of lattice-based cryptosystems reduces to the following two hard combinatorial problems:

[7]Lattice also has another meaning in mathematics that is related to partially ordered sets.

- *Shortest vector problem*: Find the shortest vector in a lattice L (represented by a basis B).
- *Closest vector problem*: Given a basis B of a lattice L and a lattice-vector $v \notin L$, find a vector $v' \in L$ that is closest to v.

These problems are believed to be hard in general case, i.e. for majority of bases B. If the basis vectors are short and almost orthogonal, then both problems become efficiently solvable. Finding such a basis in a lattice is also called *lattice basis reduction* and the most famous algorithm for the reduction is the Lenstra–Lenstra–Lovász (LLL) algorithm [126].

Based on the security arguments, the lattice-based cryptosystems are divided into two groups:

- Efficient, but not provably secure. Their efficiency is comparable to the best known algorithms. For example, the NTRU algorithm belongs to this group.
- Provably secure, but less efficient. For example the algorithms that are based on the *Learning with Errors* (LWE) combinatorial problem.

The learning with errors in rings (Ring-LWE) is a combinatorial problem that enables to construct cryptosystems that are relatively efficient and provably secure.

8.4 Multi-variate cryptography

It has been proven that the problem of solving multi-variate algebraic equations is **NP**-hard or **NP**-complete and theoreticians think that quantum computers cannot solve **NP**-complete problems in polynomial time [92]. Hence, the cryptosystems the security of which is based on the hardness of solving multi-variate equations are suitable candidates for post-quantum cryptography.

To date, only multi-variate *signature schemes* are known such as *Unbalanced Oil and Vinegar (UOV)* [116], *Hidden Field Equations (HFE)*, *Hidden Field Equation Vinegar (HFEv)* [147] and *Rainbow* [85]. These signature schemes are computationally efficient and are suitable for using in the devices of low computational power (e.g. smart-cards), but require relatively large keys (a few kilobytes).

Almost all multi-variate *encryption schemes* have been broken. For example, the Imai-Matsumoto encryption scheme that was presented in 1988 was broken by Patarin in 1995. Patarin's own encryption scheme *Little Dragon* [146] that was proposed in 1995 was broken by Coppersmith and Patarin himself in 1996 [118].

The combinatorial multi-variate problem used in cryptography is solving systems of algebraic equations over finite fields such as $\mathbb{Z}_2 = \{0, 1\}$ where the following systems of equations has to be solved:

$$\begin{array}{rcl} P_1(x_1, x_2, \ldots, x_{2n}) & = & y_1 \\ P_2(x_1, x_2, \ldots, x_{2n}) & = & y_2 \\ & \cdots & \\ P_n(x_1, x_2, \ldots, x_{2n}) & = & y_n \;, \end{array} \tag{1}$$

where $P_1, \ldots, P_n \in \mathbb{Z}[x_1, \ldots, x_{2n}]$ are polynomials of degree not higher than two.

A signature scheme that is based on this problem may use the descriptions of the polynomials $P_1, \ldots, P_n$ as the public key. The signature of a message M has $3n$ bits and consists of a $2n$-bit representation of the values of $x_1, \ldots, x_{2n}$ and of an n-bit random number r, such that:

$$H(r, M) = P_1(x_1, \ldots, x_{2n}) \| \ldots \| P_n(x_1, \ldots, x_{2n}) \ ,$$

where H is a hash function with n-bit output. To sign M, one has to compute the hash $H(r, M) = y_1 y_2 \ldots y_n$ using the random number r and then solve the system (1). The polynomials P_i are chosen in a way that once we know their secret structure – the so-called HFE (*Hidden Field Equation*) structure – the system is easy to solve, but there is no efficient ways known for discovering such a structure from the public key and from the already created signatures. Such a signature scheme was proposed by Patarin in 1996 [147] and it is still considered secure.

8.5 Hash-based cryptography

The hash-based signature schemes of Lamport and Merkle are among the very first signature schemes that were proposed in late 1970s. Hash-based signature schemes have been studied for a long time. They are considered to be secure assuming that the corresponding hash function is secure. The main drawback of most hash-based signature schemes is the limited number of possible signatures.

For signing one single bit b with the Lamport's signature scheme we need a private key that consists of two n-bit (pseudo-) random numbers k_0 and k_1. The public key is a pair $f(k_0), f(k_1)$, where f is a one-way function (such as a hash function). To sign $b \in \{0, 1\}$ the signer publishes the corresponding half k_b of the private key and deletes the other half k_{1-b}. One key can only be used for signing a single bit.

For signing a longer message M one needs a hash function H with n-bit output and n (for example, $n = 256$) private keys $(k_0^1, k_1^1), \ldots, (k_0^n, k_1^n)$. The message M is first hashed and then every bit y_i of the hash $y = y_1 y_2 \ldots y_n = H(M)$ is signed with the key (k_0^i, k_1^i) as described above. The size of the signature is $n^2 = 256 \cdot 256 = 65536$ bits (about 8 kilobytes). The size of the public key is 16 kilobytes.

Pseudorandom generators can be used to decrease the size of the private key so that all components k_j^i are computed from one single random seed.

Merkle's signature scheme is an improved version of the Lamport's signature scheme that, in order to reduce the size of the public key, uses a hash tree the root hash of which is the public key. Merkle's signature scheme enables to sign many messages with the same public key, although their number is limited. In Merkle's signature scheme that enables to sign m messages, the signature size is $n^2 + n \log_2 m$ bits which is not much larger than in the Lamport's one way signature scheme. For example, if $m = 1024$ and $n = 256$, then the size of the signature is $256 \cdot (256 + 10)$ bits (about 8.5 kilobytes). The security of the Merkle's signature scheme is based on the collision-freeness of the hash function.

In 2011, a modification of the Merkle's signature scheme, the so-called XMSS, was proposed [66] in which for $n = 256$ the size of a signature has been reduced to about 1.8 kilobytes.

In 2014, a different modification, the so-called BLT, of the Merkle's signature scheme was proposed by Buldas, Laanoja and Truu [69, 67, 68], where the signature size can be considerably reduced. A private key for signing a single message consists of just one n-bit hash, but signing is only possible in cooperation with a server. To sign a message M at time t one needs a special key k_t that is intended for signing messages only at time t. The signer first computes the hash $H(M)$ and sends the server a signing query $Q(H(M), k_t)$ that combines the hash $H(M)$ and the corresponding private key k_t. The server sends back a time stamp $S_T(Q(H(M), k_t))$. The signature is valid only if the time T in the time stamp coincides with the intended usage-time t of the private key k_t. Hence, for obtaining a correct signature requires synchronized clocks at least with one-second precision. The size of the signature, if $m \approx 2^{25}$ (the number of seconds per year) is $25 \cdot 256$ bits (about 800 bytes). Probably, the collision-freeness assumption is insufficient for proving the security of the BLT scheme, and one has to use stronger assumptions about the hash function.

The SPHINCS signature scheme [49] is the first hash-based signature scheme that is stateless, i.e. there is no need to keep track on the number of already created signatures (i.e. to manage the state of the private key) not to communicate with the server. SPHINCS uses private and public keys of size about 1 kilobyte, but the size of the signature is about 42 kilobytes. The security of the scheme is based on the collision-freeness without additional assumptions.

8.6 Code-based cryptography

Code-based cryptography uses the properties of error-correction codes. There exist both code-based encryption schemes [132] and code-based signature schemes [142].

The combinatorial problem behind the McElice's encryption scheme is the decoding problem of a general liner code, i.e. the so-called *closest code-word* problem. By a *linear code* we mean a set C of vectors is an n-dimensional vector space over the finite field $\mathbb{F}_q$, such that C is closed under addition and scalar multiplication. This also means that C is a k-dimensional subspace of the vector space $\mathbb{F}_q^n$. The dimension k of C is called the *rank* of C. The closest code-word problem is to find the closest (in terms of the Hamming distance) vector in C for a given vector x in $\mathbb{F}_q^n$. It has been proven that the most general version of this problem is $\mathbf{NP}$-hard. However, for many special cases and for limited errors, this problem is efficiently solvable, for example, if the Hamming distance of x from C does not exceed half of the *minimal distance* of the code (i.e. the minimal Hamming distance between the elements of C).

The private key in this scheme is a randomly chosen linear code, for which there is an efficient decoding algorithm. In the original version of McElice's cryptosystem, binary Goppa codes are used that are efficiently decodable via the Patterson algorithm and that are capable of correcting up to t-bit errors. The public key is obtained with masking the code with general linear code. If G is the generating matrix of the code (the row-vectors of which form the basis of the code as a subspace), then the public key G' is obtained by using randomly chosen invertible matrices S and P as follows:

$$G' = S \cdot G \cdot P \ ,$$

where $\cdot$ means matrix multiplication. Also, P is a permutation matrix each row and column of which have exactly one 1 and the rest of the elements are equal to 0.

To encrypt a message m with McElice's cryptosystem, one first encodes m as a t-bit binary string, then computes a vector $c' = mG'$, generates an n-bit random vector z with exactly t components equal to 1 and the rest of the components equal to 0, and creates a ciphertext $c = c' + z$, which is a codeword with a t-bit random error.

To decrypt the ciphertext one first computes $c' = cP^{-1}$ by using the inverse matrix P^{-1} of P, decodes c' by finding m' and the plaintext $m = m'S^{-1}$. Decryption is correct because

$$c' = cP^{-1} = mG'P^{-1} + zP^{-1} = mSG + zP^{-1} \quad ,$$

mSG is a code-word and the Hamming norm of the vector zP^{-1} does not exceed t (because P is a permutation matrix).

McElice's cryptosystem is very efficient in terms of key generation, encryption and decryption speeds, but its main drawback is a very large public key. For example, for gaining 2^{128}-security, one has to use public keys of size about 100 kilobytes.

8.7 Standards and implementations

The only post-quantum cryptosystem that has made it successfully to major standards is NTRU [4, 7]. It is also the one with the most advanced practical implementations [8] [9]. However, the most widely used cryptographic libraries (such as OpenSSL and Bouncy Castle) do not support post-quantum cryptographic algorithms.

European Telecommunications Standards Institute (ETSI) has published a set of standards concerning quantum key distribution [8]. In 2015, ETSI also launched a Quantum Safe Cryptography specification group [10] which has produced a whitepaper report [19].

[8] `http://tbuktu.github.io/ntru/`

[9] `https://github.com/NTRUOpenSourceProject/ntru-crypto`

[10] ETSI launches Quantum Safe Cryptography specification group, `http://www.etsi.org/news-events/news/947-2015-03-news-etsi-launches-quantum-safe-cryptography-specification-group`

9 Cryptographic protocols over radio connection

There are two large classes of radio frequency cards used in practical identification applications (door access, public transportation, customer loyalty, etc.).

Simpler ones, operating on 125kHz frequency range (e.g. EM Unique, HID Prox and INDALA), do not support any cryptographic capabilities like encryption or challenge-response authentication. Identification is performed by simply transmitting a pre-coded signal over the radio connection [157]. Consequently, the entity authentication solutions built on top of such cards are subjects to trivial cloning and replay attacks – these solutions should not be deployed in environments where cloning resistance is a desired objective.

Cards operating on a 13.56GHz frequency range (also called *Near-Field Communication*, or NFC cards) can be made to implement more sophisticated protocols. In this chapter we will take a look at several examples of this class of cards.

9.1 MIFARE cards

One of the first widely deployed cards in this family was MIFARE Classic, used e.g. in London's Oyster card, Netherland's OV-Chipcard, US Boston's CharlieCard, and in numerous wireless access control and ticketing systems worldwide [73].

