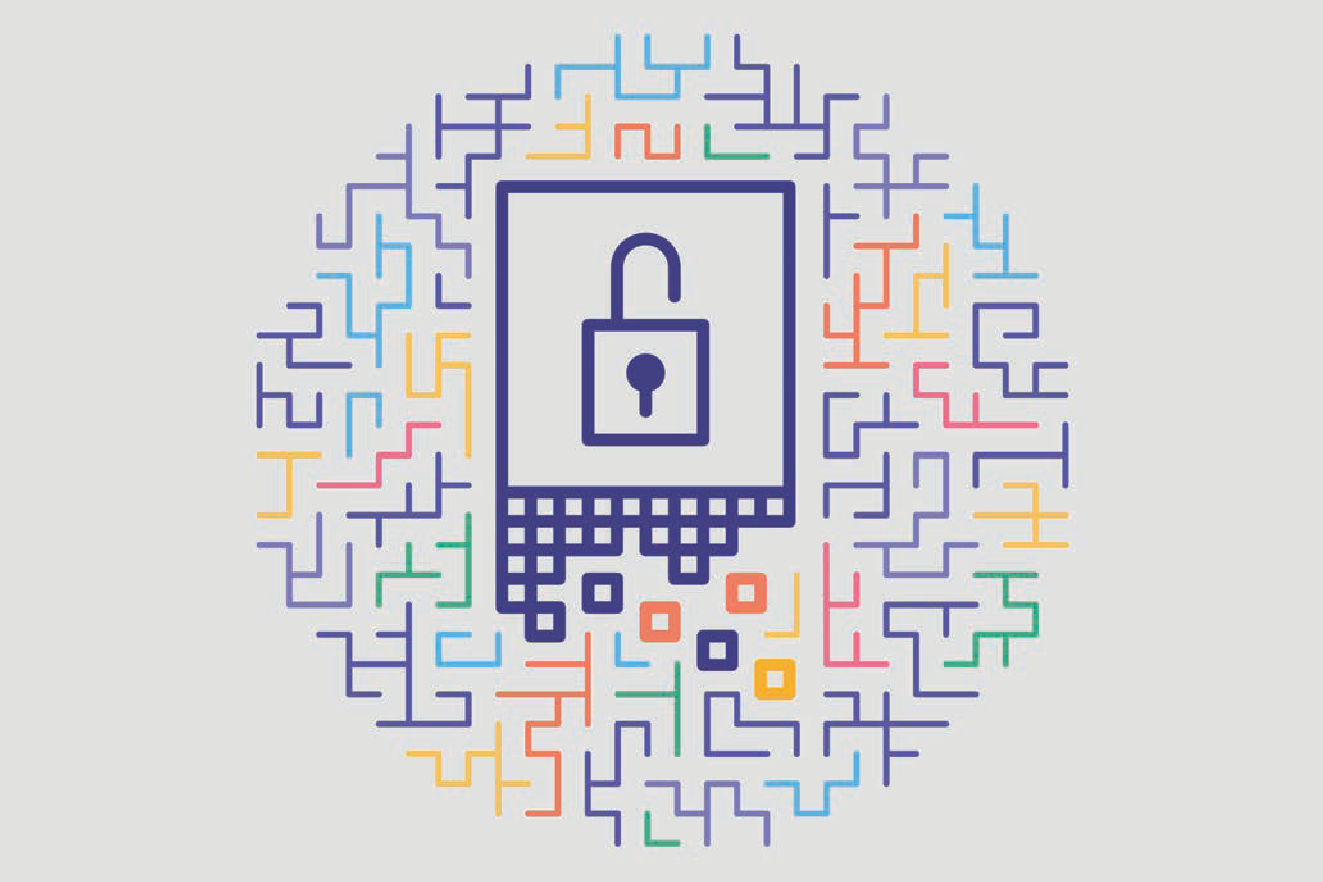
COURSE BOOK



## Cryptology

DLMCSEAITSC02



Learning Objectives

##### Introduction 9



Cryptology, the science of trusted communication, includes cryptography and cryptanalysis. Cryptography is the art of transforming information so that it is indecipherable to all but the intended recipient, and cryptanalysis is the art of deciphering ciphertext without knowledge of the key. In this course book, we also look at secret information sharing with steganogra- phy, the art of concealing a message by embedding it within another message so that nobody but the intended recipient knows of the existence of the message, in contrast to cryptography where the existence of the message is hidden.

We learn that cryptography protects information by shufﬂing data so that only additional secret information, the key, can feasibly reverse it. If the key used to encrypt and decrypt a message remains the same, we call it symmetric or single-key cryptography. By exchanging the roles of the keys using the private key to encipher and the public key to decipher, we have gained the opportunity for a digital signature of a message. For asymmetric cryptogra- phy, the security of its public-key algorithms relies on the inefﬁcient computation of mathe- matical functions on integers. For symmetric cryptographic algorithms, the security depends on the diffusion of small differences on the input and key to large differences in the output.

We distinguish between symmetric and asymmetric cryptography and discuss their biggest challenges. This can be solved by the public-key infrastructures, where private keys are certi- ﬁed either by third-party hierarchical authorities or by the web of trust, where trust is trans- ferred transitive in a network. One of the most well-known applications for enciphering are hash functions. A hash function is an algorithm that generates an output of ﬁxed size from an input of variable size. The output string of ﬁxed length that a cryptographic hash function produces from a string of any length is a kind of inimitable signature.

For prominent cryptography algorithms such as AES, DES, RSA, and ECC, we discuss their resistance to current established methods of cryptanalysis such as brute-force attacks and, in case of cryptographic hashes, brute-force attacks with rainbow tables. We will also look at their recommended key sizes calculating the effort needed to check all possible keys.

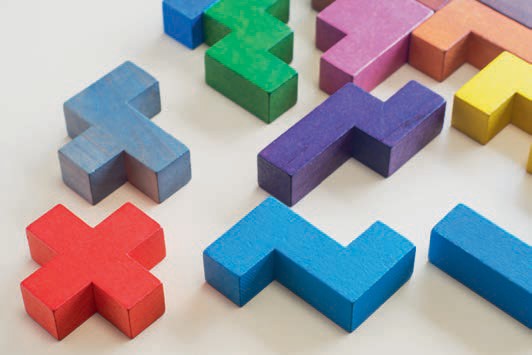
[www.iubh.de](http://www.iubh.de/)

##### 10 Introduction

Finally, we review network connections, such as VPN over the OSI stack, and prominent net- work protocols, such as TLS and its predecessor SSL, for authentication, conﬁdentiality, and authenticity for data sent through TCP. Regulatory aspects such as the GDPR and other legali- ties that affect the use of encryption will also be discussed.

As an application of cryptography to different areas, we will discuss the online banking pro- tocol FinTS, current advances in blockchain as a third trusted party by a database of entries, and the possibilities offered by quantum computing that need to be considered. Anonymous data transfer on the internet offered by onion routing in the Tor network completes our prac- tical excursion.

[www.iubh.de](http://www.iubh.de/)



# Unit 1

## Basic Concepts of Cryptology

#### STUDY GOALS

On completion of this unit, you will have learned …

… to distinguish between cryptology, cryptography, and cryptanalysis.

… about the applications of cryptology for conﬁdentiality, integrity, and availability.

… historical cryptographic algorithms and their impact.

… Kerckhoff’s criterion for best cryptographic practice.

… the principal uses of hash functions.

DL-E-DLMCSEAITSC02-U01

1. Basic Concepts of Cryptology

### Introduction

Encryption Also referred to as enciphering, this is the term for shuf- ﬂing data so that only additional secret information can feasibly undo it. The inverse of this action is known as decryption or deci-

phering.

Key This is the additional secret information that is practically indispensable for

decryption.

Symmetric cryptog-

raphy Cryptography is sym- metric when the same key is used to encrypt and decrypt.

Asymmetric cryptog-

raphy Cryptography is asymmetric when different keys are used to encrypt and decrypt. The key to encipher is public and the key to deci- pher is private.

Cryptography serves to protect information by **encryption**, the shufﬂing of data, i.e., the transformation of intelligible data into indecipherable data that only additional secret information, the key, can feasibly undo (decryption or deciphering). Shufﬂed (enci- phered) data can only be recovered (deciphered) by knowledge of the key. Since the original data are in principle still recoverable, the practice can be thought of as con- cealment. Because historically only written messages were encrypted, the source data, though a string of 1s and 0s (the viewpoint adopted in symmetric cryptography) or a number (that adopted in asymmetric cryptography), is called plaintext and the encryp- ted data the ciphertext.

###### Single and Public-Key Cryptography

Historically, the key to reverse the transformation of intelligible data into indeciphera- ble data was both necessary to decipher and to encipher. In the past, the key used to encrypt and decrypt was always the same. Symmetric cryptography was used by the Egyptians almost 2000 years BCE, during World War II on the Enigma machine, and exists today in the encryption of a wireless network (for example, by the AES algorithm).

Asymmetric cryptography was invented in the 1970s. Here, the key to encipher (the public key) and the key to decipher (the secret or private key) are different. In fact, only the key to decipher is private, while the key to encrypt is public (known to everyone). Symmetric encryption avoids the risk of compromising the key to decipher in exchang- ing the key with the cipherer, and in ownership of the cipher key (by the cipherer in addition to the decipherer).

Moreover, for a digital signature it is useful if the keys exchange their roles, i.e., the pri- vate key enciphers, and the public one deciphers. While the encrypted message will no longer be secret, every owner of the public key can check whether the private key encrypted the original message.

Today, such asymmetric cryptography algorithms are ubiquitous on the internet. Exam- ples include RSA, which is based on the difﬁculty of factoring in prime numbers, or ECC, which is based on the difﬁculty of computing points in ﬁnite curves. These protect (ﬁnancial) transactions on secure sites (those indicated by a padlock in the browser’s address bar).

###### Data Format

Up until the digital age, cryptography focused on the transformation of intelligible text into indecipherable text. Cryptography today focuses on the transformation of proces- sable (digital) data into indecipherable (digital) data. Data can be, for example, a digital ﬁle (text, image, sound, video, etc.). They are considered a bit sequence (denoted by a

Basic Concepts of Cryptology

string of 0s and 1s) or byte sequence (denoted by a string of hexadecimal pairs 00, 01,

…, FE, FF) or a number (denoted as usual by their decimal expansion 0, 1, 2, 3 …). Let us recall that every 1011*…* bit sequence is a number *n* via its binary expansion n = 1 + 0 · 2 + 1 · 22 + 1 · 23 + ... (and vice versa).

The point of view of a sequence of bits (speciﬁcally, of hexadecimal digits whose six- teen symbols 0—9 and A—F correspond to a group of four bits) is preferred in symmet- ric cryptography whose algorithms transform them, for instance, by permutation and substitution of digits. The point of view of a number is preferred in asymmetric crypto- graphy whose algorithms operate on it by mathematical functions such as raising to a power and exponentiation. The key, the additional secret information, can take various forms. While the chosen form is mainly a question of convenience, the most common are that of a number or a sequence of letters, e.g., a password, or a secret phrase (with spaces).

In the ancient scytale algorithm that uses a role of parchment wrapped around a stick, the key consists of the circumference (in letters) of the stick, a small number. Nowa- days, PIN codes (Personal Identiﬁcation Number) or passwords are ubiquitous in daily life. To facilitate memorization, the use of complete secret sentences or pass phrases is encouraged.

Asymmetric Key

-----BEGIN PGP PUBLIC KEY BLOCK -----

iQI2BCABCAAgFiEErZvSGgyHY7r19sbt31JsxV5IVU0FAl7aa3sCHQAACgkQ31Js xV5IVU0E7xAAiy24cTZz2OK+OwcoB2GxDjFC/qYV8sup3aFUqDRCpXRDRee18t+e

/N/pJUzY0CurgsNMl3bRDDMuZNPw/2/jaSYUrSpDsTHwqMI4sDAvqXOB1j6bMEUm S1/ME9EJ+LMdRhtpnmqfsHkptGfnMrrRfHAdzo43Fw5j/xmFx7eb9a8DdQMkBM9m zBtfoxB9d2/ZVqk0yoFwovuUnwVnEaOx0xLtdxdHlxRUWUF6nXJ278IHr6h4wolO JEBtZv9VN8f3rbmnaWDLEplfnOIVwAB2Becch7t1sBYbxy3rzvSU9cEHIpYgWuyQ wX2qy20+vfHEGMkwlv83l2ncacXRDeHXKAtigxhqirrAyUSThWxsAOGNJMmPq6RY aCg8IvgW09rvwU6CqZ2QuBjxXN8K0EtSH+RHH4FVTTP4wtbg3UsV/aXgNCWHF9EO V+bbhaig5bhC6lS/inosVg8ooOhW6+czl2in4G4mHSRrPZ8w74jfcxBm0ubbAdSy DUrMcNGHJ8ITjGKp23WKYpcnss5vMFYGKBa2eKlOmemPTzb3SHxlzAaHC6nef+GQ

W/wHgP5KgtVG4Y+HXtu+0tJhRCqlOPGzxVaBX0b61CYzN5jurAhZi2CCZ1E+xte+

7wEZ7j7zawT0Ev9oZpT56m00OTUcjfRPDy9tQlehuNLI6XXXLLW0F+M=

=+tEF

-----END PGP PUBLIC KEY BLOCK-----

This asymmetric key, because of its length, gets stored in a ﬁle of a couple of kilobytes called ASCII-armor.

### Terminology

The preﬁx crypto- comes from Greek *kryptós* or “hidden.” Accordingly, cryptography (from the Greek *gráphein*, “to write”) is the art of hidden writing: shufﬂing information so that it is indecipherable to all but the intended recipient. Cryptanalysis (from the Greek *analýein*, “to unravel”) or code breaking is the art of untying the hidden writing. The breaking of ciphers is accomplished by recovering or forging enciphered informa- tion without knowledge of the key.

Cryptology (from the Greek *lógos*, “word,” “reason,” “teaching,” or “meaning”) is the sci- ence of hiding, trusted communication, encompassing cryptography and cryptanalysis. The most general term is cryptology, and it is often used as a synonym for cryptography or cryptanalysis. Adjectives often used synonymously are secret, private, and conﬁden- tial. They all describe information that is not known about by other people or not meant to be known about. Something is

* secret, from Latin *secretus* “set apart,” if it is only known by a particular person or group,
* conﬁdential, from Latin *conﬁdere*, “to have full trust or reliance,” if it is to be kept secret, that is, not to be told or shared with other people,
* private, from Latin *privatus* “set apart” (from the state), if it is meant to be secret, especially regarding an individual versus the state.

Secrecy, though still important, is no longer the sole purpose of cryptology since the advent of public-key cryptography in the 1980s. To replace by electronic devices what had historically been done by hand, digital signatures and authentication were intro- duced.

Steganography from Greek *steganos*, “covered,” is the art of concealing a message by embedding it within another message (in a picture, audio ﬁle, or any binary data for- mat) so that nobody but the intended recipient knows of the existence of the message. In contrast, in cryptography the existence of the message itself is not disguised, but only its content. However, everything encrypted indicates that some, presumably

Basic Concepts of Cryptology

important, information is deliberately being hidden. Since a typical picture or audio ﬁle has hundreds of kilobytes, a few bytes can be changed to convey a secret message without noticeable change for the unprepared eye or ear (OPTED, n. d.)

Cryptography

Cryptography has long been used, for example, to conceal military messages from the enemy. Since then, (electronic binary) data have replaced text, and what used to be concealing written messages exchanged by messengers or kept secret has become securing data ﬂowing between computers or stored on a computer. Asymmetric cryp- tography allows through digital signatures that all participants in a communication can be veriﬁed undeniably, which is known as non-repudiation. This opens up many possi- bilities, especially for e-commerce.

All cryptographic methods, known as cryptosystems, are classiﬁed into symmetric or asymmetric cryptosystems. Symmetric cryptosystems are classiﬁed according to whether they operate on blocks of bits of ﬁxed length, e.g., 128 bits, (block cipher, such as AES or RSA) or single bits (stream ciphers, such as RC4). While stream ciphers are typically simpler, faster, and predestined for real time transmissions, they tend to be less secure and are therefore less commonly used (for example, a Wi-Fi network is commonly secured by a block cipher such as AES).

Code

A codiﬁcation, or an encoding, is a rule for replacing one bit of information, e.g., a letter with another one, usually to prepare it for processing by a computer. For example,

* Morse code
* the American Standard Code for Information Interchange (ASCII code) from 1963 that represents on computers 128 characters (and operations such as backspace and car- riage return) by seven-bit numbers. In ASCII, a lowercase letter a is 1100001, a lower- case letter b is 1100010, etc., and an uppercase A is 1000001, an uppercase B is 1000010.
* UTF-8 (8-bit Unicode Transformation Format). This is a variable-length character encoding by Ken Thompson and Rob Pike that represents any universal character in the Unicode standard by a sequence of between one to four bytes, and which is backwards compatible with ASCII, and is becoming the de facto standard. This

standard possibly has up to 232 ≈ 4 billion characters, and includes the alphabets

of many languages, such as English and Chinese, as well as meaningful symbols such as emoticons.

Ciphers

A cipher, like an encoding, also replaces information (which may be anything from a single bit to an entire sequence of symbols) with another one. However, the replace- ment is made according to a rule deﬁned by a key so that no one without its knowl- edge can invert the replacement.

Nowadays, information is encoded for processing by a computer and enciphered for a viable degree of information security. Information found on the internet is ﬁrst enco- ded to ASCII format, then encrypted into ciphers, e.g., by the AES (Advanced Encryption

Cipher

This rule is used when replacing information so that its inverse is only feasible with knowl- edge of the key.

Standard) algorithm and last encoded again for transmission using error-correction codes. After transmission, the recipient has to do the same in inverse order (Bellare & Rogaway, 2000).

### IT Security: Threats and Common Attacks

CIA

In information security, CIA stands for conﬁdentiality, integrity, and availa- bility of information.

A snappy acronym to summarize the fundamental aims of information security is the CIA triad. More comprehensive are “The ﬁve pillars of information assurance,” which add authentication and non-repudiation (Chen et al., 2019).

Cryptography helps to achieve all of these well. Good encryption, as achieved through thoroughly tested standard algorithms, such as AES or RSA, is practically impossible to break computationally. Instead, keys or plaintext are stolen before encryption or after decryption. While cryptography provides high technical security, human failure, for example, arising out of convenience or undue trust, is the weakest point in information security.

###### Conﬁdentiality

Conﬁdential information is meant to be kept secret and should not be disclosed to other people, e.g., information that is known only by the doctor or bank. In law, conﬁ- dentiality describes the relationship between, for example, a client and their counsel or agent, reﬂecting the trust placed in one by the other. The International Organization for Standardization (ISO) deﬁnes conﬁdentiality as ensuring that information is accessible only to those authorized to have access (2005). In IT, this means ensuring that sensitive information stored on computers is not disclosed to unauthorized persons, programs, or devices, e.g., avoiding that anyone with access to a network use common tools to eavesdrop on trafﬁc and intercept valuable private information.

###### Integrity

(Data) integrity is about the reliable, complete, and error-free transmission and recep- tion or storage of data. This means that the original data have not been altered or cor- rupted; in particular, the data are valid in accordance with expectations.

When the data have been altered, through electronic damage either by software or through physical damage to the disk, the data are unreadable. When we download a ﬁle, we verify its integrity by calculating its hash and comparing it with the published hash. Without this check, someone could, for example, package a Trojan horse into an installer on Microsoft Windows.

Basic Concepts of Cryptology

###### Availability

Though unrelated to cryptology, in information security, availability of information against threats like attacks (e.g., DoS (Denial of Service), accidents (e.g., power outages), or natural disasters (e.g., earthquakes). To achieve this, it is best to have a safety mar- gin and include redundancy, speciﬁcally

* parallel redundant failover hardware, such as a server or network. This is always running so that at any moment, upon detected failure of the primary system, pro- cessing can be automatically shifted over, and
* prevention of intrusion by monitoring network trafﬁc patterns for anomalies and block network trafﬁc when necessary.

###### Authentication

The word authentic comes from the Greek *authentes*, meaning “real or genuine.” Authentication thus is the veriﬁcation of something (or someone) as “authentic.” This might involve conﬁrming the identity of a person or the origins of an object. In IT, authentication means

* “veriﬁcation of the identity of a user or of the user’s eligibility to access an object,” i.e., convincing a computer that a person is who they claim to be after identiﬁcation, and
* “veriﬁcation that a message has not been altered or corrupted” (IBM, n. d.).

To verify their identity, a person proves who they are by showing some evidence. Multi- ple devices can carry out authentication, especially on a computer terminal. The most common methods are PINs and passwords, but smart cards or biometric identiﬁcation schemes, such as ﬁngerprints or iris scans, can carry out this task.

A common scenario is that of the man-in-the-middle attack (MITM), where the attacker assumes the identity of the other correspondent. To solve this, certiﬁcates, digital sig- natures by third parties, are used. This can be realized either as in the web of trust in OpenPGP, by signatures among persons known to each other over ends, or as on secure sites in a Web browser, by a signature of an unconditionally trusted, central certiﬁca- tion authority.

###### Non-Repudiation

Repudiation is a legal term for disavowal of a legal bind (such as an agreement or obli- gation). Someone who repudiates a) refuses to accept or be associated with a legal bind, b) refuses to recognize the validity of the legal bind (for example, their signature), and c) refuses to fulﬁll the legal bind.

Non-repudiation assures that a contract cannot later be denied by either party and the identity of the claimed sender or recipient of a given message. In computing, this means that authentication cannot be refuted afterwards. This is achieved by a digital signature. As an example, an electronic receipt proves that a particular user has sent a message such as an instruction to buy an item in an online auction. If the email says Bob sent it, then Bob can’t deny this. In today’s global economy, where face-to-face agreements are often impossible, non-repudiation is essential for safe commerce.

### Historical Overview

The history of cryptography dates back at least 4000 years. We distinguish three peri- ods (Davies, 1997):

1. Until the twentieth century, its methods were classic, mainly pen and paper.
2. In the early twentieth century, they were replaced by more efﬁcient and sophistica- ted methods on complex electromechanical machines, mainly rotor machines for polyalphabetic substitution, such as the Enigma rotor machine used by the Axis Powers during World War II.
3. Since the early twentieth century, digitalization, the replacement of analog devices by digital computers, has allowed methods of ever-greater complexity. Namely, the most tested algorithms are DES (or its threefold iteration 3DES), its successor AES for symmetric cryptography, and RSA and its successor ECC for asymmetric cryptogra- phy.

###### Classic Cryptography

From antiquity until World War I, cryptography was done by hand, thus limited in com- plexity and length to, at most, a few pages. Although principles of cryptanalysis were known, they were not really encountered due to a lack of automatization in cryptogra- phy. Therefore, given sufﬁcient ciphertext and effort, cryptanalysis was practically always successful. The Arabs were the ﬁrst to make use of cryptanalysis. They used both substitution and transposition ciphers, and knew both letter frequency distribu- tions and probable plaintext in cryptanalysis.

Scytale

A scytale (from Greek for “baton”) consists of a rod wrapped in a band of parchment on which a secret message is written. The parchment is rolled on the rod and written upon. The secret writing is legible only when the parchment is wound round an identi- cal rod. This is a transposition cipher: it shufﬂes, or transposes, the letters of the plain- text (OPTED, n. d.).

Caesar’s cipher

Caesar’s cipher is one of the simplest and most widely-known ciphers, named after Julius Caesar, who used it to communicate with his generals. This substitution cipher replaces each alphabetic letter of the plaintext by a ﬁxed alphabetic letter. Each letter

Basic Concepts of Cryptology

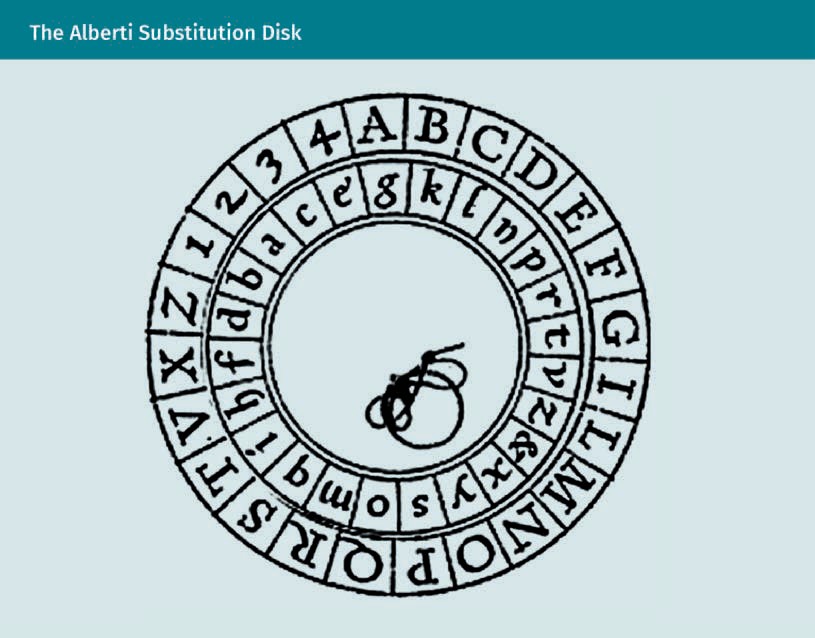
in the plaintext is shifted through the alphabet the same number of positions, that is, each letter in the plaintext is replaced by a letter some ﬁxed number of positions fur- ther down the alphabet (Bauer, 2000).

Bacon’s cipher

Francis Bacon’s cipher from 1605 is an arrangement of the letters a and b in ﬁve-letter combinations (of which there are 25 = 32) representing a letter of the English alphabet (26). Nowadays, we would call this a code, but at the time, it illustrated the important principle that only two different signs can be used to transmit any information (Tud- hope, 1993).

Alberti’s cipher disk

In 1470, Leon Battista Alberti described the ﬁrst cipher disk to shift the letters of the alphabet cyclically. He recommended changing the offset after every three or four words, thus conceiving a polyalphabetic cipher in which the same alphabetic letters are replaced by different ones. The US Army in World War I used the same device more than four centuries later (de Leeuw & Bergstra, 2007).



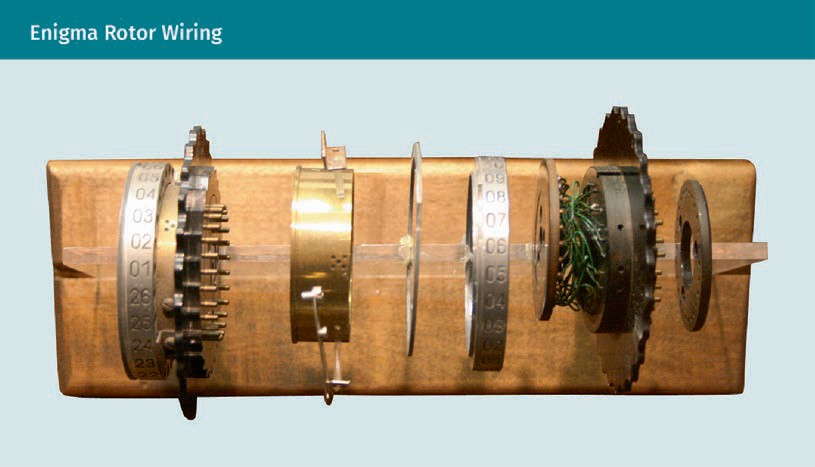
ADFGVX cipher

The most famous cipher of World War I was the German ADFGVX fractionation cipher, introduced on 5 March 1918, at the end of World War I. Believed to be unbreakable, the ADFGVX cipher was invented by Fritz Nebel, an army communications ofﬁcer, for use by

mobile military units. It replaced with a 6 x 6 matrix of the 26 letters and 10 digits into pairs of the letters A, D, F, G, V, and X. The resulting text was then written as a rectangle, and then the columns read in the order indicated by the key (Dooley, 2013, Chapter 5).

###### Electromechanical Cryptography with Rotor Machines

The ﬁrst successes of mechanization of cryptography occurred shortly after World War I. After telegraphy and radio became common in the 1920s, this created both a need and boost to cryptography and the development of the rotor cipher machines. This automi- zation step allowed faster operations of a higher complexity and lower error rates than manual encryption (Dooley, 2013, Chapter 5).



On the Enigma machine, the rotors are stacked. The rotation of one rotor causes the next one to rotate 1 / 26 of a full revolution. In operation, there is an electrical path through all rotors. Closing the key contact of the plaintext letter on a typewriter-like keyboard emits a current to one of the contacts on the initial rotor. The current then passes through the cable salad of the stacked rotors (which depends on their rota- tional positions) and ends at an indicator where it lights up the lamp of the ciphertext letter.

###### Digital Cryptography

Further development of crypto machines led to even faster rotor machines with elec- tronic substitutions of their rotors, but the concept remained the same: shifted mono- alphabetic substitutions offered a polyalphabetic substitution. The rise of the digital computer in the 1980s added no further changes in development. However, such letter per letter substitutions are still “linear over the letters,” so the ciphertext obtained from a plaintext reveals how to decrypt all letters of a plaintext of (at most) the same

Basic Concepts of Cryptology

length. Thus, a letter-per-letter substitution diffuses little and hardly spreads changes. An optimal diffusion is attained whenever the change of a single letter of the plaintext causes the change of half of the letters of the ciphertext. If the attacker has access to the ciphertexts of many plaintexts, possibly of their own choosing, then they can obtain the key from the ciphertexts of two plaintexts that differ in a single position.

DES

Instead, computers made it possible to achieve far better diffusion. Computers also allowed the combination of substitutions (such as Caesar’s cipher) with transpositions (such as the scytale), achieving far better diffusion. This lead to the creation of one of the most widely used ciphers in history, the Data Encryption Standard (DES) in 1976 (Davies, 1997).

AES

In January 1997, the US National Institute of Standards and Technology (NIST) announced a public contest for a replacement of the aging DES, the Advanced Encryp- tion Standard (AES). In October 2000, Rijndael, created by Joan Daemen and Vincent Rij- men, was chosen and became the AES (NIST, 2000).

Since improvements in computing power allowed ﬁnding the ﬁxed 56-bit DES key through an exhaustive key search, the NIST speciﬁcations for the AES demanded an increasing key length, if need be. Rijndael was not only shown immune to the most sophisticated known attacks, such as differential cryptanalysis and of an elegant and simple design, but was also both small enough to be implemented on smart cards (at less than 10 000 bytes of code) and ﬂexible enough to allow longer key lengths (Dae- men & Rijmen, 1999; 2002).

Public-key cryptography

Since the 1980s, the advent of public-key cryptography in the information age made digital signatures and authentication possible, creating ways for electronic information to replace what had historically been achieved through physical documents. Difﬁe and Hellman (1976) were the ﬁrst to publicly suggest asymmetric encryption. Conceptually, it relies on a trap function (more speciﬁcally, the modular exponential). While this inverti- ble function is easily computable, its inverse is hardly computable in the absence of additional information: the secret key (Difﬁe & Hellman, 1976).

To encrypt, the function is applied to decrypt its inverse with the secret key. For exam- ple, in the Difﬁe and Hellman (1976) approach, this function is the exponential, how- ever, over a different domain than the real numbers we are used to. In fact, they intro- duced only a scheme for exchanging a secret key through an insecure channel. The RSA cryptographic algorithm (Rivest et al., 1978) or the ElGamal algorithm (Elgamal, 1985) ﬁrst applied this approach. Not only did these algorithms enable ciphering with a pub- lic key, removing the problem of secret communication, but made digital signatures viable with the private key to cipher, possibly launching its commercial breakthrough. These algorithms still stand strong, but others, such as ECC, are currently deemed more efﬁcient and equally secure.

### Security Criteria

Kerckhoffs’ principle The ciphertext should be secure even if everything about it, except the key, is public knowl-

edge.

Ideal diffusion Shannon’s ideal states that if a bit in the plaintext or key changes, then half of

the bits in the ciphertext changes.

Hash function An algorithm that generates a ﬁxed- size output from var- iable-size input is known as a hash

function.

Checksum function An algorithm that quickly generates a ﬁxed-size output from variable-size input without colli- sions is called a checksum function.

Kerckhoff’s principle postulates the independence of a cryptoalgorithm’s security from its secrecy. While knowledge of the key compromises a single encryption, knowledge of the algorithm will compromise all encryptions ever carried out. A public algorithm guarantees the difﬁculty of decryption depending only on the knowledge of the key, but not on the algorithm. The more it is used, the more likely it becomes that the algorithm will be eventually known. For the algorithm to be useful, it thus needs to be safe even though it is public.

###### Shannon’s Criteria

Shannon’s principles of confusion and diffusion give criteria for an uninferable relation between the ciphertext and the key in respect to the plaintext. While the output of the cipher, the ciphertext, depends deterministically on the input, the plaintext, and the key, the algorithm aims to obfuscate this relationship to make it as complicated, inter- twined, and scrambled as possible. Each letter of the output (ciphertext) depends on each letter of the input (plaintext) and of the key.

Shannon (1949) put it as: “the enemy knows the system,” which is now known as Shan- non’s maxim. The opposite would be to rely on a potentially weak but unknown algo- rithm: “security through obscurity.” Ample historical evidence shows the futility of such a proposition (e.g., ADFGVX cipher).

### Hash Functions

A hash function is an algorithm that generates an output of ﬁxed (byte) size (usually around 16 to 64 bytes) from an input of variable (byte) size, for example, a text ﬁle, image ﬁle, or a compressed archive. From a variable-size input, a hash function calcu- lates a ﬁxed-size output called the hash value or, depending on its use, also (message) digest, digital ﬁngerprint, or checksum. For example, the hash md5 (of 16 bytes) of the word “key” in hexadecimal coding is 146c07ef2479cedcd54c7c2af5cf3a80.

Checksum

A (simple) hash function, or checksum function, should satisfy the following character- istics

* fast computation of a hash for any given data, and
* little possibility for two (hardly) different messages to give the same hash.

Thus, a hash function should act like a randomized function, while being deterministic and fast to calculate.

Basic Concepts of Cryptology

For example, the most naive checksum would be the sum of the bits of the input, trun- cated to the ﬁxed output size. It is almost a hash function: it is fast, and it is indeed unlikely that two different messages give the same hash. However, one easily obtains two almost identical messages with the same hash. Tiny alterations could go undetec- ted.