MIFARE Classic implemented an ISO 9798-2-inspired mutual authentication protocol relying on a proprietary CRYPTO1 stream cipher [98]. After reverse engineering, the pseudorandom bit string generated by the cipher turned out to be weak, allowing for various exploits [120, 73, 98]. As a result, MIFARE Classic should be avoided in any deployments.

There are also MIFARE cards offering stronger cryptographic algorithms; for example DESFire EV1 supporting AES-128 [112]. AES is used as the main building block in a protocol allowing to mutually authenticate the card and the reader, and to create a session key for further communication. See Section 9.2 for a more detailed description of the protocol.

As AES is a symmetric algorithm, the deployment of such cards in a large scale authentication infrastructure project presumes that the same private symmetric key is shared between many cards. This is a concern if physical security measures of the card are circumvented and the secret key is extracted from one card. In case of an earlier MIFARE DESFire card MF3ICD40, this has been shown to be possible with side-channel attacks using relatively low-cost hardware by Oswald *et al.* [144].

NXP, the company behind the MIFARE product line, has claimed that the EV1 card is Common Criteria EAL 4+ certified and that the attack by Oswald *et al.* failed for EV1.[11] However, Kasper *et al.* claim to have also a successful side-channel attack against EV1 [113].

In 2013, NXP announced the release of MIFARE DESFire EV2 generation of cards.[12] The need to share the same master key among all the users of all the services has been addressed by introducing the option of using different keys for different services. However, the only cryptographic algorithms supported by EV2 are still symmetric ones (AES, 3DES and DES). This means that all the communicating components must share the same key. Hence, the problem of extracting the key from any of them and reusing it for card cloning still remains.

A systematic way of solving this problem would be using asymmetric cryptography and public key infrastructure, but unfortunately this solution is considerably more resource-consuming.

9.2 MIFARE EV1 and Ultralight C authentication protocol

MIFARE Ultralight C was designed to replace the Classic product line in the low-end (even disposable) tag applications. Still, it provides a cryptographic authentication mechanism similar to EV1.

The authentication protocol looks roughly as follows [112, 133]. In this protocol, e_k denotes the symmetric cryptographic cipher used (AES in case of EV1 and 3DES in case of Ultralight C). Note that we have omitted some technical details (in practice, some of the generated random values are also shifted on some steps).

1. Reader → Tag: Hello.
2. Tag generates a random number r_b.
3. Tag → Reader: $e_k(r_b)$.
4. Reader decrypts r_b and generates another random number r_a.
5. Reader → Tag: $e_k(r_a|r_b)$.
6. Tag decrypts the last message and extracts r_a.
7. Tag → Reader: $e_k(r_a)$.

In the end of the protocol, both participants are expected to be convinced that the other party also knows the shared key k.

This protocol relies on the tag's ability to generate random numbers. Merhi *et al.* analysed the quality of the random number generator on Ultralight C and concluded that it is much better than the one used for MIFARE Classic [133]. They used standard randomness tests on the bit stream generated by the tag, and it passed all the tests.

Hence the attacks making use of weak pseudorandomness are less likely to succeed for Utralight C. However, we advise against the usage of the deprecated 3DES algorithm (especially since it seems to be used in 2 key setup [133], giving the efficient keylength of 112 bits).

[11]https://www.mifare.net/en/update-on-mifare-desfire-mf3icd40/

[12]https://www.mifare.net/en/products/chip-card-ics/mifare-desfire/mifare-desfire-ev2/

9.3 HID iClass

iClass by HID Global is another popular line of NFC cards working on the 13.56 MHz frequency range. iClass cards come in two varieties – Standard and High Security (the latter being also marketed as Elite).

All Standard Security cards and readers carry the same symmetric master key which is used in combination with some card- and facility-specific values to produce a diversified key. Unfortunately, the proprietary key diversification algorithm turned out to be both weak and poorly implemented, allowing for the master key to be extracted from either the cards or readers in several different ways [134, 96, 97].

After the first key recovery attacks surfaced, HID responded by modifying future models of card readers so that the firmware stored inside them could not be so easily dumped or read. However, the master key itself was not changed, making it still easy to clone the cards once the key has been recovered [122].

In High Security mode, it is possible to install a different master key for different facilities. However, key recovery is still possible and may actually be implemented more efficiently than for the Standard Security cards [97].

9.4 HID INDALA

INDALA is another popular product line by HID Global. It works on the 125kHz frequency range, and, as noted above, such cards typically do not provide any cryptographic functionality. HID INDALA is a little bit different, since it is marketed as supporting proprietary FlexSecur® encryption technology [3].

After relatively simple reverse engineering, an earlier version of this technology (known as ASP) turned out to be a simple bit rearrangement [130]. HID rolled out an updated version (known as ASP+) and claims it to be "more secure than scrambling technology used on legacy Indala ASP" [3].

However, details of the cryptographic algorithm used are still not provided. In fact, from the card copying attack point of view, these are even irrelevant, since the card itself still does not perform any computations. A constant cryptogram is written onto the card, essentially becoming a new password that can be copied and replayed easily.

It is true that this sort of a solution can be utilised to prevent accidental cross-usage of access card between different facilities (assuming these facilities use different encryption keys). However, we get a problem when, referring to this extra feature, the product is marketed as being generally secure.

The access card solution providers should be encouraged to make more precise security claims, including statements about which attacks their system does *not* counter. Also, we do not approve using proprietary encryption methods, since too many of them have been broken in the past using only a moderate amount of cryptanalytic effort.

9.5 Estonian public transportation cards

In several Estonian municipalities, public transportation cards are using NFC tags as an underlying technology. For example, the transportation cards in Tallinn are built on MIFARE Classic, whereas in Tartu MIFARE Ultralight C cards are used. However, even though both of the cards support cryptographic authentication (see Section 9.2), this functionality is not used.

In both cases, the protocol running between the card and the reader is essentially the same, consisting of transmitting the card's unique ID and a signature (RSA1024 in Tallinn and ECDSA on curve P-192 in Tartu). However, since the cards are not capable of performing asymmetric cryptography operations, the signature is pre-generated during the card initiation process, and stays constant throughout different protocol runs.

While this measure prevents unauthorised parties from issuing new cards, it does not stop the card cloning attack. The only measure against the latter is the difficulty of creating an NFC tag with a cloned ID value.

One has to take into account that the unique ID of an NFC tag/card was never designed to be a security feature, but rather an anti-collision mechanism in the case a reader has to distinguish between various tags that are presented to it at the same time. Even though off-the-shelf cheap tags do not support changing the ID, a bit more expensive ones do. Such NFC tags and the required writing equipment can easily be bought online.

When NFC-based public transportation cards were introduced in Estonia, their first application scenario was supporting free transportation for Tallinn citizens. Cloning a card that gives free ride anyway was probably not perceived as a great threat. Also, additional security measure of verifying the citizen's physical ID was planned.

However, other application scenarios emerged soon. Cloning a card that carries a monthly ticket causes direct financial loss to the transportation service provider and must hence be urgently addressed.

Even though the ID fields of transportation cards are not writeable, other fields may be. This is for example the case with Tartu bus cards that allow e.g. the signature field to be overwritten by a standard app working on a regular NFC-capable smartphone. As a result, the card will become invalid, giving us a potential Denial of Service attack. We want to emphasize that this attack can be implemented without any special equipment whatsoever.

We conclude that Estonian public transportation cards are vulnerable to various kinds of Denial of Service and cloning attacks. To avoid them, the cards should be non-writeable and use proper protocols. Basing security on the assumption of hardness of UID cloning is not a valid approach.

One possible obstacle preventing the use of a proper cryptographic authentication protocol is usability. The protocol presented in Section 9.2 requires four messages to be sent in two rounds between the tag and the reader. Both of the parties also have to perform cryptographic operations, which is especially time-sensitive for the tag.

As a result, the user experience with a card supporting a more secure protocol must change. The user can not simply wave the card in front of the reader, but must hold it there for a longer period of time. This may be unacceptable for the public transportation setting. We conclude that using NFC technology for public transportation cards may be a bad idea from the security viewpoint in general.

9.6 Deployment issues

Even if NFC cards/tags with cryptographic capabilities are used, this does not automatically mean secure deployment. Besides pseudo-cryptographic measures mentioned above, key management is another major issue.

As the performance of NFC tags is not yet sufficient to support asymmetric cryptography, mutual authentication based on symmetric keys (in the fashion of DESFire cards) is currently the best that one can achieve. However, one has to be aware of the associated weaknesses. In order to run the protocol, the reader must have access to the same symmetric key as the tag does, and this is already an assumption weakening the whole strength of authentication.

Even more, in practical deployments it is often the case that the same key is shared not only between the tag and the reader, but also among several tags, making it possible to defeat the purpose of authentication. Interview with a company installing NFC-based access control systems revealed that it is common practice to use same keys also in several installations, making e.g. door keys of one company work at the door of another company, too. This kind of practice should be strongly discouraged.

There are many reasons why access card systems using insecure protocols are deployed in practice. First, insecure systems are cheaper both to set up and maintain, as no complicated key management is required. Such a deployment also requires a lower technical skill level, thus being available to a wider range of installation service providers.

Second, the current practice of how access control systems are being acquired and integrated into infrastructure (e.g. office buildings), is principally flawed. Typically, an access control mechanism is required as a part of public tender. It would, in principle, be possible to include security requirements of such mechanisms as a part of these requirements. However, this is often omitted, since security is hard to measure, the person composing the tender does not have a sufficient level of competence, etc. Even if the access solution provider supplies some sort of a certificate that the solutions is secure, no-one guarantees that the solutions and cards deployed were deployed in the secure mode. Also, explicit security statements are often missing from the accompanying documentation.

Still, the only way to improve upon the current situation is to start from setting explicit security requirements to the newly tendered installations. In high security environments, post-installation auditing to verify the security claims made by the solution provider may also be required.

9.7 Transparency issues

A serious problem preventing auditing the access card systems is the proprietary nature of the cryptographic algorithms and protocols used in NFC applications. For example the description of the protocol used in MIFARE EV1 and Ultralight C cards (see Section 9.2) has been obtained from third-party documents and/or reverse engineering, but not from the supplier. The same holds true for HID iClass and INDALA cards, and actually all kinds of cards we studied for this report.

Even if some information about cryptographic features is available, it is typically too sparse or even controversial to draw any conclusions. For example, installation manual of ICT PRX-TSEC multi-technology card reader [29] states:

> Once you have selected the desired key slot you will be prompted to enter the 6 byte hexadecimal encryption key.

6 bytes (i.e. 48 bits) is far too little to provide a reasonable level of security in any application scenario. However, the manual does not even specify, in which scenario this sort of very weak encryption is used, what protocols are built upon it and what are the precise security goals that the system is designed to meet.

This sort of vagueness is a direct violation of what is known as the (second) Kerckhoffs's principle [115] – the system's security should rely only on the secrecy of the key, and not secrecy of the system architecture. The history of cryptography is full of attempts to violate this principle, and many of them have failed miserably.

This report strongly discourages such an approach. Instead, we suggest the following principles to be applied for future public tenders.

1. The tenders should explicitly specify desired security goals (e.g. hardness of cloning the cards/keys).
2. The providers should disclose all the measures they take to meet these goals (including the descriptions of cryptographic algorithms and protocols used).
3. Independent post-installation auditing may be required to establish that the installed solution actually corresponds to the disclosed specifications.

Unfortunately, many legacy systems have been built without taking these principles into account. Consequently, it is very hard to state or prove any security claims about them.

As the first step of improvement, security audits need to be conducted on critical legacy systems deploying RFID/NFC technology. For that, the suppliers must be mandated to reveal technical details of the installations (using legal actions, if necessary).

As the second step, solution providers must be required to improve the deployments (e.g. by rolling out firmware updates supporting more secure algorithms and protocols). If this is not possible, transitions to completely new systems must be initiated as soon as possible.