Cryptographic hash function

A should, moreover, guarantee that it is unfeasible to calculate an input that has a given hash. Thus, the output string of ﬁxed length that a cryptographic hash function produces from a string of any length (e.g., an important message) is a kind of inimitable signature. A person who knows the hash value cannot know the original message; only the person who knows the original message can prove that the “hash value” is pro- duced from that message. Heer (2008) describes it as “a hash function is weakly colli- sion resistant if ﬁnding an inverse is computationally unfeasible,” i.e., given an output, ﬁnding a (yet unknown) corresponding input. Strong collision resistance, thus, describes a hash function for which “ﬁnding a collision is computationally unfeasible” (Heer, 2008), i.e., ﬁnding two inputs with the same output. Otherwise, an attacker could substitute an authorized message with an unauthorized one

###### Applications

Hash functions are used for querying database entries for error detection and correc- tion (e.g., for data integrity checks) and in cryptography to identify data but conceal its content for data authenticity checks and authentication (e.g., store passwords, digital signatures).

In practice, even for checksums, most hash functions are cryptographic. Though slower, they are still fast enough on most hardware. Sometimes, for example when storing passwords, they are deliberately slow so that the passwords cannot be found quickly by their hash values through an exhaustive search among probable candidates. The most common cryptographic hash functions used to be MD5, usually with an output length of 128 bit, invented by Ron Rivest of the Massachusetts Institute of Technology in 1991.

SHA

The Secure Hash Algorithms (SHA) are cryptographic hash functions provided by the National Security Agency (NSA) and the NIST. SHA-1 is the successor to MD5 with a hash size of 160 bits. The MD5 was an earlier, widely used hash function, that fell victim to more and more suspicious security shortcomings (even though it is not broken, i.e., there is no known computationally feasible way to produce an input for a given hash). SHA-1 was used in the Digital Signature Algorithm (DSA) as prescribed by the Digital Sig- nature Standard (DSS) from the Internet Engineering Task Force.

In the meantime, there were three SHA algorithms SHA-1, SHA-2, and SHA-3 (released in 2015) of increasing security, mitigating the shortcomings of each predecessor. “SHA-2” permits hashes of different bit sizes. To indicate the number of bits, it is appended to the preﬁx “SHA,” for example, “SHA-224,” “SHA-256,” “SHA-384,” and “SHA-512.”

Summary

Cryptography protects information by shufﬂing data so that only additional secret information, the key, can feasibly reverse it. Up until the end of the 1970s, symmet- ric or (single-key) cryptography was common. Asymmetric cryptography followed.

The security of asymmetric cryptographic algorithms, or public-key algorithms, relies on the inefﬁcient computation of mathematical functions on the integers. The security of symmetric cryptographic algorithms depends on the diffusion of small differences on the input and key to large differences in the output. Good encryption, as achieved by standard algorithms, such as AES or RSA, is practically impossible to break computationally.

A hash function is an algorithm that generates an output of ﬁxed size from an input of variable size. We distinguish between cryptographic (or one-way) and checksum (or non-cryptographic) hash functions. For cryptographic hash functions, we also deﬁned the weak and strong collision resistance, which depend on whether it is unfeasible to ﬁnd an unknown message for a given hash and whether it is unfeasi- ble to ﬁnd two messages with the same hash.



# Unit 2

## Symmetric Cryptosystems

#### STUDY GOALS

On completion of this unit, you will have learned …

… the fundamental symmetric cryptographic algorithms: substitution and transposition.

… about the only cryptographically perfectly secure cipher the one-time pad.

… about the modern algorithms Data Encryption System and Advanced Encryption System.

… the manifold uses of (cryptographic) hash functions.

DL-E-DLMCSEAITSC02-U02

1. Symmetric Cryptosystems

### Introduction

Until the end of the 1970s, before the publication of Difﬁe and Hellman (1976) and Riv- est et al. (1978), all (known) cryptographic algorithms were symmetric (or single-key), i.e., they used the same key to cipher and decipher. Every historic algorithm, as sophis- ticated as it may have ben, be it Caesar’s cipher, the scytale or the Enigma, was sym- metric.

Symmetric algorithms operate on the input as a string (of bits or letters) by (iterated) substitutions and transpositions. In contrast, asymmetric algorithms depend on a com- putationally difﬁcult problem, such as the factorization of a composed number into its prime factors, and consider the input a natural number. The only perfectly secure cipher is the one-time pad in which the key is as long as the plaintext and the cipher- text is obtained by adding, letter by letter, each letter of the key to the corresponding (i.e., in the same position) letter of the plaintext. However, for more complex messages, e.g., text, image, or video ﬁles, a large key is impractical. Nowadays, this means that to encrypt a hard drive, another hard drive that carries the key is needed. To compensate the shorter key length, modern algorithms, ideally, create so much intertwining that they achieve almost perfect diffusion. In other words, the change of a single bit of the input or key causes the change of around half of the output bits. Modern algorithms, such as Data Encryption Standard (DES) or Advanced Encryption Standard (AES), are substitution and permutation network block ciphers, meaning that they encrypt one chunk of data at a time by iterated transpositions and substitutions.

### Substitution and Transposition

To encrypt, the two basic operations are transposition and substitution:

* A transposition changes the order (i.e., transposes or permutes) of the symbols in the text but not the symbols themselves.
* A substitution replaces every symbol in the text with another (group of) symbol(s), but not the order of the symbols.

The historical prototypical algorithms for these two operations are:

* Caesar's substitution cipher, that advances every letter in the plaintext by the three positions, e.g., encrypts A as D, B as E, and
* the permutation of the plaintext by the scytale (baton of Licurgo) where the parch- ment is wrapped around the baton and the text is written on it horizontally.

Symmetric Cryptosystems

We will see that even with many possible keys, an algorithm, such as that given by an arbitrary permutation of the alphabet which has almost 280 keys, can be easily broken if it preserves regularities, such as the frequency of the letters. Substitution and per- mutation networks join and iterate these two complementary prototypical algorithms to reach this goal.

###### Substitution Ciphers

In a substitution cipher, the key determines substitutions of the plaintext alphabet (considered as a set of units of symbols such as single letters or pairs of letters) by the ciphertext alphabet. For example, if the units of the plaintext and ciphertext are both the letters of the Latin alphabet, then a substitution permutes the letters of the Latin alphabet. If the substitution cipher is monoalphabetic (such as Caesar’s cipher), then the same substitution is applied to every letter of the plaintext independent of its position. If the substitution cipher is polyalphabetic (such as the Enigma), then the substitution varies with the position of the letter in the plaintext. To encrypt, each alphabetical unit of the plaintext is replaced by the substituted alphabetical unit, and inversed to decrypt. Monoalphabetic substitution is considered insecure because the frequencies of symbols that can be found in the plaintext and in the ciphertext remain the same. In English, for example, around 25 letters of ciphertext sufﬁce for cryptanaly- sis. The main approach to reduce the preservation of the single-letter frequencies in the ciphertext, known as polyalphabetic substitution, is to use several cipher alpha- bets.

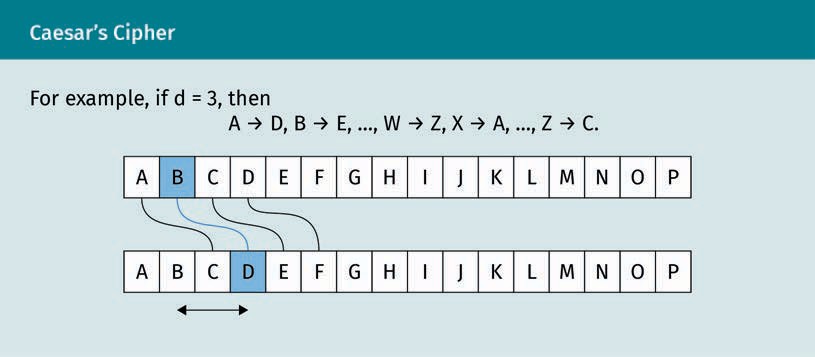
Shift by a ﬁxed distance

The simplest substitution cipher is a cyclical shift of the plaintext alphabet called Cae- sar’s cipher. The Roman emperors Caesar (100—44 BCE) and Augustus (63—14 BCE) used this method (Bauer, 2016). Fix a distance d between letters in alphabetical order, i.e., a number between 0 and 25, and shift (forward) each letter of the (Latin) alphabet by this distance d. We imagine that the alphabet is circular, i.e., the letters are arranged in a ring, so that the shift of a letter at the end of the alphabet results in a letter at the beginning of the alphabet.

Substitution cipher This cipher replaces each alphabetical unit of the plaintext with a corresponding alphabetical unit.

Caesar’s cipher This substitution cipher shifts the

alphabetical position of every plaintext letter by the same distance.



There are 26 keys (including the trivial key d = 0). Caesar's cipher displaces each letter of the alphabet by the same distance. To decipher, each letter is shifted by the negative distance –d, that is, d positions backwards.

Substitution by permutation of the letters of the alphabet

Instead of replacing each letter by one shifted by the same distance d, let’s replace each letter with an arbitrary letter, as in the example below.

A B … Y Z

…

E Z … G A

We shufﬂe the letters among themselves. This way we obtain 26 · 25 · 1 = 26! > 1026

keys (which is around the number of passwords that are possible with 80 bits).

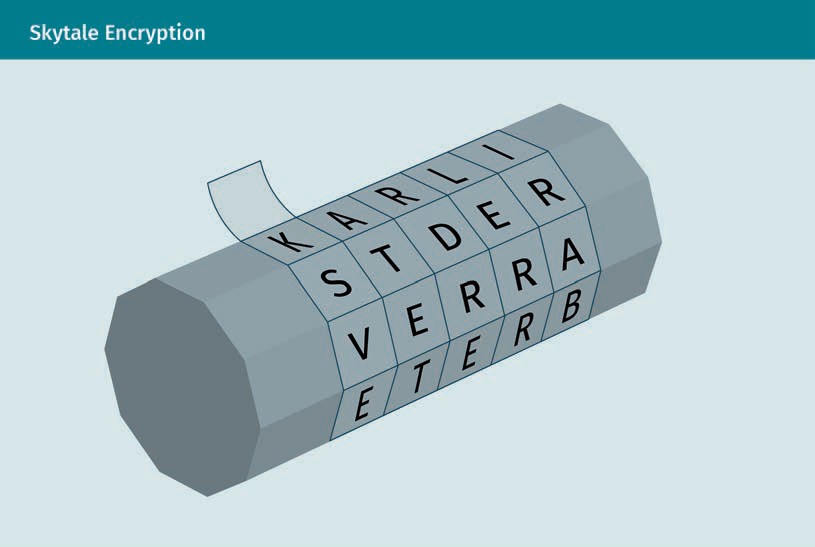
###### Transposition (or Permutation) Ciphers

A transposition cipher encrypts the plaintext by permuting its units (and decrypts by the inverse permutation). Each alphabetical unit stays the same; the encryption depends only on the positions of the units.

The scytale is a baton with which the Spartans enciphered as follows: (1) wrap the stick into a narrow strip, (2) write on this strip horizontally, that is, along the larger edge, and

(3) unroll the strip (Bauer, 2016). The letters transposed on the cloth could only be deci- phered by a stick with the same circumference (the key) in the same way as the text was encrypted: wrap the staff into the strip and read this strip horizontally, along the larger edge.

Symmetric Cryptosystems



Here, the key is the number given by the number of letters that ﬁt around the circum- ference of the stick.

###### Security of Historical Examples

Let us apply the established security criteria to the substitution ciphers.

Caesar’s cipher

This simple substitution cipher violates all desirable qualities. For example, Kerckhoff’s principle states that the algorithm be public once the method is known. Considering the small number of 25 keys, the ciphertext gives way in short time to a brute-force attack.

Substitution by arbitrary permutation of the letters of the alphabet

A substitution by an arbitrary permutation of the letters of the alphabet has 26 · 25 · 1 = 26! > 1026 keys, so a brute-force attack is computationally infeasible.

However, it violates the goals of diffusion and confusion. If the key (permutation of the alphabet) exchanges the letter α for the letter β, then there’s

* poor confusion because the substitution of β in the key implies only the substitu- tion of each letter β in the ciphertext,
* poor diffusion because the substitution of a letter α in the plaintext implies only the substitution of the corresponding letter β in the ciphertext.

Brute-force attack This exhaustive key search checks each possible key.

###### Product Ciphers

Product cipher A composition of ciphers where the output of one cipher serves as the input

of the next.

A product cipher composes ciphers, i.e., if the product is two-fold, then the output of one cipher is the input of the other. The ciphertext of the product cipher is the cipher- text of the ﬁnal cipher. Combining transpositions only with transpositions or substitu- tions only with substitutions, the obtained cipher is again a transposition or substitu- tion, and hardly more secure. However, mixing them, a transposition with substitutions, can indeed make the cipher more secure.

A fractionation cipher is a product cipher that substitutes every symbol in the plaintext by a group of symbols (usually pairs) and transposes the obtained ciphertext. The most famous fractionation cipher was the ADFGVX cipher used by the German forces during World War I (Bauer, 2016). In this cipher, the 26 letters of the Latin alphabet and ten dig- its were arranged in a 6 × 6-table and replaced by the pair of letters among A, D, F, G, V, and X that indicate the row and column of the letter or digit. The resulting text was written as usual from left to right into the rows of a table and then each column read in the order indicated by a keyword.

We will see how modern ciphers reﬁne this idea of a product cipher to obtain good dif- fusion.

### Block Ciphers

Classic ciphers usually replaced single letters and sometimes pairs of letters. Systems that operated on trigrams or larger groups of letters were regarded as too tedious and never widely used.

Instead, it is safer to substitute a whole block (of letters instead of a single letter) according to the key. However, the alphabet of this replacement would be gigantic, so this ideal is practically unattainable, especially on hardware as limited as a smart card with an eight-bit processor. For a block of 16 bytes, for example, this substitution table would have a horrendous 2128 • 16 bytes. However, in modern single-key cryptography, the units of information is commonly 128 bits, about 27 alphabetic characters, whereas two-key cryptography based on the RSA algorithm commonly uses units of 2048 bits, about 620 alphabetic characters.

For example, AES only replaces each byte, each entry in a block, a replacement table of 28 = 256 entries of 1 byte (and afterwards transposes the entries.) We will see that these operations complement each other so well that they are practically as safe as a substitution of the whole block.

Symmetric Cryptosystems

###### Block and Stream Ciphers

A block cipher partitions the plaintext into blocks of the same size and enciphers each block by a common key; while a block could consist of a single symbol, normally it is larger. For example, in DES the block size is 64 bits, and in the AES it is 128 bits.

A stream cipher partitions the plaintext into units, normally of a single character, and then encrypts the i-th unit of the plaintext with the i-th unit of a key stream. Examples are the one-time pad, rotor machines (such as the Enigma), and DES used in Triple DES (in which the output from one encryption is the input of the next encryption). In a stream cipher, the same section of the key stream that was used to encipher must be used to decipher. Thus, the sender’s and recipient’s key stream must be synchronized initially and constantly thereafter.

###### Feistel Ciphers

A Feistel cipher or a substitution and permutation network (SPN) groups the text (byte sequence) into n-byte blocks (e.g., n = 16 for AES) and enciphers each block by itera- tion (e.g., ten times in AES, and ﬁve times in our prototypical model) of the following three steps:

* + 1. Add (XOR) the key,
    2. Substitute the alphabet (which operates in sub-blocks of the block, for example, on each byte), and
    3. Permute of all the sub-blocks in a block.

After the addition (XOR) of the key is applied, the substitution of the alphabet, e.g., in the AES algorithm (each byte, pair of hexadecimal letters, by another), and the permu- tation of the text (from the current step, called “state”), e.g., in AES, the entries in each row (and the columns) are permuted.

These two simple operations, the substitution of the alphabet and the permutation of text, complement each other well. They generate high confusion and diffusion after a few iterations. In the ﬁrst and last round, the steps before and after the addition of the key are omitted because they do not increase the cryptographic security. Since the algorithm is public (Kerckhoff’s principle), any attacker is capable of undoing all those steps that do not require knowledge of the key. Though seemingly a Feistel cipher dif- fers from classical cryptosystems, it is a product cipher, made up of transpositions and substitutions.

Stream cipher versus block cipher

A stream cipher operates on single characters (e.g., sin- gle bytes), while a block cipher oper- ates on groups of characters (e.g., each 16 bytes large).

Substitution and permutation network This cipher itera- tively substitutes and permutes each block after adding a key.

### Data Encryption Standard (DES)

Data encryption

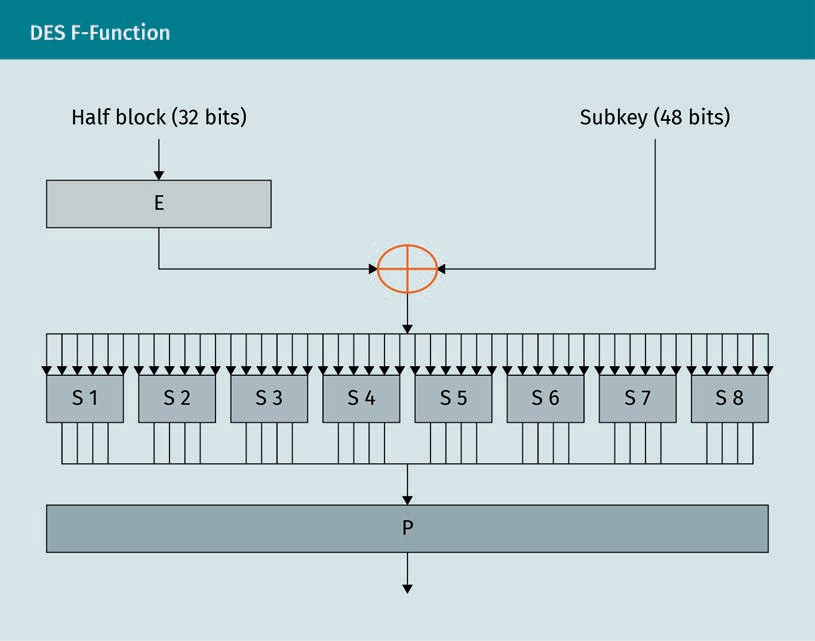
standard This block cipher has a key length of 56 bits. It was con- ceived by Horst Feis-

tel from IBM.

The data encryption standard (DES) became a public standard in 1977 after it won a public competition announced by the US National Bureau of Standards (now known as the National Institute of Standards and Technology, NIST). The DES contains 16 itera- tions where substitution and permutation (transposition) are executed. Its block size is 64 bits, and the key consists of 64 bits. However, only 56 of these can be chosen by the user and constitute the key: the remaining eight are redundant parity check bits.

As the name of its inventor suggests, it is a Feistel cipher, or substitution and permuta- tion network, similar to the prototype discussed above. It groups the text (a byte sequence) into 32-bit blocks with sub-blocks of four bits and enciphers each block in 16 iterations of the following three steps, called the Feistel function, or F-function, in the given order

1. Add (XOR) the key,
2. Substitute each four-bit sub-blocks of the block by the S-box (in hexadecimal nota- tion), and
3. Permute all the sub-blocks.



Symmetric Cryptosystems

At each round i, the output from the preceding round is split into the 32 left-most bits, L(i) , and the 32 right-most bits, R(i). R(i) will become L(i + 1), whereas R(i + 1) is the output of a complex function, L(i) + f(R(i)*,* K(i + 1)) whose input is the i + 1-th block of the key bits, K(i + 1), and the entire preceding intermediate cipher. This process is repeated 16 times.

Essential for the security of DES is the nonlinear S-box of the F-function f speciﬁed by the NIST. It is both nonlinear, i.e., f(A) + f(B) ≠ f(A + B), and maximizes confusion and diffusion as intended by the Shannon criteria for a secure cipher.

###### Key Size and the Birth of 3DES

The security of the DES like any algorithm is no greater than the effort to search 256 keys. Considered an infeasible computational task in 1977, a special purpose DES search engine achieved this in three days two decades later. A workaround, called Triple DES or 3DES, was devised that effectively gave the DES with a 112-bit key double the key size of DES keys by just using two normal DES keys.

The encryption would be E(1) (D(2) (E(1) (1) encrypt with the ﬁrst key, (2) decrypt with the second key, and (3) encrypt with the ﬁrst key.

Decryption would be D(1) (E(2) (D(1) *(*1) decrypt with the ﬁrst key, (2) encrypt with the second key, and (3) decrypt with the ﬁrst key.

If the two keys coincide, then this cryptosystem becomes an ordinary single-key DES. Thus, triple DES is retroactively compatible with equipment implemented for (single) DES. DES was the ﬁrst cryptographic algorithm to fulﬁll Kerckhoff’s principle.

### Advanced Encryption Standard (AES)

In January 1997, the NIST announced a public contest for an advanced encryption standard (AES) to replace the former symmetric encryption standard, the Data Encryp- tion Standard (DES). Since improvements in computing power ﬁnding the ﬁxed 56-bit DES key with a brute-force attack, the NIST speciﬁcations for the AES demanded an increasing key length. The AES algorithm was the winner of this competition (Bauer, 2016).

###### Evaluation of Security

The creators of AES could demonstrate that these two operations complemented each other so well that, after several iterations, they almost compensate for the absence of a replacement of the entire block by another (Daemen & Rijmen, 1999; 2002). As was the

3DES

This term refers to the triple application of DES with double the key size of the DES algorithm.

Advanced encryption standard

This substitution and permutation network with a (vari- able) key length of usually 128 bits became a crypto- graphic standard in 2000, succeeding DES.

case with DES, the AES, decades after its introduction, still stands strong against any attacks of cryptanalysis, but is not expected to yield to developments in computing, as happened to the DES, given its adjustable key size.

###### Applications

Rijndael stood out for its simplicity and particularly computational economy in imple- mentation. Not only was it secure, but thanks to its elegant and simple design, also small enough to be implemented on smart cards (at less than 10,000 bytes of code). To this day, it is considered the safest algorithm. There is no need for another standard symmetric cryptographic algorithm. Indeed, it is used everywhere. For example, to encrypt a wireless network, one key is used, so the encryption is symmetrical. The saf- est option, and therefore most recommended, is encryption by AES.

###### Algorithm

AES is a block cipher, i.e., it operates on blocks of bytes, usually of 16 bytes (AES-128).

|  |  |
| --- | --- |
| SubBytes | Substitute each byte of the block according to a substi- tution table (S-box) with 28 = 256 entries of 1 byte. |
| ShiftRows | Shift the entries of the last three rows cyclically. |
| MixColumns | Add the columns between them. |

###### Encipherment in Blocks

The AES algorithm groups the plaintext (and the keys) into 4×B byte rectangles where B is number of columns in the block: 4, 6, or 8. Commonly, and for us from now on, B = 4, i.e., the rectangle is a square. On a hexadecimal basis, such a square has for instance the shape

A1 13 B1 4A A3 AF 04 1E 3D 13 C1 55

B1 92 83 72

Symmetric Cryptosystems

###### The Rijndael Binary Field

To understand the AES, some mathematical deﬁnitions of groups and ﬁelds, as well as a basic knowledge of calculations with polynomials, are necessary. The mathematics below are meant to explain the AES functions SubBytes, ShiftRows, and MixColumns in more detail.

Groups and ﬁelds

A group is a set G with

* an operation • : G × G → G that satisﬁes the associativity law,
* a neutral element e such that x • e = x = e • x for all x in G), and
* an inverse element y for every element x such that x • y = e = y • x.

Generally, the operation is denoted by •, the neutral element by 1, and the inverse of x

in G by x-1.

Example

The set of nonzero rational numbers \*\* with the multiplication operation • is a group. If the group G is commutative (the operation satisﬁes the commutativity law), then com- monly the operation is denoted by +, the neutral element by zero, and the inverse ele- ment of x in G by –x.

Example

The set of rational numbers \* with the addition operation + is a commutative group. A “ﬁeld” is a set F with an addition and multiplication operation + and • such that

* the set F with + is a commutative group,
* the set F\* = F - { 0 } with • is a commutative group, and
* the distributivity law is satisﬁed.

Example

The set of rational numbers \* with addition + and multiplication • is a ﬁeld.

Bytes as polynomials with binary coefﬁcients of degree 7

A byte, a sequence b7 … b1, b0 of eight bits in {0, 1} is considered a polynomial with binary coefﬁcients by

b7 . . . b1b0 b7X7 + ! + b1X+ b0

For example, the hexadecimal number 0 x 57, or binary number 0101 0111, corresponds to the polynomial x6 + x4 + x2 + x + 1.

All additions and multiplications in AES take place in the binary ﬁeld IF28 with 28 = 256 elements, which is a set of numbers with addition and multiplication that satisﬁes the associativity, commutativity and distributivity law (e.g., \*) deﬁned as follows. Let

IF2 = 0,1

be the “ﬁeld of two elements” with the addition of 1 + 0 = 0 + 1 = 1 and 0 + 0 = 1 + 1 = 0 (which is the ⊕ addition given by XOR), and (the natural) multiplication 1 • 0 = 0

• 1 = 0 • 0 = 0 and 1 • 1 = 1.

Let

IF2 X = +/2+ = the polynomials on IF2,

that is, the ﬁnite sums a0 + a1 X + a2 X2 + ·+ an Xn to a0, a1, …, an to IF2 and be

IF28: = IF2 X / X8 + X4 + X3 + X + 1 .

That is, the result of both operations + and • in IF2[X] is the rest of the division by X8 + X4 + X3 + X + 1.

Addition

The + addition of two polynomials is the addition in IF2 coefﬁcient to coefﬁcient. That is, as bytes, the + addition is given by the XOR addition.

Multiplication

The multiplication • is given by the natural multiplication followed by the division with rest by the polynomial

m x = x8 + x4 + x3 + x + 1 .

For example, in hexadecimal notation, 57• 83 = C1 because

x6 + x4 + x2 + x + 1 x7 + x + 1 = x13 + x11 + x9 + x8 + x6 + x5 + x4 + x3 + 1

and

x13 + x11 + x9 + x8 + x6 + x5 + x4 + x3 + 1 = x5 + x3 + 1 x8 + x4 + x3 + x + 1 + x7

+ x6 + 1

The multiplication by the polynomial 1 does not change anything: it is the neutral ele- ment. For any polynomial b(x), Euclid’s extended algorithm, calculates polynomials a(x) and c(x) such that

Symmetric Cryptosystems

b x a x + m x c x = 1 .

That is, in the division with the remaining a(x) b(x) for m(x) left over 1. This means that

a is the inverse multiplicative in IF 8,

2

b−1 x = a x in IF28 .

When we invert a byte b into IF 8, we mean byte a = b–1.

2

###### Rounds

The AES enciphers each block iteratively, in rounds. Let R be the number of rounds, which depends on B. There are R = 10 rounds for B = 4 columns, R = 12 rounds for B

= 6 columns, and R = 14 rounds for B = 8 columns.

For us, R = 10. In these rounds, keys are generated and the plaintext is replaced and transposed by the following operations:

|  |  |  |
| --- | --- | --- |
| 1. | Round r = 0 | AddRoundKey to add (by XOR) the key to the plain- text block |
| 2. | Rounds r = 1,  …, R – 1 | To encrypt, apply the following functions   1. SubBytes to replace each byte (= sequence of eight bits) with a better distributed sequence of bits 2. ShiftRows to permute the entries of each line of the square 3. MixColumn to add the boxes of the columns in the square between them 4. AddRoundKey to generate a key from the previous round’s key and add it (by XOR) to the square |
| 3. | Round r = R | To encrypt, apply the following functions   1. SubBytes 2. ShiftRows 3. AddRoundKey |

Compared to previous rounds, the MixColumn function is omitted. It turns out that Mix- Column and AddRoundKey, after a slight change of AddRoundkey, can change the order without changing the result of both operations. In this equivalent order, the operation MixColumn does not increase cryptographic security, as the last operation invertible without knowledge of the key. Thus, the last operation MixColumn can be omitted.

The CrypTool 1 offers in its menu “Individual Procedures > Visualization of Algorithms > AES” an “animation” entry of the rounds, and an “inspector” entry to experiment with the values of plaintext and key. The functions used are described in more detail below.

SubBytes

SubBytes substitutes each byte of the block by another byte given by the S-box substi- tution table.

To calculate the value of the entry by which the S-box substitutes each byte:

* + 1. Calculate its multiplicative inverse B in IF 8,

2

* + 1. Calculate ai = b1 + bi+4 + bi+5 + bi+6 + bi+7 + ci

where i = 0, 1, …, 7 is the index of each bit of a byte, and

* + B = b7 b6 b5 b4 b3 b2 b1 b0 is the entry byte,
  + A = a7 a6 a5 a4 a3 a2 a1 a0 is the output byte of the operation and
  + c is the constant byte 01100011.

In matrix form,

a0 b0

1 0 0 0 1 1 1 1 1

a1 1 1 0 0 0 1 1 1 b1 1

a2 1 1 1 0 0 0 1 1 b2 0

a3 1 1 1 1 0 0 0 1 b3 0

= +

a4 1 1 1 1 1 0 0 0 b4 0

a5 0 1 1 1 1 1 0 0 b5 1

a6 0 0 1 1 1 1 1 0 b6 1

0 0 0 1 1 1 1 1 0

b

a7 7

ShiftRows

ShiftRows shifts the line number l of the square l positions to the left (where line num- bers are counted starting from zero, that is, l runs through 0, 1, 2, and 3 and the shift is cyclic). *N.B.* the ﬁrst line is not shifted.

MixColumn

MixColumn multiplies each column of the block by a ﬁxed matrix:

Symmetric Cryptosystems

* if Bj (with coefﬁcients b0,j, b1,j, b2,j and b3,j) corresponds to the column j of the input block, and
* if Aj (with coefﬁcients a0,j, a1,j, a2,j and a3,j) corresponds to the column j of the out- put block of the operation,

then

a0,j

a1, j

=

a2, j

a3, j

02 03 01 01

01 02 03 01

01 01 02 03

03 01 01 02

b0,j

b1, j

.

b2, j

b3, j

For example, the byte a0,j is computed by

a0,j = 2 · b0, j + 3 · b1, j + b2, j + b3, j .

AddRoundKey

AddRoundKey adds, by the XOR operation, the key W(r) of the current round r to the B square of the ciphertext that is,

B ⊕ W r .

The key is generated column by column. We denote them by W(r)0, W(r)1, W(r)2 and

W(r)3, i.e.,

W r = W r 0 ∣ W r 1 ∣ W r 2 ∣ W r 3 .

Since the key has 16 bytes, each column has four bytes. That is, the last column of the previous round key, plus the result of ScheduleCore, is applied to the ﬁrst column of the previous round key (which we denote by T). Here ScheduleCore is the composition of transformations:

* SubWord substitutes each of the 4 bytes of T according to the S-box of SubBytes.
* RotWord shift T one byte to the left (in a circular manner, that is, the last byte becomes the ﬁrst).
* Rcon(r) adds (by XOR) to T the constant value, in hexadecimal notation, [(02)r–1 00 00 00] (where the power, i.e., the iterated product, is calculated in the Rijndael ﬁeld IF 8). The only byte that changes is the ﬁrst by adding either the value 2r–1 (for r ≤ 8) or the value 2r–1 in F 8 for r = 9, 10.

2

2

1. The ﬁrst round key W(0) is given by the initial W key.
2. For r = 1, …, R (where R is the total number of rounds, R = 10 for us), the four col- umns W(r)0, W(r)1, W(r)2 and W(r)3 of the new key are generated from the columns of the old W(r–1) key as follows:
   1. The ﬁrst column W(r)0 is given by

W r 0 = W r − 1 0 ⊕ ScheduleCore W r − 1 3 ;

* 1. The next columns W(r)1, W(r)2 and W(r)3 are given, for i = 1, 2 and 3, by

W r i = W r i − 1 ⊕ W r − 1 i;

* 1. That is, the previous column of the current round key plus the current column of the previous round key.

###### Diffusion

We note that the only transformation that is not afﬁne is the inversion in the ﬁeld IF 8 in the SubBytes operation. In the operation SubBytes, the following are applied: the inversion in IF 8, a linear application, and the translation by a constant vector (a) Shif- tRows is a linear permutation, (b) MixColumn is a linear addition, and (c) AddRoundKey is the translation by the round key.

2

2

Regarding the goals of ideal “diffusion” and “confusion,” notice that in each step about half of the bits (in SubBytes) or bytes (in MixColumn and ShiftRows) are replaced and transposed. The complementarity of the simple operations for high security, that is, their generation of high confusion and diffusion after a few iterations is evident through the substitution of the alphabet and the permutation of text. For the latter, this is noticeable in particular in the permutation between the entries of every line, and of the permutation between the columns. The Cryptool by Forma Estudio (2007) exempli- ﬁes this characteristic.

We see how this small initial difference spreads out, generating very different results after just four rounds! This makes plausible the immunity of AES against differential cryptanalysis.

In case all key and plaintext entries are equal to 00, we also understand the impact of adding the Rcon(r) constant to the key in each round. That’s where all the confusion comes from!

### Cryptographic Hash Functions

Let us recall that a hash function transforms arbitrary data (such as a ﬁle), i.e., a varia- ble-length string into a ﬁxed-length string (which is usually 16 or 32 bytes long). It transforms a large amount of data into a small amount of information.

Symmetric Cryptosystems

###### Hash as ID

A hash function should be applicable to all possible ﬁxed-length sequences and simi- lar to a uniformly random variable, i.e., the probability of each of the values of the function is the same.

If, for example, the output has 256 bits, then ideally each value should have the same probability 2–256. That is, the output identiﬁes the input practically uniquely (with a col- lision chance of ideally 2–256). So one might think of a data hash, for example, from a ﬁle as its ID card. A hash identiﬁes many data with little information.

Since the length of the hash sequence is limited (rarely ≥ 512 bits), and the length of the input sequence is unlimited, collisions exist, i.e., equal hashes from different ﬁles. However, the algorithm minimizes the probability of collisions by distributing their val- ues as evenly as possible, or intuitively, makes them as random as possible. More accu- rately, every possible ﬁxed-length sequence is a value, and the probability of each of the values is the same.

###### Cryptographic Hash Functions

For a cryptographic or one-way hash function the algorithm should resist

1. The creation of an inverse image, i.e., the function is unidirectional. Given an output, the quickest way to ﬁnd an input with this output is by brute force, i.e., proving all possible inputs.
2. The creation of a second reverse image, i.e., given an input, the quickest way to ﬁnd another input with the same output is by brute force.
3. The creation of collisions, i.e., the fastest way to ﬁnd two (arbitrary) entries with the same output is by brute force.

According to Kerckhoff’s principle, the algorithm should also be public. In practice, the most important characteristic is resistance against the attack to create a second reverse image, and the least important is that against collisions attacks. There are sev- eral algorithms (e.g., MD4, MD5, and SHA-1) that do not resist collisions attacks but are still in use. For example, the CRC algorithm is a (not cryptographic) hash function. Other common cryptographic hash functions include SHA-256 and SHA-3.

For example, the output of the hash function SHA-256 of IUBH is 0a65cfb2c33b58749182d3eb965155336a52219793ff633016cb3942b2809a3f. The output of a hash function, but also message digest or digital ﬁngerprint, depending on the input, are used, for example, for message integrity and authentication.

###### Common Cryptographic Hash Functions and Applications

The most commonly used (cryptographic) hash algorithms even today are 16 bytes such as MD4, MD5, or SHA-1, which uses 20 bytes. Although none of these withstands colli- sion attacks, they remain popular. Their implementation details are described in request for comments (RFC). In a text ﬁle, an RFC publicly speciﬁes the details of a pro- posed internet standard or of a new version of an existing standard and are commonly drafted by university and corporate researchers to collect feedback. An RFC is discussed online and in formal meetings of the working group tasked by the Internet Engineering Task Force (IETF). For example, networking standards such as IP and Ethernet have been documented in RFCs.

The Message-Digest algorithm (MD4) was developed in 1991 by Ron Rivest, one of the three creators of the RSA algorithm. Described in RFC 1320, it is fast but vulnerable to pre-image creation.

RSA Data Security developed MD5. Described in RFC 132, it is vulnerable to collisions, but not to creating a second reverse image. It is often used for integrity checks, by soft- ware with peer-to-peer protocol (P2P) and as password storage.

The secure hash algorithm (SHA-1) was developed by NIST. It is vulnerable to collisions, but not to creating a second reverse image. The more recent versions, such as SHA-256 and SHA-3, are recommended over the ancient SHA-1 or MD4 and MD5.

Digital signatures

If the roles of the public and private key are ﬂipped, then the encryption is a digital signature. While the encrypted message will no longer be secret, every owner of the public key can check whether the private key encrypted the original message. However, in theory, signing a ﬁle (using the RSA algorithm) that was encrypted using the RSA algorithm would decrypt it. In practice, just as a signature is sufﬁcient to unequivocally identify the signed ﬁle (but its content often secret, for example, when signing a secret key), usually a cryptographic hash is encrypted by the private key.

(H)MAC

For message authentication, a message authentication code (MAC) is used. If this MAC is secured by a hash function, it is called a keyed-hash message authentication code (HMAC). The American National Standards Institute (ANSI) elaborated a message digest, a hash of a message, with a password as hash-key using DES algorithm as a new stand- ard in cryptography. Thus, a HMAC is generated by a message and a secret key used in a cryptographic hash function, such as SHA-1 or MD5. The MAC value protects the integrity and authenticity of a message by allowing intended receivers to detect any changes to the message content. Finally, the HMAC delivers to a user both an integrity and an authenticity check (Krawczyk et al., 1997).

Symmetric Cryptosystems

Data storage and integrity

Hash functions (both cryptographic and non-cryptographic) are used for fast data query, i.e., at ﬁxed time, in a hash table, and in a Merkle tree. They are also used to ensure the integrity of a ﬁle in the face of accidental modiﬁcations, i.e., to detect differ- ences between the ﬁle and a reference version.

Passwords

Cryptographic hash functions are used to distribute values evenly (key stretching), intuitively making them less predictable, known as Key Derivation Function (KDF), and to generate and store passwords, as PBKDF (Password Based Key Derivation Function). Cryptographic hash functions are also used to ensure the integrity of a ﬁle against tam- pering, that is, to detect differences between the ﬁle and a reference version (typically, the one before the ﬁle is transported). In particular, to ensure the authenticity of a ﬁle, to detect differences between the ﬁle and a version that was under the control of a speciﬁc person.

Note the difference between authenticity and authentication. The former guarantees the equality of data received and sent from a person, e.g., in the digital signature, the latter the identity of that person, e.g., in a secure site access.

The cryptographic hash algorithms listed above distribute the values evenly, but are fast, so they’re not appropriate for password creation because they are vulnerable to brute-force attacks. To prevent these, the PBKDF algorithms are (a) deliberately slow, such as bcrypt, (b) deliberately require a lot of memory to compute, such as scrypt algorithm, or (c) used only once for each entry, guaranteed by a salt, an additional, unique, usually random, argument. Without salt, the algorithm is prone to attacks by “rainbow tables,” tables of the hashes of the most common passwords.

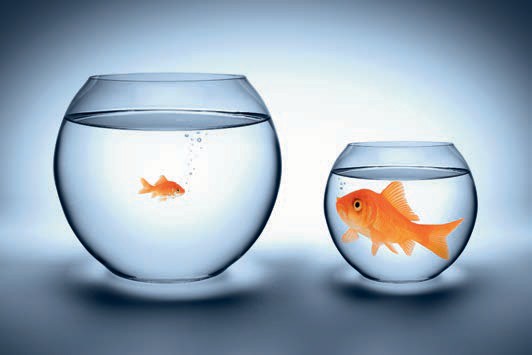
Summary

There are two basic operations in ciphering: transposition and substitution. Trans- positions rearrange the text without changing symbols; substitutions replace the symbols of the text without changing the order. Feistel ciphers are insecure because they preserve the statistical data of the plaintext. In contrast, modern ciphers combine substitution and permutation ciphers, known as substitution and permutation networks.

While no method of cryptanalysis is faster than an exhaustive key search, modern ciphers such as DES (1976) or AES (2000), in practice achieve the same. In other words, no cryptanalytic method faster than exhaustive key search is known. The key criterion is high diffusion as deﬁned by Shannon.

Hash functions transform arbitrary data of variable-length into a ﬁxed-length out- put. The output hash should be well diffused: the inversion of one input bit causes changes to about half of the output; applicable to all possible ﬁxed-length sequen- ces; and similar to a uniformly random variable, i.e., the probability of each of the

values of the function is the same. For a ﬁxed length output of 256 bits ideally each value should have the same probability 2-256. A hash function is cryptographic (or one-way) when reversion is computationally infeasible and when similar data yield dissimilar hashes. The recent versions SHA-256 and SHA-3 are recommended.



# Unit 3

## Asymmetric Cryptosystems

#### STUDY GOALS

On completion of this unit, you will have learned …

… the advantages and limitations of asymmetric cryptography.

… about trapdoor functions and modular arithmetic.

… the most common asymmetric cryptographic algorithms and their underlying trapdoor functions.

DL-E-DLMCSEAITSC02-U03

1. Asymmetric Cryptosystems

### Introduction

The big practical problem of single-key cryptography is key distribution, to secretly pass the same key to all correspondents, usually far away from each other, before they can communicate securely. Also problematic is the large number of keys needed for a group of correspondents to communicate securely.

In 1976, Difﬁe and Hellman found that the key distribution problem could be solved by an algorithm that satisﬁed (computationally)

* + easy creation of a matched pair of keys for encryption and decryption,
  + easy encryption and decryption,
  + infeasible recovery of one of the keys despite knowledge of the algorithm, the other key, and any number of matching plaintext and ciphertext pairs, and
  + infeasible recovery of the plaintext for almost all keys k and messages x.

A user of this algorithm, Alice keeps her decryption key secret but makes her encryp- tion key public, for example, in a public directory. Anyone who wishes to communicate securely with Alice need only to look up her public key to send her a ciphertext that only she can decrypt, i.e., a message can be encrypted without any need for secrecy. Everyone who uses the corresponding public key can decipher a ciphertext encrypted with Alice’s secret key, i.e., a sender can be identiﬁed without any need for secrecy.

The security of two-key cryptographic algorithms relies on the computational difﬁculty of a mathematical problem, e.g., factoring a number that is the product of two large primes. Ideally, computing the secret key is equivalent to solving the hard problem so that the algorithm is at least as secure as the underlying mathematical problem is difﬁ- cult. This has not been proven for any of the standard algorithms, although it is believed to hold for each of them.

###### Arithmetic Trapdoors

To see the usefulness of (modular) arithmetic in cryptography, recall that asymmetric cryptography is based on a trapdoor function, which must be easily computable, but its inverse must be practically incomputable without knowledge of a shortcut: the key!

The ease of calculating the function corresponds to the ease of encryption, while the difﬁculty of calculating the inverse corresponds to the difﬁculty of decryption, i.e., inverting the encryption. For example, RSA uses as an encryption function the raising to an n-th power, and as a decryption function its inverse, the root extraction.

While both the function itself and even its inverse are easily computed using the usual multiplication of numbers, cryptographic algorithms (such as RSA) use modular arith- metic to entangle the computation of the inverse function without knowledge of the key.

Asymmetric Cryptosystems

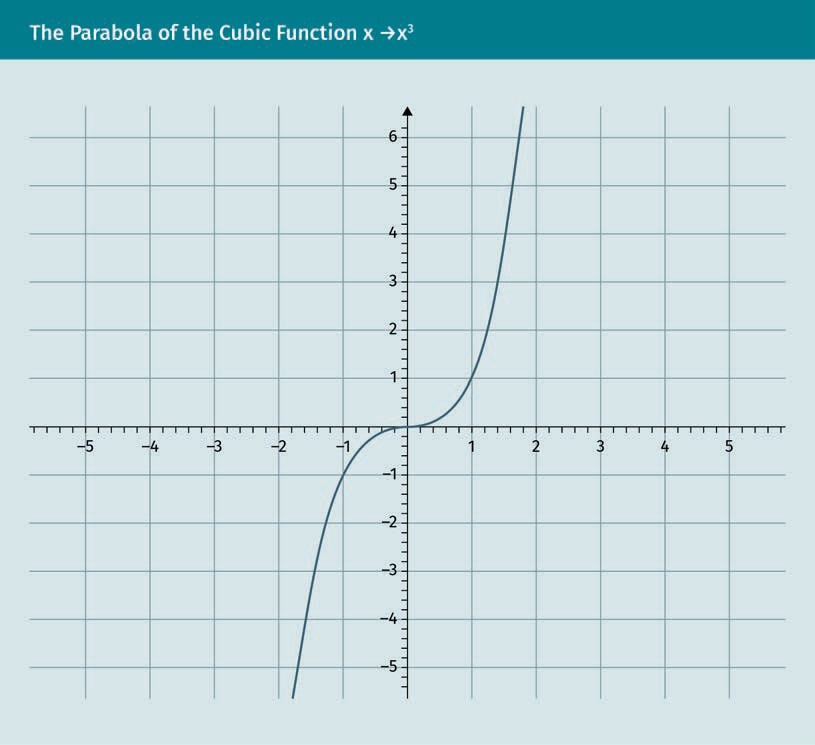
We already know modular or circular arithmetic from the arithmetic of the clock, where m = 12 is considered equal to 0. Because the indicator restarts counting from 0 after each turn, for example, three hours after 11 is two o’clock, 11 + 3 = 14 = 12 + 2 = 2. Over these domains, called ﬁnite rings, (the graphs of) these functions become irregu- lar and practically incomputable, at least without knowledge of a shortcut (the key).

Modular arithmetic as randomization

The difﬁculty of calculating the inverse corresponds to the difﬁculty of decryption, i.e., inverting the encryption. In an asymmetric cryptographic algorithm, the ease of encrypting a number, and the difﬁculty of deciphering a number are based on an inver- tible function such that it is easily computable, but its inverse function is difﬁcult to compute. For example, the inverses of the trapdoor functions raising to a power x /xe (in the RSA algorithm), and exponentiation x /gx (in the Difﬁe-Hellman algorithm) are given by the root extraction x /x1/e, and the logarithm log*g*. They are easily computa- ble in the domain of real numbers 0 (for example, by the bisection method for continu- ous functions thanks to the connectedness of 0), but on their ﬁnite cryptographic domains they are almost incomputable. Let us introduce these ﬁnite domains.

Functions on discrete sets

Their domain is not the set of integers + (or that of the real numbers 0 that includes them) because both functions, exponentiation and raising to a power, are continuous over 0.



If the domain of these functions were 0, their inverses could be approximated over 0, for example, by iterated bisection where the inverse point is besieged in intervals that are halved at every iteration.

Finite ring This is a ﬁnite set contains 0 and 1 over which a sum is explained that obeys the laws of associa- tivity and commuta-

tivity.

Finite rings

To avoid the iterative approximation of the zero and thus complicate the computation of the inverse function (besides facilitating the computation of the proper function), the domain of a trapdoor function is a ﬁnite ring denoted by

+/m+ = 0,1, . . . , m − 1

for a natural number m. In such a ﬁnite ring,

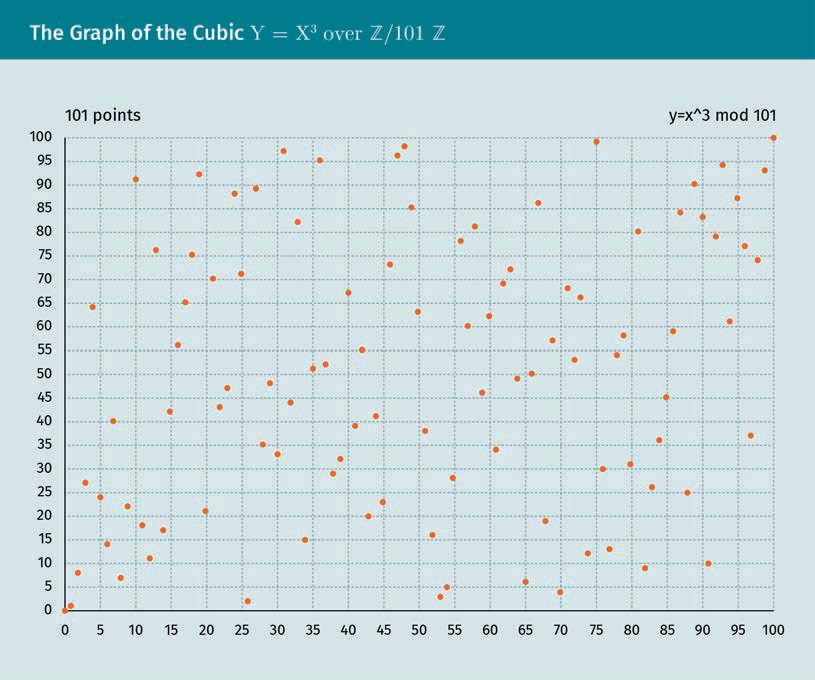
m = 1 + ! + 1 = 0

Asymmetric Cryptosystems

and necessarily every addition (and thus every multiplication and every raising to a power) has result < m. So + / m + its addition is different from that on + (or 0). For example, for m = 7,

22 = 2 · 2 = 4 and 32 = 3 · 3 = 7 + 2 = 0 + 2 = 2 .

We will introduce these ﬁnite rings ﬁrst by the examples + / 12 + and + / 7 + the ring given by the numbers of the hours on a clock and of weekdays), then for any m.



When we look at the graphs of the functions that are so regular on 0, we note that both, over the ﬁnite ring + /101 +, the parabola is initially as regular over + / 101 + as over +. However, starting from x = 11 (because 112 = 121 > 100), it begins to behave erratically (except for the symmetry of the parabola on the central axis x = 50.5 due to the equality ((–x)2 = x2).

Two integers a and b are congruent modulo m

a ≡ b mod m

if m | a – b, that is, if their difference a – b is divisible by m.

Congruence

Two integers a and b are congruent mod- ulo m if they leave the same remainder after division by m.

The number m is the modulus. Alternatively, phrased differently, a ≡ b mod m if a and

b leave the same remainder divided by m.

### The Difﬁe-Hellman Key Exchange

Difﬁe-Hellman key

exchange This protocol was an overt agreement on a common secret key whose security relied on the infeasi- bility of computing the logarithm mod- ulo a large number.

The ﬁrst published protocol to overtly agree on a mutual secret key is the Difﬁe-Hell- man key exchange protocol (Difﬁe & Hellman, 1976). This is not yet two-key cryptogra- phy because both correspondents know the single secret key. The asymmetric crypto- graphic algorithms that build on this protocol (e.g., El Gamal and ECC) generate a unique key for every message.

###### Key Exchange Protocol

Let us denote in every asymmetric encryption a public number with an uppercase letter and a secret number with a lowercase letter. For two parties, Alice and Bob, to overtly agree on a secret key, they ﬁrst combine an appropriate prime number p (the modu- lus), and an appropriate natural number g (the base).

First, Alice generates one half of the key by (1) choosing a number a, (2) calculating A

≡ga mod p, and (3) transmittting A to Bob. Next, Bob generates the other half of the key by (1) choosing a number b, (2) calculating B ≡gb mod p, and (3) transmitting B to Alice. The secret mutual key between Alice and Bob is

b a

b a aab ba b

c ( A mod p = g mod p mod p = g mod p = g mod p = g mod p mod p = B

mod p .

This protocol shows how to build a mutual secret key overtly. This key can then be used to encrypt all further communication, such as AES. However, it shows neither how to encrypt a message, nor how to sign one.

###### Security

The security of the Difﬁe-Hellman key exchange relies on the difﬁculty of computing the logarithm modulo p. An eavesdropper would obtain the secret key Ab = Ba from A and B, if they could compute

a = loggA or b = loggB mod p .

While a power is easily computable, even more so in modular arithmetic, its inverse, the logarithm, the exponent for a given power, is practically incomputable for appropri- ate choices of p and g.

Asymmetric Cryptosystems

###### Logarithm

Since initially (for x < logg p) the values gx over + / p + equal the values gx over +, the secret numbers a and b should be large enough, that is, > logg p. To ensure this, in practice these numbers are artiﬁcially increased, that is, the message is padded.

At present, the fastest algorithm to calculate the logarithm x from gx, is an adaption of the general number ﬁeld sieve (Gordon, 1993). Roughly, the number of operations to calculate the logarithm of an integer of n bits is

exp logn1/3 .

### The RSA Algorithm

The best-known public-key algorithm is the Rivest-Shamir-Adleman (RSA) cryptoalgor- ithm (Rivest et al., 1978). A user secretly chooses a pair of prime numbers p and q so large that factoring the product N = p q is beyond estimated computing powers for the lifetime of the cipher. The number N will be the modulus, that is, our trapdoor function will live on + / N + = { 0, 1, …, N-1 }.

N is public, but p and q are not. If the factors p and q of N were known, then the secret key could be easily computed. For RSA to be secure, the factoring must be computa- tionally infeasible, currently 2048 bits. The difﬁculty of factoring roughly doubles for each additional three digits in N.

The RSA algorithm creates a public key to encrypt, and a private key to decipher. Com- pared to the Difﬁe-Hellman protocol, this algorithm has the advantage of being com- pletely asymmetric. There is no need to share a mutual secret key (and the secret key is kept in a single place only). Instead, a single correspondent has access to the secret key. However, in this case, the communication is encrypted only towards the owner of the secret key. To encrypt in both directions, either each correspondent creates an asymmetric RSA key, or the other correspondent enciphers and sends a symmetric key (known as hybrid encryption, as it combines an asymmetric with a symmetric crypto- system).

###### Number Theory

Euler’s formula lets p and q be different prime numbers. If a is divisible by neither p

nor q, then

a p − 1 q − 1 ≡ 1 mod pq .

RSA

This algorithm encrypts by raising to a power. Its secur- ity relies on the computational infea- sibility of factoring a product of prime numbers.

Taking roots

Let p and q be different prime numbers, N = p q and φ(N) = (p – 1)(q – 1). For every exponent n such that

n ≡ 1 mod 3 N

we have

an ≡ a mod N for every integer a .

If m ≡1 mod φ(N), then by Euler’s Formula am ≡a mod N, that is, taking to the power is the identity function,

m· ≡ id mod N .

In particular, if m = E d is the product of two whole numbers E and d, that is,

Ed ≡ 1 mod 3 N ,

then

a = am = aEd = aE d .

That is, • d = • 1/E mod N. Calculating a power is much easier than a root!

Example

If p = 3 and q = 11 then

N = pq = 33 and 3 N = p − 1 q − 1 = 20 .

If E = 7 and d = 3, then n = Ed = 21 ≡1 mod 20. For example, for base two, we check

2E = 27 = 128 = 29 + 3 · 33 ≡ 29 mod N

and

29d = 293 = − 4 3 ≡ − 64 = 2 − 2 · 33 ≡ 2 mod N .

Therefore,

E 29 = 2 = 29d mod N .

Asymmetric Cryptosystems

Euclidean algorithm

A common divisor of two whole numbers a and b is a natural number that divides the two. The greatest common divisor of two whole numbers is the greatest natural number that divides the two. For example, the greatest common divisor of 12 and 18 is 6. Denote by mdc(a,b) the largest common divisor of a and b,

mdc a, b = the greatest natural number that divides a and b .

Iterated division with rest yields an efﬁcient algorithm to calculate the largest common divisor, i.e. the Euclidean algorithm.

Extended Euclidean algorithm

For the computation of the exponent of the decryption function, we need more infor- mation than the largest common divisor (calculated by the Euclidean algorithm). In the extended algorithm, one observes that in each step of the Euclidean algorithm the largest common divisor mdc(x, m) of x and m is a linear combination (or sum of multi- ples) of x and m, that is,

mdc x, m = λx + µm for integers x and m .

As an example, we have mdc(528, 220) = 44 and, indeed,

44 = 5 · 748 − 7 · 528 .

###### Encryption Algorithm

Recall that an uppercase letter denotes a public number (and vice-versa), whereas a lowercase letter denotes a secret number. We have a look at two parties: Alice secretly sends the message m to Bob through an insecure channel.

1. To generate the key, Bob chooses two prime numbers p and q, and an exponent E

relatively prime to φ(N) := (p – 1)(q – 1).

1. To transmit the key, Bob sends Alice the product N := pq (the modulus) and the exponent E (the public key).
2. To cipher, Alice calculates M = mE mod N, and transmits M to Bob.
3. To decipher, Bob calculates (with the Euclidean extended algorithm) d such that Ed

≡1 mod (p – 1)(q – 1) (because E is relatively prime to φ(N)), and calculates Md = mEd = m mod N (with Euler’s Formula).

In sum, raising to the power y = xE mod N encrypts where the exponent E is the public key. Correspondingly, its inverse, taking the E-th root x = y1/E mod N, decrypts. It is practically incomputable. However, modulo N, with Euler’s formula, there is d such that

y1/E = yd mod N

for a number d that the Euclidean algorithm calculates from E as well as p and q. Therefore, the secret key is d, or, sufﬁciently, the knowledge of the prime factors p and q of N.

###### Security

Since the modulus N, the exponent E*,* and the encrypted message M (= mE), are all public, the computational security of RSA is based solely on the difﬁculty of ﬁnding a root modulo a large number

m ≡ E M = M1/E mod N .

### Elliptic Curves

Denote IFp := + / p +. Among all curves, the clue of the elliptic curves (given by an equation y2 = x3 + ax + b) is that one can add points on them: p + q + r = 0 if a line passes for p*,* q, and r. By restricting the solutions to points (x, y) in (IFp × IFp) for a large prime number p and ﬁxing a point P on the curve while it is easy to compute the exponential, i.e., for n compute

Difﬁe-Hellman over elliptic curves

An analog of the Difﬁe-Hellman protocol, in which iterated multiplication of a number modulo p is replaced by iterated addition of a point on a ﬁnite elliptic curve. The Difﬁe- Hellman protocol (over IFp) has an analog over elliptic curves. Instead of *multiplying* repeatedly (n times) the base g in IF \*, i.e., computing

p

There is an advantage to using the logarithm over a ﬁnite elliptic curve (the function that for a given point G and Y determines the scalar x in 5 such that Y = x G) instead of the logarithm over IFp (the function that given numbers g and y determines the exponent x such that y ≡gx mod p). Depending on the number of bits n of p (regarding the fastest presently known algorithms), the time to compute the logarithm over an elliptic curve increases linearly and takes about n/2 operations, while the time to com- pute the multiplicative logarithm increases sublinearly and takes about n1/3 operations.

For example, the security obtained by a 2048-bit key for the multiplicative logarithm equals approximately that of a 224-bit key for the logarithm over an elliptic curve. A 512-bit key for an elliptic curve has a length of 15360 bits of an RSA key.

###### Elliptic Curves

An elliptic curve E over a ﬁnite ﬁeld (in which 0 ≠2, 3) is an equation

y2 = x3 + ax + b

Asymmetric Cryptosystems

for coefﬁcients a and b such that the curve is not singular, i.e., its discriminant is non- zero, 4 a3 + 27 b2 ≠ 0.

The equation y2 = x3 + ax + b is the form of Weierstraß, but there are several others that have proved to be computationally more efﬁcient, such as Montgomery

By2 = x3 + Ax2 + x where B A2 − 4 ≠ 0 .

If the characteristic is *2*, that is, IFq with q = 2n, then the equation is y2 + cxy + dy = x3

+ ax + b.

After choosing a domain (e.g., +, \*, 0, 6 or IFp for a prime number p) the points (x, y) that solve this equation, E(x,y) = 0, form a curve in the plane. This plane, 0 is for the usual Cartesian plane, + is for a lattice of points, and + / m + is for the ﬁnite lattice of points inside the square of length m whose bottom left corner is at the origin.

In addition to the points in the plane, there is also the point at inﬁnity (or ideal point) that is denoted as 0. Thus, the points of the elliptic curve are given by

E: = all points x, y such that E x, y = 0 ∪ 0

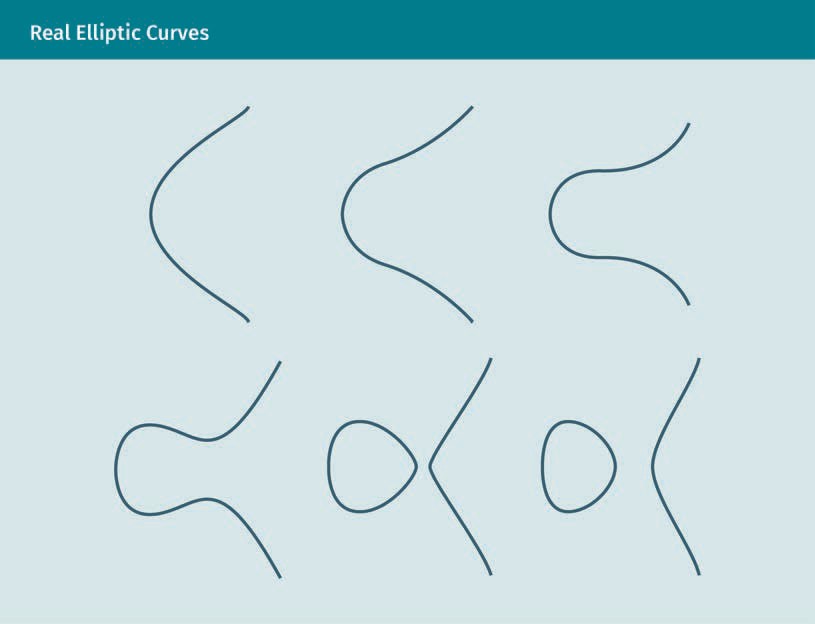
where the notion of point depends on the domain. On a ﬁnite ﬁeld IFq, the number of points # E is limited by q + 1 – t where t ≤ 2 √{q}, that is, asymptotically equal to # IF \* = q –1. Schoof’s (1999) algorithm can compute this in about n5 operations for n *=* log2 q the number of binary digits of q.

q

Continuous and discrete ﬁnite curves

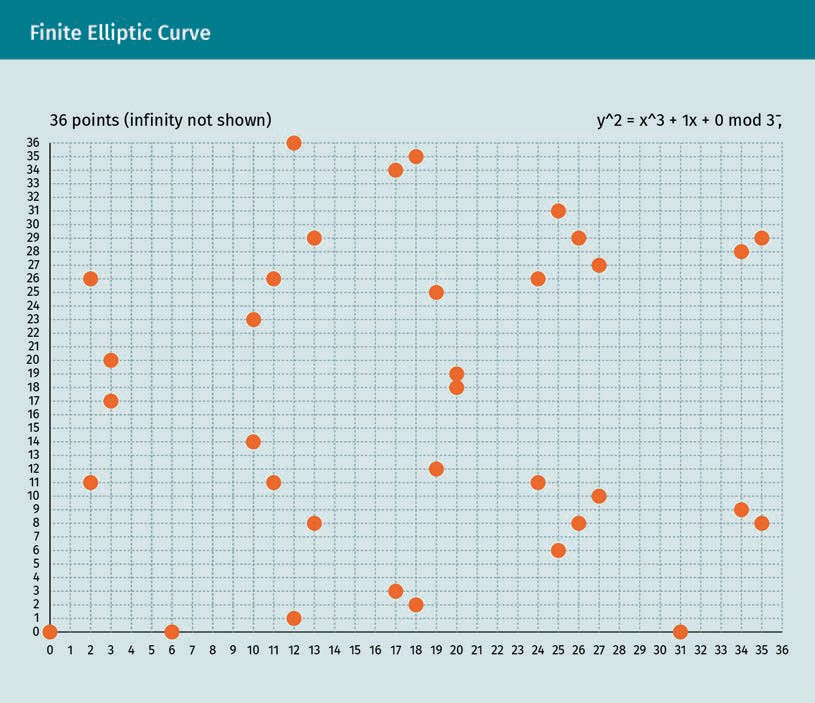
For the domain 0, the curves take the following forms in the real plane for different a

and b parameters:



While on ﬁnite ﬁelds, we obtain a discrete set of points (symmetrical around the mid- dle horizontal axis).

Asymmetric Cryptosystems



Curves in cryptography

For the cryptographic algorithm on this curve to be safe, that is the computation of the logarithm on it takes time, there are restrictions on the choices of q = pn and the ellip- tic curve (on its deﬁning coefﬁcients a and b). For example, as a guideline, q ≥ 2224 to be as secure as RSA for a key size of 2048 bits. A safe choice is, for example, the Curve25519 given by

y2 = x3 + 486662x2 + x

over IFq with q = p2 where p = 2255 – 19 (which explains its name). Its number of points is # E = 2252 + 277423177773585193790883648493. This curve became popular as an unbiased alternative to the recommended, and soon distrusted, curves from the NIST.

###### Key Exchange Using Elliptic Curves

ECC uses Difﬁe-Hellman key exchange to build a secret key, turn it into a cryptographic hash, and use it to encrypt communication by a symmetric cryptographic algorithm.

Encryption by the ECC is standardized by the ECIES (Elliptic Curve Integrated Encryption Scheme), a hybrid procedure (asymmetric cryptography with symmetric cryptography).

Once the mutually secret c key (a point on the ﬁnite elliptic curve) is agreed upon, Alice and Bob derive a key to a symmetric cipher such as AES or 3DES. The derivation func- tion that transforms a secret information into the appropriate format is called a Key Derivation Function (KDF). Such a standardized function is ANSI-X9.63-KDF with the SHA-1 option. For example, the TLS protocol uses the x coordinate of the point c, con- catenates to it numbers relating to the connection, and calculates a cryptographic hash of this concatenated number.

Let us transfer the Difﬁe-Hellman protocol from multiplication in a ﬁnite ﬁeld to addi- tion on a ﬁnite elliptic curve. Denote G a point on the curve, and

xG = G + ! + G

the x-fold iterated addition over the ﬁnite elliptic curve (instead of g and gx = g ·g for a ﬁnite ﬁeld).

Setup

The critical cryptographic number is the order n of the base point G which should be big enough.

Here an example of a base point. The elliptic curve Curve25519 with

y2 = x3 + 486662x2 + x

over IFq with q = p2 where p = 2255 – 19, uses as base point G = (xG, yG) uniquely determined by

Steps

1. In the ECDH (Elliptic Curve Difﬁe-Hellman) protocol, for Alice and Bob to overtly build a secret key, they combine a q power from a prime number p appropriate, an elliptic curve E appropriate over IFq, and a G point appropriate in E.
2. Alice, to generate one half of the key, chooses a number a, calculates A = a G, and

transmits A to Bob.

1. Bob, to generate another half of the key, chooses a number b, calculates B ≡b G, and transmits B to Alice.

The secret mutual key between Alice and Bob is

c: = bA = baG = abG = aB .

We note that for both of them to compute the same key c, the addition must satisfy the associative and commutative law, i.e., it is indispensable that E be a group.

The ECDHE protocol, where the additional ﬁnal E stands for “ephemeral,” uses the same key exchange as the ECDH protocol, but discards the keys (which are necessarily signed by permanent keys to testify the identity) after the session (Corbellini, 2015b).

Asymmetric Cryptosystems

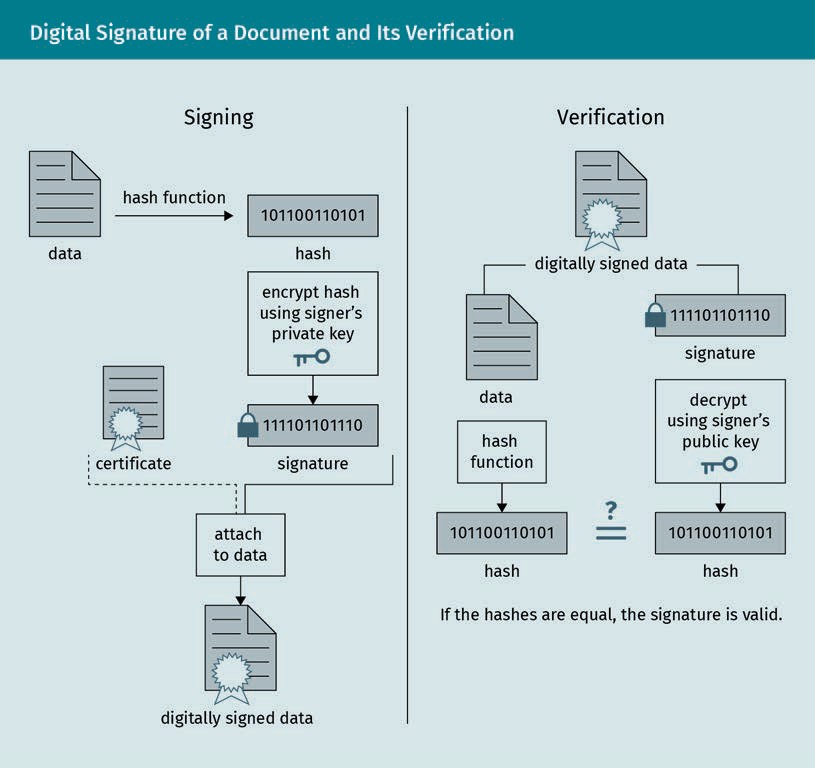
### Signatures

Again, public key cryptography uses two keys: a public key and a private key. Usually the public key is used to encrypt, while the private key is used to decipher. Thus, a text can be transferred from the encipherer (Alice) to one person only, the decipherer (Bob). The roles of the public and private keys can be reversed. The private key is used to encrypt, while the public key is used to decipher. Thus, the encipherer can prove to all decipherers (those who have the public key) their ownership of the private key: the digital signature.