The canonical place to state security requirements for Estonian information systems is the baseline protection manual ISKE. Currently, the relevant measure M1.80w "Access control system and authorisation management" does not specify explicit security requirements.[13] It does refer to the standard DIN EN 50133-1 that was accepted in Estonia as EVS-EN 50133-1. However, the standard is now obsolete, and has been replaced by EVS-EN 60839-11-1:2013.

This standard defines four grades of risk for access control systems, ranging from low to high. Token-based access control systems do not have any special requirements for grades

[13] `https://iske.ria.ee/7_06/ISKE_kataloogid/7_Kataloog_M/M1/M_1.80`

1 and 2. However, on grade 3, encrypted data communication is mandated, and on grade 4 the token must support mutual authentication. Also, cryptographic protection is compulsory for card data manipulation operations.

We criticise such a state of affairs. Reference to an obsolete standard should be updated and the relevant requirements to the access control systems should be included in ISKE directly, connecting them with ISKE security levels.

10 Mobile-ID protocol analysis

The TLS handshake protocol has a number of variations, providing different kinds of guarantees to the client and the server about each other's identity. In the most common scenario, the server will be authenticated to the client, while the identity of the client will not become known to the server through cryptographic means (e.g. by using a client certificate). Subsequently, the client may present credentials (e.g. password) to the server using the secure channel that the TLS handshake protocol has established, thereby allowing the server to assume that the provided credentials have been provided indeed by a specific client and to authorize that user for that session. A variety of attacks apply to this scenario.

This is not the case for Estonian ID-card, where the client is ultimately identified to the server by the private key in the user's ID-card. During the years after ID-card launch and adaption, it became clear that an alternative identification and signing method is needed for an unlikely but theoretically possible scenario, where the ID-card infrastructure fails for some unforeseeable reason.

Since 2007, Estonian mobile operators have issued their customers mobile phone SIM cards that contain an USIM application which enables them to sign hashes presented sent to them via DigiDocService and OTA platform by using encrypted SMS messages [22]. The signing is done with either RSA or ECC private keys stored inside the SIM card trust zone; the private keys themselves never leave the SIM card. The Mobile-ID protocol allows a user having a personified Mobile-ID enabled SIM card, to identify oneself to a service provider, say a bank or a government institution, but also to sign documents digitally.

The *authentication procedure* is initiated by a service provider's server with a request to DigiDocService, which is ultimately the backend and backbone of the Mobile-ID service. DigiDocService forwards the server- and DigiDocService-generated challenge to the USIM application in the to-be-authenticated user's SIM card by a SMS message (through mobile service provider's network). This SIM card signs the challenge with the private key inside it and returns the hash by SMS message and through DigiDocService to the service provider's server.

The *digital signing procedure* is initiated by the signing application available from `https://installer.id.ee` or an application running in service provider's server. This application sends the hash of the to-be-signed document to DigiDocService, which then forwards that hash to the SIM card, where the USIM application prompts the user with the control code, computed from the received hash. If the user approves the control code (that is also shown by the signing application or service provider) and enters correct PIN, the USIM application signs the hash with according private key and returns the signature by SMS message and through DigiDocService to the service provider's server or the signing application running in user's computer.

Over the years, this protocol has been thoroughly analysed; this report is based on the results. While the Mobile-ID protocol is used also in other countries besides Estonia [10, 11], the current report focuses strictly on Estonian implementation. The conclusions for other deployments may be different than the ones given here.

10.1 Authentication with Mobile-ID

Ideally, the device containing the SIM card (usually user's phone) is different from the device (running a web browser) that's used to access the service (bank's website), which the user needs to be authenticated for. Such a design would ideally lower the amount of trust placed onto each single device that is part of the trusted computing base in that case. To securely authenticate the user, there must be means to transport the server-generated challenge to the phone, and means for the phone to verify that the user wanted to authenticate himself to that particular service.

The communication between the server and the phone is mediated by the DigiDocService web service and OTA platform [22], managed by AS Sertifitseerimiskeskus; the messages between OTA and the SIM card are encrypted using 3DES [24]. In the Mobile-ID protocol, the phone generates a four-digit *control code* from the received challenge, which was jointly generated by the server and DigiDocService. The server also sends the same control code to the client device. The user must then verifty that the four-digit control codes match. If these do match, the user allows the phone to sign the challenge, by unlocking that functionality of the USIM application with the PIN associated with the needed private key. The signature is sent back to server via DigiDocService. The entire protocol is described in [22].

In 2009, Laud and Roos analysed the Mobile-ID protocol for its security against network-based attacks [124]. No large changes (e.g. changing the contents of messages) have been made to the underlying protocol as of 2016 [24], hence the inferred results should still be relevant. The following issues were found in the protocol [124].

- When the USIM application signs the challenge, then indeed only the challenge is signed. The prudent engineering practice for cryptographic protocols [31] suggests to include the name of the verifier under the signed message, if this signature is intended for a particular verifying party. This reduces the chance of successful man-in-the-middle attacks.
- One part of the challenge is generated by DigiDocService. As the four-digit control code has only 10,000 possible values, DigiDocService may cause collisions in them.
- The control code shown in the client device is (in some use cases) computed solely by DigiDocService, and not with the server.

Each of these weaknesses allow DigiDocService to perform man-in-the-middle attacks. I.e., the maintainers of DigiDocService are able to masquerade as other users to servers. Put otherwise, the DigiDocService is part of the trusted computing base for authentication with Mobile-ID. Hence it is also important for service providers to carefully verify the identity of DigiDocService when establishing TLS sessions and exchanging messages with DigiDocService. If all the weaknesses listed above were corrected, then the DigiDocService would stay out of the trusted computed base for user authentication.

These weaknesses do not extend to the attacks performed by a malicious mobile operator or through a man-in-the-middle attack: as the messages between DigiDocService and the SIM card are encrypted [24], the attacker has no way to change them undetectably.

We have no better suggestions for the removal of weaknesses than the introduction of necessary changes to the protocol. However, this would break interoperability with the already deployed SIM cards.

Our analysis has also shown that the Mobile-ID protocol is as vulnerable to man-in-the-middle (between the client device and service provider's server) attacks as password-based authentication. This may become significant in scenarios where the user does not carefully determine the identity of the service provider's server. The authentication using the ID card does not have this type of vulnerabilities.

10.2 Digitally signing documents with Mobile-ID

Similarly to Estonian ID-cards, the Mobile-ID SIM cards contain separate private keys for authentication and signing. Signing protocol is simpler than the authentication protocol, as there is no server depending on the outcome. In signing protocol, the signing application sends the hash(es) of the to-be-signed document to DigiDocService, which calculates the hash for signing (according to the XAdES specification [6]) and sends it via the OTA platform to the SIM card, where the USIM application shows the control code, computed from that hash. If the user approves the control code (that is also shown by the signing application) and enters the correct PIN, the USIM application signs the hash and sends the signature back to the DigiDocService, which forwards it to the signing application.

Obviously, DigiDocService can change the hash that is going to be signed. It is easy to change it without changing the control code, because these have only 10,000 possible values, meaning that collisions are easy to find. Such changes cannot be detected by the signer before the hash has been signed by the USIM application. Hence DigiDocService belongs to the trusted computing base when signing with Mobile-ID. To detect such attacks, the signer should verify the constructed signature after the signing.

In the current Estonian context, the deployment details significantly reduce the impact of the weaknesses of both the authentication and signing protocols — the DigiDocService is operated by the same entity that issues the certificates for authentication and signing, identifying the users of Mobile-ID. This entity is a part of the trusted computing base anyway, and the additional attacks it could perform through DigiDocService give it no additional power.

10.3 Other security considerations

While [124] considered only the network security of the Mobile-ID protocol, there are other important technical and organisational details of the supporting infrastructure to consider. A list of these details has been discussed in [167], mostly in the form of issues that should be more thoroughly considered. The next few paragraphs give an overview of those issues.

There are important technical aspects surrounding the SIM card and its keys, and the USIM application on it. The private keys on the card have to be generated securely. A good source of randomness must be employed for this generation, and the generation process itself is rather resource-intensive (at least for certain cryptographic primitives). If such randomness source is unavailable inside the card, or the card is computationally too weak to generate the keys itself, then these keys have to be generated outside of the card. The generated keys have to be loaded onto the card and the path from the generation place to the card must be protected through organisational means. Similar issues pertain to the generation of PINs protecting the use of private keys, and their transport to the user.

The Mobile-ID application on the SIM card is responsible for receiving the challenge (or the hash of a document to be signed), checking the PIN and actually computing the signature. This USIM application is a security-critical component of the whole Mobile-ID system. Hence it should be thoroughly tested, its code quality should be evaluated, not only for functionality, but also for security and the absence of possible side-channel attacks on that specific hardware/software combination. To start with, the application should be well-specified and its code should be checked against the specification.

The organisational security methods around the Mobile-ID platform have to be carefully selected. The questions pertaining to the procurement of chips, the operating system and the application, as well as the personalisation and storage of the chips may be similar to the ID-card infrastructure or to the normal operations of a mobile telecommunications provider. Hence the same organizational methods may be applicable for securing all these processes. But the issue of managing secure channels between the DigiDocService and the mobile operators is unique to Mobile-ID.

The questions listed in previous paragraphs on the basis of [167] have been considered during the intervening years [24] and deemed to have been answered in satisfactory manner [16]. The generation of the keys for the SIM card is regulated in [5, Sec. 6.1.2], requiring the SIM card producer to give assurance to the mobile operator that the keys have been generated according to best practice and are unique. The protection of PINs is similarly regulated. The attacks against a specific person are also made harder by the SIM card being bound to a person only after it has been issued [23]. The current USIM application on the SIM card has been specified and is currently being tested whether it matches the specification and satisfies the security requirements.

Certain security claims of Mobile-ID, particularly about the lack of a single point of failure, are based on the independence of the client device and the phone. Such scenario may have been the case in 2008 when Mobile-ID was introduced and used with “feature phones”, but nowadays a smartphone or a tablet often fulfils both roles, enabling the service providers to lure users to use apps [25, 26, 27], that allow the user to access a service, say a user’s bank account on the same device that the user uses for authentication and digital signing. Such convergence was already warned against in [167].

We conclude that while the network security aspects of the Mobile-ID protocol have not significantly changed from the time it was analysed, the applied technical and organisational security measures have significantly evolved and become more concrete. Also, the usage patterns of Mobile-ID have changed and may still be changing [170]. In particular, the boundaries between devices may be dissolving, especially if the same cloud-based services are used from both the computer and the phone. We suggest that the appropriate Estonian government institutions analyse Mobile-ID security measures to check how they hold today, as the typical Mobile-ID use cases have changed since the last thorough analysis of Mobile-ID protocol. While financial or other private institutions may accept their risks or lower these with insurance, government institutions do not always have that luxury.

11 Conclusions and recommendations

After publication of the previous, 2015 version of the report, no major cryptanalytic breakthroughs have occurred. This means that the recommendations given in [18] are still valid. We will summarise them here as well.

- As the standard choice for symmetric encryption, AES block cipher is recommended. In mid-term perspective (up to 10 years), all the standard key lengths (128, 192 and 256 bits) may be used. For long-term security (30-50 years), AES-256 is recommended. Camellia cipher can also be considered secure. However, RC4, DES and 3DES are obsolete and their usage should be terminated.
- In case of RSA and discrete-logarithm-based systems (like Diffie-Hellman key exchange, ElGamal and DSA), usage of 1024-bit keys should be stopped urgently. Existing deployments of 2048-bit keys may be continued for 5 years. New installations and installations requiring mid-term security should use at least 3072-bit keys.
- Usage of hash functions MD5 and SHA-1 should be discontinued as soon as possible. As replacements, SHA-2 and SHA-3 families of hash functions are recommended. New installations should avoid implementing SHA-224.
- After the NSA announcement about Suite B recommendation updates, the situation with elliptic key cryptosystems is unclear. We estimate that implementation of a practical quantum computer capable of breaking current asymmetric encryption systems is still more than 5 years away. Hence, using standard elliptic curves (like P-256 and P-384) is fine for at least this time period.