The theory behind the encryption by the public key (digital messages) or private key (digital signature) is almost the same. Only the roles of the trap function arguments are reversed. For example, in the RSA algorithm, this exchange of variables is indeed all that happens. In practice, however, the private key and a cryptographic hash usually encrypt the paddings of the plaintext to avoid pathologies that reveal the key when the text is too short. While in public key encryption, the function used to ﬁrst transform the text (the padding) is easily invertible, in private key encryption, the function used to ﬁrst transform the text (the hash) is hardly invertible.

Digital signature This encryption of a message by the pri- vate key is followed by decryption by the public key to check whether the original message was

encrypted by the pri- vate key.



###### RSA Signature Algorithm

In the RSA signature algorithm, the only difference is that the exponents E and d

exchange their roles, i.e., the signed message is M = md (instead of mE).

Signing and the deciphering are both given by • d for the private key d. Thus, signing an encrypted document (for the public key E that corresponds to d) is equivalent to deci- phering it! In practice, different key pairs are used to encrypt / decrypt, to sign / verify, and a cryptographic hash h(d) of the document d is signed, i.e., a small number that identiﬁes the document.

We note that instead of the original message, it signs a cryptographic hash (e.g., with the algorithm MD5) of the original message, and with additional information, such as the name of the signer and the algorithms used to encipher and calculate the hash.

###### Digital Signature Algorithm (DSA)

ElGamal (1985) was the ﬁrst to show how to build an encryption and signature algo- rithm based on the Difﬁe-Hellman protocol. While the encryption algorithm is rarely employed (the standard cryptography command-line tool GnuPG offers it as a ﬁrst alternative to RSA), the signature algorithm is behind that of the Digital Signature Algo- rithm (DSA), which is used in the US government’s Digital Signature Standard (DSS), issued in 1994 by the NIST. Elliptic Curve DSA (ECDSA) is a variant of the DSA which uses points on ﬁnite (elliptic) curves instead of integers.

### Key Exchange and Public-Key Infrastructures

Compared to symmetric cryptography, asymmetric encryption avoids the risk of com- promising the key to decipher that is involved in exchanging the key with the cipherer. This secure communication with anyone through an insecure channel is a great advant- age compared to symmetric cryptography. Let us recall the classic methods to exchange a symmetric key, before looking at its asymmetric counterpart. While asymmetric cryp- tography made it possible to exchange a secret key overtly, this convenience obscures the identity of the key holder, making it prone to a man-in-the-middle attack which public-key infrastructure work around with the use of certiﬁcates (digital signatures by third parties of public keys).

###### Symmetric Cryptosystems

A symmetric key must be passed secretly. Possible methods are

Asymmetric Cryptosystems

* derivation from a base key using a Key Derivation Function (KDF), a cryptographic hash function which derives a secret key from secret—and possibly other public— information, for instance, a unique number,
* creation of a key from key parts held by different persons, for example, as an ana- logue to the one-time pad. If *s* is the secret (binary) number, then s = s1 ⊕s2 ⊕…⊕sn for the partial secrets s1, s2, … Reconstruction of *s* is only possible if all s1, s2, … are

combined.

* transmission through a different channel, for example, in person, a sealed letter, by telephone, or by quantum entanglement in which quantum particles are linked so that if the state of one changes, instantly the other changes, irrespective of spatial separation, even though information cannot travel faster than the speed of light.

###### Man-In-The-Middle Attack (MITM)

The great beneﬁt of asymmetric communication is that secrecy can be separated from authentication, i.e., for a created ciphertext nothing than the desired public key is nee- ded for deciphering. This goes along with the fact that there’s no information at all about the decipherer. For the communication of multiple parties, say Alice, Bob, and Eve, the public keys must be authenticated, i.e., the decipherers Bob and Eve must be authenticated in the directory of the cipherer Alice. If not, Eve could trick Alice into thinking she is communicating with Bob by substituting her public key with Bob’s.

In an MITM, the attacker places themselves between the correspondents, assuming toward each of them the identity of the other to intercept messages. As an example,

1. Bob sends his public key to Alice. Eve intercepts it, and sends Alice her own public key that claims Bob as its owner. If Alice sends a message to Bob, then she uses, without realizing it, Eve’s public key!
2. Alice enciphers a message with Eve’s public key and sends it to Bob.
3. Eve intercepts the message, deciphers it with her private key. She can read the mes- sage and alter it.
4. Eve enciphers the message with Bob’s public key.
5. Bob deciphers the message with his private key and suspects nothing.

Both Alice and Bob are convinced they have used each other’s public key, but they are actually using Eve’s!

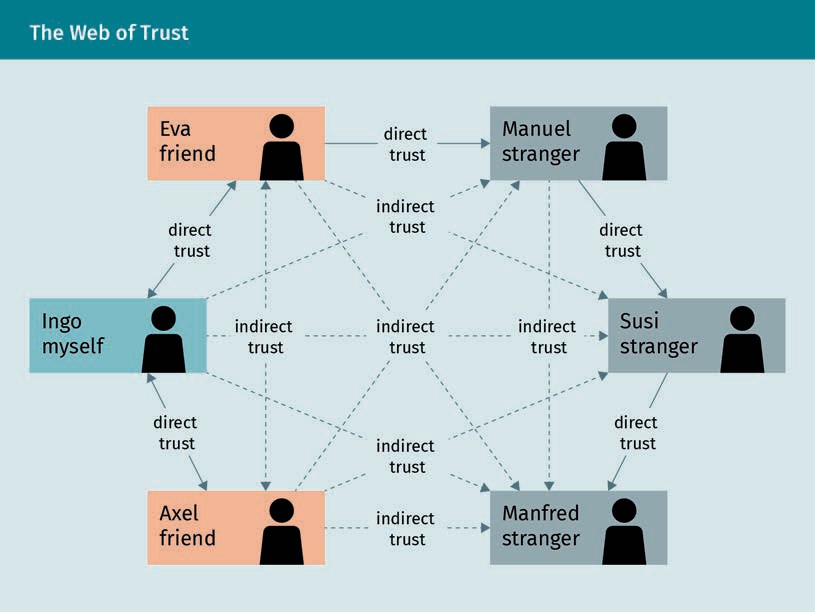
###### PKI

A public-key infrastructure (PKI) of a network establishes trust among its spatially sep- arated users by ﬁrst authenticating them, then authorizing their public keys by signing them (digital certiﬁcates) and distributing them. In institutions and corporations, a PKI is often implemented as “a trust hierarchy” of certiﬁcation authorities, whereas in looser communities, it can be decentralized and trust mutually established by the users themselves. A PKI includes

* + (digital) certiﬁcates. These are public keys signed to authenticate their users. Other than the name and key, they contain additional personal data, such as an email address, and, usually, an expiry date.
  + certiﬁcate revocation list (CRL). This is a list of certiﬁcates that have been revoked before their validity expires because the key has been compromised or the key owner is no longer trusted because of their departure.
  + directory service. This is a searchable database of the emitted certiﬁcates. Examples include a trust hierarchy an LDAP server (Lightweight Directory Access Protocol, a standard used by large companies to administer access of users to ﬁles, printers, servers, and application data) and the web of trust, a server that hosts a database searchable by a web form.

###### Philosophy of Solutions

Third parties, i.e., other identities with private keys that conﬁrm with their digital signa- tures, conﬁrm the identity of the key owner. However, the problem of the public key's identity arises again: How can we ensure the identities of the private key owners? There are two solutions: hierarchical authorities and the web of trust.



While in the web of trust the connections built by trust form a graph, in the approach by hierarchical authorities they form a tree.

Asymmetric Cryptosystems

Hierarchical authorities

In the hierarchical authorities approach, private key owners are distinguished by hier- archical levels. At the highest level lie the root authorities that we trust unconditionally. For example, VeriSign, GeoTrust, and Commode, are major US certifying companies. A look at the /etc/ssl/certs folder in the Linux distribution openSUSE reveals one Ger- man (TeleSec of Deutsche Telekom AG, the former national telecommunications opera- tor), three Spanish (Firmaprofesional, ACCVRAIZ1—*Agencia de Tecnología y Certiﬁcación Electrónica*, and ACC RAIZ FNMT—*Fábrica Nacional de Moneda y Timbre*), and many US authorities.

Web of trust

In the web of trust, private key owners cannot be distinguished from each other. The absence of root authorities, unconditionally trusted entities, is compensated for by the trust initially established by having obtained the public key personally (for example, at key-sign parties) or by having obtained the key through a different channel (website, email, etc.) and having communicated the checksum through another channel (phone, SMS, instant messenger, etc.). Trust is transitively passed in the established network.

###### Standardization of Philosophies on the Internet

On the internet, the system of trust by hierarchical authorities has been standardized by the scheme X.509, principally used to encrypt the communication between a user and a (commercial) website (but also between users in corporate environments, such as, S/MIME email encryption), and the OpenPGP scheme (as implemented by the GnuPG program), with its main use of encrypting emails. This scheme radically rejects any hierarchy. The user can publish a public key with an email address on a public-key server without even conﬁrming that they have access to this email address account.

###### DANE

The Internet Engineering Task Force (IETF) proposed (in RFC 63941: DANE use cases and RFC 66982: DANE protocol) the DANE protocol that aims to cryptographically harden the TLS, DTLS, SMTP, and S/MIME protocols using DNSSEC. With DNSSEC, a DNS resolver can authenticate a DNS resolution, that is, whether it is identical to that on the authorita- tive DNS server, by checking its signature (of the authoritative DNS server). Instead of relying, like these protocols do, entirely on certiﬁcate authorities (CAs), domain holders can restrict the CAs that validate the domain’s certiﬁcate, and can emit certiﬁcates for themselves, without reference to CAs.

Using CAs, there is no restriction on the certiﬁcates issued. If an attacker can gain con- trol of a single CA among the many CAs that the client trusts, then they can emit fake certiﬁcates for every domain. DANE allows clients to ask the DNS servers which certiﬁ- cates are trustworthy so that the domain holder can restrict the scope of a CA. When the user is passed a domain name certiﬁcate (as part of the initial TLS handshake), the client can check the certiﬁcate against a TLSA resource record (TLSA RR) published in the DNS for the service name which is authenticated by the authoritative DNS server.

Web of trust Private key owners

conﬁrm other’s iden- tities by successively passing trust to each other among equals.

Key-sign parties These are meetings where participants exchange and sign their public key mutually.

The most common standard for a PKI is the hierarchy of X.509 certiﬁcate authorities.

X.509 was ﬁrst published in 1998 and is deﬁned by the International Telecommunica- tions Union’s Standardization sector (ITU-T). X.509 establishes in particular a standard format of electronic certiﬁcate and an algorithm for the validation of certiﬁcation path. The IETF developed the most important proﬁle, PKIX Certiﬁcate and CRL Proﬁle (PKIX) as part of RFC 3280, currently RFC 5280. It is supported by all common web browsers, such as Chrome and Firefox, which come with a list of trustworthy X.509 certiﬁcate authori- ties.

In more detail, the TLSA RR contains the entry Certiﬁcate Usage, whose value restricts the authority allowed to validate the certiﬁcate for the user. The lower the level, the more restrictive it is.

* + Level 0: PKIX-TA (CA constraint). The client’s trust resides in a PKIX authority.
  + Level 1: PKIX-EE (Service Certiﬁcate Constraint). The client’s trust resides in a PKIX certiﬁcate.
  + Level 2: DANE-TA (Trust Anchor assertion). The client’s trust resides in an authority which, in contrast to PKIX-TA, does not have to be a PKIX certiﬁcate authority.
  + Level 3: DANE-EE (Domain-issued certiﬁcate). The client’s trust resides in a certiﬁcate which, in contrast to PKIX-EE, does not have to be a PKIX certiﬁcate.

The DANE check serves to conﬁrm certiﬁcates issued by public certiﬁcation authorities. With DANE values (two and three), the domain holder has the option of creating their own even self-signed certiﬁcates for their TLS-secured services, without having to involve a certiﬁcation authority known to the client. By choosing between “Trust Anchor” (TA) and “End Entity” (EE), the domain owner can decide for themselves whether to anchor DANE security to a CA or server certiﬁcate.

###### Hybrid Cryptosystems

Hybrid encryption A two-key algorithm is used to authenti- cate the correspond- ents by digitally signing the mes- sages or to exchange a key for single-key cryptography for efﬁ- cient communication

thereafter.

Between two parties it is common to create a hash, e.g., of a message, as encryption key. MITM is avoided by the authentication of the corresponding public keys at a certiﬁ- cation authority. Since single-key cryptographic algorithms are more efﬁcient than two- key cryptographic algorithms by a considerable factor, the main use of two-key encryp- tion is thus hybrid encryption where the two-key algorithm is used to authenticate the correspondents by digitally signing the messages, or to exchange a key for single-key cryptography for efﬁcient secure communication thereafter.

For example, in the TLS (Transport Layer Security; former SSL) protocol, which encrypts secure sites on the World Wide Web, a cryptographic package such as TLS\_RSA\_WITH\_3DES\_EDE\_CBC\_SHA (identiﬁcation code 0x00 0x0a) uses RSA to authen- ticate and exchange the keys, 3DES in CBC mode to encrypt the connection, and SHA as a cryptographic hash.

Asymmetric Cryptosystems

Summary

Symmetric cryptography suffers from the key distribution problem. Asymmetric cryptography solves this problem seemingly at once, by enabling the use of differ- ent keys to encrypt and decrypt. However, the identity of the key owner must be conﬁrmed. This can be done personally or by third parties, such as identities with private keys that conﬁrm ownership with their digital signatures. However, the problem of the private key identity arises again: How can we ensure the identities of these private key owners? There are two solutions. In the approach via hierarchi- cal authorities, private key owners are distinguished by hierarchical levels. At the highest level lie the root authorities which we trust unconditionally. In the web of trust, trust is transferred from one to the other.

Asymmetric cryptography relies on a trapdoor function, which must be easy, but its inverse must be practically incomputable without knowledge of a shortcut, the key!

The difﬁculty of calculating the inverse corresponds to the difﬁculty of decryption, i.e., inverting the encryption. Complication of the computation of the inverse func- tion is done using modular (or circular) arithmetic.

The most current cryptography widely in use uses elliptic curves. The Difﬁe-Hell- man protocol (over IFp) has an analog over elliptic curves. The advantage of using elliptic curves are shorter key sizes; small keys for the ECC achieve the same level of security as large keys for the RSA or DH. As an example, the security of a 224-bit key from the ECC corresponds to that of a 2048-bit key from the RSA or DH. This fac- tor in reducing key sizes corresponds to a similar factor in reducing computational costs.



# Unit 4

## Authentication

#### STUDY GOALS

On completion of this unit, you will have learned …

… about user authentication by password, personal identiﬁcation number (PIN), smart card, and biometric identiﬁers.

… how authentication is most securely achieved over distance.

… how the secret key is never revealed by challenge and response protocols.

… how information is not leaked by zero-knowledge proofs.

… how Kerberos mediates between users and servers without either one revealing their password to the other.

DL-E-DLMCSEAITSC02-U04

1. Authentication

### Introduction

Authentication is the identiﬁcation of a person or data or the conﬁrmation that the user is who they claim to be, for example, when logging into a server by entering a username and password. Authentication also checks that the message, such as an instruction sent to a bank by email, is authentic, i.e., unchanged between the time that the message was sent and its time of arrival.

This unit exclusively covers the former type of authentication, the identiﬁcation of a person, the user, which is particularly important on the internet as the person is far away. In this sense, identiﬁcation tells a computer or a network who the user is, usually by their user (or account) name. This is followed by authentication, which convinces a computer or network that a person is who they claim to be.

To authenticate, the user can use information that only they know as proof of identity. The user could give their password, a personal identiﬁcation number (PIN), or present an item, such as a software certiﬁcate containing public or private keys, a hardware cer- tiﬁcate, such as a token or a smart card, which has a microprocessor that stores a key and runs cryptographic algorithms, or a mobile device or email account that receives a code. There are also biometric identiﬁers, such as ﬁngerprints, facial recognition, or iris scanning.

Authentication should not be confused with authorization, the ﬁnal conﬁrmation of authentication that determines the user’s eligibility to access certain contents.

The authentication protocol can be simple, two-factor, one-way, or mutual. When a sin- gle proof sufﬁces such as a PIN or password, it is simple. It is two-factor, when more than one proof is necessary, for example, a PIN and a smart card. It is one-way if authentication is carried out in just one direction, e.g., party A authenticates them- selves to party B, or mutual, when authentication is bidirectional, e.g., likewise party B authenticates themselves to party A. Most operating systems (such as Linux) and appli- cations store a hash of the authentication data rather than the authentication data itself. During authentication, the hash of the authentication data entered must match the hash stored in order to be veriﬁed. Even if an intruder knows the hashes, it is prac- tically impossible to determine which authentication data matches a given hash.

### Passwords

Passwords are stored on a system data of authorized users. It is the most common approach for authentication: free, convenient, and private. It should be easy to memo- rize but difﬁcult to guess.

Authentication

###### Conveniences

Comparing authentication by what one knows (such as a password) to what one is (bio- metric data such as a ﬁngerprint), the advantages are that it does not require sophisti- cated hardware, it is securely stored, and it cannot be forged.

Comparing authentication by what one knows (such as a password) to what one has (such as a smart card), the advantages are that it does not have to be carried around, it is transparently stored, and it cannot be lost, stolen, or extorted.

###### Criticisms

The more meaningful, the more easily guessed (for example, a word in the user’s lan- guage), but the less of a pattern, the harder to remember. A compromise is a pass- phrase, i.e., a complete sentence instead of a single word. Though longer, its content is more meaningful and, thus, more easily remembered than a non-patterned sequence of symbols. To shorten it, the ﬁrst letter of each word is taken. For example, “Better to light one candle than to curse the darkness” can become “B2l1ct2ctd.”

Another workaround is a password manager. This program stores all passwords in a ﬁle encrypted by a master password. With this program, the passwords need no longer to be remembered and can be complex. This ease comes with risk: the master password is still a password whose exposure when used for an insigniﬁcant account endangers that of all other accounts, including the most critical ones.

###### Attacks

Attacks during the entry of another person’s password include

* spying while entering the password. Workarounds are inconvenient, for example, by masking the typed letters, covering the keyboard, or using a screen keyboard.
* keylogging
* login spooﬁng where one user’s account a login entry form is faked so that the next user’s entered login data are stored, display an error message and logging the ﬁrst user out.
* asking a user for the password, either through email or on the phone as a system administrator.
* asking a system administrator for the password by posing as a purported user who has forgotten their password.
* asking a user to change their password on a purported entry form.

Attacks on another person’s stored password principally exploits the following deﬁ- ciency: Since passwords have to be memorized, they tend to follow a pattern. For exam- ple, they can be built from (birth) dates and particular names. Common strategies include reversing the spelling or changing the capitalization of

Password

A secret sequence of letters attached to a user identity that grants access to a system (such as a computer) is known as a password.

* + the user name,
  + the ﬁrst, middle, or last name,
  + spouse, children, friends, or pet names,
  + telephone numbers or house addresses

These more likely candidates can be guessed ﬁrst. Alternatively, the attacker can use those already leaked.

### Challenge-Response and Zero-Knowledge Protocols

Challenge-response

protocol This protocol poses a task that can be solved only by a user who has the authen-

tication data.

Zero-knowledge pro-

tocol This protocol (origi- nally presented by Goldwasser, Micali, and Rackoff) proves knowledge of a secret but disclose no other informa-

tion.

Nonce Used in a crypto- graphic protocol (such as one for authentication), this stands for (a ran- domly generated) number used once.

Challenge-response protocols are usually one of two tasks. First, they can be a crypto- graphic hash function or an enciphering function of a symmetric cryptographic algo- rithm whose secret key is shared among the claimant and veriﬁer. The veriﬁer gener- ates a random number, and the claimant responds with the result of applying the hash function on that number. Alternatively, it can be a digital signature algorithm where the claimant signs with their private key a message generated by the veriﬁer, which the veriﬁer checks with the public key.

A zero-knowledge protocol goes, in theory, further, as it shows how a claimant can prove knowledge of a secret to a veriﬁer such that no other information (than this proof of knowledge) is disclosed (Goldwasser et al., 1985).

###### Challenge-Response

A challenge-response protocol poses a task that can be solved only by a user with additional authentication data. An example of a challenge would be some (randomly) generated value encrypted using the password for the encryption key. As an example response, a similarly encrypted value that depends on the original value is given, thus proving that the user could decrypt the original value. This exemplary protocol is very familiar on smart cards.

This (randomly) generated value is a nonce and avoids a replay attack where the exchanged data is recorded and sent again later.

For example, in CRAM-MD5 or DIGEST-MD5, the challenge is the (iterated) hash of the password. Randomly generated, the response is the hash of the password, and a value that depends on the original value. A possible scenario could be

1. A server sends a unique challenge value challenge to the client.
2. The client computes the response as a hash (challenge + secret) and sends it to the server.
3. The server calculates the expected value of response and veriﬁes that it coincides with the client’s response.

Authentication

Such encrypted or hashed information does not reveal the password itself, but may supply enough information to deduce it with a dictionary (or rainbow table) attack, i.e., by probing many values. Information that is randomly generated enters each exchange, a salt, such as the current time.

Nonce, salt, and IV (Initialization Vector) are all numbers that are (usually) randomly generated, disclosed, and used once in a cryptographic process to improve its security by making it unique. A nonce is a number used once in a cryptographic protocol to make the exchange unique. A salt is used as additional input to a hash function to make its input unique (so that the same original input hashed gives a different output). An initialization vector is a number used as additional input to an enciphering (of a block cipher) to make its input to make an enciphering of a block cipher’s input unique so that the same original input encrypted with the same key gives different output.

Challenge-response protocols such as those presented below are used, e.g., in object- relational databases such as PostgreSQL or email clients such as Mozilla Thunderbird.

###### Digest-MD5

Digest-MD5 was a common challenge-response protocol that used the MD5 hash func- tion speciﬁed in RFC2831. It was based on the HTTP Digest Authentication (as speciﬁed in RFC2617) and was made obsolete by Melnikov (2011).

###### Challenge-Response Authentication Mechanism (CRAM)

CRAM-MD5 is a challenge-response protocol based on HMAC-MD5, that hashes for mes- sage authentication using the MD5 hash function (Krawczyk et al., 1997). The RFC draft by Zeilenga (2008) recommends obsoleting it by CRAM. The CRAM steps are (1) the server sends the client a nonce, (2) the client is supposed to respond with HMAC (secret, nonce), and (3) the server again calculates HMAC (secret, nonce) and checks whether it coincides with the client’s response to be convinced that the client knew the secret.

The weaknesses of CRAM include that no mutual authentication, i.e., the server’s iden- tity is not veriﬁed, and that the used hash function MD5 is quickly computed, and thus facilitates dictionary attacks. Instead, key stretching, i.e., using a hash function that is deliberately computationally expensive is preferable. The ﬁnal weakness is the pass- word storage, where some implementations store the user’s plain password, while oth- ers (such as Dovecot) store an intermediate hash value of the password. While this pre- vents storage of the plain password, for authenticating with CRAM-MD5, knowledge of the hash value is equivalent to that of the password itself.

Salt

A (randomly) gener- ated number that adds to the input of a hash function to make its output unique especially if the additional input is secret information such as a password.

###### Salted Challenge-Response Authentication Mechanism (SCRAM)

Salted Challenge-Response Authentication Mechanism (SCRAM) is a challenge- response protocol for mutual authentication (Menon-Sen et al., 2010), that supersedes CRAM-MD5 (Zeilenga, 2008).

While in CRAM the client password is stored as hash on the server, now knowledge of the hash (instead of the password) sufﬁces to impersonate the client on further authentications. The burden has just been shifted from protecting the password to its hash. SCRAM prevents this by demanding additional information on top of the authen- tication information (StoredKey) stored on the server that was initially derived from the client’s password (ClientKey).

The advantages of SCRAM compared to older challenge-response protocols are that the authentication information stored on the server is insufﬁcient to impersonate the cli- ent (Menon-Sen et al., 2010). In particular, (a) a dictionary attack (rainbow tables) after an authentication-database leakage is prevented by salting the information, (b) the server cannot impersonate the client to other servers because it stores only partial authentication information, and (c) password reuse after data breach is prevented by binding the hash to a single server. Only the salted and hashed version of a password is used during login and the salt on the server is immutable.

SCRAM also supports mutual authentication (by the client and server).

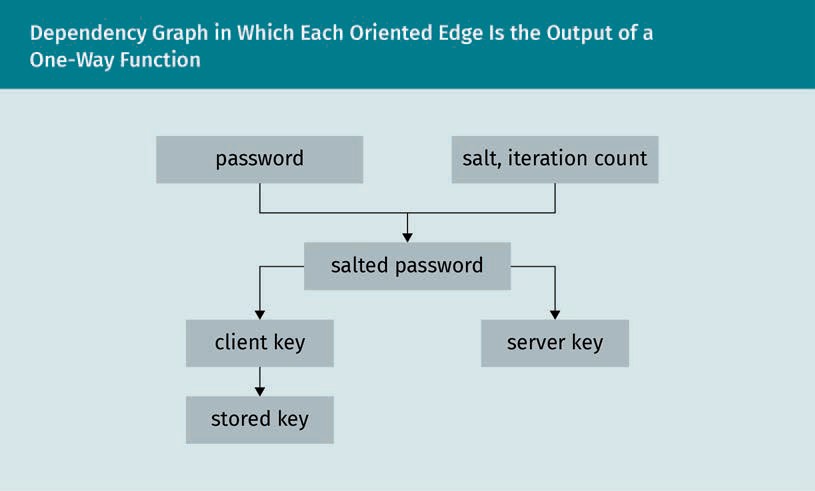
Key creation, transmission, and storage

When the client creates a password, the server stores derived keys StoredKey and Serv- erKey together with the parameters used for its derivation, as follows.

* 1. The client computes SaltedPassword by applying the password hashing function PBKDF2 (by default it is PBKDF2, but nowadays bcrypt is recommended) with Itera- tionCount many times on the input. The input is given by the password and the salt, i.e., SaltedPassword := PBKDF2(password, salt, IterationCount)
  2. The client computes ClientKey respectively ServerKey by applying the HMAC function on SaltedPassword with the public constant strings “Client Key” respectively “Server Key”, i.e., ServerKey := HMAC(SaltedPassword, "Server Key") and ClientKey := HMAC(SaltedPassword, "Client Key")
  3. The client computes StoredKey by hashing ClientKey, i.e., StoredKey := H(ClientKey) and sends ServerKey and StoredKey to the server (but not ClientKey).

The server stores StoredKey, ServerKey, salt, and IterationCount to later check proofs from clients and issue proofs to clients. ClientKey is used ﬁrst to authenticate the client against the server, and ServerKey is used later by the server to authenticate against the client.

Authentication



The server only stores the public part of the root (salt and IterationCount) and the leaves (StoredKey and ServerKey) of this tree., i.e., the password is never sent to the server. Only the following are sent: the salt, iteration count, ServerKey, and StoredKey, i.e., HMAC(SaltedPassword, "Server Key") and H(HMAC(SaltedPassword, "Client Key")).

Thus, after a database breach, i.e., after an attacker has stolen a ServerKey, a client’s password does not need to be replaced, only the salt and iteration count need to be changed and ClientKey and ServerKey replaced.

Client authentication to the server

For the server to authenticate, the client sends to the server an authenticator (contain- ing their client-name and a client-nonce). The server sends to the client a salt (salt), iteration count (ic) and a server nonce (server-nonce). Therefore, both, the client and server know AuthMessage := client-name, client-nonce, salt, ic, server-nonce. The client creates proof of their knowledge of StoredKey by computing

1. ClientSignature := HMAC(StoredKey, AuthMessage) ClientProof := ClientKey ⊕ ClientSignature

The server recovers the ClientKey by computing the ClientSignature (by knowing Store- dKey from storage and AuthMessage from this exchange), and “deciphering” the Client- Key' from the one-time pad ClientProof by computing ClientKey' = ClientProof ⊕ Client- Signature.

It computes StoredKey' by H(ClientKey), and checks whether the computed StoredKey' coincides with the stored StoredKey. If so, the client is successfully authenticated.

If only the ClientSignature were sent, then an attacker who knows the StoredKey could impersonate the client. Instead, ClientProof additionally requires the client to know the ClientKey. Therefore, the value of the ClientKey' that is calculated on the server should be immediately and irreversibly deleted after veriﬁcation.

Caveat

If an attacker knows the StoredKey from the server, and the AuthMessage and Client- Proof from an authentication exchange, then they can calculate the ClientSignature, thus the ClientKey, and can impersonate the client to the server.

###### Zero-Knowledge Proofs

A zero-knowledge protocol shows how a claimant can prove knowledge of a secret to a veriﬁer such that no other information is disclosed other than the validity of the claim (to anyone, including the veriﬁer).

Proof is meant probabilistically, i.e., the probability of the claim being true is beyond any reasonable doubt. Because the proofs are independent of each other, the probabil- ity can be increased as much as desired by increasing their number. The impossibility of gathering information about the secret from that exchange is linked to the computa- tional difﬁculty of solving a mathematical problem.

This characteristic has two beneﬁts. First, the veriﬁer cannot obtain any information even if they do not adhere to the protocol. Every proof is independent of each other. Second, the veriﬁer cannot impersonate the claimant to a third party. A recording of the proof does not help in convincing a third party because the sequence could have been mutually ﬁxed in advance.

Comparison to classic protocols

While claiming to know the secret alone is unconvincing, the leakage of information during classical protocols in which a claimant (C) proves knowledge of a secret to veri- ﬁer (V) still compromises the secret. If C transmits their password to V, then V and everyone who eavesdropped on this transmission obtains all data to impersonate C from this moment. In a challenge-response protocol, a new challenge is used in every occurrence of the protocol. At every occurrence, V and an eavesdropper can accumulate new information on the secret, eventually breaking it. For example, if the challenge is the encryption of a plaintext, and the attacker can choose this plaintext, then this is a chosen-plaintext attack.

Public key

To weigh the advantages and inconveniences between a zero-knowledge proof and a digital signature with a private key (veriﬁed by its corresponding public key) like any challenge-response protocol, every signature, i.e., encryption of a document by the user’s private key, leaks information. In the extreme case that the attacker can choose this plaintext, this is a chosen-plaintext attack. However, zero-knowledge protocols

Authentication

require less computation than public-key protocols. Since digital signatures are practi- cally secure and many devices (e.g., smart cards) have little computing power, digital signatures are more common.

The security of both, most zero-knowledge protocols and public-key protocols, depends on the unproved assumption that cryptanalysis is computationally as difﬁcult as a mathematical problem (such as the computation of quadratic residuosity, the decom- position of an integer into its prime factors and discrete logarithm).

Ali Baba’s cave

Ali Baba’s cave illustrates the principles behind a zero-knowledge proof. In a circular cave, there is a door invisible from the entrance that bars any passage if not opened by a password (such as “Sesame”). For C to prove to V that they know the password with- out disclosing it, C enters the cave unobserved from V until standing in front of the door’s left or right. V demands C to return to the entrance coming from the left or right. Because the probability that C entered the cave on the same side as V asked for to leave on is one half and all proofs are independent, for example, after 10 proofs the chance C does not know the password is 2–10, less than a thousandth. While V is con- vinced that C knows the password, he cannot convince anybody else. Even if V recorded the sequence, it could have been mutually ﬁxed in advance.

Schnorr’s sigma protocol

Schnorr presented a zero-knowledge protocol simple enough to run on smart cards (Schnorr, 1991). Knowledge of discrete logarithms is proved, i.e., the prover P knows an integer x, and the veriﬁer V knows gx mod p, where p is a prime number. For P to prove knowledge of *x* without revealing it.

1. P chooses some integer a and sends ga to V.
2. V tosses a coin and sends the result c in {0, 1} to P.
3. P sends to V either a, whenever c = 0, or a + x, whenever c = 1.

This is a zero-knowledge protocol because

* if c = 0, then nothing about x is revealed (but only a),
* if c = 1, then the veriﬁer learns a + x mod p, but again, as long as nothing about a is revealed (where we count on the difﬁculty of computing log mod p), neither any- thing about x is revealed.

If c = 0, then no knowledge of x is needed. If c = 1 however, then V can verify whether P knows a + x by ga+x = ga gx where both values on the right-hand side are known. Since the probability that c = 1 is 1/2 and all proofs are independent, after, say, 10 proofs the chance V does not know x is 2–10 < 1/1000.

### Biometric Authentication

Biometric authentication identiﬁes the user of a computer by either physical or behav- ioral human characteristics. Physical characteristics include ﬁngerprints (which have been around for a long time), facial characteristics, such as the relative positions of the eyes, nose, lips, jaw, and eye characteristics, such as the pattern of blood vessels in the iris. Behavioral characteristics include areas such as typing manner, e.g., the speed of keystrokes or the occurrence of typos (particularly useful to supplement a log-in dia- logue); handwriting style, either static where an image is used or dynamic where the traces on a tablet are evaluated by the functions (of time) like *x* and *y*-coordinates, pressure, and inclination; and voice properties (speaker recognition is particularly use- ful to verify the identity of telephone customers). Each spoken word is decomposed into its formants, the dominant frequencies, and then physiological and behavioral characteristics identify the physiological characteristics (describing the shape of the person’s vocal tract, i.e., of their nose, mouth, jaw, tongue) or the behavioral character- istics (describing the movement of the nose, mouth, jaw, tongue changing accent, tone, pitch, or pace).

Authentication by what one is (e.g., a ﬁngerprint) and what one knows (e.g., a password) has the advantages that even when known requires some effort to replicate and cannot be forgotten. Comparing authentication by what one has (e.g., a token) has the advan- tages that nothing has to be carried around nor can anything be lost or stolen.

This type of authentication requires sophisticated hardware that is exposed and thus imitable. For example, the German Chaos Computer Club (CCC) and its collaborators have demonstrated repeatedly the ease of forging ﬁngerprints, for example, using wood glue and sprayable graphene. The many successful forgery attacks on biometric identi- ﬁcation, such as ﬁngerprints and photo ID, leads to the conclusion to treat them as a complement instead of a replacement for a password, a smart card, a second factor, or merely as a replacement for a user.

### Authentication in a Distributed System

In a distributed system, if the identity of the system against which authenticating is guaranteed, and nobody is possibly eavesdropping, then the secret itself for authenti- cation could be securely sent. Otherwise, to ensure that none other than the intended recipient sees the secret for authentication, the secret itself must never be sent. Instead, both the user and the system convince each other they know the shared secret (usually a password). Thus, the identiﬁcation data itself is never sent, but only proof that the user has access to it. In practice, for systems that are password-based, the most popular approach is the challenge-response system.