These recommendations should be taken into account when deciding upon the TLS cipher suite. Additionally, one has to select the key exchange algorithm and block cipher mode of operation.

To provide forward security, ephemeral Diffie-Hellman key exchange should be used; the corresponding cipher suites have either DHE or EDH in their names.

As classical block cipher modes of operation do not provide integrity assurance, message authentication codes must be deployed together with them. TLS version 1.2 provides an additional GCM mode that also adds an integrity mechanism to encryption. There are several implementation caveats of the GCM mode. For example, Böck *et al.* have recently shown how to attack it in case of nonce reuse [57]. However, their findings also show that recent versions of OpenSSL have been properly patched against this vulnerability.

In the upcoming TLS version 1.3, both non-ephemeral key exchange and non-authenticated block cipher modes are planned to be disabled [154]. We recommend implementing these updates already today, by selecting for example one of the following cipher suites:

- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`,
- `TLS_DHE_RSA_WITH_AES_256_GCM_SHA384`.

During deployment, one may discover that not all the devices and applications support these cipher suites. In this case the above recommendations should be taken into account in the largest possible extent that still provides the required level of interoperability. In any case, usage of weak cryptographic algorithms (like RC4, DES and MD5) should be prohibited. An interested reader can find many practical tips for secure system configuration of various applications in the Applied Crypto Hardening report [65].

An important aspect to take into account when building security-critical systems is modularity of the cryptographic primitives used. This is especially important in the light of the upcoming transition to post-quantum algorithms. Search for the next generation of cryptographic primitives has only started and no-one can really know how long they will last. This means that the only way to ensure continuous security of the systems is to make the primitives easy to change.

However, one has to keep in mind that no currently available practical cryptographic technique guarantees security for a very long time span. If privacy is required for, say 70 years (which it is for some document types), cryptographic measures alone are not sufficient, and additional physical and organisational measures need to be applied.

Another general recommendation is to use the most up-to date versions of cryptographic libraries like OpenSSL, BoringSSL or BouncyCastle to prevent implementation-level vulnerabilities from being misused.

When planning for critical systems and infrastructure, security requirements must already be addressed in the public tender phase. Explicit security requirements should be stated in the call and the providers should open the technical specifications of their proposals to the extent that will allow for independent post-installation auditing. The corresponding requirements should also be included into the Estonian baseline security framework ISKE.

During the reporting period (2015-16), the most significant advancements have been made in cryptanalysis of hash functions. Finding a collision of SHA-1 has become almost accessible, hence the recommendation to phase out its usage as soon as possible.

At the same time a significant amount of resources have been invested into creating a sufficiently large quantum computer that would break almost all of the currently deployed asymmetric algorithms. However, there are still major physical obstacles to overcome (like a relatively short life-time of qubits). Unless NSA knows something the rest of the world does not, a sufficiently powerful quantum computer still seems at least 10 years away.

Bibliography

[1] PKCS #1: RSA Encryption Standard, versioon 1.5, 1. november 1993. http://www.rsa.com/rsalabs/node.asp?id=2125.

[2] PKCS #1: RSA Encryption Standard, versioon 2.0, 1. september 1999. http://www.rsa.com/rsalabs/node.asp?id=2125.

[3] HID Indala FlexSecur® Technology, 2006. https://www.hidglobal.com/sites/default/files/indala-flexsecur-wp-en.pdf.

[4] IEEE Standard Specification for Public Key Cryptographic Techniques Based on Hard Problems over Lattices. *IEEE Std 1363.1-2008*, pages C1–69, March 2009.

[5] Mobiil-ID sertifitseerimispoliitika, December 16th 2009. Version 2.0.

[6] XML Advanced Electronic Signatures (XAdES), June 2009. ETSI TS 101 903 v1.4.1, http://webapp.etsi.org/WorkProgram/Report_WorkItem.asp?WKI_ID=28064.

[7] Lattice-based polynomial public key establishment algorithm for the financial services industry, 2010. ANSI X9.98-2010.

[8] Quantum Key Distribution (QKD), 2010. ETSI GS QKD family of standards, version 1.1.1.

[9] 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).

[10] Changes related to DigiDocService web service, 2013. https://www.sk.ee/en/News/dds-en (checked 22.06.2016).

[11] EMT Helps to Launch Mobile ID in Azerbaijan, 2013. EMTHelpstoLaunchMobileIDinAzerbaijan (checked 22.06.2016).

[12] 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).

[13] OpenSSL TLS protocol downgrade attack (CVE-2014-3511). OpenSSL Security Advisory [6 Aug 2014], August 2014. https://www.openssl.org/news/secadv_20140806.txt.

[14] The Heartbleed Bug, 2014. http://heartbleed.com.

[15] Vulnerability in Schannel Could Allow Remote Code Execution (2992611). Microsoft Security Bulletin MS14-066, November 2014. https://technet.microsoft.com/library/security/ms14-066.

[16] Audit Attestation 2015 for the CA- and TSA- Management System of SK, 2015. Available from http://sr.riik.ee/en/sk.html.

[17] Cryptography today, August 2015. https://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

[18] 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).

[19] Quantum safe cryptography and security. etsi white paper no. 8, June 2015. http://www.etsi.org/images/files/ETSIWhitePapers/QuantumSafeWhitepaper.pdf.

[20] SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions, August 2015. FIPS PUB 202, http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

[21] Commercial National Security Algorithm Suite and Quantum Computing FAQ, January 2016. https://www.iad.gov:8443/iad/library/ia-guidance/ia-solutions-for-classified/algorithm-guidance/cnsa-suite-and-quantum-computing-faq.cfm.

[22] DigiDocService specification, versioon 3.9.1, January 22nd 2016. http://www.sk.ee/upload/files/DigiDocService_spec_est.pdf.

[23] How to apply for Mobiil-ID?, 2016. http://www.id.ee/index.php?id=36913.

[24] Interviews with specialists from AS Sertifitseerimiskeskus, May–June 2016.

[25] Mobiiilipank, 2016. https://www.lhv.ee/en/mobile-bank/?l3=en (checked 20.06.2016).

[26] Mobile Bank, 2016. http://www.seb.ee/eng/everyday-banking/service-channels/mobile-bank (checked 20.06.2016).

[27] Mobile bank, 2016. https://www.swedbank.ee/private/d2d/mobile/mobile (checked 20.06.2016).

[28] National Income and Product Accounts. Gross Domestic Product: Fourth Quarter and Annual 2015 (Third Estimate). Corporate Profits: Fourth Quarter and Annual 2015, March 2016. Bureau of Economic Analysis, http://www.bea.gov/newsreleases/national/gdp/2016/gdp4q15_3rd.htm.

[29] PRX-TSEC tSec Multi-Technology Card Reader Installation Manual, May 2016.

[30] SK time-stamping service to restrict the use of SHA-1 and SHA-224, January 2016. https://sk.ee/en/News/sk-time-stamping-service-to-restrict-the-use-of-sha-1-and-sha-224/.

[31] Martín Abadi and Roger M. Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[32] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. Imperfect forward secrecy: How diffie-hellman fails in practice. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 5–17, New York, NY, USA, 2015. ACM.

[33] Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 99–108, New York, NY, USA, 1996. ACM.

[34] Nadhem AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. On the security of rc4 in tls. In *Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13)*, pages 305–320, Washington, D.C., 2013. USENIX.

[35] Nadhem J. AlFardan and Kenneth G. Paterson. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, pages 526–540. IEEE Computer Society, 2013.

[36] David W. Archer, Dan Bogdanov, Benny Pinkas, and Pille Pullonen. Maturity and performance of programmable secure computation. Cryptology ePrint Archive, Report 2015/1039, 2015. http://eprint.iacr.org/.

[37] Sanjeev Arora and Boaz Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[38] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. DROWN: Breaking TLS using SSL v2. http://drownattack.com, March 1st 2016.

[39] 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.

[40] Gregory V. Bard. The Vulnerability of SSL to Chosen Plaintext Attack. Cryptology ePrint Archive, Report 2004/111, 2004. http://eprint.iacr.org/.

[41] Gregory V. Bard. A Challenging but Feasible Blockwise-Adaptive Chosen-Plaintext Attack on SSL. In Manu Malek, Eduardo Fernández-Medina, and Javier Hernando, editors, *SECRYPT 2006, Proceedings of the International Conference on Security and Cryptography, Setúbal, Portugal, August 7-10, 2006, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 99–109. INSTICC Press, 2006.

[42] Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, and Joe-Kai Tsay. Efficient padding oracle attacks on cryptographic hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 608–625, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[43] R. Barends, J. Kelly, A. Megrant, D. Sank, E. Jeffrey, Y. Chen, Y. Yin, B. Chiaro, J. Mutus, C. Neill, P. O'Malley, P. Roushan, J. Wenner, T. C. White, A. N. Cleland, and John M. Martinis. Coherent Josephson Qubit Suitable for Scalable Quantum Integrated Circuits. *Phys. Rev. Lett.*, 111:080502, Aug 2013.

[44] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols. In *Proceedings of the Twenty-second Annual ACM Symposium on Theory of Computing*, STOC '90, pages 503–513, New York, NY, USA, 1990. ACM.

[45] Tal Be'ery and Amichai Shulman. A Perfect CRIME? Only TIME will tell. Technical report, Black Hat Europe, 2013. `https://media.blackhat.com/eu-13/briefings/Beery/bh-eu-13-a-perfect-crime-beery-wp.pdf`.

[46] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 784–796, New York, NY, USA, 2012. ACM.

[47] Mihir Bellare and Chanathip Namprempre. Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm. In Tatsuaki Okamoto, editor, *Advances in Cryptology - ASIACRYPT 2000*, volume 1976 of *Lecture Notes in Computer Science*, pages 531–545. Springer Berlin / Heidelberg, 2000.

[48] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: A system for secure multi-party computation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pages 257–266, New York, NY, USA, 2008. ACM.

[49] Daniel J. Bernstein, Daira Hopwood, Andreas Hülsing, Tanja Lange, Ruben Niederhagen, Louiza Papachristodoulou, Michael Schneider, Peter Schwabe, and Zooko Wilcox-O'Hearn. SPHINCS: practical stateless hash-based signatures. In *Eurocrypt 2015*. Springer, 2015. ilmumas.

[50] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P. Y. Strub, and J. K. Zinzindohoue. A messy state of the union: Taming the composite state machines of tls. In *2015 IEEE Symposium on Security and Privacy (SP)*, pages 535–552, May 2015.

[51] K. Bhargavan, A. Delignat-Lavaud, A. Pironti, A. Langley, and M. Ray. Transport Layer Security (TLS) Session Hash and Extended Master Secret Extension, 2015. IETF RFC7627, `http://tools.ietf.org/html/rfc7627`.

[52] Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Alfredo Pironti, and Pierre-Yves Strub. Triple handshakes and cookie cutters: Breaking and fixing authentication over TLS. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 98–113. IEEE Computer Society, 2014.

[53] Karthikeyan Bhargavan and Gaëtan Leurent. Transcript Collision Attacks: Breaking Authentication in TLS, IKE and SSH. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[54] Simon Blake-Wilson, Don Johnson, and Alfred Menezes. Key agreement protocols and their security analysis. In Michael Darnell, editor, *Crytography and Coding: 6th IMA International Conference Cirencester, UK, December 17–19, 1997 Proceedings*, pages 30–45, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[55] G.R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the 1979 AFIPS National Computer Conference*, pages 313–317, Monval, NJ, USA, 1979. AFIPS Press.

[56] Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In Hugo Krawczyk, editor, *Advances in Cryptology — CRYPTO '98: 18th Annual International Cryptology Conference Santa Barbara, California, USA August 23–27, 1998 Proceedings*, pages 1–12, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[57] Hanno Böck, Aaron Zauner, Sean Devlin, Juraj Somorovsky, and Philipp Jovanovic. Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS. Cryptology ePrint Archive, Report 2016/475, 2016. http://eprint.iacr.org/.

[58] Dan Bogdanov. *Sharemind: programmable secure computations with practical applications*. PhD thesis, University of Tartu, 2013.

[59] Dan Bogdanov, Liina Kamm, Baldur Kubo, Reimo Rebane, Ville Sokk, and Riivo Talviste. Students and Taxes: a Privacy-Preserving Social Study Using Secure Computation. *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2016(3), 2016. (to appear).

[60] Dan Bogdanov, Liina Kamm, Sven Laur, and Ville Sokk. Rmind: a tool for cryptographically secure statistical analysis. Cryptology ePrint Archive, Report 2014/512, 2014. http://eprint.iacr.org/.

[61] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A Framework for Fast Privacy-Preserving Computations. In *Proceedings of the 13th European Symposium on Research in Computer Security - ESORICS'08*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2008.

[62] Dan Bogdanov, Riivo Talviste, and Jan Willemson. Deploying secure multi-party computation for financial data analysis (short paper). In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security. FC'12*, volume 7397 of *Lecture Notes in Computer Science*, pages 57–64. Springer, 2012.

[63] Peter Bogetoft, DanLund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, JanusDam Nielsen, JesperBuus Nielsen, Kurt Nielsen, Jakob Pagter, Michael Schwartzbach, and Tomas Toft. Secure multiparty computation goes live. In *Financial Cryptography and Data Security*, volume 5628 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2009.

[64] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption Without Bootstrapping. In *Proceedings of the 3rd Innovations*

in Theoretical Computer Science Conference, ITCS ’12, pages 309–325, New York, NY, USA, 2012. ACM.

[65] Wolfgang Breyha, David Durvaux, Tobias Dussa, L. Aaron Kaplan, Florian Mendel, Christian Mock, Manuel Koschuch, Adi Kriegisch, Ulrich Pöschl, Ramin Sabet, Berg San, Ralf Schlatterbeck, Thomas Schreck, Alexander Würstlein, Aaron Zauner, and Pepi Zawodsky. Applied Crypto Hardening, 2016. https://bettercrypto.org/static/applied-crypto-hardening.pdf.

[66] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. XMSS – A Practical Forward Secure Signature Scheme Based on Minimal Security Assumptions. In *Post-Quantum Cryptography*, number 7071 in Lecture Notes in Computer Science, pages 117–129. Springer Berlin Heidelberg, 2011.

[67] Ahto Buldas, Risto Laanoja, and Ahto Truu. Efficient Implementation of Keyless Signatures with Hash Sequence Authentication. Cryptology ePrint Archive, Report 2014/689, 2014. http://eprint.iacr.org/.

[68] Ahto Buldas, Risto Laanoja, and Ahto Truu. Efficient Quantum-Immune Keyless Signatures with Identity. Cryptology ePrint Archive, Report 2014/321, 2014. http://eprint.iacr.org/.

[69] Ahto Buldas, Risto Laanoja, and Ahto Truu. Security Proofs for the BLT Signature Scheme. Cryptology ePrint Archive, Report 2014/696, 2014. http://eprint.iacr.org/.

[70] Martin Burkhart, Mario Strasser, Dilip Many, and Xenofontas Dimitropoulos. SEPIA: privacy-preserving aggregation of multi-domain network events and statistics. In *Proc. of USENIX conference on Security*. USENIX Association, 2010.

[71] Zhengjun Cao and Lihua Liu. Comment on "Realization of a scalable Shor algorithm". Cryptology ePrint Archive, Report 2015/1133, 2015. http://eprint.iacr.org/.

[72] Adrian Cho. Quantum or not, controversial computer yields no speedup. *Science*, 344(6190):1330–1331, 2014.

[73] Nicolas T. Courtois, Karsten Nohl, and Sean O’Neil. Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards. Cryptology ePrint Archive, Report 2008/166, 2008. http://eprint.iacr.org/.

[74] Mike Czumak. Exploiting MS14-066 / CVE-2014-6321 (aka “Winshock”), November 2014. http://www.securitysift.com/exploiting-ms14-066-cve-2014-6321-aka-winshock/.

[75] Ivan Damgård, Kasper Damgård, Kurt Nielsen, Peter Sebastian Nordholt, and Tomas Toft. Confidential benchmarking based on multiparty computation. Cryptology ePrint Archive, Report 2015/1006, 2015. http://eprint.iacr.org/.

[76] Ivan Damgård, Martin Geisler, Mikkel Krøigaard, and Jesper Buus Nielsen. Asynchronous multiparty computation: Theory and implementation. In *Public Key Cryptography – PKC 2009*, volume 5443 of *Lecture Notes in Computer Science*, pages 160–179. Springer, 2009.

[77] Ivan Damgård, Valerio Pastro, Nigel Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In *Advances in Cryptology – CRYPTO*

2012, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662. Springer, 2012.

[78] Giacomo De Meulenaer, François Gosset, Guerric Meurice De Dormale, and Jean-Jacques Quisquater. Integer factorization based on elliptic curve method: Towards better exploitation of reconfigurable hardware. In *Field-Programmable Custom Computing Machines, 2007. FCCM 2007. 15th Annual IEEE Symposium on*, pages 197–206. IEEE, 2007.

[79] Antoine Delignat-Lavaud and Karthikeyan Bhargavan. Network-based origin confusion attacks against https virtual hosting. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 227–237, New York, NY, USA, 2015. ACM.

[80] Daniel Demmler, Thomas Schneider, and Michael Zohner. ABY - A framework for efficient mixed-protocol secure two-party computation. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015*. The Internet Society, 2015.

[81] Bert den Boer and Antoon Bosselaers. Collisions for the compression function of MD5. In Tor Helleseth, editor, *Advances in Cryptology – EUROCRYPT '93*, volume 765 of *Lecture Notes in Computer Science*, pages 293–304. Springer Berlin Heidelberg, 1994.

[82] T. Dierks and C. Allen. The TLS Protocol Version 1.0, 1999. IETF RFC2246, http://tools.ietf.org/html/rfc2246.

[83] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1, 2006. RFC4346, http://www.ietf.org/rfc/rfc4346.txt.

[84] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol, Version 1.2, 2008. IETF RFC5246, http://tools.ietf.org/html/rfc5246.

[85] Jintai Ding and Dieter Schmidt. Rainbow, a New Multivariable Polynomial Signature Scheme. In John Ioannidis, Angelos Keromytis, and Moti Yung, editors, *Applied Cryptography and Network Security*, volume 3531 of *Lecture Notes in Computer Science*, pages 164–175. Springer Berlin Heidelberg, 2005.

[86] Hans Dobbertin. Cryptanalysis of MD5 Compress, May 1996. Rump session presentation at Eurocrypt 1996.

[87] Léo Ducas and Daniele Micciancio. FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second. In *Advances in Cryptology – EUROCRYPT 2015*, volume 9056 of *Lecture Notes in Computer Science*, pages 617–640. Springer, 2015.

[88] Thai Duong and Juliano Rizzo. Here Come The ⊕ Ninjas. http://packetstormsecurity.com/files/105499/Browser-Exploit-Against-SSL-TLS.html, May 13th 2011.

[89] Thai Duong and Juliano Rizzo. The CRIME Attack, Sep 2012. https://docs.google.com/presentation/d/11eBmGiHbYcHR9gL5nDyZChu_-lCa2GizeuOfaLU2HOU/.

[90] E.A.Grechnikov. Collisions for 72-step and 73-step SHA-1: Improvements in the Method of Characteristics. Cryptology ePrint Archive, Report 2010/413, 2010. http://eprint.iacr.org/.

[91] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of RC4. In Serge Vaudenay and Amr M. Youssef, editors, *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16–17, 2001 Revised Papers*, pages 1–24, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[92] Lance Fortnow. On NP in BQP. Computational Complexity Blog, `http://blog.computationalcomplexity.org/2007/02/on-np-in-bqp.html`, February 2007.

[93] Jens Franke, Thorsten Kleinjung, Christof Paar, Jan Pelzl, Christine Priplata, and Colin Stahlke. SHARK: a realizable special hardware sieving device for factoring 1024-bit integers. In *Cryptographic Hardware and Embedded Systems–CHES 2005*, volume 3659 of *LNCS*, pages 119–130. Springer, 2005.

[94] Kris Gaj, Soonhak Kwon, Patrick Baier, Paul Kohlbrenner, Hoang Le, Mohammed Khaleeluddin, and Ramakrishna Bachimanchi. Implementing the elliptic curve method of factoring in reconfigurable hardware. In *Cryptographic Hardware and Embedded Systems-CHES 2006*, volume 4249 of *LNCS*, pages 119–133. Springer, 2006.

[95] Steven D. Galbraith and Pierrick Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 78(1):51–72, 2016.

[96] Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult. Exposing iClass key diversification. In *5th USENIX Workshop on Offensive Technologies*, 2011.

[97] Flavio D. Garcia, Gerhard de Koning Gans, Roel Verdult, and Milosch Meriac. *Computer Security – ESORICS 2012: 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, chapter Dismantling iClass and iClass Elite, pages 697–715. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[98] Flavio D. Garcia, Gerhard Koning Gans, Ruben Muijrers, Peter Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. *Computer Security - ESORICS 2008: 13th European Symposium on Research in Computer Security, Málaga, Spain, October 6-8, 2008. Proceedings*, chapter Dismantling MIFARE Classic, pages 97–114. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[99] Christina Garman, Kenneth G. Paterson, and Thyla Van der Merwe. Attacks only get better: Password recovery attacks against rc4 in tls. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 113–128, Washington, D.C., August 2015. USENIX Association.

[100] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. `https://crypto.stanford.edu/craig`.

[101] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In *Public Key Cryptography – PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.

[102] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *Advances in Cryptology – EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.

[103] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[104] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In *Advances in Cryptology - CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.

[105] Martin Georgiev, Subodh Iyengar, Suman Jana, Rishita Anubhai, Dan Boneh, and Vitaly Shmatikov. The most dangerous code in the world: Validating ssl certificates in non-browser software. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 38–49, New York, NY, USA, 2012. ACM.

[106] Yoel Gluck, Neal Harris, and Angelo Prado. BREACH: reviving the CRIME attack. `http://www.breachattack.com`, July 12th 2013.

[107] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 218–229, New York, NY, USA, 1987. ACM.

[108] Shai Halevi and Victor Shoup. Design and Implementation of a Homomorphic-Encryption Library. `http://people.csail.mit.edu/shaih/pubs/he-library.pdf`, 2013. Accessed May 27, 2016.

[109] Wilko Henecka, Stefan K ögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. TASTY: Tool for Automating Secure Two-party Computations. In *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pages 451–462, New York, NY, USA, 2010. ACM.

[110] Clemens Hlauschek, Markus Gruber, Florian Fankhauser, and Christian Schanes. Prying open pandora's box: Kci attacks against tls. In *9th USENIX Workshop on Offensive Technologies (WOOT 15)*, Washington, D.C., August 2015. USENIX Association.

[111] Yan Huang, David Evans, Jonathan Katz, and Lior Malka. Faster Secure Two-party Computation Using Garbled Circuits. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, pages 35–35, Berkeley, CA, USA, 2011. USENIX Association.