As best practice, even though the secret itself is never sent, it is still advisable to encrypt all communication to establish authentication, e.g., by public-key encryption. Other approaches include revealing a partial secret, e.g., in a single-use system the

Authentication

identiﬁcation data is only used once (TANs in banking). However, if the identiﬁcation data can be eavesdropped and its use for authentication, and thus invalidation, be pre- vented (e.g., by logging into a forged copy of the bank’s website), then it can be used later. Another approach is to send additional secret information through a second channel, e.g., the sending of an SMS in the mobile TAN (mTAN) system.

###### FIDO 2

The FIDO (Fast IDentity Online) Alliance was ofﬁcially founded in February 2013 to develop open and license-free industry standards for authentication on the internet in arrangement with many companies such as Google or Microsoft. On December 9, 2014, the ﬁrst FIDO standard was released that speciﬁes the standard U2F (Universal Second Factor) for hardware and software for two-factor authentication and the standard UAF (Universal Authentication Framework) for the associated network protocol for pass- word-less authentication. These standards aim to facilitate authentication on the inter- net by accepting a user’s belongings (what they have), such as security tokens, or prop- erties (what they are), such as a ﬁngerprint, instead of knowledge (what they know), such as passwords or personal identiﬁcation numbers. Thus, a user no longer needs to memorize numerous secure passwords, at the cost of some drawbacks. In comparison with previous methods of two-factor authentication, such as SMS veriﬁcation codes, FIDO2 requires the key, such as the smartphone, to be physically near the computer.

FIDO2 (2020) consists of the

1. W3C Web Authentication Standard (WebAuthn) which allows accesing the internet with biometric information, mobile devices, or FIDO security key. WebAuthn is sup- ported by various operating systems, such as Windows 10 and Android, but also browsers, such as Google Chrome, Mozilla Firefox, Microsoft Edge, or Apple Safari.
2. Client to Authenticator Protocol (CTAP) of the FIDO Alliance that is based on U2F. The release of FIDO2, U2F was renamed CTAP1. CTAP is, among other use cases, for authentication in desktop applications and web services.

Personal data and private keys are always and exclusively in the hands of the user and are not stored on public servers. To register an account via FIDO2, the servers sends a request, and the FIDO2 key generates a public and a secret key from a secret initial key and the server address. This public key is transmitted to the server, which stores it and can uniquely identify the FIDO2 key in the future.

This way, the FIDO2key can identify itself with an individual key at each server without the server obtaining information on the key pairs for the same FIDO2 key on other serv- ers. The FIDO2 key generates a separate key pair for each service. Based on the domain of the other party, e.g., Ebay and Google, they cannot determine which of their users have the same FIDO2 key. In practice, this is an advantage (on the server side!) of authentication by a password where a user often uses similar passwords on different servers.

Key

A FIDO2 key (or authenticator or token) is the device by which to authenticate oneself to a service. It can be either an external device to connect to the user’s PC or smart- phone via USB, NFC, or Bluetooth, such as a security token to be inserted into a USB port, a smart card to be inserted into a card reader, an NFC token, smartphones with Bluetooth-Interface IEEE 802.15.1, and Bluetooth Tokens (Bluetooth V4.0 Low Energy 2,45 GHz). But it can also be an internal authenticator, i.e., software that uses the crypto chip of the PC, smartphone, or tablet for FIDO2, supported by Windows 10, Android 7, and above. To protect against misuse of the FIDO2 key, it can be additionally secured bio- metrically or with a password/PIN. If the stick is lost, then either a registered backup key is available or one has to identify oneself again by the mobile phone number in combination with an email address or alike.

Adaption

A FIDO2 key can either be used instead of a password or in addition to it, as a second factor. Depending on how a service has implemented FIDO2, the key sufﬁces for logging in (one-factor authentication) or entering an additional password (two-factor authenti- cation) is necessary. FIDO2 one-factor authentication is already available for Micro- soft.com, Outlook.com, Ofﬁce 365, and OneDrive in the Edge browser. FIDO2 two-factor authentication works, for example, with Google, GitHub, Dropbox, and Twitter.

###### Kerberos

Kerberos is named after Cerberus, the three-headed beast who guards the Adamantine entrance Gate to Hades. Cerberus authenticates dead souls, and Kerberos mutually authenticates users over a network (Neuman & Ts’o, 1994). There is one solution to the problem of key distribution, i.e., of secretly passing the same key to all correspondents, and the many keys needed for a group of correspondents to communicate securely to each other. All correspondents must trust a central authority unconditionally. Moreover, each user must be able to securely obtain a session key for each correspondence so that each correspondent has only to protect one key, while the responsibility to protect all the keys among the correspondents is shifted to the central authority.

Authentication and ticket-emission components

Kerberos is a network protocol (as speciﬁed in RFC 1510—The Kerberos Network Authen- tication Service V5) which permits users to securely authenticate each other over an insecure network by a trusted third party, the Key Distribution Center (KDC). Once a user is authenticated (Kerberos), they are authorized by access protocols such as LDAP (Lightweight Directory Access Protocol) (Neuman & Ts’o, 1994).

The KDC stores the symmetrical keys of all registered users (client or server) to authen- ticate them as an intermediary third-party. Due to its critical role, it is good practice to have a secondary KDC as a fallback.

Kerberos groups users into clients and (service) servers (SSs) that host services for the clients. The protocol authenticates a client only once so that they are trusted by all SSs for the rest of his session. This is achieved by two factors:

Authentication

1. a ticket, a one-use credential emitted by the KDC to authenticate a client to a server from which they are requesting a service, encrypted using the server’s key. It con- tains the server’s and the client’s ID, the client’s network address, a timestamp, a lifetime, and a session key encrypted using the client’s key.
2. an authenticator, a credential, encrypted using the session key shared between the client and the server, that accompanies the ticket to authenticate the client. It con- tains the client’s ID, the client’s network address, and a timestamp.

The KDC is split up into an Authentication Server (AS) and a Ticket-Granting Server (TGS). The AS authenticates each user in the network, the AS stores a symmetric key for each user, be it client or SS, and which is only known to itself, the AS, and the user. After the AS has authenticated a client with the client’s key, the AS sends him a Ticket- Granting Ticket (TGT) and a session key encrypted using the client’s key. The TGS gener- ates a session key as part of a ticket between two users of the network after authenti- cation. After a client has sent their TGT, been authenticated by the TGS, and requests a service of an SS, the TGS emits a ticket and two copies of a session key. One is encryp- ted using the user’s TGT session key; the other is encrypted using the SS key, to authen- ticate the client to the SS and secure their communication.

Authentication protocol

To allow a user (or client) to securely communicate with another user (Service Server or Application Server (SS or AP)), via the KDC, the Kerberos protocol deﬁnes ten messages, among them (Neuman & Ts’o, 1994)

|  |  |
| --- | --- |
| Code | Meaning |
| KRB\_AS\_REQ | Kerberos Authentication Service Request |
| KRB\_AS\_REP | Kerberos Authentication Service Reply |
| KRB\_TGS\_REQ | Kerberos Ticket-Granting Service Request |
| KRB\_TGS\_REP | Kerberos Ticket-Granting Service Reply |
| KRB\_AP\_REQ | Kerberos Application Request |
| KRB\_AP\_REP | Kerberos Application Reply |

The authentication and subsequent ticket granting between a user and a server is mediated by a Kerberos server (Pixis, 2019):

1. The client, to authenticate the AS,
   1. authenticates the authentication server (AS) (KRB\_AS\_REQ) using a long-term shared secret (client’s key), and
   2. receives a short-term shared secret (session key) and a (ticket-granting) ticket from the authentication server (KRB\_AS\_REP).
2. The client, to authenticate the SS via the AS,
   1. authenticates the AS using their TGT, and
   2. requests (KRB\_TGS\_REQ) a ticket from the TGS (KRB\_TGS\_REP) that contains a session key between the client and the SS.
3. The TGS, to create the ticket,
   1. generates a client-to-server session key;
   2. encrypts, using the client key, the session key;
   3. encrypts, using the SS key, the client-to-server ticket that contains the client’s ID, the client’s network address, a timestamp, a lifetime, and the session key; and
   4. sends the results of both encryptions to the client.
4. The client, to authenticate directly to the SS,
   1. decrypts the client-to-server session key using their own key, and
   2. sends to the SS (KRB\_AP\_REQ and KRB\_AP\_REP)
      * the client-to-server ticket, encrypted using the SS’s key, and
      * an authenticator that contains the client’s ID and a timestamp *N*, encrypted using the client-to-server session key.
5. The SS
   1. retrieves the client-to-server session key by decrypting the client-to-server ticket using their own key,
   2. decrypts, using the session key, the authenticator and checks it. If the check suc- ceeds, then the server can trust the client.
   3. sends the timestamp incremented by 1 encrypted via the session key to the cli- ent.

With the session key, the client can decrypt the timestamp to check it. If succeeding trust is established and service requests to the server can begin. The server of course provides the requested services to the client (Pixis, 2019).

### Smart Cards

A smart card is shaped like a credit card but contains a microprocessor to securely store and process information. In contrast, a magnetic-stripe card only stores little information (around <100 bytes) and cannot process it. Information processing on a smart card is secured by security algorithms; a successful authorization is mandatory to access the data (Borst et al., 2001).

###### Components

Most of the processing in smart cards is dedicated to cryptographic operations, in par- ticular, encryption between on-chip components. For use, a smart card needs external power and avvclock signal provided through contact with a smart card reader (that

Authentication

usually can also write). The operating system on most smart cards implements a stand- ard set of control commands such as those standardized in ISO 7816 or CEN 726. The components of a smart card are

1. Random-Access Memory (RAM) reads and writes data, but only stores information temporarily as long as there is electricity. Typically, a smart card has 1 kByte of RAM.
2. Read-Only Memory (ROM) stores information permanently. The operating system and encryption algorithms are stored. Typically, a smart card has 32 kByte of ROM. For improved security, the ROM is buried in lower layers of silicon.
3. Electrically Erasable Programmable Read Only Memory (EEPROM) stores information permanently but is slow and one can only read/write to it a limited number of times (around 100 000 times). Typically, a smart card has 8K—128 kByte of EEPROM. For improved security, the EEPROM is shielded in a metal coating.
4. The processor used to be an 8-bit microcontroller, but increasingly more powerful 16 and 32-bit chips are being used. A coprocessor is often included to improve the speed of encryption computations.

There is a single input/output port controlled by small data packets called APDUs (Application Protocol Data Units). Because data ﬂows only at around 9600 bits per sec- ond and half-duplex, i.e., the data can either ﬂow from the reader to the card or from the card to the reader, but not bidirectional. A smart card can be read out slowly, while both reader and smart card authenticate each other by a challenge-response protocol using a symmetric key encryption. For encryption purposes, the card generates a ran- dom number, shares it with the reader, which in return encrypts the number with a mutual key and returns this to the smart card. Finally, the card compares the encryp- tion result with its own encryption. Once mutually authenticated, each exchanged mes- sage is veriﬁed by a message authentication code (HMAC) which is calculated using as input the message, encryption key, and a random number (Borst et al., 2001).

###### SIM Cards

The Universal Integrated Circuit Card (UICC) or Universal Subscriber-Identity Module (USIM) is quite popular for mobile devices in GSM and UMTS networks. It’s a smart card of a few hundred kB and ensures security and integrity of personal data. UICCs Sub- scriber Identiﬁcation Module (SIM) is an application which stores the authentication information. UICC store not only the mobile phone service subscriber’s identifying key and the subscription information, but also their preferences, text messages, contacts, and last dialed numbers. Compared to credit cards, a larger EEPROM is available.

###### Advantages

Cryptographic keys do not need to be remembered and can be arbitrarily complex, but cryptographic keys cannot be acquired over distance. The procedure is less sophistica- ted, uses less expensive hardware, and cryptographic keys are stored securely on a device. Thus forgeries with ﬁngerprints or vein scanners are avoided. Keys stored on a

smart card are much harder to read and leave fewer traces. For example, they never reveal the key, but provide only the information that was asked for and are immune to keyloggers that record keystrokes.

### Identity and Anonymity

Anonymity comes from the Greek word for “name-less” (OPTED, n.d.). Colloquially, it means that a person’s identity is unknown. Here, that an element (e.g., a person or a computer) within a set (e.g., a group or network) is unidentiﬁable (within this set). Pro- tecting one’s identity from being disclosed is not only necessary for someone who breaks the law, e.g., when attempting to exploit a computer in a network, but also as a precaution to a possible abuse. Similar countermeasures are available:

* + a proxy server between the user and the internet that, among other tasks such as caching frequently used data and restricting access to some users, can hide IP addresses (which identify a computer in a network),
  + the Tor project which encrypts all the user’s trafﬁc and routes it through many relays, which do not know of each other, and
  + Virtual Private Networks, such as OpenVPN or IPsec, which encrypt all the user’s traf- ﬁc and routes it towards a central server.

However, these countermeasures have inconveniences, such as involved setup and slower data transfer. A less compromising practical measure is to adapt best practices for protecting one’s privacy on the World Wide Web. These could include using browser add-ons that ﬁlter out trackers, e.g., from cookies, referrers (the URL of the previous webpage from which a link was followed), and requests to centralized content delivery networks (CDNs), such as uBlock Origin, Privacy Badger, Don't track me Google, and Decentraleyes.

###### Identity Theft

Identity, from the Latin *identitas*, (*idem*, same and *entitas*, entity), are the characteris- tics by which someone or something is uniquely recognizable, i.e., held by no other person or thing (OPTED, n. d.). Identity theft is the assumption of another person’s identity, usually another person’s name or other personal information such as PINs, social security numbers, driver’s license, or banking details, to commit fraud, such as to open a fraudulent bank or credit card account. With the advent of digitalized records and anonymity on the internet, identity theft has become increasingly common.

For example, in SIM-card swapping, the attacker obtains a victim’s mobile phone num- ber to assume, temporarily, their online identity. The attacker initially obtains personal data about the victim, usually their name, mobile phone number, and mailing address. They then exploit the fact that mobile operators usually offer their customers a new SIM card, for example, after the phone is lost, and transfer the previous phone number.

Authentication

The attacker now pretends to be the real customer on the phone with the customer service center. Indeed, it sufﬁces to know the mobile phone number to reset the pass- word of an Instagram account.

Summary

Identiﬁcation is telling a computer who the user is, usually by their user (or account) name. This is followed by authentication, the veriﬁcation of the user’s identity, i.e., convincing a computer that a person is who they claim to be. To authenticate, the user proves their identity by information that only they know (such as a password), only they have (hardware such as a smart card), and only describes them (such as biometric identiﬁers).

Each method has its proper advantages and disadvantages. In particular, pass- words, the most common method, have to be easy to remember, which in practice weakens them. The FIDO2 standard aims at replacing (or at least complementing) them by hardware and biometric authentication.

Instead of revealing the secret itself when authenticating, it is safer to prove only its knowledge. In a challenge-response authentication protocol, the successful response to the challenge requires its knowledge, such as encryption and decryp- tion of random data with the secret key. In a zero-knowledge protocol, in contrast to a challenge-response protocol, no information whatsoever can be won on the secret key provided the computation of a mathematical function is assumed infea- sible.

The Kerberos protocol mediates between users and servers through a central server that stores symmetric keys of all parties. Instead of the parties mutually proving the knowledge of their symmetric keys, it creates a session key for each correspond- ence of limited validity.



# Unit 5

## Cryptanalysis — How to Break Encryption

#### STUDY GOALS

On completion of this unit, you will have learned …

… which scenarios we need to protect against with cryptanalysis.

… how to estimate the duration of a brute-force attack and accelerate it.

… the role of differential cryptanalysis and frequency analysis in breaking ciphers.

DL-E-DLMCSEAITSC02-U05

1. Cryptanalysis — How to Break Encryption

### Introduction

Cryptanalysis Recovering or forg- ing enciphered infor- mation without knowledge of the key is known as crypt-

analysis.

Cryptanalysis is the art of untying the hidden writing, i.e., of breaking ciphers. The pre- ﬁx “crypto” comes from the Greek for “hidden” and “analysis” from the Greek “to untie” (OPTED, n. d.). History delivers plenty of cryptanalytic success stories, among many oth- ers, the decryption of the German Enigma rotor machine by Polish and British forces in World War II. We will present some established principles of cryptanalysis. In practice, the cryptanalyst’s intuition and ability to recognize subtle patterns in the ciphertext were paramount, but difﬁcult to convey. Today, however, cryptanalysis is based on mathematics and applied through the efﬁcient use of extensive computing power (Dooley, 2013, Chapter 5).

### Frequency Analysis

Cryptanalysis of single-key cryptosystems relies on patterns in the plaintext carrying over to the ciphertext. For example, in a monoalphabetic substitution cipher the fre- quencies of the occurrences of the letters in the plaintext alphabet are the same as those in the ciphertext alphabet. This can be put to good cryptanalytic use by recogniz- ing that the cipher is a monoalphabetic substitution cipher and giving the likeliest can- didates of plaintext letters.

###### Latin Alphabet

A substitution by any permutation of the letters of the alphabet, such as,

A B … Y Z

…

E Z … G A

has

26 · 25!1 = 26! > 1026

keys, so a brute-force attack is computationally infeasible. However, it violates the goals of diffusion and confusion. If the key (i.e., the given permutation of the alphabet) exchanges the letter α for the letter β, then there’s bad confusion. This is because the substitution of β in the key implies only the substitution of each letter β in the cipher- text. Similarly, the substitution of a letter α in the plaintext implies only the substitu- tion of the corresponding letter β in the ciphertext. In fact, the algorithm allows statisti- cal attacks on the frequency of letters, bigrams (pairs of letters) and trigrams (triples of letters). In English, for example, the most frequent letter is “e,” the most frequent bigram is “th,” and the most frequent trigram is “the.”

Cryptanalysis — How to Break Encryption

Thus, substituting

* the most frequent letter from ciphertext to the most frequent letter in English (e),
* the most frequent bigram from coded text to the most frequent bigram in English (th), and
* the most frequent trigram of the ciphertext to the most frequent trigram in English (the)

is a good starting point to decipher the text. The more ciphertext, the more likely that this substitution coincides with what the text was enciphered.

Using these frequencies, for example, on the ciphertext *ACB ACBGA ACSBDQT* gives *THE THE\*\* TH\*E\*\*\** for yet to be deciphered letters marked by \*. By the restrictions of Eng- lish vocabulary and sentence structure, yielding *THE THEFT THREAPS.*

###### Homophones

To hide the frequencies of the alphabetic letters, one approach is to use multiple homophones relative to the frequency of the letter in the plaintext, e.g., twice as many symbols for E as for S, so that each cipher symbol occurs on average equally often in the ciphertext. The German mathematician and astronomer Carl Friedrich Gauss assumed that his introduction of homophones rendered such a cipher unbreakable. However, other frequencies in the plaintext (partially) withstand encryption, such as digraphs. “TH” occurs most often, about 20 times as frequently as “HT,” and so on, still considerably facilitating cryptanalysis given sufﬁcient ciphertext (Jagetiya & Krishna, 2020).

### Brute-Force Attacks

In practice, the security of a cryptosystem relies foremost on its resistance to the most efﬁcient (known) methods of cryptanalysis and the computational effort needed to check all keys, a brute-force attack. Given the time and computational resources, the right key will eventually be found. If only the ciphertext is available then a brute-force attack would decrypt the ciphertext block-by-block and key-by-key in order to get some meaningful text. Compared to single-key cryptography, by relying on computationally difﬁcult mathematical problems (i.e., whose runtime grows exponentially in the bit- length of the input), the cryptanalysis of two-key cryptography is that of computational mathematics. The task is ﬁnding an algorithm to shorten computation time, ideally, one with polynomial runtime in the number of input bits.

However, in practice, for the prime factor decomposition used in RSA or the discrete logarithm in the Difﬁe-Hellman key exchange, we can ﬁnd algorithms whose runtime grows slower than exponentially in the number of input bits (sub-exponential), but none with polynomial run-time. Moore’s Law originally stipulated that computing power approximately doubles every year. Since its introduction the speedup in prime

Homophones Multiple cipher sym- bols for the same plaintext letter are known as homo- phones.

factor decomposition has been around six additional binary digits each year. Thus, as the computational effort doubles for each three additional digits, the speedup quadru- pled each year. To be conﬁdent of a key’s security beyond 2020, currently at least 2048 binary digits are called for.

###### Comparison of Key Sizes

This table compares the key sizes in bits to a security level comparable between a sym- metrical algorithm such as the AES, an asymmetric algorithm by elliptic curves, and an asymmetric algorithm such as Difﬁe-Hellman or RSA (Lenstra & Verheul, 2001).

|  |  |  |
| --- | --- | --- |
| Symmetric Key | Asymmetric Elliptic Key | Classic Asymmetric Key |
| 80 | 160 | 1024 |
| 112 | 224 | 2048 |
| 128 | 256 | 3072 |
| 192 | 384 | 7680 |
| 256 | 512 | 15360 |

The numbers in the table are estimated by the fastest known algorithm to solve the cryptographic problem. Given an input key with n bits,

* For a symmetric algorithm like AES, the fastest algorithm is to try out all possible keys, whose complexity (= the number of operations) is O(2n).
* For the logarithm over a ﬁnite elliptic curve, the fastest algorithm today is the generic baby step, giant step algorithm (or, slightly faster, Pollard’s ρ-algorithm) whose complexity is roughly O(2√n).
* For classic asymmetric algorithms, either RSA or Difﬁe-Hellman, on a ﬁnite ﬁeld, the fastest algorithm is the “general number ﬁeld sieve” whose complexity, roughly, for a large n is O(2√3n).

In practice, the smaller ECC keys speed up cryptographic operations by a factor of > 5 compared to RSA and Difﬁe-Hellman (in addition to facilitating their exchange among people and saving bandwidth). However, there are also disadvantages to ECC compared to RSA, e.g., its signature algorithm depends on a pseudo Random Number Generator that, when badly programmed, reveals the private key.

Cryptanalysis — How to Break Encryption

### Rainbow Tables

Rainbow tables are a method of password cracking that compares a password hash to precomputed hashes of the most likely passwords, i.e., a time-memory trade-off of more memory for less computation.

A rainbow table is a table of cryptographic hashes of the most common passwords. Thus, more likely passwords are revealed sooner. The generation of the table depends on the cryptographic hash function and the character set used, the password length, the number of table entries, and so forth. Common cryptographic hash algorithms such as MD4/5 and SHA are fast, thus they are unsuitable for password creation because they are vulnerable to brute-force attacks. For example, MD5 as a cryptographic hash function is designed to be fast and thus lends itself towards a rainbow table attack, while hash functions such as PBKDF1, PBKDF2, bcrypt, scrypt, and the recent Argon2 were designed to prevent this kind of attack by being deliberately slow (bcrypt) and/or deliberately memory hungry (scrypt). A rainbow attack can however most effectively be prevented by making the used hash function unique for each password. This is ach- ieved by adding a salt.

###### Lookup Table

A lookup table (LUT) is a data structure used to replace runtime computations with a simple array of indexing operations. In the LUT, all entries are precomputed and then only looked up to save runtime because array-indexing operations are in general faster than computations (Kumar et al., 2013).

###### Meet-In-The-Middle Attack on DES

The key size of the symmetric industry standard cryptoalgorithm DES was merely 56 bits, so little that it ceded to brute-force attacks shortly after its vetting. Therefore, a twofold encryption for two different keys was thought to effectively double the key size to 112 bits. However, the meet-in-the-middle attack by Difﬁe and Hellman trades off memory for time to ﬁnd the key in only 2n+1 encryptions (using around 2n stored keys) instead of the expected 22n encryptions (van Oorschot & Wiener, 1996).

Assume the attacker knows a plaintext P and its ciphertext C, i.e.

C = EK: EK′ P ,

where E denotes the encryption using K’ respectively K’’. The attacker (1) computes EK(P) for all possible keys K and stores the results in memory, (2) decrypts the cipher- text by computing DK(C) for every K, and (3) looks for matches between these two sets, whose keys likely match those used to encrypt P to C.

Therefore, triple encryption, 3DES was necessary to effectively double the key size and harden decryption for the computing power to come.

### Known/Chosen Plain/Ciphertexts

Asymmetric cryptography uses mathematical methods, more exactly modular arith- metic, to encipher. The security, the difﬁculty of deciphering, of asymmetric cryptogra- phy is based on computational mathematical problems that have been recognized as difﬁcult for centuries. Symmetric cryptography (e.g., hash functions) uses more artisa- nal methods of ciphering, which aim to maximize diffusion and confusion, mainly by iterated substitution and permutation. The security of symmetric cryptography is sim- ply based on its resistance to years of ongoing attacks, i.e., it is satisfactory from a prac- tical standpoint, but less so regarding the struggle for eternal truths.

###### Perfect Security

Perfect security This is the probabil- ity that a plaintext and a key resulting in a given ciphertext is the same for all plaintexts and all

keys.

Plaintexts generally do not occur with the same probability. It depends, for example, on the language, jargon, or protocol used. A cipher is perfectly secure if none of its cipher- text reveals anything about the corresponding plaintext. Moreover, for every plaintext, its probability is (stochastically) independent of any ciphertext.

Let p denote a plaintext and by P(p) its probability. In formulas, for every plaintext p and every ciphertext c, we have P(p|c) = P(p). In practice, this means that if an attacker intercepts a ciphertext c, then they have no advantage, i.e., their probability of knowing the plaintext is the same as if they do not know c.

Shannon (1949) proved the following theorem on the conditions for a cipher to be per- fectly secure: Given a ﬁnite number of keys and plaintexts with positive probabilities, i.e., P(p) > 0 for every plaintext p, the cipher is perfectly secure, if

* the probability distribution is uniform, i.e., all probabilities are equal, and
* for each plaintext p and every ciphertext c, there is a unique key k to get c from p.

Statistical deviations tend to weaken the cryptosystem. In particular, it is important to use a completely random number generator for the keys.

###### One-Time Pad

The only perfectly secure cryptosystem is the one-time pad. Such a perfectly secure cryptosystem is, however, impractical. For real-time applications, such as on the inter- net, it is little used. The one-time pad adds (by the XOR operation) each bit of the plaintext t with the (positionally) corresponding bit of a key c that is of the same length, and is discarded after use, i.e., it will not be used to encrypt other plaintexts.

Cryptanalysis — How to Break Encryption

Thus the ciphertext T = (T1, T2, …) is

T = t ⊕ c = t1 ⊕ c1, t2 ⊕ c2, . . . .

This cipher is as safe as theoretically possible!

If the plaintext has a single block t, then this simple (XOR) addition of a key, the one- time pad, is a secure algorithm. However, it is often inconvenient or even close to impossible to have a key as large as the plaintext, e.g., to encrypt communication over a network. How much data will be transferred must be known beforehand.

It is a bad idea (though natural) to use the same key for two different blocks. If, for example, the plaintext has two blocks b’ and b’’, then, with this algorithm, the sum (XOR)

One-time pad

In a one-time pad, the key is as long as the plaintext; the keys are added letter by letter (or bit by bit) to obtain the ciphertext.

b′ ⊕ c ⊕ b: ⊕ c = b′ ⊕ b:

of the two cipher blocks b’ ⊕ c and b’’ ⊕ c equals the sum b’ ⊕ b’’ of the two clear blocks. This is because the addition XOR is by deﬁnition auto-inverse, i.e., x ⊕ x = 0 regardless of whether the binary digit is x = 0 or x = 1!

It can happen that the ciphering of the ﬁrst block by one-time pad is the second block. Unfortunately, the second block is not a good key because far from being random, usu- ally its content is similar to that of the ﬁrst block, i.e., the key is predictable.

###### Proven Security

As perfect security is unfeasible, security is just demonstrated by resistance against known attacks or by reducing the computational difﬁculty to that of a (computational mathematical) problem recognized as difﬁcult: proven security. Although there are pro- ven secure symmetric cryptosystems, the most efﬁcient and widely used algorithms, such as AES, prove their resistance only against known attacks, such as those of differ- ential or linear cryptanalysis.

Formally proven security

The mathematical problems on which the difﬁculty of the decryption in asymmetric cryptographic algorithms are based, are all NP-complete, i.e., their solutions are veriﬁa- ble in polynomial runtime (in the bit-length of the input) and all other such problems can be reduced to it. Thus, every cryptographic algorithm (enciphers or) deciphers a message with the key in polynomial runtime in the bit-length of the key.

In contrast, all known algorithms for deciphering without the key take exponential time in the bit-length of the key. By the P-versus-NP conjecture, there is no algorithm that takes polynomial runtime (in the bit-length of the key). For now, the conjecture being unresolved, there may theoretically exist polynomial algorithms for deciphering without the key in polynomial runtime. However, after decades of continuous efforts by the community of cryptanalysts, it is assumed unlikely.

The initial example of such a proven secure cryptosystem was Goldwasser et al.’s (1982) semantic security, which reduces the difﬁculty of decipherment to that of the computa- tion of the quadratic residue. Given x and N a product of two primes, it is difﬁcult to determine whether x is quadratic modulo N (i.e., whether there is y such that x = y2 mod N or not) if, and only if, the Jacobi symbol for x is +1 and the prime factors of N are unknown.

The Goldwasser-Micali cryptosystem consists of:

1. Key generation algorithm that produces the private key as two primes p and q, and the public key N = p q and a number x which is quadratic neither modulo p nor modulo q (such that Jacobi’s symbol of x for N is +1, and for both their factors p and q is –1). For example, if p, q ≡3 mod 4, then x = N–1 will do.
2. A probabilistic enciphering algorithm. If m = (m1, m2, …) are the bits of the plain-

text, then numbers y1, y2, … that are indivisible by p and q are generated, and the enciphered message is M = (M1, M2, …) with M1 = y 2 xm1, M1 = y 2 xm1, …

1

1

1. A deterministic decipherment algorithm. If M = (M1, M2, …) is the enciphered mes- sage, then m1 = 0 if, and only if, M1 is quadratic modulo N, … which is quickly deter- mined by knowledge of both factors p and q of N.

Paradox

Theoretical security remains an insufﬁcient idealization for reality. For example, Ajtai and Dwork (1997) presented and proved a theoretically secure cryptosystem that was broken a year later. “Proven” does not mean “true”; a proven secure system is not nec- essarily truly secure because the proof is made in a formal model which assumes cer- tain operational principles, attackers, a set security objective, and the difﬁculty of the problem to which the proof is reduced.

For example, the implemented cryptosystem differs from the formal cryptosystem. A partial objective is already sufﬁcient for the attacker. If, for example, the security objec- tive is that the attacker does not derive the entire plaintext from the ciphertext, then it may already be enough for them to derive a passage of the plaintext. Besides, the proof may be wrong! Despite this uncertainty, a proof of security is a useful criterion (though theoretically necessary, but practically insufﬁcient) for the security of a cryptosystem.

###### Attacking Scenarios

What does security mean? The criterion that the attacker cannot derive the plaintext from the ciphertext is insufﬁcient because they could acquire other useful (partial) information about the plaintext. But even the impossibility to derive useful information on the plaintext is insufﬁcient in some circumstances. If the public-key encryption is deterministic (i.e., if the same input always returns the same output, as is the case with RSA encryption as implemented in a textbook), and the attacker can limit the number of possible plaintexts (they know, for example, that the ciphertext is “yes” or “no”), then they can encrypt all these possible plaintexts with the public key and compare the

Cryptanalysis — How to Break Encryption

ciphertexts to the encrypted texts. For an asymmetric algorithm, we should assume that the attacker knows the public key. Thus, they can encrypt any plaintext of their choice and compare it to the ciphertext, i.e., they can mount a Chosen-Plaintext Attack (CPA).

Generally, the attacking scenarios are categorized by how much the cryptanalyst knows about the ciphertext. Do they know only the ciphertext or are there probable or known pairs of ciphertexts and plaintexts available? Is a chosen plaintext or ciphertext also available? For example, to break a monoalphabetic cipher, the ciphertext alone usually sufﬁces thanks to frequency analysis. Often, however, the cryptanalyst either will know or guess some of the plaintext, such as a preamble of a letter (such as a formal greet- ing) or a computer ﬁle format (such as an identiﬁer). Lastly, most opportunely, they can either ask the sender to encrypt a plaintext that they chose or the recipient to decrypt a ciphertext that they chose.

Ciphertext-only attack

The attacker has the ciphertexts of several messages that were encrypted by the same algorithm. Their task is to recover as much plaintext as possible or, better, to recover the used algorithms and keys.

Probable-plaintext attack

The attacker has the ciphertext and suspects that the plaintext contains certain words (a crib) or even whole sentences. Their task is to recover as much plaintexts as possible or, better, to recover the algorithms and keys that were used. For example, Enigma, the cryptographic electromechanical rotor-machine used by the Axis Powers in World War II, was broken by the repetitiveness of the messages it enciphered: the weather report was sent on a daily basis and announced as such at the beginning of every message.

Known-plaintext attack

The attacker has a ciphertext and the corresponding plaintext. Their task is to recover the (algorithm and) key that was used, e.g., linear cryptanalysis falls into this scenario. For example, an attack from 2006 on the Wired Equivalent Privacy (WEP) protocol for encrypting a wireless LAN exploits the predictability of parts of the encrypted mes- sages, namely the headers of the 802.11 protocol.

Chosen (or adaptive) plaintext attack (CPA)

The attacker has the ciphertexts of the plaintexts that they can freely choose. The attacker can freely adapt the plaintext depending on the text obtained after each deci- pherment and analyze the resulting changes in the ciphertext. Their task is to recover the algorithm and key that was used. Differential cryptanalysis falls into this scenario.

This is the minimal attacking scenario to be prepared against for asymmetric cryptogra- phy! Since the encryption key is public, the attacker can encrypt messages at will. Therefore, if the attacker can reduce the number of possible plaintexts, e.g., because they know that these are either “Yes” or “No,” then they can encrypt all possible plain- texts with the public key and compare them to the intercepted ciphertext. The classic RSA algorithm, for example, suffers from this attack. Therefore, to protect against this CPA attack, every implementation of this algorithm must pad the plaintext with random data before encryption.

Chosen (or adaptive) ciphertext attack

The attacker has a ciphertext c and the plaintexts of the ciphertexts (except c) that they can freely choose. The attacker can freely adapt the plaintext depending on the text obtained after each decipherment and analyze the resulting changes in the ciphertext. Their task is to recover the algorithm and key that was used. For example, the attacker has to analyze a cipher machine black box, i.e., whose inner workings are unknown.

Few practical attacks fall into this scenario, but it is important for proofs of security. If resistance against the attacks of this scenario can be proven, then resistance against every realistic attack of chosen ciphertext is granted.

###### Semantic Security

If an asymmetric algorithm is used, then the attacker should be assumed to know the public key. Thus, they can encrypt any plaintext of their choice and compare it to the ciphertext, i.e., they can mount a CPA.

IND-CPA secure To be “indistinguish- able under chosen- plaintext attack,” means that no attacker has a prob- ability signiﬁcantly higher than a half to distinguish two ciphertexts.

A cipher is secure against IND-CPA if no attacker can distinguish which one of two plaintexts, previously selected, corresponds to the ciphertext received afterward. Spe- ciﬁcally, the cryptosystem is indistinguishable under CPA if every probabilistic polyno- mial-time attacker has only an insigniﬁcant “advantage” over random guessing (Abdalla et al., 2016).

The four steps of the game IND-CPA with polynomial-runtime restriction (in the bit- length of the key k) on the attacker’s computations (carried out on creating the two plaintexts, step two, and on choosing the plaintext that corresponds to the ciphertext, step four).

1. A pair of keys is created, one secret and one public, both with k bits. The attacker receives the public key.
2. The attacker computes two plaintexts M0 and M1, of the same size.
3. The cipher machine randomly chooses a bit b ∈ {0, 1}, enciphers Mb, and passes the ciphertext to the attacker.
4. The attacker chooses a bit b’ ∈ {0, 1}.

The attacker who chooses the bit b’ in the fourth step randomly is right with a proba- bility of ½, where the probability P(b = b’) – 1/2 is insigniﬁcant, if the attacker has an insigniﬁcant "advantage.” This is if they win with probability ≥1 / 2 + =(k), where = is an insigniﬁcant function in k, i.e., for every (nonzero) polynomial function p there is k0 such that =(k) < 1/p(k) for every k > k0. An insigniﬁcant difference should be granted because the attacker easily increases their probability of success above 1/2 by guess- ing a secret key and trying to decipher the ciphertext with it. Although this game is for- mulated for an asymmetric cryptosystem, it can be adapted to the symmetric case by replacing the public-key cipher with a cryptographic oracle, a black box function that retains the secret key and encrypts arbitrary plaintexts at the attacker’s request.

Cryptanalysis — How to Break Encryption

### Side-Channel Attacks

Physical implementations used by side-channel attacks are e.g., measures of timing, power consumption, and electromagnetic or sound emissions. We will focus on timing attacks. These can be carried out remotely; however, the measurements often suffer from noise, i.e., random disturbances from sources such as network latency, disk drive access times, and correction of transmission errors. Most timing attacks require the attacker to know the implementation. However, these attacks can also be used to reverse engineer.

We will focus on the example of a timing attack that measures the time for computing integer powers. For this, we ﬁrst have to understand how integer powers are computed.

###### Exponentiation by Squaring

Exponentiation by squaring (also known as square-and-multiply algorithm or binary exponentiation) is an algorithm used to quickly compute large integer powers of a number by binary expansion of the exponent. This is especially useful in modular arithmetic. To compute bn, instead of b multiplying b by itself n times, only 2 • log2 n multiplications are needed.

Given a nonnegative integer base b and exponent e, to compute be mod M, expand the exponent binary, i.e.,

e = e0 + e12 + e222 + ! + es2s with e0, e1, . . . , es ∈ 0,1 ,

and compute

Side-channel attack This attack uses information about the physical imple- mentaion of a cipher machine.

Timing attack

An attack that meas- ures the runtime of cryptographic opera- tions and compares them to estimated ones.

1 2 22 2s

b , b ,b , . . . , b

mod M .

Because of b^{2n+1} = b^{(2n)2} = (b^{2n})2, i.e., each power is the square of the previous one (and at most M), each power, one after the other, is easily computable, yielding

e e + e 2 + e 22 + ! + e 2s

e 2 e1

22 e2

2s e

b = b 0 1 2

s = b 0 b b

! b s .

In hindsight, only powers with e0, e1, …, es equal to one count; the others can be omit- ted.

For example, to calculate 35 mod 7, expand

5 = 1 + 0 · 21 + 1 · 22

and calculate

31 = 3, 32 = 9 ≡ 2, 322 = 32 2 ≡ 22 = 4 mod 7

yielding

35 = 31 + 22 = 31 · 322 = 3 · 4 ≡ 5 mod 7 .

###### Vulnerable Algorithms

For the execution of binary exponentiation, we have a look at the exponent, especially the number of bits equal to one. Its runtime depends linearly on this number. Though only this number is insufﬁcient for ﬁnding a key, statistical correlation analysis on exponentiations using different bases helps derive the exponent.

Cryptoalgorithms that encipher using exponentiation modulo a large prime number, and are vulnerable to this attack. This includes RSA, Difﬁe-Hellman, and ElGamal. For example, in RSA, the message is the base b and the key is the exponent e. Brumley and Boneh (2005) demonstrated a network-based timing attack on SSL-enabled web servers using RSA that successfully recovered the private key in a day. This led to the wide- spread deployment of blinding techniques to conceal correlations between key and encryption time.

###### Example for the Exponential (as used in Difﬁe-Hellman)

Kochis (1996) exposed a ﬂaw in the following algorithm to compute modular exponen- tiation, i.e., to compute R(y) = yx mod n for n public and y known, but x secret. The attacker, by computing R(y) for several values of y and knowing n, y and the computa- tion time, can derive x as follows.

Let w be the bit length of x and put s\_0 = 1. For k ranging from 0 to w – 1: if the k-th bit of x is 1, then put R\_k = (s\_k \* y) mod n; otherwise, put R\_k = s\_k. Put s\_{k+1} = R\_k^k mod n

End (of For loop) Return (R\_{w-1})

According to the value of the k-th bit of x, either (sk × y) mod n or nothing is compu-

ted. Therefore, the execution time of the algorithm, for different values of y*,* will even- tually yield the value of the k-th bit.

Cryptanalysis — How to Break Encryption

### Modern Cryptanalytic Algorithms

To understand the reasons behind the design choices of each step of a block cipher algorithm, such as AES, one must understand which attacks it deﬁes. A powerful mod- ern cryptanalytic algorithms is differential cryptanalysis, applicable to block ciphers. It assumes a CPA; the attacker sends pairs of (slightly) differing plaintexts, whose cipher- texts they receive. They then study how differences in the input propagate (on differen- tial trails) through the network of encipherment transformations to differences at out- put. Daemen and Rijmen (1999) proved the resilience of AES against wide trails.

###### Prototypical Feistel Cipher

Let us demonstrate this technique in the toy model of a Feistel cipher (Heys, 2002), that divides the plaintext into blocks of 16 bits and subdivides each block into four blocks of four bits.

For each round, there is a corresponding (independent) key. In each one of the ﬁrst three rounds,

1. Add the round key C to the block B / B ⊕ C.
2. Substitute each of the 4 sub-blocks bits according to the table (in hexadecimal notation)

|  |  |
| --- | --- |
|  | |
| 2. | 0 |
| 3. | 1 |
| 4. | 2 |
| 5. | 3 |
| 6. | 4 |
| 7. | 5 |
| 8. | 6 |
| 9. | 7 |
| 10. | 8 |

|  |  |
| --- | --- |
| 11. | 9 |
| 12. | A |
| 13. | B |
| 14. | C |
| 15. | D |
| 16. | E |
| 17. | F |
| 18. | E |
| 19. | 4 |
| 20. | D |
| 21. | 1 |
| 22. | 2 |
| 23. | F |
| 24. | B |
| 25. | 8 |
| 26. | 3 |
| 27. | A |
| 28. | 6 |
| 29. | C |
| 30. | 5 |

Cryptanalysis — How to Break Encryption

|  |  |
| --- | --- |
| 31. | 9 |
| 32. | 0 |
| 33. | 7 |

1. Swap bit i from the sub-block j with j from the sub-block i.
2. In the penultimate fourth round, add the round key to the block, B /B ⊕ C, and substitute each of the four sub-blocks of four bits with the table.
3. In the last ﬁfth round, add the round key to the block, B /B ⊕ C.

In the fourth round, the last step, the permutation, is omitted as it would only permute the last ﬁfth round key. From a cryptographic point of view, it is superﬂuous.

In the last ﬁfth round, the last two steps (substitution and permutation) are omitted because the algorithm, being public (following Kerckhoff’s principle), can be undone by any decipherer without knowledge of the key. From a cryptographic point of view, they are superﬂuous.

The substitution table originates from the DES algorithm and is commonly called the substitution box (S-box).

###### Differential Cryptanalysis

A cryptanalyst’s dream is to learn whether a part of the chosen key is correct, i.e., whether it coincides with the corresponding part of the key used to encrypt the text. For example, in the Heys cipher, the key has 16 bits. If the cryptanalyst learns whether half (eight bits) of the tried key matches the corresponding half of the correct key, then they only need to test out all possible combinations of these eight bits and the remaining eight bits. Thus, the number of combinations that need to be tested is reduced from 216 = 65536 to 2 • 256 = 512.

Criterion for decipherment

In a brute-force attack, the cryptanalyst deciphers the enciphered text with each possi- ble key. To know whether the tried key is correct, they check whether the content is intelligible. They could do this, for example, by a criterion such as counting the fre- quencies of the letters, pairs, and triples of the deciphered text, and comparing them to the frequencies of the likely language of the plaintext. If they come close, then the plaintext is probably intelligible and the tried key was used by the encipherer.

If the cipher has a single round, then this criterion is applicable. However, if the cipher has two or more rounds, and the decipherer executes the last round of the decryption algorithm with a certain key, then this criterion is no longer applicable because the text obtained is the output of the encryption algorithm (with the same key) from the penul-

timate round. Instead, the criterion of differential cryptanalysis for having found the correct key is probabilistic. The key tested out is probably correct if, for a certain “incoming” difference ∆X and a certain “outgoing” difference ∆Y, plaintext pairs with difference ∆X result with a certain probability in cipher pairs with difference ∆Y.

For differential cryptanalysis to be applicable, the cryptanalyst must be able to encrypt any number of freely chosen plaintexts with the same key and examine the encrypted texts.

A differential is a pair D = (∆X, ∆Y) of input respectively output differences ∆X

respectively ∆Y. Differential cryptanalysis exploits the high probability of a difference

∆X := X’ ⊕ X’’ between two plaintexts X’ and X’’ propagating to a difference ∆Y = Y’ ⊕ Y’’ between the two ciphertexts Y’ and Y’’ (for X’ and X’’) (in the penultimate round). Here X’ ⊕ X’’ is the addition XOR, bit per bit, i.e., ∆X, indicates all the bits in which X’ and X’’ differ). The pair D = (∆X, ∆Y) is the differential. The difference of X’ and X’’ is ∆X := X’ ⊕ X’’*.*

For differential cryptanalysis to be efﬁcient, there must be a differential D with high probability pD. Therefore, among all incoming pairs with difference ∆X, the probability of an outgoing pair having difference ∆Y (in the penultimate round) is pD. More exactly, the encipherer will encipher a statistically signiﬁcant number of pairs (> 1/pD) of plaintexts with difference ∆X to count the number of the enciphered pairs of cipher- texts with difference ∆Y.

Frequency of differences for a substitution table

An afﬁne transformation A is the composition of a linear application, i.e., A(x ⊕ y) = A(x) ⊕ A(y) for all x and y, and of a translation, i.e., A(x) = x ⊕ x0 for some ﬁxed x0.

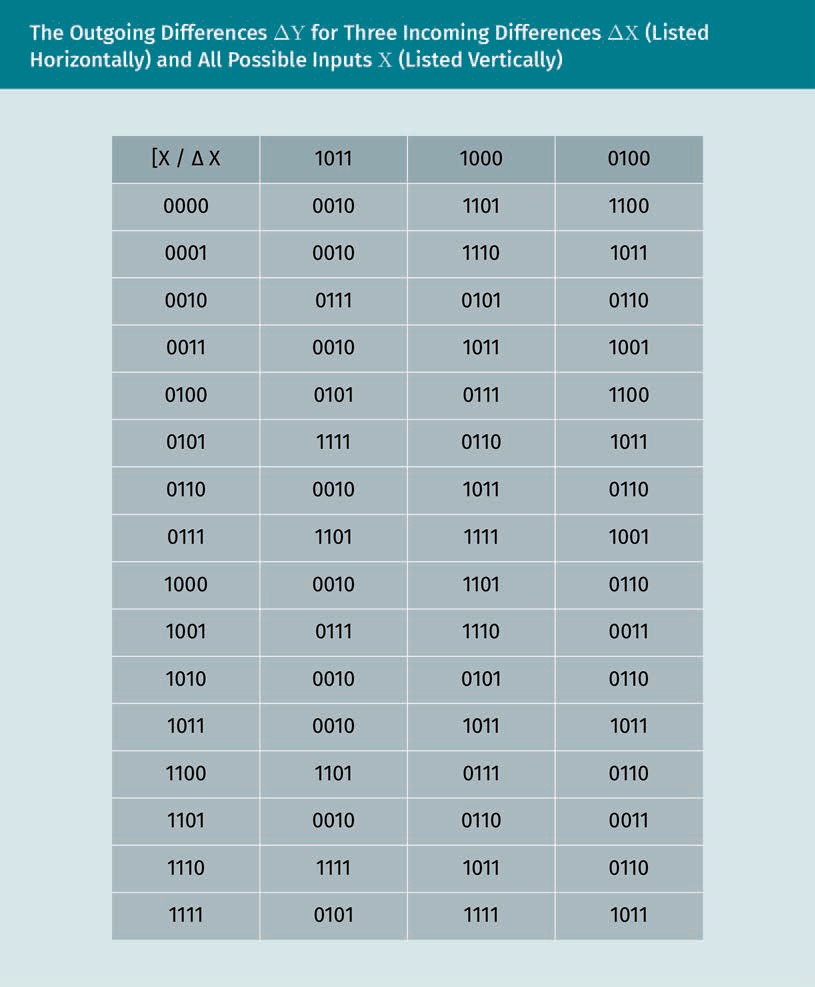
For an afﬁne transformation A, the outgoing difference ∆Y is independent of the incoming pair X’ and X’ (but only depends on ∆X). The transformation A

* if it is linear, then always, i.e., for every incoming pair with difference ∆X, the outgo- ing difference is ∆Y = A(∆X), and
* if it is a translation, then always ∆Y = ∆X.

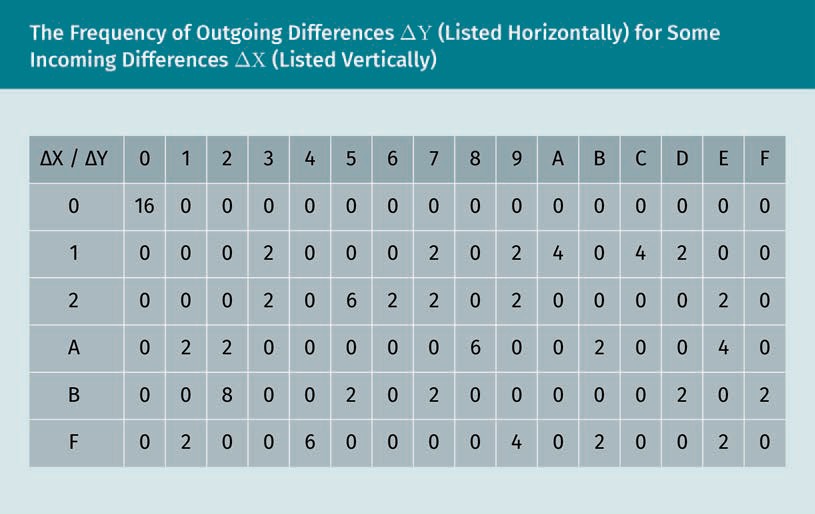
Regarding the ﬁrst and second function of each round of a Feistel cipher, the addition of the key is a translation, and the permutation is linear.

Therefore, the outgoing difference is independent of the incoming pair. However, the outgoing difference ∆Y of the substitution is not determined by the incoming differ- ence ∆X alone, but it depends on X’ and X’’! We examine the substitution table to ﬁnd a differential D of high probability pD, i.e., to ﬁnd an incoming difference ∆X with a large number of pairs X’ and X’’ that yield an outgoing difference ∆Y. Given ∆X, there are 24 = 16 possible inputs X’ (which determines X’’ = X’ ⊕ ∆X), and we count the frequencies of the 24 = 16 possible outgoing differences ∆Y = 0, 1, …, F.

Cryptanalysis — How to Break Encryption



Let us count for every incoming difference ∆X how many times each outgoing differ- ence ∆Y appears among all the incoming pairs X’ and X’’ such that X’ ⊕ X’’ = ∆X.



The entries for each line add up to 16, the number of all possible pairs for a given dif- ference. The ﬁrst line conﬁrms that two equal inputs result in two equal outputs. The highest number is 8 and reached for ∆X = B and ∆Y = 2. In addition, the number 6 comes up ﬁve times. We will choose our differentials among those with these high fre- quencies.

Example

In the frequency table

* of a translation, such as the addition of the secret key, all boxes are null except those in the ﬁrst column which have value 16.
* for a linear operation, such as a permutation of bits, in each line all the entries are null except one of value 16.

Differential trails

A differential trail is a tuple of differences so that every entry ∆Ui is the input of the S- box of the i-th cipher round. Given a Feistel cipher and a differential trail (∆U1, ∆U2,

…), and given the outgoing difference ∆Vi of the S-box of round i, the incoming differ- ence of the next round ∆Ui+1 is the result of applying the permutation to ∆Vi. (The key addition, as a translation, does not change the difference.)

We want to ﬁnd the most probable differential trail D in the Heys cipher

D = ∆U1, ∆U2, ∆U3, ∆U4

Cryptanalysis — How to Break Encryption

or at least a trail in which each differential (∆Ui, ∆Vi) is among the most probable. Every differential consists of four sub-differentials, corresponding to the four sub- blocks of four bits that constitute a block of 16 bits. To ﬁnd such a probable differential trail, in each round maximize the frequency of each sub-differential, i.e., the number of times the S-box transforms the incoming difference (of the sub-differential) into the outgoing difference. In particular, minimize the number of (active) nonzero sub-differ- entials.

An example of such a trail D is the following: let the difference in the ﬁrst round be

∆U1 = 0000 1011 0000 0000 ,

which is by S-box 2 replaced by

∆V1 = 0000 0010 0000 .

By the subsequent permutation, we obtain as difference entering the second round

∆U2 = 0000 0000 0100 0000

which is by S-box 3 replaced by

∆V2 = 0000 0000 0110 0000 .

Because the bits number 2 and 3 are nonzero, we obtain by the subsequent permuta- tion as incoming difference of the third round that with two active sub-differentials

∆U3 = 0000 0010 0000

which is by S-boxes 2 and 3 replaced by

∆V3 = 0000 0101 0000 .

Finally, by the subsequent permutation, the fourth round input is

∆U4 = 0000 0110 0000 0110 .

Let Si,j denote the substitution of the sub-block j by the S-box in the i-th round. On our differential trail, we enlist for each round i = 1,2,3 and for each sub-differential j = 1,2,3,4 different from zero, the probability of substitution Si,j transforming the incoming difference X (in hexadecimal notation) into outgoing difference Y:



If we suppose that the differentials of one round are independent of the differentials of the previous round (which is a negligibly inaccurate simpliﬁcation), then the proba- bility pD of the concatenated substitutions transforming

∆U1 = 0000 1011 0000 0000

into

∆U4 = 0000 0110 0000 0110 .

is the product of the probabilities of each substitution,

pD = 8/16 · 6/16 · 6/16 · 6/16 = 27/1024 .

To ﬁnd the key, for every possible combination of K5,5, …, K5,8 and K5,13, …, K5,16, an (integer) multiple m of 1/pD ≈ 38 pairs of plaintexts U’1 and U’’1 with difference ∆U1, the cryptanalyst

* 1. Enciphers the pair U’1 and U’’1,
  2. Reverses the cipher up to the S-box input in the fourth round by the round key

K5 = 0000 K5,5, . . . , K5,8 0000 K5,13, . . . , K5,16

to obtain the pair υ’4 and υ’’4 with difference υ4, and

* 1. Compares the difference ∆υ4 to ∆U4. If they match, then they increment the count n

by 1.

If for a combination of sub-blocks K5,5, …, K5,8 and K5,13, …, K5,16 the count yields n/m

≈ pD, i.e., the ratio between the number n of matched pairs and the number m of total pairs are close to the probability pD, then these sub-blocks are probably the sub-blocks 2 and 4 of the round key 5 used by the cipher.

Cryptanalysis — How to Break Encryption

To conclude that we found the correct sub-blocks, we use these hypotheses (a) that the differentials of a round are independent of the differentials of the previous round, and

(b) that a probability of matching pairs close to that calculated indicates the correct key.

Both have no rigid mathematical foundation, but are only plausible because, respec- tively, each round tries to diffuse as much as possible, i.e., to make the value of each output bit practically independent of all input bits. There is unlikely a key which is dif- ferent but reproduces the same probability.

Summary

Cryptanalysis, the art of breaking ciphers, is currently based on mathematics and applied by efﬁcient use of extensive computing power.

The security against brute-force attacks of a cryptosystem, and the recommended key sizes, relies foremost on its resistance to the most efﬁcient (known) methods of cryptanalysis (using a back door), and the computational effort needed to check all keys (using the front door) by checking the decrypted output for probable patterns of a plaintext.

If the secret information is stored as cryptographic hashes, a brute-force attack uses a rainbow table. This is promising against quickly computed hash functions such as MD4/5 and can be prevented by making the used hash function unique for each password, and adding a salt.

The only perfectly secure cryptosystem is the one-time pad where a key of the same size as the plaintext is added to the plaintext. Asymmetric cryptography uses mathematical methods, speciﬁcally modular arithmetic, to encipher. Symmetric cryptography uses more artisanal methods of ciphering, maximizing diffusion and confusion by iterated substitution and permutation. The security of symmetric cryptographic algorithms is based simply on its resistance against years of ongoing attacks.

A side-channel attack uses information from the physical implementation of a cipher machine such as measures of timing, power consumption, electromagnetic emissions, or sound emissions. A timing attack measures the runtimes of crypto- graphic operations and compares them to the estimated ones, e.g., one exploits that the runtime of the computation of a power depends on the number of nonzero bits of its exponent (In RSA and Difﬁe-Hellman, this is the key.).

Differential cryptanalysis applies to block ciphers assuming a chosen-plaintext attack. The attacker sends pairs of differing plaintexts whose ciphertexts they receive. They then study how differences on input propagate on differential trails through the network of encipherment transformations to differences at output.

# Unit 6

## Cryptology and the

## Internet



#### STUDY GOALS

On completion of this unit, you will have learned …

… about the history of the internet.

… the purpose of a Virtual Private Network and Transport Layer Security protocols.

… how emails are secured.

… to retrieve, authenticate, and encrypt internet domains.

DL-E-DLMCSEAITSC02-U06

1. Cryptology and the Internet

### Introduction

Before the internet age, it was unimaginable to use cryptography in everyday life. Today, everyday life on the internet would be unimaginable without (public-key) cryptography, e.g., for securely shopping online. Besides ciphering, cryptography establishes trust where previously paper documents were used for signing, identity authentication, granting authority, license, or ownership. Cryptography achieves this more securely; while a written signature is imitable, a digital signature is uniquely linked to the signed document.

To secure transactions on the internet, e.g., in electronic banking, commerce, or mailing, a cryptographic protocol (such as Transport Layer Security (TLS), formerly Secure Sock- ets Layer (SSL)) establishes trust and encrypts all trafﬁc. Trust is established when a client connects to a web server. The server’s identity then has to be guaranteed to avoid a man-in-the-middle attack. Up to now, central authorities issue digital X.509 cer- tiﬁcates. Trafﬁc is encrypted, e.g., in an open wireless network, such as those using log- in portals in public places, no trafﬁc between the router (that connects to the internet) and the client is being encrypted, neither by the router nor by the client.

### Internet Protocols

The various protocols that standardize the processing of exchanged data on the inter- net can be grouped into layers, ordered according to how highly structured the pro- cessed data are. The lower layers format the raw data and serve as interfaces for the upper layers, whereas the upper layers are closer to the user’s applications and handle more abstract data. Among these protocols, the two most important protocols (and those that were deﬁned ﬁrst) are the Transmission Control Protocol (TCP), and the internet Protocol (IP). They specify how data should be formatted, addressed, transmit- ted, routed, and received at the destination. Though the TCP/IP protocol reliably deliv- ers data packets over the internet, it neither guarantees security, conﬁdentiality, authenticity, nor manages sessions between a client and the server (e.g., suspension, termination, and restart of a session).

OSI model The model Interna- tional Standards Organization (ISO) model stacks the various protocols of the internet protocol suite into seven abstraction layers.

Most internet applications rely on a higher (application) layer, such as the HTTP proto- col for Web servers. The best known stacks of such layers of internet protocols are the Open Systems Interconnection (OSI) reference model as the international standard for the architecture of computer networks and the Internet Protocol Suite speciﬁed in Sec- tion 1.1.3 of Braden (1989). It is sometimes called the DoD(-Layer) model because devel- opment began in the late 1960s with a study under the supervision of the United States Department of Defense (DoD).

Cryptology and the Internet

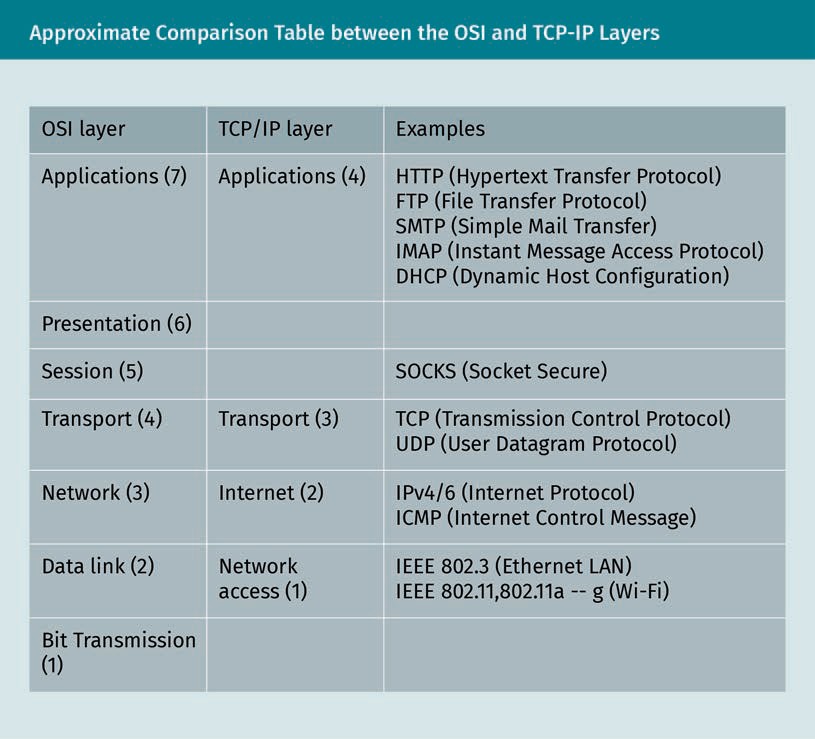
###### Comparison between the OSI and TCP/IP Reference Model

Since the internet runs on the TCP/IP reference model, which has only four layers. The standard ISO seven-layer stack is more of a theoretical abstraction than a practical standard. Unlike the standard ISO seven-layer stack, the TCP/IP four-layer stack evolved by being used rather than drafted and, thus, is known to work, even cross-plat- form. While the Internet Protocol Suite is descriptive, the OSI reference model was intended to be prescriptive. The OSI model is a wonderfully abstract construction, though the network, which exists and works, does not fully follow it. However, the OSI model is of historical and conceptual interest; as it precedes the older mode, the same principles apply.

When the models are (incorrectly) used interchangeably, both are referred to as the internet reference model.

Internet Protocol Suite

TCP/IP stacks the various protocols of the Internet Protocol Suite into four lay- ers.



1. The physical layer provides only the means to transmit raw bits. Electrical speciﬁca- tions such as network hardware or physical cabling are speciﬁed.
2. The network access layer (Link Layer) does not contain protocols of the TCP/IP fam- ily, but subsumes those for data transmission from point-to-point, to connect differ- ent subnets, such as Ethernet, Point-to-Point\_Protocol (PPP), or 802.11 (Wire- less\_LAN).
3. The internet layer comprises all protocols for the forwarding and routing of packets, i.e., determining the next intermediate destination for a received packet and to for- ward the packet there. The core of this layer is the internet Protocol in its version IPv4 or Ipv6 responsible for tasks like segmenting, error detection, and correction.
4. The transport layer comprises all protocols for establishing, prioritizing, maintaining, and terminating the communication between two computers on a network and checking that data sent from one computer to another has correctly reached its destination. Most important are the Transmission Control Protocol (TCP) to reliably send data streams, but also unreliable protocols such as the User Datagram Proto- col (UDP).
5. The session layer keeps track of the progress of data transfers, for session hiberna- tion (checkpointing), suspension, termination, and restart procedures, e.g., after a transmission error or interruption.
6. The presentation layer is responsible for the formatting of messages, converting data into a format understandable by an application such as a Web Server, e.g., encryption and decryption (or code set conversions, say from ISO Latin-1 to UTF-8).
7. The application layer comprises all protocols to exchange application-speciﬁc data over the network, e.g., the Mozilla HTML engine used by Firefox and Chrome or the SMTP protocol used by email programs.

However, different application protocols (e.g., HTTP, FTP, or IMAP) implement the func- tions of Layer 5, 6, and 7 differently, and do not separate these layers strictly. Hence, practitioners such as network engineers, subsume all those layers as Layer 5+, the application layer, just as the Internet Protocol Suite does.

### IPsec

A private IP network is a network, commonly made up of a local area networks (LANs) in residential and commercial environments whose computers have IP addresses that fall into the ranges speciﬁed by IPv4 (in RFC 1918, analogue ones exist in IPv6)

* from 10.0.0.0 to 10.255.255.255,
* from 172.16.0.0 to 172.31.255.255, and
* from 192.168.0.0 to 192.168.255.255.

These addresses can be used without approval from an internet registry. We often use “computer” to mean an endpoint of the network. This term also includes tablet, smart- phones, and other network devices.

Cryptology and the Internet

A Virtual Private Network (VPN) is used to connect networks between two companies (to form an extranet), various networks within the same company (intranets), and a sin- gle client via the internet to an intranet (remote access), which is the most common use of VPNs for the end-user.

To establish privacy, the connections between two closed networks use authentication and encryption. The parties mutually authenticate using a previously shared secret such as a password or certiﬁcate. Then the exchanged data is encrypted and decrypted at the endpoints.

###### Gateway

Brieﬂy, a gateway is a device that links two networks such as a router that distributes network trafﬁc ﬂow. An internet connection at home usually uses a router to deliver internet data packets to the devices at home. The ﬁrst device that connects to the internet is also known as a default gateway. By convention, the gateway has the lowest IP address in the subnet, an original address group.

A ﬁrewall ﬁlters data packets to protect an inner (private) network from an outside (public) network. It is usually located on a gateway, or possibly as software on a user’s computer, e.g., as part of the operating system such as the Microsoft Windows ﬁrewall.

Many consumer devices are both a router and a ﬁrewall. Consequently, these three terms, gateway, router, and ﬁrewall, are sometimes used interchangeably.

###### NAT

Network Address Translation (NAT) allows an Internet Protocol (IP) network to translate public IP addresses into private ones. NAT intercepts both incoming and outgoing IP data packets and changes the source or destination address in the packet header (and adjusts the checksums) to translate public IP addresses into private ones. Typically, NAT is implemented on gateways at the network’s boundary. NAT secures the private net- work by preventing external computers from accessing the internal network IP space. Typically incoming connections are only allowed as replies to outgoing connections (made by a computer inside the private network) (Schell & Martin, 2006).

###### Endpoints

The VPN can connect

VPN

A private network made up of two or more (spatially sepa- rate) closed net- works connected via an open network (such as the inter- net) is called a VPN.

* gateway-to-gateway. This is the simplest case, where both endpoints are directly accessible with hard-coded addresses and ports, e.g., on the same LAN or both pub- licly accessible.
* host-to-gateway (remote access). This is the common case of one of the two parties behind a gateway, e.g., a sales representative's notebook or a home ofﬁce computer. They are behind a (public or private) router that connects to a public server such as the company’s central server via the internet, which can be reached under a ﬁxed IP address or domain on the internet.
* host-to-host (peer-to-peer). NAT allows an IP network to translate public IP addresses into private ones of hosts behind a gateway, often by mapping many pri- vate IP addresses to a single public one by assigning different ports to them (Port Address Translation, PAT). This case is less common.

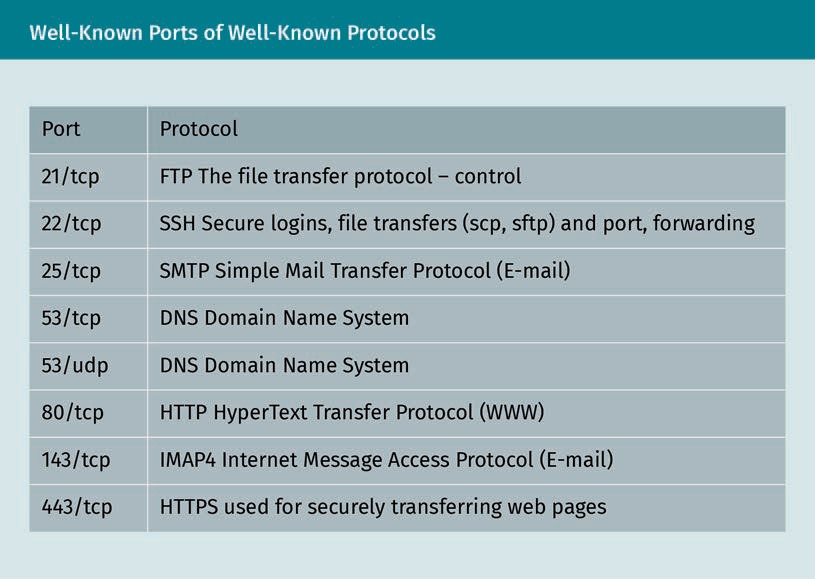
###### Underlying Transport Protocols

A VPN connection commonly uses either the TCP or the UDP protocol. The TCP protocol is more reliable but slower. It is useful for obfuscating VPN trafﬁc to look like regular HTTPS trafﬁc, decreasing the changes of being blocked. In UDP, packets are sent without any conﬁrmation, i.e., no guarantee that sent data arrived correctly. This duty is shifted to the applications that use the protocol, e.g., VOIP applications. User Datagram Proto- col (UDP) is the faster and preferable option for connecting a VPN if the above two restrictions that TCP circumvents do not apply.