[112] Ralph Jacobi and Josh Wyatt. MIFARE DESFire EV1 AES Authentication With TRF7970A, December 2014. `http://www.ti.com.cn/cn/lit/an/sloa213/sloa213.pdf`.

[113] Timo Kasper, Ingo von Maurich, David Oswald, and Christof Paar. Cloning cryptographic RFID cards for 25$. In *5th Benelux workshop on information and system security. Nijmegen, Netherlands*, 2010.

[114] John Kelsey. Compression and information leakage of plaintext. In Joan Daemen and Vincent Rijmen, editors, *Fast Software Encryption: 9th International Workshop, FSE 2002 Leuven, Belgium, February 4–6, 2002 Revised Papers*, pages 263–276, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[115] Auguste Kerckhoffs. La cryptographie militaire. *Journal des sciences militaires*, IX:5–38, January 1883.

[116] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar Signature Schemes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 206—222. Springer Berlin Heidelberg, 1999.

[117] Vlastimil Klíma, Ondrej Pokorný, and Tomáš Rosa. Attacking RSA-based sessions in SSL/TLS. In Colin D. Walter, Çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2003: 5th International Workshop, Cologne, Germany, September 8–10, 2003. Proceedings*, pages 426–440, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[118] Neal Koblitz. *Algebraic Aspects of Cryptography*. Number 3 in Algorithms and Computation in Mathematics. Springer, 1998.

[119] Neal Koblitz and Alfred J. Menezes. A Riddle Wrapped in an Enigma. Cryptology ePrint Archive, Report 2015/1018, 2015. `http://eprint.iacr.org/`.

[120] Gerhard Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. *Smart Card Research and Advanced Applications: 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008, London, UK, September 8-11, 2008. Proceedings*, chapter A Practical Attack on the MIFARE Classic, pages 267–282. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[121] Hugo Krawczyk. The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?). In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer Berlin / Heidelberg, 2001.

[122] Brian Krebs. How Secure is Your Security Badge?, August 2014. `http://krebsonsecurity.com/2014/08/how-secure-is-your-security-badge/`.

[123] Adam Langley. The POODLE bites again, December 2014. `https://www.imperialviolet.org/2014/12/08/poodleagain.html`.

[124] Peeter Laud and Meelis Roos. Formal Analysis of the Estonian Mobile-ID Protocol. In Audun Jøsang, Torleiv Maseng, and Svein J. Knapskog, editors, *Identity and Privacy in the Internet Age, 14th Nordic Conference on Secure IT Systems, NordSec 2009, Oslo, Norway, 14-16 October 2009. Proceedings*, volume 5838 of *Lecture Notes in Computer Science*, pages 271–286. Springer, 2009.

[125] John Launchbury, Iavor S. Diatchki, Thomas DuBuisson, and Andy Adams-Moran. Efficient lookup-table protocol in secure multiparty computation. In *ACM SIGPLAN International Conference on Functional Programming, ICFP'12*, pages 189–200. ACM, 2012.

[126] A.K. Lenstra, H.W. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[127] Yehuda Lindell and Benny Pinkas. A Proof of Security of Yao's Protocol for Two-Party Computation. *Journal of Cryptology*, 22(2):161–188, 2009.

[128] Lior Malka. Vmcrypt: Modular software architecture for scalable secure computation. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 715–724, New York, NY, USA, 2011. ACM.

[129] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay—a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20, Berkeley, CA, USA, 2004. USENIX Association.

[130] Josh Mandel, Austin Roach, and Keith Winstein. MIT Proximity Card Vulnerabilities. http://web.mit.edu/keithw/Public/MIT-Card-Vulnerabilities-March31.pdf.

[131] Nikos Mavrogiannopoulos, Frederik Vercauteren, Vesselin Velichkov, and Bart Preneel. A Cross-protocol Attack on the TLS Protocol. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 62–72, New York, NY, USA, 2012. ACM.

[132] R.J. McEliece. A public-key cryptosystem based on algebraic coding theory. The Deep Space Network Progress Report, DSN PR 42-44, pages 114–116, January and February 1978.

[133] Mohamad Merhi, Julio Cesar Hernandez-Castro, and Pedro Peris-Lopez. Studying the Pseudo Random Number Generator of a low-cost RFID tag. In *RFID-Technologies and Applications (RFID-TA), 2011 IEEE International Conference on*, pages 381–385. IEEE, 2011.

[134] Milosch Meriac. Heart of darkness-exploring the uncharted backwaters of hid iclasstm security. In *27th Chaos Communication Congress*, 2010. http://www.openpcd.org/images/HID-iCLASS-security.pdf.

[135] B. Moeller and A. Langley. TLS Fallback Signaling Cipher Suite Value (SCSV) for Preventing Protocol Downgrade Attacks, 2015. IETF RFC7507, http://tools.ietf.org/html/rfc7507.

[136] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. This POODLE Bites: Exploiting the SSL 3.0 Fallback, September 2014. https://www.openssl.org/~bodo/ssl-poodle.pdf.

[137] Thomas Monz, Daniel Nigg, Esteban A. Martinez, Matthias F. Brandl, Philipp Schindler, Richard Rines, Shannon X. Wang, Isaac L. Chuang, and Rainer Blatt. Realization of a scalable shor algorithm. *Science*, 351(6277):1068–1070, 2016.

[138] G. Eric Moorhouse. Shor's Algorithm for Factorizing Large Integers. http://www.uwyo.edu/moorhouse/slides/talk2.pdf.

[139] Jeffrey Morris. Implications of Quantum Information Processing On Military Operations. *The Cyber Defense Review*, May 2015. http://www.cyberdefensereview.org/2015/05/29/quantum/.

[140] Randall Munroe. Heartbleed Explanation, 2014. http://xkcd.com/1354.

[141] Moni Naor, Benny Pinkas, and Reuban Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 129–139, New York, NY, USA, 1999. ACM.

[142] H. Niederreiter. Knapsack-type cryptosystems and algebraic coding theory. *Problems of Control and Information Theory*, (15):159–166, 1986.

[143] H. Orman. The OAKLEY Key Determination Protocol, 1998. IETF RFC2412, http://tools.ietf.org/html/rfc2412.

[144] David Oswald and Christof Paar. *Cryptographic Hardware and Embedded Systems – CHES 2011: 13th International Workshop, Nara, Japan, September 28 – October 1, 2011. Proceedings*, chapter Breaking Mifare DESFire MF3ICD40: Power Analysis and Templates in the Real World, pages 207–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[145] Arnis Paršovs. Practical issues with TLS client certificate authentication. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[146] Jacques Patarin. Asymmetric Cryptography with a Hidden Monomial. In Neal Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 45–60. Springer Berlin Heidelberg, 1996.

[147] Jacques Patarin. Hidden fields equations (HFE) and isomorphisms of polynomials (IP): Two new families of asymmetric algorithms. In *Eurocrypt '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 33–48. Springer, 1996.

[148] Henning Perl, Michael Brenner, and Matthew Smith. Poster: an implementation of the fully homomorphic Smart-Vercauteren crypto-system. In *ACM Conference on Computer and Communications Security*, pages 837–840. ACM, 2011.

[149] A. Popov. Prohibiting RC4 Cipher Suites, 2015. IETF RFC7465, http://tools.ietf.org/html/rfc7465.

[150] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014.

[151] E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3, 2015. Internet Draft, https://tools.ietf.org/html/draft-ietf-tls-tls13-11.

[152] E. Rescorla, M. Ray, S. Dispensa, and N. Oskov. Transport Layer Security (TLS) Renegotiation Indication Extension, 2010. IETF RFC5746, http://tools.ietf.org/html/rfc5746.

[153] Eric Rescorla. Understanding the TLS Renegotiation Attack, 11 2009. http://www.educatedguesswork.org/2009/11/understanding_the_tls_renegoti.html.

[154] Eric Rescorla. TLS 1.3, 2015. http://www.realworldcrypto.com/rwc2015/program-2/RWC-2015-Rescorla-TLS.pdf.

[155] Ivan Ristic. Defending against the BREACH Attack. Qualys Blogs, https://community.qualys.com/blogs/securitylabs/2013/08/07/defending-against-the-breach-attack, Aug 7th 2013.

[156] Juliano Rizzo and Thai Duong. Practical Padding Oracle Attacks. In *4th USENIX Workshop on Offensive Technologies (WOOT 10)*, Washington, D.C., August 2010. USENIX Association.

[157] Tomáš Rosa. RFID Security. Selected Areas of LF and HF Applications, 2012. Soom – Hacking & Security 2012, Prague, http://crypto.hyperlink.cz/files/rosa_soom_v1a.pdf.

[158] Adi Shamir. How to share a secret. *Communications of the ACM*, 22:612–613, 1979.

[159] P. W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Foundations of Computer Science, 1994 Proceedings., 35th Annual Symposium on*, pages 124–134, Nov 1994.

[160] 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/.

[161] N.P. Smart and F. Vercauteren. Fully homomorphic SIMD operations. *Designs, Codes and Cryptography*, 71(1):57–81, 2014.

[162] Marc Stevens. New collision attacks on SHA-1 based on optimal joint local-collision analysis. In *EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 245–261. Springer, 2013.

[163] Marc Stevens, Pierre Karpman, and Thomas Peyrin. Freestart collision for full sha-1. Cryptology ePrint Archive, Report 2015/967, 2015. http://eprint.iacr.org/. Accepted to Eurocrypt 2016.

[164] Marc Stevens, Arjen Lenstra, and Benne de Weger. Chosen-Prefix Collisions for MD5 and Colliding X.509 Certificates for Different Identities. In Moni Naor, editor, *Advances in Cryptology - EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 1–22. Springer Berlin Heidelberg, 2007.

[165] Marc Stevens, Arjen K. Lenstra, and Benne De Weger. Chosen-prefix Collisions for MD5 and Applications. *Int. J. Appl. Cryptol.*, 2(4):322–359, July 2012.

[166] Marc Stevens, Alexander Sotirov, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Osvik, and Benne de Weger. Short Chosen-Prefix Collisions for MD5 and the Creation of a Rogue CA Certificate. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 55–69. Springer Berlin / Heidelberg, 2009.

[167] Jaak Tepandi. Mobiil-ID turvauuring, 2008.

[168] Mathy Vanhoef and Frank Piessens. All your biases belong to us: Breaking rc4 in wpa-tkip and tls. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 97–112, Washington, D.C., August 2015. USENIX Association.

[169] Serge Vaudenay. Security Flaws Induced by CBC Padding - Applications to SSL, IPSEC, WTLS... In Lars Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 534–545. Springer Berlin / Heidelberg, 2002.

[170] Anto Veldre. eID kasutusmallid, February 22th 2016. Blog post, available from https://blog.ria.ee/eid-kasutusmallid/.

[171] Xiaoyun Wang, Xuejia Lai, Dengguo Feng, Hui Chen, and Xiuyuan Yu. Cryptanalysis of the Hash Functions MD4 and RIPEMD. In Ronald Cramer, editor, *EUROCRYPT*, volume 3494 of *LNCS*, pages 1–18. Springer, 2005.

[172] Xiaoyun Wang, Andrew C Yao, and Frances Yao. Cryptanalysis on SHA-1, 2005. NIST cryptographic hash workshop, `http://csrc.nist.gov/groups/ST/hash/documents/Wang_SHA1-New-Result.pdf`.

[173] Xiaoyun Wnag, Yiqun Lisa Yin, and Hongbo Yu. Finding Collisions in the Full SHA-1. In Victor Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 17–36. Springer Berlin Heidelberg, 2005.

[174] Meggie Woodfield. Patch tuesday: Winshock vulnerability, November 2014. `https://blog.digicert.com/winshock-vulnerability/`.

[175] David Wu and Jacob Haven. Using Homomorphic Encryption for Large Scale Statistical Analysis. `https://crypto.stanford.edu/people/dwu4/papers/FHE-SI_Report.pdf`, 2012. Accessed May 27, 2016.

[176] Andrew Chi-Chih Yao. Protocols for secure computations. In *Foundations of Computer Science, 1982. SFCS '08. 23rd Annual Symposium on*, pages 160–164, Nov 1982.

[177] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 162–167, Oct 1986.

[178] Yuval Yarom, Daniel Genkin, and Nadia Heninger. Cachebleed: A timing attack on openssl constant time rsa. Cryptology ePrint Archive, Report 2016/224, 2016. `http://eprint.iacr.org/`.

[179] Ralf Zimmermann, Tim Güneysu, and Christof Paar. High-performance integer factoring with reconfigurable devices. In *Field Programmable Logic and Applications (FPL), 2010 International Conference on*, pages 83–88. IEEE, 2010.

A Shor's algorithm

Shor's quantum algorithm is capable of finding a period r of any integer function $f\colon \mathbb{Z} \to \mathbb{Z}_{2^m}$ on a quantum computer. By the period, we mean the smallest positive integer r so that $f(x+r) = f(x)$ for every $x \in \mathbb{Z}$. If $r < 2^n$ and $m = O(n)$, the time for finding r is $O(n^2)$.

A.1 Basic Principles of Quantum Mechanics

State space The state of any closed physical system (from a single particle to the universe as a whole) is completely described by a point in a vector space V, where the coordinates are complex numbers and where an inner product $\langle\cdot,\cdot\rangle$ is defined (hence, V is a *Hilbert space*). States of the system are the vectors Ψ for which $||\Psi|| = \sqrt{\langle\Psi,\Psi\rangle} = 1$, i.e. the *unit vectors*. Everything that can be known about the system is "encoded" into Ψ. The state space may sometimes have infinite number od dimensions. For the purpose of quantum computation, finite-dimensional state spaces are sufficient.

Dynamics If $\Psi(t)$ is a state at time t and $\Psi(t')$ is the state at some later time t', then the evolution of the state is described by a unitary linear operator U:

$$\Psi(t') = U_{t,t'}\Psi(t) \ .$$

The unitarity of U means that $UU^\dagger = 1$, where $U^\dagger$ is the *Hermitian conjugate* of U, i.e. the unique operator that satisfies $\langle U\Psi,\Psi'\rangle = \langle\Psi, U^\dagger\Psi'\rangle$ for any $\Psi,\Psi' \in V$. The exact form of such a transformation depends on the particular system. Often, $U_{t,t'}$ is described as a solution (integration from t to t') of a differential equation $\mathrm{i}\hbar\frac{\partial}{\partial t}\Psi = \mathcal{H}\Psi$, which is called the *Schrödinger's equation* of the system. Here $\mathcal{H}$ is the so-called Hamiltonian operator that describes the energy of the particular system, $\hbar$ is the Planck's constant and i is the imaginary unit.

Measurement A measurement is described by a set of mutually orthogonal subspaces V_i that (together) span V. The orthogonality of V_i and V_j means that $\langle\Psi_i,\Psi_j\rangle = 0$ for every $\Psi_i \in V_i$ and $\Psi_j \in V_j$. There is one-to-one correspondence between possible results r_i of the measurement and the subspaces V_i. Let $P_i\colon V \to V_i$ denote the projection operators. After the measurement, with probability $p_i = ||P_j\Psi||^2$, the result of the measurement is r_i and the state Ψ changes to

$$\frac{1}{||P_j\Psi||}P_j\Psi \ .$$

A.2 Quantum Bit and Quantum Register

A quantum bit (qubit) is the simplest possible quantum system that is represented by a two-dimensional complex vector space V with base vectors $|0\rangle$ and $|1\rangle$, i.e. in general, a qubit can be in the superposition state

$$\Psi = \alpha|0\rangle + \beta|1\rangle \ ,$$

where α and β are complex numbers with $|\alpha|^2 + |\beta|^2 = 1$. Measuring Ψ gives $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. A two-bit quantum system (register) has a four-dimensional state space $V \otimes V$, the *tensor-square* of V, with the base vectors $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. The two-bit quantum register may, in general, be in any state of the form:

$$\Psi = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle \ ,$$

where $\alpha, \beta, \gamma, \delta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1$. In general, the state space of an n-qbit quantum register is a 2^n dimensional complex vector space $\underbrace{V \otimes V \otimes \ldots \otimes V}_{n}$, i.e. the n-*th tensor power* of V with the basis consisting of 2^n base vectors $|0..00\rangle, |0..01\rangle \ldots |1..11\rangle$. The exponentially growing complexity of the state of an n-qubit quantum register illustrates the difficulty of simulating quantum processes with classical computers.

A.3 Quantum Gates

Like any classical computation can be represented as an evaluation of a Boolean circuit, also any quantum computation can be represented as a sequence of evaluating quantum gates. An m-bit quantum gate is a device that transforms input qubits $x_0, \ldots, x_{m-1}$ to output qubits $y_0, \ldots, y_{m-1}$. Every quantum gate is a linear and unitary transformation of the state space and can be represented as a matrix. For example, a single qubit quantum gate is a linear unitary transformation U (represented as a matrix $\begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix}$) that converts a qubit $\alpha|0\rangle + \beta|1\rangle$ to a new state $\alpha'|0\rangle + \beta'|1\rangle$ of the form:

$$\begin{bmatrix} \alpha' \\ \beta' \end{bmatrix} = \begin{bmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} u_{00}\alpha + u_{01}\beta \\ u_{10}\alpha + u_{11}\beta \end{bmatrix} \ .$$

In classical circuits, there are only two one-bit gates: the identity gate (just a wire) and the NOT gate. In quantum circuits, there are many more one-qubit gates. In the following, we will go through some of the most important and relevant to this chapter quantum gates. However, there exist many more.

Single Qubit Gates

NOT Gate Quantum NOT gate just swaps the complex coefficients α and β of the qubit $\alpha|0\rangle + \beta|1\rangle$ and transforms it to the qubit $\beta|0\rangle + \alpha|1\rangle$. The NOT-gate is represented as a matrix $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$. Analogous to the classical NOT gate, a quantum NOT applied twice is equivalent to the identity gate, i.e.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I \ .$$

Hadamard Gate Hadamard gate converts $|0\rangle$ to the sum $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|1\rangle$ to the sum $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. It is represented by the matrix $H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$. It is easy to verify that, analogous to the NOT gate, H applied twice is the identity gate, i.e. $HH = I$.

Phase Shift Gate The phase shift gate transforms a qubit $\alpha|0\rangle + \beta|1\rangle$ to a qubit $\alpha|0\rangle + e^{i\phi}\beta|1\rangle$, i.e. geometrically, it rotates the complex coefficient β by an angle ϕ on the complex plane. The phase transform gate is represented by the matrix $R_\phi = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix}$.

Two Qubit Gates

Controlled NOT (XOR) Gate is a two-bit gate that is an analogue of the classical XOR gate. It inverts the second qubit if the first qubit appears to be 1, i.e. $|00\rangle \mapsto |00\rangle$, $|01\rangle \mapsto |01\rangle$, $|10\rangle \mapsto |11\rangle$, and $|11\rangle \mapsto |10\rangle$.

Swap Gate Swap gate inverses the order of the bits. For example, for two-qubit swap gate, $|01\rangle \mapsto |10\rangle$ and $|10\rangle \mapsto |01\rangle$, but $|00\rangle \mapsto |00\rangle$ and $|11\rangle \mapsto |11\rangle$.

Controlled Phase Shift Gate Controlled phase shift gate is a two-qubit quantum gate. If the first qubit appears to be $|1\rangle$, the ordinary phase shift gate with factor $e^{i\phi}$ is applied to the second qubit, i.e. $|00\rangle \mapsto |00\rangle$, $|01\rangle \mapsto |01\rangle$, $|10\rangle \mapsto |10\rangle$, and $|11\rangle \mapsto e^{i\phi}|11\rangle$.

The matrices of the described more complex gates are shown in Table 7.

Quantum Circuits

For any classical circuit (say with AND and NOT gates) that implements a function $f\colon \{0,1\}^m \to \{0,1\}^m$ can be implemented as a quantum circuit U that transforms a $2m$-bit quantum input register so that $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$, which means that $|x\rangle|0^m\rangle \mapsto |x\rangle|f(x)\rangle$

It can be shown that for the modular exponential function $f(x) = a^x \bmod N$, where $N \approx 2^n$, there exists a quantum circuit F with $O(n^2)$ quantum gates, that computes $|x, y\rangle \mapsto |x, y \oplus f(x)\rangle$.

A.4 Quantum Fourier Transform

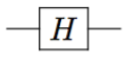
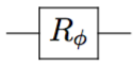
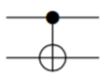
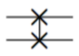
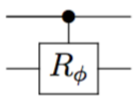
The classical Fourier transforms a vector $(x_0, \ldots, x_{N-1}) \in \mathbb{C}^N$ to the vector $(y_0, \ldots, y_{N-1}) \in \mathbb{C}^N$ according to the formula:

$$y_k = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} x_j e^{2\pi i \frac{jk}{N}} \quad . \tag{2}$$

Similarly, the quantum Fourier transform maps a quantum state $\sum_{i=0}^{N-1} x_i|i\rangle$ to a quantum state $\sum_{i=0}^{N-1} y_i|i\rangle$ according to formula (2). For example, for a single bit, i.e. if $N = 2$, the quantum Fourier transform maps $x_0|0\rangle + x_1|1\rangle$ to $\frac{x_0+x_1}{\sqrt{2}}|0\rangle + \frac{x_0-x_1}{\sqrt{2}}|1\rangle$, which in matrix form is

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = H \cdot \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} \quad .$$

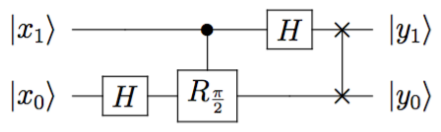
Table 7. Notation and matrix representation of gates in quantum circuit diagrams.

Gate	Matrix	Notation
Hadamard	$\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$	
Phase Shift	$\begin{bmatrix}1 & 0\\ 0 & e^{\mathrm{i}\phi}\end{bmatrix}$	
Controlled NOT (XOR)	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}$	
Swap	$\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}$	
Controlled phase shift	$\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&e^{\mathrm{i}\phi}\end{bmatrix}$	

Using the notation $\omega = e^{\frac{2\pi \mathrm{i}}{N}}$, the matrix representation (for $N = 4$) of the quantum Fourier transform is the following:

$$\frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \omega & \omega^2 & \omega^3\\ 1 & \omega^2 & \omega^4 & \omega^6\\ 1 & \omega^3 & \omega^6 & \omega^9\end{bmatrix} = \frac{1}{2}\begin{bmatrix}1 & 1 & 1 & 1\\ 1 & \mathrm{i} & -1 & -\mathrm{i}\\ 1 & -1 & 1 & -1\\ 1 & -\mathrm{i} & -1 & \mathrm{i}\end{bmatrix}.$$

For $N = 4$, the quantum Fourier transform can be computed using the following quantum circuit:



Tis operations corresponds to the product of unitary matrices:

$$\underbrace{\begin{bmatrix}1&0&0&0\\0&0&1&0\\0&1&0&0\\0&0&0&1\end{bmatrix}}_{\text{swap}}\cdot\underbrace{\begin{bmatrix}1&1&0&0\\1&-1&0&0\\0&0&1&1\\0&0&1&-1\end{bmatrix}}_{\text{second } H}\cdot\underbrace{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&\mathrm{i}\end{bmatrix}}_{\text{phase-shift}}\cdot\underbrace{\begin{bmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{bmatrix}}_{\text{first } H} = \begin{bmatrix}1&1&1&1\\1&\mathrm{i}&-1&-\mathrm{i}\\1&-1&1&-1\\1&-\mathrm{i}&-1&\mathrm{i}\end{bmatrix}.$$

A general recurrent scheme for quantum circuit implementation of QFT_n in case $N = 2^n$ is presented in Figure 3, where QFT_n is built from QFT_{n-1} (where $N = 2^{n-1}$). Note that the schemes are represented without the final swap of bits.