Ports

A port is a software concept. It is a number (between 0 and 65535) that stands for a data channel into and out of a computer in a network. In the internet Suite of Proto- cols, it connects the TCP or the UDP to a higher-level (application) protocol.

Cryptology and the Internet



###### IPsec

Internet Protocol Security (IPsec) is a collection of protocols that secures IP communi- cations by authenticating and optionally encrypting each packet over public and inse- cure networks. It is mainly used for VPNs and is, at least in the business market, the most established protocol (Friedl, 2005).

Tunnel mode is usually used between gateways through the internet and connects two networks between two gateways. The end devices themselves connected via the two networks do not have to support IPsec. The security of the connection is only provided on the partial route between the two gateways. A new external IP header is used. The IP addresses of the two communication endpoints are located in the inner protected IP header.

Transport mode is usually used when the ﬁnal destination is not a gateway. It uses an additional IPsec header between the IP header and the transported data. It is less secure than tunnel mode.

Tunnel mode

This mode encrypts the whole IP packet.

Transport mode This mode encrypts the data portion but

leaves the original IP addresses as plain- text.

###### IPsec Protocols

IPsec essentially consists of the Internet Key Exchange (IKE) and Encapsulated Security Payload (ESP) protocol. IKE is the technical implementation of the Internet Security Association and Key Management Protocol (ISAKMP) framework. IKE uses UDP at port 500 for the initial key exchange and port 50 for the IPSEC encrypted data. ESP (for NAT traversal) uses UDP port 4500 and TCP port 10, 000.

IKE uses UDP at port 500 for the initial key exchange and establishes a common secret key either manually through the exchange of public keys, or automatically through the certiﬁcates from a trusted certiﬁcate server. To be used with VPNs for maximum secur- ity, IKEv2 is paired with IPSec. In comparison to other VPN protocols, the single most important beneﬁt of IKEv2 is its ability to reconnect quickly after VPN connection loss.

ESP (speciﬁed in RFC 3948) encrypts critical information by encapsulating the TCP or UDP data section by an ESP header. In Tunnel Mode, where the entire original IP packet is encapsulated with a new packet header added, ESP protection is afforded to the whole inner IP packet (including the inner header while the outer header (including any outer IPv4 options or IPv6 extension headers) remains unprotected (Manualzz, 2020, p. 63)

###### Other VPNs

OpenVPN is a popular unstandardized open source VPN protocol over the UDP or TCP protocol that uses TLS for key exchange and OpenSSL for encryption. It supports dynamically assigned IP addresses behind NAT gateways. Using TLS, it is incompatible with IPSec. It is implemented as software which is is stable and secure, thanks to the use of OpenSSL, runs on all common operating systems, such as Windows, Linux, macOS, Solaris, OpenBSD, and Android, and can scale up to thousands of clients.

WireGuard is a minimalist and modern open source VPN software built into Linux ker- nel 5.5 (and above). It is simple, user-friendly, and easy to set up. It is secure thanks to latest cryptographic algorithms and best practices. The software has short source code (initially around 4000 lines in comparison to hundreds of thousands in OpenVPN) and only allows UDP on IPv4 or IPv6. TCP support is missing. WireGuard does not verify the identity of the server by certiﬁcates nor can it manage IP addresses dynamically. For this reason, many providers refrain from using WireGuard for fear of risking their cus- tomers’ privacy (despite zero-log policies).

###### VPN Software

To securely use a shared VPN, we must trust its operator and users. Although most users are well-behaved citizens, maybe one individual is not, and law enforcement could eventually scrutinize all network trafﬁc. To set up our own VPN, there are several software options such as SoftEther (“Software Ethernet”) VPN. SoftEther is free and

Cryptology and the Internet

open source and can use many popular VPN protocols such as SSL-VPN (HTTPS), OpenVPN, and IPsec. The software also supports NAT traversal via SSL-VPN Tunneling to run VPN servers behind ﬁrewalls by using HTTPS so that even deep packet inspection (that looks at the metadata as well as data) is unable to detect SoftEther’s VPN trans- port packets.

### Transport Layer Security

TLS and its predecessor SSL encrypt data in both directions and guarantees the iden- tity of the server and the client. Originally developed by the Netscape Corporation, it is now supported by all the major browsers and the most common security protocol used on the World Wide Web (Lee et al., 2007)

For an application programmer, both provide a protocol that can be accessed almost like plain TCP. For a user, they establish a safe channel over the internet to allow the user’s private information, such as credit card or banking account numbers, to be safely transmitted via certiﬁcates: public and symmetric keys (indicated by a small padlock on the web browser’s address bar). TLS sits on top of the transport layer (in the OSI refer- ence model, layer 4) as it requires reliable data transfer.

###### X.509 Certiﬁcates

The X.509 certiﬁcate establishes authentication and encryption. These are organized hierarchically and pass trust from the upper to the lower level. Those at the top, which are trusted unconditionally, are called root authorities. In practice, this unconditional trust is achieved by the deployment of their self-signed certiﬁcates, e.g., as part of an internet browser installation.

The signature of a X.509 certiﬁcate is the encryption by the private key of the hash of the concatenation (V,SN,AI,CA,TA,A,KA)

where

* V is version X.509,
* SN is the serial number of the certiﬁcate,
* AI is the algorithm identiﬁer number,
* CA is the name of the certifying authority,
* TA is the validity interval time of the certiﬁcate,
* A is the name of subject, and
* KA is the subject’s public key.

The scheme of hierarchical authorities was created to establish trust through machines and allow the key exchange to be automated.

TLS/SSL

Cryptographic trans- port protocols TLS/SSL provide authentication, con- ﬁdentiality, and authenticity for data transmitted over a reliable transport, typically TCP.

X.509 certiﬁcate This ﬁle is signed by a certiﬁcate author- ity and contains the name, address, and public key of the Web site.

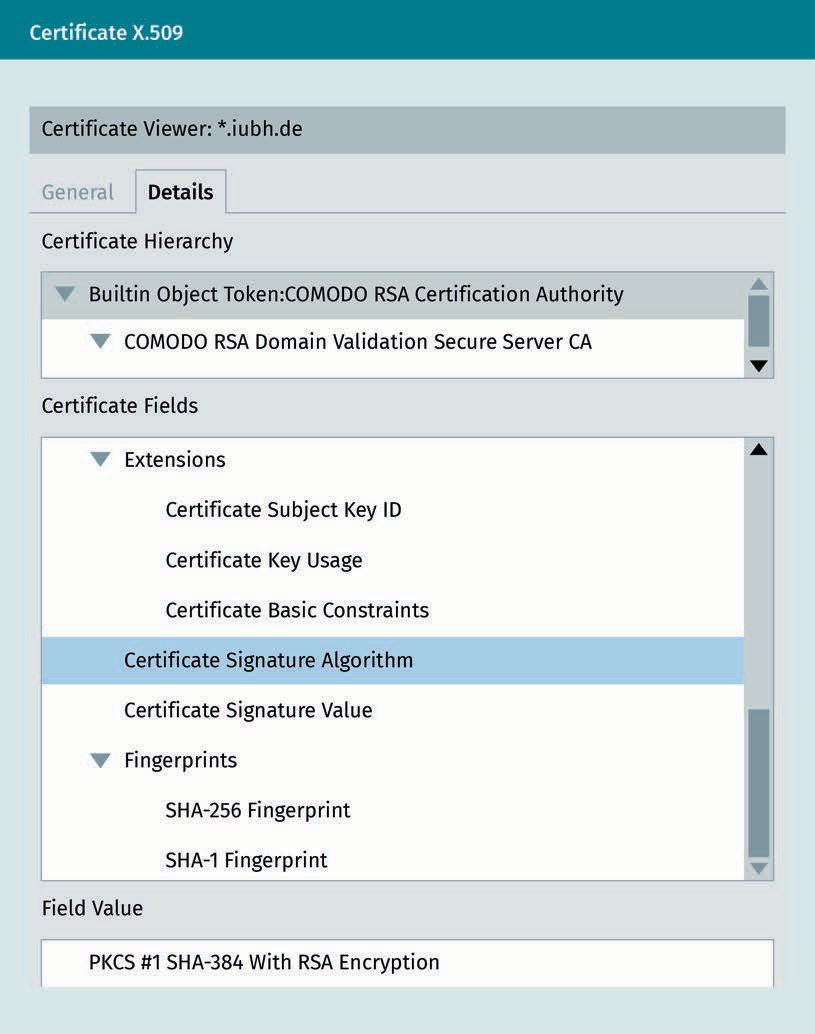
The padlock in the browser’s address bar reﬂects the level of security. Whereas “Let’s Encrypt” certiﬁcates only verify via email the ownership of the domain, companies offer an Extended Validation (EV) certiﬁcate that veriﬁes the identity of the owner. When issuing a common certiﬁcate, such as a free one from “Let's Encrypt,” the veriﬁcation is completely automated without any personal ofﬂine veriﬁcation. To obtain the certiﬁ- cate, access to the domain sufﬁces, e.g., by uploading a received ﬁle. When issuing an EV certiﬁcate, the veriﬁcation of the site owner is done in person.

###### Handshake

The heart of the TSL/SSL protocol is the handshake that sets up the session encryption. The deﬁnitive reference is RFC5426, a step-by-step illustration that traces every single exchanged byte (Driscoll, 2019).

The following steps are the ﬁrst steps between a client and the server (e.g., an e-com- merce site) to establish an encrypted connection (e.g., to receive credit card data from the client).

Cryptology and the Internet



In the image above, the server is www[.]iubh[.]de. The certiﬁcate is (a) signed by the Comodo RSA Domain Validation Secure Server CA intermediate authority, (b) signed by the root authority Comodo RSA Certiﬁcation Authority, and (c) self-signed.

The client creates a pre-secret, a random (pseudo-)number of 48 bytes, enciphers it using the public key (using the asymmetric algorithm initially agreed on), and sends it to the server. The server then deciphers the pre-secret using its private key.

The client and server calculate the secret (master secret), a number of 48 bytes, by a function PRF, master\_secret = PRF(pre\_master\_secret, ClientHello.random + ServerHello.random), which uses as input the pre-secret, and the nonces, that were communicated during the “Hello,” from the client server.

The client and the server derive four symmetric keys, for the algorithm initially agreed on, e.g., if it is AES, each one 16 bytes long, form the secret. Among these, the ﬁrst two serve to check data authenticity, and the last two serve to encrypt the data.

### Secure Email

Most internet protocols, among them the ones for emailing, such as POP3, IMAP, and SMTP, initially ignored security concerns and exchanged all data in plain text. Since then, various approaches have emerged to encrypt the data either only during trans- port (e.g., TLS), which is more convenient because it is easier to set up and use, or from end-to-end (e.g., S/MIME or OpenPGP). In general, the end-to-end encryption offers a higher security level. As the name indicates, it also encrypts and decrypts the data at the endpoints: the sender and recipient. An email sent with end-to-end encryption is unreadable to the mail servers, and no third party can scan email for malware. Instead, it has to be done by the user’s computer directly after decryption.

However, end-to-end protocols require additional effort and still only provide partial protection. They require the user to set up pairs of public and private keys and publish the public keys. Moreover, they protect only the content of the email, but no metadata, so that a third party can still observe who sent email to whom.

###### TLS

STARTTLS

This is a TLS over the plaintext protocol similar to IMAP4 and POP3, as deﬁned in RFC2595. It allows email servers to

encrypt all exchanged data between the servers as well as between servers and clients.

The most common email protocol for encryption during transport is STARTTLS. How- ever, certiﬁcate veriﬁcation is optional because a failure of veriﬁcation is considered less harmful than failure of email delivery, i.e., most email delivered over TLS provides only opportunistic encryption.

Use of STARTTLS is independent of whether the email’s contents are encrypted. An eavesdropper cannot see the encrypted email contents, but it is decrypted and thus visible at each intermediate email relay. In other words, the encryption takes place between the servers, but not between the sender and the recipient.

Cryptology and the Internet

###### S/MIME

The Secure Multipurpose Internet Mail Extension (S/MIME) is a protocol that standard- izes public-key encryption and the signing of emails. S/MIME’s IETF speciﬁcation enhances the Privacy Enhanced Mail (PEM) speciﬁcations of the 1990s. Although RSA public-key encryption was used initially, ECC has also been used since RFC5753 (Rams- dell, 2009)

Most email clients, such as Microsoft Outlook or Mozilla Thunderbird, support S/MIME secure email. Before use, one must install an individual key certiﬁcate. Before installing it, create a (set of) key pair(s), and send it to the CA (in-house or public) for them to sign. After successful veriﬁcation, the CA creates a certiﬁcate of the key by signing it with its private signature key.

###### Web of Trust

The web of trust is based on propagation of personal trust. It has the principal advant- age that it is a peer-to-peer system; it is independent of any particular third party such as an authority. However, as its major inconvenience, it needs personal maintenance. In practice, instead of the web of trust, it is more feasible to establish trust through another impersonal channel (by exchanging the ﬁngerprints of public keys), e.g., by post, phone, or a messenger.

###### OpenPGP

Since trust is a personal matter, the automatic unconditional trust of the user placed into root certiﬁcate authorities (principally companies) is unsatisfactory. Still, few peo- ple use OpenPGP. Most regard the concrete personal effort needed for maintaining the keys, which inherently requires the user’s estimate of their trust in the keys, dispropor- tionately large for the abstract beneﬁt of greater privacy and security received.

###### Examples of OpenPGP Programs

We present some programs that use the OpenPGP protocol, such as

* the command line program GPG to create keys and (de)encrypt and sign/authenti- cate for them,
* the extension Enigmail for the email client Thunderbird, and
* the extension Mailvelope for Firefox and Chrome to encrypt emails on Web interfa- ces.

S/MIME

An email encryption protocol that uses certiﬁcate authori- ties to establish trust for key distri- bution.

OpenPGP

This email encryp- tion protocol uses the web of trust for key distribution.

GnuPG

Gnu Privacy Guard (GnuPG or GPG) was written to offer open and free cryptographic methods to the public. It is a command-line program for encrypting and decrypting data (e.g., emails), and creating and verifying digital signatures to ensure authenticity of data. It underlies the cryptographic functionality of many cryptographic applications with a graphical user interface.

Enigmail

The Enigmail program is an extension to the graphic email program Thunderbird that adds to it the functions to encrypt and decrypt, as well as to sign and check email sig- natures. The user can access these functions by buttons in Thunderbird itself. To imple- ment these, it uses GnuPG underneath. The Thunderbird 78 release, planned for sum- mer 2020, has the functionality for email encryption and digital signatures using the built-in OpenPGP standard and replaces the Enigmail add-on and the dependency on GnuPG, whose installation was a hassle for beginners (Snipes, 2019).

Mailvelope

Mailvelope is an extension for the browsers Firefox and Chrome, developed by the Mail- velope GmbH, which adds encryption and decryption functions to the web interface of common email providers. Mailvelope is open source and based on OpenPGP.js, an OpenPGP library for JavaScript. It is comfortable, but comfort comes at the expense of security and thus it is safer to use an email client, such as Thunderbird. It is potentially vulnerable to Cross-Site Scripting (XSS) attacks, where one site accesses local data stored for another.

###### Automatic Key Exchange

Programs such as the extensions AutoCrypt (Thunderbird), prettyeasyprivacy (Outlook), or the messenger Delta-Chat (Android) offer only “Opportunistic Security—Some Protec- tion Most of the Time,” protection against passive, but not active, eavesdroppers where the encryption only protects the user as long as nobody is interested in them (Dukhovni, 2014). Such a program, precisely because of the lack of veriﬁcation of the owner of the private key that corresponds to the public key, is vulnerable to the MITM attack in which the attacker interposes themselves between the two communicating parties. It is perfectly possible to use someone else’s name on an email or WhatsApp account. Encryption of the communication only prevents it from being read by a third party, but does not guarantee the other correspondent’s identity. To avoid this attack, one must personally or via another channel, such as a telephone, check the ﬁngerprint (a cryptographic checksum) of the other correspondent’s public key. This is tedious, but unavoidable.

Cryptology and the Internet

### Secure DNS

The Domain Name System (DNS) is a distributed database analogous to a phone book of the internet. DNS translates human-friendly alphabetic internet domain address, such as https//www [.]iubh[.]de, to a computer-friendly numeric IP address, such as 194.6.193.105 (as can be found by the Unix command-line program nslookup). This inter- net domain address takes the form of the domain name of a machine, followed by a top-level domain (TLD), separated by dots, e.g., iubh.de has the domain name iubh and the TLD de.

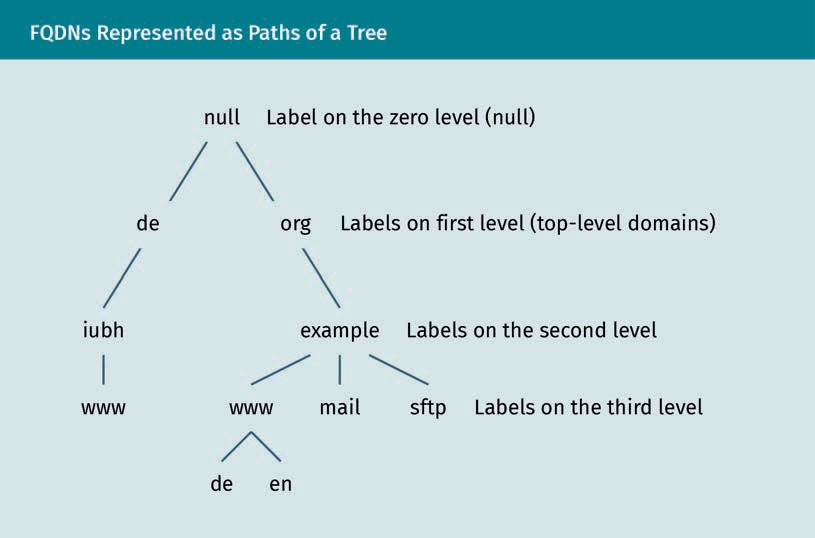
A fully qualiﬁed domain name (FQDN) is a unique worldwide name to address an IP address on the internet, which can be freely chosen under the rules determined by the Internet Corporation for Assigned Names and Numbers (ICANN).

Every FQDN ends in a top-level domain (such as “.de”). A subdomain (“www”) can be added to a FQDN (“iubh.de”) by prepending it with a dot. Common subdomain labels are [“www](http://www/).” for Web servers and “mail.” ; “smtp.,” “pop3.,” and “imap” are common for (outgoing and incoming) mail servers. Each domain (or label such as “iubh”) may con- tain at most 63 characters. The entire FQDN [(”www.iubh.](http://www.iubh.de/)de”) is limited to 255 characters.

The FQDNs are put in correspondence with the IP address by the entries of name serv- ers. The labels of FQDNs are represented as nodes of a tree. A FQDN is then a path of the tree. The highest node is the null or root label that represents an empty name. Below the highest node are those that represent a top-level domain (“de”). Below the nodes on the ﬁrst level are those that represent a domain (“iubh”). Below the nodes on the second level are those that represent a subdomain of the domain (“www”).

DNS

A database of all internet domain names, the DNS is distributed on hier- archically organized servers over the internet.



DNS servers use a set of databases distributed on servers over the internet that are organized hierarchically. DNS uses the UDP or TCP protocol on Port 54 (Mockapetris, 1987).

A DNS zone is the subset, often a single domain (such as iubh.de), of the DNS hierarchy which is described by a zone text ﬁle. A list of entries called resource records (RRs) map a FQDN to its IP address. The zone ﬁle format, as originally speciﬁed for the Berkeley Internet Name Domain (BIND) software, is used by most DNS server software. A root name server (root server) is the name server that resolves all FQDNs of a top-level domain (TLD) such as “.com.”

###### DNSsec

DNSsec This protocol authenticates the resolutions of a domain name to an

IP address.

The DNSsec resolution is signed on registration by the DNS server responsible for the zone ﬁle. This provides authenticity of the resolution, but neither conﬁdentiality nor authentication of the DNS server. To this end, further secure DNS protocols, such as DNScrypt, DNS-over-TLS, and DNS-over-HTTPS have been devised.

Protocol

DNSsec is a protocol that extends DNS. It includes all entries (Resource Records or RRs) of a DNSSEC packet. RFC 4034 speciﬁes the addition of the four RRs DNSKEY, RRSIG, NSEC, and DS.

The DNSKEY record passes a public key between the resolver and the name server. This public key is the one associated with the private key with which the authority server will sign hashes of RRSET records. The resolver will use the public key in the DNSKEY record to authenticate the message of the authority server by verifying its signature. RRSIG record contains the signer’s name and the signature of the record sent by the authority server, the signature that the resolver will later verify. There is one RRSIG record for each zone record in the signed zone ﬁle.

Signatures

DNSsec authenticates each RR with a digital signature. The RR owner is the primary authoritative name server as deﬁned in the Start-of-Authority RR (SOA-RR) entry of the zone ﬁle.

* 1. For each zone a ZSK, a pair of a public and private key, is generated.
  2. The public key is added to the zone ﬁle in the DNSKEY-RR.
  3. Each RR of the zone ﬁle is then hashed and signed by the private zone signing key.
  4. The resolver obtains the authoritative name server’s public key by the DNSKEY record.
  5. The resolver can then verify authenticity of the passed RR by verifying whether the decrypted hash of the RR using the public key (corresponding to the primary name server’s private key) matches the hash of the RR.

Cryptology and the Internet

If no server matches the request, then DNSsec proves that no such RR exists and uses a new type of record that sends the name of the ﬁrst (in alphabetic order) existing domain.

###### Virtual Hosting

Virtual hosting allows multiple DNS names to be hosted on a single server (usually a web server) on the same IP address.

The distinction between domain on the server was requested is made at the applica- tion level. The requested domain is sent, always unencrypted, in the following

* + - in the SMTP protocol, during the SMTP handshake.
    - in the HTTP protocol, by the HTTP header ﬁeld Host sent by the client as is obliga- tory since HTTP/1.1 (which is commonplace today, but whose inclusion cannot be enforced by the server).
    - in the HTTPS protocol, for a suitable assignment of certiﬁcates to domains, both, cli- ent and server, must support Server Name Indication (SNI)

###### Reverse DNS

A reverse DNS (rDNS) lookup or resolution works in a reverse order. It’s a query to the DNS to check whether a given IP address is in use and what domain name it is used for. This can be used for identiﬁcation purposes on the internet because it is recommen- ded that every host should match to a name (RFC 1912).

###### Secure DNS

DNSsec authenticates DNS replies with a signature from the authoritative DNS server on which the domain name was registered, not necessarily the DNS server that answered the DNS request. DNSsec thus offers authenticity of the DNS entries, but nei- ther conﬁdentiality as all data exchanged are unencrypted, nor authentication, as the correct DNS server does not need to authenticate.

Secure DNS protocols prevent the former, encrypt all exchanged data, and prevent the latter, authenticate the DNS server to the DNS client. The four principal contenders for encrypted and authenticated DNS queries and replies are NSCrypt protocol, DNS-over- TLS (or DoT), DNS resolution over TLS, and DNS-over-HTTPS (or DoH), DNS resolution over HTTPS (Hounsel et al., 2020).

Neither Windows, macOS, nor Linux support encrypted DNS queries by default at the time of writing. However, Android 9 supports DoT. Firefox supports encrypted DNS quer- ies by DoH, and Google is testing DoH over Chrome. DNSCrypt has less backing from the big companies.

###### Privacy Considerations

Because all of the other sent metadata usually include the requested DNS, encrypting DNS queries mainly shifts it from one party (the DNS server without encryption) to another (the DNS server with encryption). Since most requests on the internet leak the domain name, e.g., reverse DNS lookup or by protocol headers, the merit of secure DNS queries lies less in the protection against eavesdropping, but more in the authentica- tion of the DNS server. However, authentication of the requested domain is usually already provided by TLS. The downside is centralization of all DNS queries (away from the ISPs) to the DNS provider permits their bundled processing. While the data between the client and the server are encrypted during transport, they are decrypted at each end.

DNSCrypt Unstandardized secure DNS protocol without trusted cer- tiﬁcate authorities is

referred to as DNSCrypt.

DNSCrypt trusts a public signing key instead of X.509 certiﬁcates of an CA. The public key obtained from a provider veriﬁes keys by conventional DNS queries, known as short-term public keys used for key exchange or cipher identiﬁers. While servers are willing to reuse generated short-term key pairs, clients look for new generated key- pairs on each query (DNSCrypt, n. d.). The protocol sits over the TCP (mandatory) and UDP (optionally) transport protocols. Although the protocol has been around since 2013, it is not standardized (DNSCrypt, n. d.).

DNS-over-HTTPS was speciﬁed in RFC8484 and uses HTTPS to be indistinguishable from any other HTTPS trafﬁc and is thus practically never blocked. Firefox has added support for DoH through the DNS servers of Cloudﬂare. Google plan to test DOH with Chrome.

DNS-over-TLS, thanks to the IETF standardization in RFC7858, is the most widely suppor- ted in software. DoT clients authenticate the service they connect to using Simple Pub- lic-Key Infrastructure (SPKI) which is a joint effort from the IETF to simplify traditional

X.509 PKI and a supported standard of establishing trust.

DoT is plain DNS trafﬁc within a TLS connection using a dedicated port 853 (and occa- sionally on port 443). Up until the encryption by TLS, it is the same as DNS over TCP/IP instead of UDP. Since TLS is the encryption protocol used to secure almost all other internet services, the technology is well understood and constantly improved.

Summary

The protocols that standardize communication on the internet can be stacked into layers: the Open Systems Interconnection (OSI) reference model with seven layers and the Internet Protocol Suite with four layers. The layers are ordered according to how much structure the processed data have; the higher the layer, the closer to the user’s applications. The two most important internet protocols are TCP and IP, which specify how data should be formatted, addressed, transmitted, routed, and received at the destination.

Cryptology and the Internet

A VPN is a private network made up of two or more closed networks connected through an open network. The IPsec VPN uses the transport mode to establish point-to-point communication between two endpoints or the tunnel mode to con- nects two networks via two gateways. In IPSEC tunnel mode, IP packets are encap- sulated (tunneled) in other IP packets.

The cryptographic transport protocol TLS and its predecessor SSL offer authentica- tion, conﬁdentiality and authenticity for data sent through TCP. Authentication and encryption are established by X.509 certiﬁcate of an certiﬁcate authority. These are organized hierarchically and pass trust from the upper to the lower level, where the topmost are called root authorities.

Email is encrypted either only during transport or from end-to-end, which is more secure. In the latter, data are encrypted and decrypted at the endpoints, the sender and recipient, so emails sent are unreadable to the mail servers.

The DNS is a distributed database on the internet that resolves human-readable domain names into machine-readable IP addresses. DNS offers neither privacy, as all data exchanged is unencrypted, nor trust, as the DNS server does not need to authenticate. DNSsec offers authenticity by signing all DNS records, whereas more recent secure DNS protocols encrypt all exchanged data and authenticate the DNS server to a client.



# Unit 7

## Practical Aspects of Cryptology

#### STUDY GOALS

On completion of this unit, you will have learned …

… where, why, and how random numbers in cryptography are generated as well as the distinction between physical and pseudo-random number generation.

… how to ensure long-term security regarding future strength of key lengths and using means.

… to consider and counter possible future threats through quantum computing.

… best practices for using cryptography in application development.

… to conform to legal regulatory requirements by using encryption for data protection.

… to be aware of government trapdoors and security holes.

DL-E-DLMCSEAITSC02-U07

1. Practical Aspects of Cryptology

### Introduction

Even though the cryptographic algorithms are secure in theory, in practice, a lot may go wrong when implementing cryptography. Thus, as a software developer, caution must be taken and best practices adopted.

Most importantly, use what has been vetted such as software libraries that implement cryptographic functions such as encryption, decryption, signing, and veriﬁcation. Besides the cryptographic algorithms, the implementation of the random number gen- erator, which is notorious for exploits, is critical. As we saw in elliptic curve cryptogra- phy (ECC), if the same ephemeral key (usually randomly generated) is used twice to sign different documents with the same private signature key, then the ephemeral becomes known and reveals the secret signing key.

Even well-known time-proven cryptographic open source libraries, e.g., OpenSSL have security holes. The Heartbleed bug made common Web servers reveal on request pro- cessed secret data, such as passwords or server keys. Although it was quickly ﬁxed, the question remains if intelligence agencies exploit these, breaches through back doors built into cryptographic software.

Even the best cryptographic practice does not stop Moore’s law. This law predicts a doubling of the computing power every year that progressively weakens keys. Besides continual progress, there may be technological leaps, such as the quantum computer, which could break many common asymmetric ciphers such as Difﬁe-Hellman, RSA, ECC, and more involved alternatives.

### Random Number Generation

Cryptography requires random numbers to generate keys for (symmetric and asymmet- ric algorithms), and nonces, salts, and IVs (Initialization Vectors). Here, IVs are numbers that are usually randomly generated, disclosed, and used once in a cryptographic proc- ess to improve its security by making it unique. Examples of this include a session between a client and a server or a hash function used to store a password.

True randomness is critical for the generation of keys. This is less critical with dispen- sable nonces, where a key is sometimes used as a nonce, for which uniqueness can be sufﬁcient. As an example, the GnuPG strong random number generator for crypto- graphic keys builds a pool of 600 entropy physical disorder bytes and hashes these by SHA-1, or the GnuPG nonce generator adds 20 bytes containing the process ID number (PID) and the time in seconds and eight bytes taken at random from a strong random number generator, and hashes these with SHA-1.

Practical Aspects of Cryptology

###### Secrets

A secret, i.e., a secret sequence of bits, or, equivalently, a secret number, must often be generated by the computer

* if the secret needs to be generated automatically. For example, to extend the Difﬁe- Hellman key exchange to a public-key algorithm, the El Gamal algorithm generates for every plaintext to be encrypted or signed an ephemeral secret key. The popular signature algorithms DSA and its analogue over elliptic curves, ECDSA, which the El Gamal signature algorithm underlies, generate an ephemeral key pair for every plaintext to be signed.
* if the secret needs to satisfy a speciﬁc format. For example, in RSA with modulus N

= p q for prime numbers p and q, one key E is a number without any (prime) factor in common with (N) = (p – 1)(q – 1) .

* if the secret needs to be guaranteed to be random. To ease memorization, humans choose passwords with patterns.
* if the bit length of the secret needs to be signiﬁcantly larger than what a retainable password can provide. For example, for decryption of a ciphertext encrypted using RSA to take as long as an exhaustive search of a 112-bit secret key, the secret RSA key must be 2048 bits long, around 340 ASCII Armor letters.

For secrecy, it is necessary that the output of the generation is unpredictable. Since the generation algorithm is usually known, its input must be unpredictable. This excludes, e.g., using as input computer times or a number of a known sequence.

###### Uniqueness

As an example, if the secret ephemeral key used by the signature algorithms DSA and its analogue over elliptic curves ECDSA is known, then the signee’s permanent secret key can be inferred, i.e., an attacker can forge the signee’s signatures:

In particular, if two of the same signee’s ephemeral public keys used for different docu- ments coincide, then the secret ephemeral key used by either of these signature algo- rithms can be inferred. For example, the standard Java library for the generation of a random number on Android generated repetitive random numbers, thus allowing to forge signatures of users of an Android Bitcoin app (Ducklin, 2013). Consequently, in this case, the generated number must be unique.

###### Random Number Generators (RNGs)

While true randomness of numbers is critical for security-sensitive applications and can be generated using hardware random number generators, pseudorandom numbers are often sufﬁcient for less conﬁdential ones, e.g., for probabilistic experimental simu- lations such as the Monte Carlo method.

###### Pseudorandom Number Generator (PRNG)

Pseudorandom num-

ber generator A PRNG produces sequences of num- bers which appear independent of each other, i.e., which sat- isfy statistical tests for randomness.

A pseudorandom number generator requires careful mathematical analysis, such as the BigCrush Test Suite (L’Ecuyer & Simard, 2007), but are produced by a deﬁnite mathe- matical procedure. However, this apparent randomness is sufﬁcient for most purposes. The random number generators provided by most software libraries are pseudoran- dom, as can be seen in a list of such pseudo-generators (“List of random number gen- erators,” 2020). An ancient inﬂuential and simple one is the linear congruential genera- tor, which for a multiplier a, offset c and modulus m produces the sequence X0, X1, … inductively given by

Xn + 1 ≡ a Xn + c mod m .

Its successor that uses linear feedback shift registers, replaces arithmetic in /m by that in the binary polynomial ring 2 [X]. Xorshift+ 128 an efﬁcient option that passes the Big- Crush Test Suite. It is an adaption to pass the test suite of Xorshift that iteratively mul- tiplies a nonzero initial n -bit string by an invertible matrix of order 2n – 1. Other gener- ators that pass the test suite are the hash function SHA-1 and the symmetric algorithm AES with initial values.

###### Hardware Random Number Generator (HRNG)

A hardware random number generator is a computer device that generates random numbers from a physical process which is theoretically completely unpredictable, e.g., thermal noise, voltage ﬂuctuations in a diode circuit, or quantum optics. In Unix oper- ating system, randomness is gathered from the devices /dev/random and /dev/uran- dom. GnuPG, in absence of these, uses process statistics, but also supports the hard- ware RNGs inside the Padlock engine of VIA (Centaur) CPUs and x86 CPUs with the RDRAND instruction.

### Long-Term Security

How do we ensure that data encrypted today will still be secure in the decades to come? This is particularly relevant for certain long-term applications such as health data. In practice, the security of a cipher, and thus the recommended key sizes, relies foremost on its resistance to the most efﬁcient (known) methods of cryptanalysis (using a back door), and the computational effort needed to check all keys (taking the front door) by checking the decrypted output for probable patterns of a plaintext.

One has to take into account the increasing computing power to apply the strongest known cryptanalysis algorithms, and new algorithms (less predictable) and devices (more predictable) on the horizon, in particular quantum computing. Moreover, preven-

Practical Aspects of Cryptology

tive measures can be taken by diversifying the keys used for encryption so that a single compromised key compromises as little ciphertext as possible. Perfect Forward Security generates a new key (pair) for every new session, i.e., an exchange of ciphertexts.