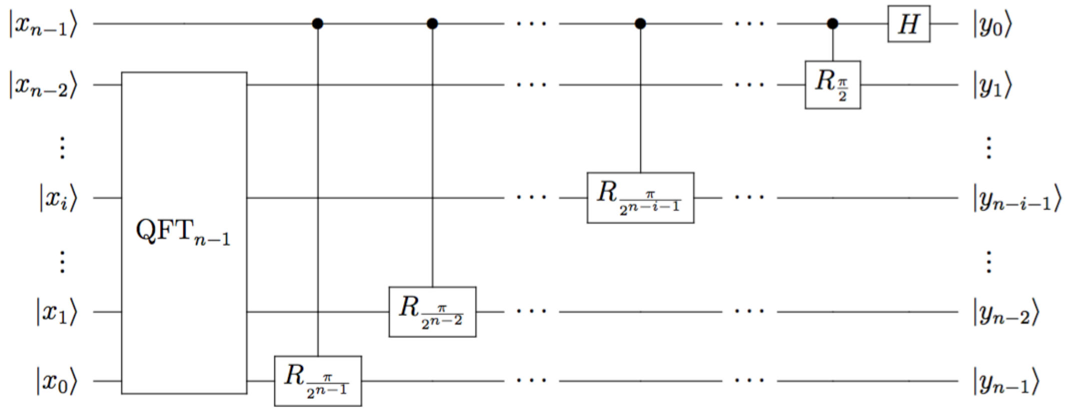


Figure 3. Implementation of QFT_n by using Hadamard- and controlled phase shift gates.

One important application of the quantum Fourier transform is that if QFT_m is applied to the state $|00..0\rangle$, then we obtain the uniform superposition

$$\Psi = \frac{1}{2^{m/2}}(|0..00\rangle + |0..01\rangle + \ldots + |1..10\rangle + |1..11\rangle)$$

of all 2^m possible base states. If $f\colon\{0,1\}^m \to \{0,1\}^m$ is any one-to-one function, then we may implement f as a quantum circuit and compute the superposition

$$\Psi' = \frac{1}{2^{m/2}}(f(|0..00\rangle) + f(|0..01\rangle) + \ldots + f(|1..10\rangle) + f(|1..11\rangle))$$

of all possible values of f just by executing the quantum circuit once. This is called *quantum parallelism*. If f is not invertible, we are able to implement a $2m$-qubit quantum circuit F that computes an equally distributed superposition of $|x\rangle|f(x)\rangle$, so that $F\colon|x\rangle|0\rangle \mapsto |x\rangle|f(x)\rangle$.

A.5 Shor's Period-Finding Algorithm

Say $F\colon|x,0\rangle \mapsto |x, f(x)\rangle$ is a quantum circuit for an integer function $f\colon\mathbb{Z} \to \mathbb{Z}_{2^m}$ with the period $r < 2^n$. We use two registers of size $2n$ and m qubits, that are initially set to $|0,0\rangle$. Let $N = 2^{2n}$. Then we:

1. Apply QFT_{2n} to the first register and obtain the superposition $\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i,0\rangle$.
2. Apply F to obtain the superposition $\frac{1}{\sqrt{N}}\sum_{i=0}^{N-1}|i,f(i)\rangle$.
3. Measure the second part of the register to obtain a result $k = f(s)$, where we may choose s so that $s < r$. Then the register will be in the superposition of all $\lceil N/r\rceil$ possible values $s, s+r, s+2r, \ldots$ that will result in k as the output of f. The reduced superposition will be:
$$\Psi' = \frac{1}{\sqrt{p}}\sum_{j=0}^{p-1}|s+jr,k\rangle \ ,$$
where $p = \lceil N/r\rceil$.
4. We apply QFT_{2n} again to obtain:
$$\mathrm{QFT}_{2n}\Psi' = \frac{1}{\sqrt{pN}}\sum_{i=0}^{N-1}\left(\sum_{j=0}^{p-1}e^{2\pi\mathrm{i}\frac{i(s+jr)}{N}}\right)|i,k\rangle = \frac{1}{\sqrt{pN}}\sum_{i=0}^{N-1}e^{2\pi\mathrm{i}\frac{is}{N}}\left(\sum_{j=0}^{p-1}e^{2\pi\mathrm{i}\frac{ijr}{N}}\right)|i,k\rangle \ .$$

The absolute values of the coefficients have sharp peaks in case i is a multiple of $\frac{N}{r}$. This is because:

$$\lim_{p\to\infty} \frac{1}{p} \sum_{j=0}^{p-1} e^{2\pi \mathrm{i} \alpha j} = \begin{cases} 1 & \text{if } \alpha \in \mathbb{Z} \\ 0 & \text{if } \alpha \notin \mathbb{Z} \end{cases} .$$

5. Finally, measure the first register, and get a value c close to $\lambda\frac{N}{r}$ with $\lambda \in \mathbb{Z}_r$, i.e. $\frac{c}{N} = c \cdot 2^{-2n} \approx \frac{\lambda}{r}$. This is equivalent to finding a rational approximation $\frac{a}{b}$ with $a, b < 2^n$ for the fixed point binary fractional number $c \cdot 2^{-2n}$. In such approximation, λ and r are uniquely determined only if the rational fraction $\frac{\lambda}{r}$ is irreducible, i.e. if $\gcd(\lambda, r) = 1$. Sufficient condition for this is that λ happens to be a *prime number*. By the prime-number theorem, the probability of a random $\lambda \in \mathbb{Z}_p$ being prime is about $\frac{1}{\ln r} = \frac{1}{O(n)}$, which means we need a linear (in n) number of trials to find r.

B Support for elliptic curves in various libraries

Table 8 summarises the elliptic curve support in various libraries.

Table 8. Support for elliptic curves in various libraries and frameworks

Curve	OpenSSL 1.0.2	GnuTLS 3.4.10	NSS 3.23	Java SE 7 / 8 (SunEC)	Bouncy Castle 1.54	.NET 4.5 / 4.6	BoringSSL	WolfSSL 3.8.0
brainpoolP160r1	+	-	-	-	+	-	-	-
brainpoolP160t1	+	-	-	-	+	-	-	-
brainpoolP192r1	+	-	-	-	+	-	-	-
brainpoolP192t1	+	-	-	-	+	-	-	-
brainpoolP224r1	+	-	-	-	+	-	-	-
brainpoolP224t1	+	-	-	-	+	-	-	-
brainpoolP256r1	+	-	-	-	+	-	-	-
brainpoolP256t1	+	-	-	-	+	-	-	-
brainpoolP320r1	+	-	-	-	+	-	-	-
brainpoolP320t1	+	-	-	-	+	-	-	-
brainpoolP384r1	+	-	-	-	+	-	-	-
brainpoolP384t1	+	-	-	-	+	-	-	-
brainpoolP512r1	+	-	-	-	+	-	-	-
brainpoolP512t1	+	-	-	-	+	-	-	-
c2pnb163v1	+	-	+	-	+	-	-	-
c2pnb163v2	+	-	+	-	+	-	-	-
c2pnb163v3	+	-	+	-	+	-	-	-
c2pnb176v1	+	-	+	-	+	-	-	-
c2pnb208w1	+	-	+	-	+	-	-	-
c2pnb272w1	+	-	+	-	+	-	-	-
c2pnb304w1	+	-	+	-	+	-	-	-
c2pnb368w1	+	-	+	-	+	-	-	-
c2tnb191v1	+	-	+	+	+	-	-	-
c2tnb191v2	+	-	+	+	+	-	-	-
c2tnb191v3	+	-	+	+	+	-	-	-
c2tnb239v1	+	-	+	+	+	-	-	-

Table 8. Support for elliptic curves in various libraries and frameworks

Curve	OpenSSL 1.0.2	GnuTLS 3.4.10	NSS 3.23	Java SE 7 / 8 (SunEC)	Bouncy Castle 1.54	.NET 4.5 / 4.6	BoringSSL	WolfSSL 3.8.0
c2tnb239v2	+	-	+	+	+	-	-	-
c2tnb239v3	+	-	+	+	+	-	-	-
c2tnb359v1	+	-	+	+	+	-	-	-
c2tnb431r1	+	-	+	+	+	-	-	-
prime192v1 (P-192)	+	+	+	+	+	-	-	+
prime192v2	+	-	+	+	+	-	-	-
prime192v3	+	-	+	+	+	-	-	-
prime239v1	+	-	+	+	+	-	-	-
prime239v2	+	-	+	+	+	-	-	-
prime239v3	+	-	+	+	+	-	-	-
prime256v1 (P-256)	+	+	+[14]	+	+	+	+	+
secp112r1	+	-	+	+	+	-	-	+
secp112r2	+	-	+	+	+	-	-	-
secp128r1	+	-	+	+	+	-	-	+
secp128r2	+	-	+	+	+	-	-	-
secp160k1	+	-	+	+	+	-	-	-
secp160r1	+	-	+	+	+	-	-	+
secp160r2	+	-	+	+	+	-	-	-
secp192k1	+	-	+	+	+	-	-	-
secp224k1	+	-	+	+	+	-	-	-
secp224r1 (P-224)	+	+	+	+	+	-	+	+
secp256k1	+	-	+	+	+	-	-	-
secp384r1 (P-384)	+	+	+[14]	+	+	+	+	+
secp521r1 (P-521)	+	+	+[14]	+	+	+	+	+
sect113r1	+	-	+	+	+	-	-	-
sect113r2	+	-	+	+	+	-	-	-
sect131r1	+	-	+	+	+	-	-	-
sect131r2	+	-	+	+	+	-	-	-
sect163k1	+	-	+	+	+	-	-	-
sect163r1	+	-	+	+	+	-	-	-
sect163r2	+	-	+	+	+	-	-	-
sect193r1	+	-	+	+	+	-	-	-
sect193r2	+	-	+	+	+	-	-	-
sect233k1	+	-	+	+	+	-	-	-
sect233r1	+	-	+	+	+	-	-	-
sect239k1	+	-	+	+	+	-	-	-
sect283k1	+	-	+	+	+	-	-	-

[14]These curves are supported by default. Support for additional curves may be achieved by setting the appropriate compilation flags.

Table 8. Support for elliptic curves in various libraries and frameworks

Curve	OpenSSL 1.0.2	GnuTLS 3.4.10	NSS 3.23	Java SE 7 / 8 (SunEC)	Bouncy Castle 1.54	.NET 4.5 / 4.6	BoringSSL	WolfSSL 3.8.0
sect283r1	+	-	+	+	+	-	-	-
sect409k1	+	-	+	+	+	-	-	-
sect409r1	+	-	+	+	+	-	-	-
sect571k1	+	-	+	+	+	-	-	-
sect571r1	+	-	+	+	+	-	-	-
wap-wsg-idm-ecid-wtls1	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls3	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls4	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls5	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls6	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls7	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls8	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls9	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls10	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls11	+	-	-	-	-	-	-	-
wap-wsg-idm-ecid-wtls12	+	-	-	-	-	-	-	-
Curve25519/X25519	-	-	-	-	+	-	+	+
Ed25519	-	-	-	-	-	-	+	+
FRP256v1	-	-	-	-	+	-	-	-