###### Key Lengths

The world’s fastest supercomputer is IBM’s Summit and covers 520 square meters in the Oak Ridge National Laboratory, in Tennessee, USA. It has around 122 petaﬂops, i.e.,

1.22 · 1017 ﬂoating point operations per second. The number of ﬂops needed to check a key depends e.g., on whether the plaintext is known or not, but be optimistically assumed to be 1000. Therefore, Summit can check approx. 1. 22 · 1014 keys per second, and, a year having 31536000 seconds, approximately 3.85 · 1023 keys a year.

To counter the increasing computing power, we prudently apply Moore’s Law. Every twenty years computing power increases by a factor 220 = 1.05 · 106. To ensure that in, for example, sixty years, a key is not to be found during a yearlong search by world’s fastest supercomputer at least 4.44 · 1041 key combinations have to be used.

For a key of bit length n , the number of all possible keys is 2n. If n = 80 , then there are 280 possible key combinations. While this number is sufﬁcient for now, the projected fastest super computer in twenty years will likely ﬁnd it.

AES

For the symmetric algorithm AES, the fastest known algorithm currently is the exhaus- tive key search, to try out all possible keys, whose complexity (= the number of opera- tions) is 2n. The minimal AES key length is 128 bits, i.e., there are 2128 possible key com- binations. Therefore, the chance that the world’s fastest supercomputer in sixty years ﬁnds the secret key is around 10–8, a millionth percent. We conclude that the minimal AES key length is safe against brute-force attacks for the years to come.

Asymmetric algorithms

In single-key cryptography, its cryptanalysis exploits statistical patterns through its reli- ance on computationally difﬁcult mathematical problems, i.e., whose runtime grows exponentially in the bit-length of the input. The cryptanalysis of two-key cryptography is that of computational mathematics. To ﬁnd an algorithm that quickly computes the solutions of the difﬁcult mathematical problem. For the classic asymmetric algorithms, the prime factor decomposition used in RSA, or the discrete logarithm in the Difﬁe- Hellman key exchange, the fastest algorithm is the General Number Field Sieve (Len- stra, 2006) whose number of operations, roughly, for a large number of input bits M, putting N = log (2) M, is L(N, 1 / 3 , (64 / 9)1 / 3^) where

L N. r, c . = exp cNr logN 1 − r .

To compare this to the number of operations 2m required for exhaustive key search of a key of bit length m , we put n = log (2) m , equate both numbers of operations, and obtain

Nlog N 2 ≈ 9n/64 ≈ n3/7

which must be solved numerically for N. For example, for m = 80 , i.e., n we ﬁnd M = 1024 , i.e., N, to satisfy log (N)2 = 43: N log (N)2 500 and 553 / 7 000. Thus at least as much computational effort is needed for ﬁnding a private 1024 bit long key for RSA and Difﬁe-Hellman by the General Number Field Sieve as for ﬁnding a secret 80 bit long key (say, for AES) by exhaustive key search.

For the logarithm over a ﬁnite elliptic curve, the fastest algorithm today is the generic baby step, giant step algorithm (or Pollard’s ρ-algorithm) whose complexity is roughly 2√n. For the computational effort in ﬁnding a private key of bit length N for ECC to be comparable to that for ﬁnding a secret key of bit length n (say, for AES) by exhaustive key search, the key length must double, i.e., N 2n.

With these formulas, we can calculate that the key length of a (private) RSA or Difﬁe- Hellman key must be at least 3072 bits to be as secure as that of an AES key of 128 bits. Moreover, the key length of an ECC key must be at least 256 bits to be as secure as that of an AES key of 128 bits.

Indeed, these key sizes are sufﬁcient for the decades to come, assuming that no faster algorithm than those known is discovered.

###### Quantum Computing

Quantum researchers hope to build computers that harness a phenomenon known as superposition where the quantum system can have multiple states until a measure- ment breaks it down into a single state.

Quantum computer A quantum com- puter uses qubits, which can be “entangled,” in between on and off so that it can carry out multiple calcula- tions at the same

time

A classical computer stores information in bits. Each bit is either on or off. A quantum computer that uses qubits has a ﬁnal output that depends on the interferences gener- ated by them (Steane, 1998). In order to build a useful quantum computer, we face some challenges. A quantum computer must keep its qubits entangled long enough to complete a computation. Errors caused by inevitable interactions with the environment need to be ﬁltered out and corrected. In addition, the state of a quantum system gets interfered by measuring what needs to be remedied.

Quantum states

The properties measured on one particle depend on the operations carried out on all the others. If a particle can take two states (denoted as 0 and 1), a system of two parti- cles can take e.g., states 00 or 11, or even a superposition of these states. For example, each of the two particles considered in isolation is measured randomly as a 0 or a 1, but the particles are “twins,” i.e., the measurement of a state of one of the two particles forces the other particle into the same state. The violation of a statistical inequality proves that this is a characteristic of the particles and not due to a (hidden) link (Bell, 1964; Aspect et al., 1982).

Practical Aspects of Cryptology

A physical quantity in quantum mechanics is called observable and is usually in a superimposed state. Only in special (eigen)states it has a uniquely determined (eigen)value. In general, the eigenstate is the result of applying an observable to a (superimposed) state and choosing an eigenvalue.

Formally, a superimposed state is a complex unit vector, i.e., a vector of length one with complex entries and an observable A is a (self-adjointed) operator on the vector space of all states. Because A is self-adjoint, there is a basis bi such that A bi = ai bi for real ai. Each bi is an eigenstate and ai its eigenvalue.

An eigenstate bi is an eigenvector of A and interpreted as the possible outcome of a measurement for a given observable. A general state |= ci bi is a linear combination of eigenstates. The complex coefﬁcient ci of bi is called the probability amplitude of bi; and its absolute value |ci | in [0, 1] is the probability of the measurement of bi.

Quantum bits

While a digital computer processes bits, that can be in one of two states, 0 or 1, a quan- tum computer processes quantum bits (qubits) whose states superpose. The state space, the set of all possible states, of a computer with n bits is given by the set of all strings

Q . = 0,1 n = 0, . . . , 0 , . . . , 1, . . . , 1

n

of n bits, of which there are 2n. The state space of a quantum computer with n qubits is given by all probability amplitudes (or Schrödinger wave functions) on Qn, i.e., all strings of 2n entries of complex numbers of unit length. A probability amplitude is a vector of unit length with complex entries indexed by Qn, i.e., a string Qn of 2n complex numbers

λ = λ 0, . . . , 0 , . . . , λ 1, . . . , 1

such that ~q Qn ~ |(q)|2 = 1.

The basis states are those probability amplitudes that have a single nonzero entry of value 1, of which there are 2n. The state whose entry of value 1 is in Qn is denoted by |. For example, a single qubit superimposed between 0 and 1 is denoted by a | 0 + b | 1 such that a2 + b2 = 1, the probability amplitude of two qubits together superimposed between 00, 01, 10, and 11 is denoted by

α · 00 > + β · 01 > + γ · 10 > + δ · 11 >,

with ||2 + ||2 + ||2 + ||2 = 1.

Each elementary operation on the state space is then described by a (unitary) matrix of 2n columns (orthogonal to each other). Each column is the probability amplitude obtained by applying the operation to the corresponding basis state.

###### Quantum Speed-Up and Post-Quantum Cryptography

A quantum computer offers many possibilities for solving classical problems that a classical computer does not have (Di et al., 2004). To ﬁnd an item among an unordered list of N items on a classical computer, on average - , i.e., O(n) operations are needed.

2

However, a quantum algorithm exists that achieves this in only about

π , i.e.,

4

N

O operations. Applications also exist where a quantum computer is superior, such

N

as solving Simon’s problem (Brassard & Hoyer, 1997) or prime factorization and discrete logarithms with Shor’s algorithm (Shor, 1994).

Unsurprisingly, given a sufﬁciently large quantum computer, current public-key encryp- tion schemes are easily broken. For example, Shor’s factoring algorithm can be used to encounter RSA encryption, while factoring with classical algorithms is quite time-con- suming. But quantum mechanical effects are also an enabler for quantum encryption. Although quantum computing allows the solving of many classical problems much more quickly than with a classical approach, not all of them are conjectured to remain hard even on a quantum computer, e.g., problems that are NP-hard such as ﬁnding the closest vector to a lattice.

Lattice ciphers

The lattice cipher or Goldreich-Goldwasser-Halevi (GGH) cipher uses a one-way trap- door function based on the closest vector problem, which is NP-hard (Goldreich et al., 1997). Given any basis of a lattice, it is easy to generate a vector close to a lattice point by adding a small error vector to the latter. For returning back to the lattice point, we can take a particular kind of basis. The private key consists of a lattice basis b, i.e., an invertible matrix with integer entries which consists of nearly orthogonal vectors. Fur- ther, it consists of a unimodular matrix u, i.e., an invertible matrix with integer entries whose inverse has integer entries, which transform vectors with integer entries.

The public key is the basis B = u b of the lattice L. If a message m (a vector (m1 , …, mn ) of integers) and a public key B are given, then to encrypt the plaintext m we com- pute the lattice point vector v = m B, choose a small error vector e, e.g., one with entries in 0, and compute the ciphertext c = v + e (= m B + e). If a ciphertext c and the private key consisting of a lattice basis b and unimodular matrix u are given, then to decrypt the ciphertext c computes

c · b−1 = m · B + e b−1 = m · u · b · b−1 + e · B−1 = m · u + e · B−1 .

While m u is a vector of integers, the error term e B–1 is fractional. Thus, the integral summand m u can be distinguished from the fractional summand e B–1 by removing the fractional error term e B–1 to obtain the integer vector m u. Finally, we compute the plaintext m = (m u) u–1.

Practical Aspects of Cryptology

Code-based cryptography

Code-based post-quantum ciphers are asymmetric ciphers that are based on error cor- recting codes to transmit bits over a noisy channel. For example, to avoid that Alice sends, say 01 , but Bob receives 11, a simple solution would be that Alice repeats each bit thrice, 000111, and Bobs takes for each group of three the bit that appears most often, e.g., 100110 would be decoded to 10. However, this encoding scheme is limited to one erroneous bit in each group of three bits. Instead, a linear code multiplies the bit vector v with a matrix M, i.e., computes w = v M. For any number of erroneous bits n, the matrix M can be chosen such that, at most, n erroneous bits in w are correctable (to v). A well-known example, the McEliece cipher uses linear error-correcting codes (McEliece, 1978).

The private key is made up of three matrices, an error correction matrix c, and inverti- ble matrices S and P. The public key is C = S c P can correct n error bits. To encrypt, compute w = v C + e for an error term e containing n errors. To decrypt knowing P–1 and *c* allows one to remove S e and then apply S–1.

Hash-based cryptography

Hash-based cryptography relies on the security of cryptographic hash functions rather than on the hardness of mathematical problems. A one-time signature scheme is derived from a one-way function such as a cryptographic hash function (Lamport, 1978).

Let k be the private key and ﬁx an integer L. The public key is K = HashL (k), the L - fold nested application of the hash function to k.

HashL k = Hash Hash . . . Hash k .

The signature S of a message, given by an integer M < L, with the private key k is the M -fold nested application of the hash function to k , i.e., S = HashM (k). The signature S is checked by the equality

HashL − M S = HashL k = K .

This simple scheme is insecure, e.g., signatures can be forged. From the signature S of

M , one derives the signature of M + 1 by

Hash Hash K M+ 1 = Hash S .

Therefore, we must not only sign M by k, but also L – M by a different key. Still, this scheme is computationally too expensive because messages need to be split into shorter ones so that encryption is computationally feasible. Moreover, the scheme can only be used to securely sign one message per key. If a private key signs more than one message, then a signature can be forged. If messages m and M are signed with key k’

and L – m and L – M with key k’’, then further iterated applications of H to the signa- ture (s’ or S’ corresponding to the smaller value between m and M for k’ or k’’) forge signatures for messages between m and M.

###### Perfect Forward Secrecy

Perfect forward secrecy is when correspondents, who have exchanged their (perma- nent) public keys and established mutual trust, create an (ephemeral) session key and sign it with their (permanent) private keys. This is done to avoid a man-in-the-middle attack. After correspondence each correspondent deletes the ephemeral private key. This way, even if the correspondence was recorded, it cannot be deciphered later on, i.e., it cannot be deciphered by obtaining a correspondent’s private key.

One example is the TLS protocol version 1.2, which encrypts much of the communica- tion over the internet. More speciﬁcally, in the handshake between client and server, after the client has received (and trusted) the server certiﬁcate, the server and the cli- ent exchange an ephemeral public key which serves to encrypt the communication of this correspondence. This ephemeral key is signed by the permanent public key of the server. The creation of this asymmetric key in perfect forward secrecy makes the crea- tion of a symmetric preliminary key by the client in the TLS handshake superﬂuous.

The mutual secret key can be established, e.g., by the DHE (Difﬁe-Hellman Ephemeral) or ECDHE (Elliptic Curve Difﬁe-Hellman Ephemeral) protocol. This key is then used as input to generate a secret symmetric key, e.g., for the AES encryption algorithm.

### Incorporating Cryptography into Application Development

###### Risk Analysis

Risk analysis measures the likelihood that an organization’s security will be breached, e.g., by an intruder exploiting network vulnerabilities. Risk analysis calculates the inﬂic- ted damages from such a breach be they material (such as data privacy violation) or immaterial (such as a reputation loss). Risk is often measured as ﬁnancial risk and countered similar to insuring against threats such as theft. Risk analysis assesses the threats (by an intruder or company’s insider) to the company’s computer network. According to the CSI/FBI Survey, the biggest ones are, in order, (1) viruses, (2) employee abuse of networks, (3) denial of service (DoS), and (4) stolen intellectual property.

Risk analysis assesses the values of the company’s assets, consisting of material val- ues, such as the company’s computer network, and immaterial values, such as the company’s reputation. Within a risk analysis, the company’s vulnerability, i.e., where and how a security breach could happen, are discovered. Preventive measures, such as ran-

Practical Aspects of Cryptology

dom generation and regular revision of passwords, anti-virus software, and intrusion detection systems, such as ﬁrewalls, are prescribed. Finally, risk analysis helps to build plans to quickly recover from an exploit.

###### Don’t Roll Your Own Crypto!

The ﬁrst rule of cryptography is: don’t implement cryptography yourself in production code. Instead, leave it to the experts and use a proven library that has stood the test of time under the scrutiny of cryptanalysts rather than a homemade one.

In practice, a cryptographic algorithm is secure if it has proved resilient by defying all attacks. An established cryptographic algorithm should be used and proven ciphers, such as AES and RSA, are recommended. Security cannot be assured if a new one is used. In particular, it is a bad idea to design a new cryptographic algorithm. Similarly, a software library is secure if it has proved resilient by defying all attacks. An established software library should be used and common software libraries, such as Libsodium (or the widely-used OpenSSL), are recommended.

###### Authenticated Encryption

Programmers often confuse encryption and authentication. While encryption ensures conﬁdentiality, authentication ensures integrity. When they have to be done separately, ﬁrst encrypt, then authenticate. For example, compute a MAC, Message Authentication Code, and verify the MAC before decryption. *A priori*, there are three options for authenticated encryption.

1. Encrypt-and-Authenticate. The plaintext is encrypted with a MAC of the plaintext and appended to the ciphertext (as in SSH).
2. Authenticate-then-Encrypt. The computed MAC of the plaintext and the plaintext are encrypted (as in SSL).
3. Encrypt-then-Authenticate. The plaintext is encrypted and a MAC of the ciphertext appended (as in IPsec).

This way, one can verify the MAC and discard texts without decryption. Thus, the amount of DoS attacks is reduced while also minimizing the offered “attack surface.”

Only Encrypt-then-Authenticate is proven secure, i.e., secure against IND-CCA (indistin- guishability of ciphertext for chosen ciphertexts). From here, an attacker can ask for any pair of ciphertexts to be deciphered (excluding the ciphertext in question), and still cannot distinguish which one between two plaintexts corresponds to the ciphertext.

The fact that neither Encrypt-and-Authenticate nor Authenticate-then-Encrypt are secure in theory, but Encrypt-then-Authenticate is, does not in practice mean that the former two are insecure, nor that the latter is secure. However, there have been a num- ber of vulnerabilities for the former, while none for the latter. Therefore, it is best to

use a library that handles authenticated encryption as a whole, instead of only offering these functions separately, where one has to be manually composed correctly. Stick to the golden rule that as much as possible is reused instead of reimplemented.

###### Password Storage

To store a password, hash it using a key derivation hash function and erase the plain- text password from memory as soon as it is received. Possible hash functions would be scrypt, that uses a nonce and iteration count to counter rainbow attacks, PBKDF2, in addition to a password. Do not use a fast hash function like MD5, which is more vulner- able to rainbow table attacks. Take great care even with less sensitive applications because some users might reuse these passwords for more sensitive ones.

Finally, we recall that encoding (e.g., base64 encoding) and compression (e.g., Zip com- pression) aren’t encryption as they hardly obfuscate information. Encoding and com- pression algorithms are both reversible, keyless transformations of data. Encodings transform data for better processing, while compression reduces data as much as pos- sible.

### Legal and Regulatory Aspects

General Data Protec- tion Regulation (EU)

2016/679

The GDPR governs the exposure of per- sonal data that is processed and stored by hand or by computers. Since 25 May 2018, it applies to all companies within the European

Union.

In addition to the GDPR, other national data protection laws may apply. In Germany, this is the federal data protection act *Bundesdatenschutzgesetz* (BDSG) and the data protection acts of the German federal states (Datenschutz, 2020).

GDPR wages the interests of companies and consumers in the digital age and protects every citizen’s fundamental right to informational autonomy by granting the concerned citizen transparency and ultimate authority in the processing of their personal data. In other words, the processing of personal data must always have a clear purpose that has been clearly conﬁrmed by the concerned person. Additionally, the data must be protected, and they must be deleted as soon as the bespoken purpose ends expires (e.g., credit bureaus such as the SCHUFA in Germany).

###### Personal Data Protection

Every staff member who processes personal data must be trained on data secrecy. In general, forwarding personal data to third parties is inadmissible without the consent of the concerned person. If extraordinary admission is granted, then data must be encrypted and sent separately for each purpose, so that third parties neither eaves- drop nor collect data.

Personal data according to the GDPR are

Practical Aspects of Cryptology

* general personal data (e.g., name, birth dates, address)
* identiﬁcation numbers (e.g., tax ID, health insurance number)
* physical characteristics (e.g., gender, hair and eye color)
* ownership characteristics (e.g., vehicle and real estate ownership)
* value judgements (e.g., school and work records)
* bank data (e.g., credit information, account balances)
* customer data (e.g., orders, mailing addresses)
* online data (e.g., IP address, location data)

Special data that needs special protection is in general (but listed similarly, e.g., in Paragraph 4, 9 of the GDPR): ethnic origin, political opinions, religious beliefs, sexual orientation, or syndicate membership and biometric data, such as gene data or health records.

Consequently, in health care, patient information is strictly conﬁdential by law. In Ger- many, only encrypted documents may be transmitted, and only data necessary for treatment can be collected. The patient data must be securely stored and kept conﬁ- dential by the staff. In Germany, the unauthorized disclosure of patient data subject to professional secrecy can be punished by Section 203 of the Criminal Code (StGB) with a monetary fee or up to one year of prison.

###### Government Trapdoors

Leaked documents show that the American National Security Agency (NSA) searched for vulnerabilities through the Heartbleed bug. The NSA has sabotaged international encryption standards by purposefully weakening algorithms for later decryption. For example, in the speciﬁcation of the ﬁrst US Data Encryption Standard (DES) in the 1970s, the NSA was suspected of weakening DES by shortening the key length (de Leeuw & Bergstra, 2007).

The Escrowed Encryption Standard (EES) is a chip-based symmetric encryption system developed by the NSA. It was developed as part of a US government project to provide electronic devices sold to the general public with a security chip. The encryption key was to be provided to the government, which would then be able to eavesdrop on com- munications if necessary.

The main difference from other encryption methods is that, if necessary, US authorities can get access to the keys used by two users to exchange data. The procedure is speci- ﬁed in such a way that two keys are required for eavesdropping, which are deposited with different authorities and which should only be released at the same time by court order. This ofﬁcial access possibility is not achieved by a built-in back door in the tech- nical sense, but by depositing two partial keys. If the legal conditions are met, the two parts of the key are issued and joined together.

The Swiss Crypto AG was an internationally active company in the ﬁeld of information security. Between 1960 and 1990, at the height of the Cold War, Crypto AG was a leading company for encryption devices serving more than 130 countries. The US Central Intelli-

gence Agency (CIA) was concerned about being unable to decipher foreign messages and approached European manufacturers, including Crypto AG. The (West) German for- eign intelligence service BND and the CIA secretly bought the company in 1970. They arranged for many states to be supplied with machines with weaker encryption that could be decrypted by the intelligence agencies in Operation Rubikon (Leeuw & Berg- stra, 2007; Theveßen et al., 2020).

The company enabled these two services to decipher encrypted messages between the 1960s and 2010. Although the Soviet Union and China were never among Crypto’s cli- ents, the CIA could learn about some of their exchanges thanks to third countries equipped with the tampered devices. The CIA estimates that it could, for example,

* + read around 85% of the Iranian coded messages sent in the late 1980s,
  + spy on Egyptian communications during the 1978 Camp David negotiations,
  + spy on Argentinian messages during the 1982 Falklands War, and
  + gather decisive information during the 1989 invasion of Panama.

###### Heartbleed

Heartbleed was a vulnerability in the SSL implementation OpenSSL, used in the popu- lar web servers Apache and nginx, which ran two-thirds of the web pages at the time of the bug. The Heartbeat allows server and client to keep a TLS connection alive by send- ing a message of any content (payload) from one end to the other, which is then sent back in the same way to show that the connection is alive.

The RFC 6520 Heartbeat Extension tests TLS/DTLS secure communication links by allow- ing a computer at one end of a connection to send a “Heartbeat Request,” which con- sists of a payload, along with the length of that payload as a 16-bit integer. The receiver must then send the same payload back to the sender.

Versions of OpenSSL subject to the Heartbleed bug allocate a memory buffer for the message to be returned based on the length ﬁeld in the request message, regardless of the actual payload size of that message. Because of this failure to check the appropri- ate limits, the returned message consists of the payload, possibly followed by whatever else is allocated in the memory buffer (“Heartbleed,” Behavior section, 2020).

The problem with implementing the TLS heartbeat feature in OpenSSL was that the program does not check how long the received payload is. The attacker can write arbi- trary values in the payload\_length ﬁeld provided for this purpose in the header of the payload packet and thus read the memory of the remote peer.

Although this Heartbleed attack works in both directions, let us assume that a client is attacking a server.

Practical Aspects of Cryptology

1. The attacker sends the server a heartbeat payload that has 1 byte, but claims that it has, e.g., 16 kilobytes.
2. The server writes the attacker’s byte into its memory in a buffer called pl. Since the actual size of the payload is not compared, the server assumes the size speciﬁed by the attacker (payload) when the payload is returned.
3. The server reserves 16 kilobytes of memory (and a little more for administrative information): buffer = OPENSSL\_malloc(1 + 2 + payload + padding) (bp is the buffer.)
4. The server then copies the payload size given by the attacker (about 16 Kilobytes) at this place for the response: memcpy(bp, pl, payload)k

However, the source pl to be copied only has a single byte from the incoming heart- beat. The following bytes consist of any other data that the server is currently process- ing, such as passwords, data of another user that was just decrypted, or secret keys for the server. The server then sends the data packet (bp) to the client, who can repeat this attack at will.

The tapping leaves hardly a trace on an attacked computer. It is therefore not certain to what extent the error has been exploited. Although it was reported that the NSA used the Heartbleed bug from the beginning, this was immediately denied by the NSA direc- tor: “This administration takes seriously its responsibility to help maintain an open, interoperable, secure and reliable internet […] It is in the national interest to responsi- bly disclose the vulnerability rather than to hold it for an investigative or intelligence purpose” (Hosenball & Dunham, 2014, para. 5 and 11).

Summary

Many things can go wrong when implementing cryptography. As a software devel- oper, it is recommended to use libraries that are already available and vetted. Open source software libraries that implement cryptographic functions such as encryp- tion, decryption, signing, and veriﬁcation are waiting to develop an original crypto- graphic algorithm.

Even though most has already been implemented by cryptography professionals, still great care has to be taken in choosing the right implementation. For example, the implementation of random number generators is notorious for exploits and audits a system. In ECC, if the same ephemeral key, usually randomly generated, for signing is used twice to sign different documents, then the ephemeral becomes known and reveals the key.

Even when reusing well-known proven cryptographic open source libraries such as OpenSSL, bugs might creep in, such as the Heartbleed bug, that made common web servers reveal on request currently processed secret data. These bugs, once known, are quickly ﬁxed. The question remains whether intelligence agencies will exploit these.

Furthermore, governments make laws that oblige the use of encryption, but govern- ment agencies might also work against it, e.g., by demanding back doors be built into cryptographic software or devices.

Finally, even with best cryptographic practices, Moore’s law predicts that continuous technological progress doubles the computing power about every year and thus weakens the future security of keys. Therefore, a security margin for long-term security, approximately twenty years, has to be added. There may also be techno- logical leaps, such as the looming construction of a quantum computer, which would break many common asymmetric ciphers such as Difﬁe-Hellman, RSA, and ECC, as well as more involved alternatives.



# Unit 8

## Applications

#### STUDY GOALS

On completion of this unit, you will have learned …

… how cryptography is used in online banking and voting.

… about authentication and access control via a one-time password.

… the basic anatomy of the blockchain that stores and secures the transactions of cryptocurrencies.

… how steganography embeds a secret message into a public message and could be statistically detected.

… about the Tor project guarantees anonymity on the internet.

DL-E-DLMCSEAITSC02-U08

1. Applications

### Introduction

The development driver of cryptology has long been its application. Although previ- ously it involved eavesdropping on war time communication, today this is almost all business transactions. Its main applications, facilitating secure communication and establishing secure communication channels, can be found in current applications, e.g., secure emails via S/MIME, establishing secure connections over insecure channels, e.g., the internet via SSL/TLS and HTTPS, or setting up virtual private networks via IPSec. But many other applications are quite common, including

* data hiding and secret sharing,
* password protection,
* identiﬁcation,
* biometric security,
* digital signatures,
* establishing and sharing of public-keys, or
* zero-knowledge techniques.

Further prominent and more demanding examples of cryptology can be found in elec- tronic commerce (e-commerce), mobile commerce (m-commerce), and electronic vot- ing. Secure communication over insecure channels for buying and selling, online bro- kerage, money transactions, and voting has to be established. Thus, banking applications, mapping all the workﬂows into online transactions, have become one of the main drivers in the further development of cryptology.

Blockchain technology, one of the most innovative technology currently in develop- ment, is also one of the driving forces. Blockchain, a synonym for new, alternative, and disruptive business models, will be the game changer for technologies from the ﬁnan- cial to the industrial sector. A distributed ledger of information that increases transpar- ency and efﬁciency, while reducing complexity and costs, sounds quite promising. Blockchain leverages PKIs to offer trust in form of conﬁdentiality, privacy, and security for its managed information ﬂows. With the advent of quantum computing, which could decode current cryptography systems, enterprises from all sectors have to enhance their systems to quantum-resistant cryptography in order to keep their data and pro- cesses safe.

### Online Banking

The term online banking covers the services offered by banks, which consist of ena- bling access to an account to perform passive operations (e.g., balance and recent ﬁnancial transaction checks), but also active operations (e.g., carrying out transactions or making transfer orders). Online banking is offered through an user interface

Applications

accessed by a Web browser, mobile phone app (mobile banking), or telephone (tele- phone banking). The access through the interface offered on ATMs and terminals, or a computer, are assigned to the more general term electronic banking.

###### Requirements

Like any web application running on a web browser or just through an insecure chan- nel such as the internet, the users of online banking services can be exposed to iden- tity theft caused by, e.g., a key logger technique or phishing. Basic skills in using web and mail in a secure manner are expected.

Because of these threats, the general requirements on information security have to be fulﬁlled by banks as the institution offering the services. Online banking has to guaran- tee

* conﬁdentiality, with encryption of data so that ciphertexts can’t be read without the key and knowledge of the correct cipher used;
* integrity, by avoiding unauthorized changes to the content trough adequate identiﬁ- cation and authentication processes; and, of course,
* availability, by a redundant and secure server infrastructure.

Contracts can be signed with an electronic signature completely online; documents classiﬁed as conﬁdential, e.g., identity documents, account information, invoices, and deposits, can be shared secure in online banking instead of through the mail.

###### Security

Online banking offers bank information over the internet as an insecure channel. The security and privacy of exchanged information is a major concern of online banking. Nevertheless, the system is an obvious target for attacks related to customer or user authentication. Online banking reduces banking transaction costs and increases bene- ﬁts to customers by offering integrated services, where many information and transac- tion services are available through the internet. The utmost security is strived for to avoid possible fraudulent transactions of any kind. Data encryption in information and transaction is the ultimate medium to guarantee security and privacy in online banking through passwords, biometric, PINs, digital signatures, steganography, and so on. Banks have to invest more in data and information protection and security due to the contin- uous surge in usage of the digital channels and associated threats. Cryptography has become the major hammer to protect all transactions. Its main tasks are

* making messages unintelligible or unreadable to non-legitimated users,
* authenticating the source of messages,
* ensuring that a message has not been altered during processing,
* cross checking of transaction on data,
* avoiding data manipulation, and
* securing transactions.

Online banking makes use of various techniques of cryptosystems, offering encrypting and decrypting of data, and hiding information with keys used in cryptographic algo- rithms. Cryptosystems are the ultimate weapon in avoiding violent security attacks on communication channels. It is the constant work of improving the security of these sys- tems that builds trust between the customer and their bank. Beneath a simple authen- tication by password, ID, and PIN, strong authentication methods such as tokens, user certiﬁcates and private keys are used. In addition to the encrypted data transmission by the SSL protocol, methods for a digital signature are offered.

###### Authentication

In addition to the encryption of conﬁdential content, cryptography is also used to verify a user’s identity and to guarantee data integrity by preventing data from unauthorized changes. For online communication, it is essential that there is trust between the both communication partners. For this purpose, a digital signature can be used. This signa- ture data is associated or logically connected to the owner and guarantees its authen- ticity.

OTP (one-time pass-

word) This password is used once to prove identity and is only valid for one login or one transaction.

Authentication or access control in online or mobile banking can be done with an OTP. An OTP is sent in an SMS to a previously registered mobile number. This OTP can be used for banking transactions or login. Additionally, the PIN of the mobile device must be known to read the OTP and further transactions are only possible after the authenti- cation to unlock the mobile device. This offers an end-to-end-encryption for the OTP SMS. This provides two levels of authentication because both the PIN and the OTP must be correct to proceed. The generated OTP is usually encrypted with the symmetric AES with key lengths of usually 128, 192, or 256 bits. This has the drawback of a large system load for encryption and decryption but offers a higher security level compared to a static, memorized password (Rao et al., 2011).

Transactions carried out over the internet are quite common but exposed to threats, which can be handled by a layered security concept. The OTP sent as plaintext SMS, although secured by the symmetric AES, is also vulnerable to attacks among the mobile communication channel. Of course, OTP used in a layered security concept is safer than standalone OTP. Critical transactions require a two-factor authentication routine, e.g., a combination of the user ID and an encrypted OTP. For security purposes a digital signa- ture could be used, generated by an OTP over SMS or a hardware token. Beyond a two- factor authentication routine there is still room left for an advanced security by using a three-layer system with a combination of authentication by username and PIN code, mobile security with an OTP, and biometric security using ﬁngerprint or iris recognition.

An alternative two-factor authentication scheme is the embedded crypto-biometric authentication, a combination of cryptography and a biometric technique. Used espe- cially for person authentication, the biometric image, e.g., ﬁngerprint or iris scan, is immediately encrypted with a symmetric key and, using a 3D chaotic map, sent to a server. The encryption keys are extracted from the random pixel distribution in the bio- metric raw image of some stable biometric features and a pseudo number generator. This is done with multiple iterations and different keys used per iteration.

Applications

###### FinTS / HBCI

The Home Banking Computer Interface (HBCI) is an open standard for banking transac- tions over the internet. It deﬁnes transmission protocols, message formats, and secur- ity procedures to be used for home or mobile banking. HBCI is a European standard, correlated to the US standards IFX (Interactive Financial Exchange) and OFX (Open Financial Exchange) (Kubicek & Diederich, 2015). The main attributes of HBCI are that it

* is independent of the operation system,
* supports accounts on multiple banking companies,
* uses DES and RSA encryption and signatures, and
* stores the key on a chip card.

The FinTS (Financial Transaction Services) speciﬁcation succeeds the speciﬁcation for the HBCI 3.0 standard (*Deutsche Kreditwirtschaft*, 2020). First published as version 3.0 in 2002 FinTS supports online banking with SWIFT, protocols, and standards for interna- tional payment systems between thousands of ﬁnancial member institutions in 194 countries, among them the central banks of most countries.

Version 4.0 from 2004 included some updates. First, all data are encoded into the uni- versal XML (Extensible Markup Language) format. Second, data are exchanged by the HTTP, HTTPS (the synchronous case of a permanent connection between client and bank), and SMTP (the asynchronous case of an unstable connection between client and bank) protocols to facilitate integration with other payment systems (and ensure com- munication with clients behind a ﬁrewall). The underlying transport protocols are chosen according to the application protocol. While unencrypted protocols (such as the PIN/TAN method) must use an encrypted HTTPS protocol, encrypted protocols (such as RAH-7 or RAH-9) may use the unencrypted HTTP protocol.

In 2014, version 4.1 was published containing improvements improved from years of practical experience. It adopted SEPA (Single Euro Payments Area), a self-regulatory ini- tiative from the European banking sector represented in the European Payments Coun- cil to set (technical) standards that standardize the payment transactions in the Euro- pean Union (BIC, IBAN, etc.). In 2015, the European Union passed the revised legal framework Payment Services Directive (PSD2) that all payment service providers of the member states had to respect until 2018 to make online payments more secure, and in particular to protect the customers’ rights.

PSD2 was supplemented in 2017 by technical regulatory standards (2018/389) for (two- factor) client authentication.

###### Roles

FinTS distinguishes between the following roles (*Deutsche Kreditwirtschaft*, 2020).

SWIFT

Founded in 1977, the Society for World- wide Interbank Financial Transac- tions, or SWIFT, is owned by the mem- ber banks.

* + The messenger signs and transmits the message and does not need to know about its (possibly encrypted) contents.
  + The issuer signs the order part of a user message and possibly encrypts (and can be the messenger as well).
  + The witness signs the order part of a user message in addition to the issuer if the issuer’s signature alone is insufﬁciently authorative.
  + The intermediary mediates between the customer and bank as a technical interface endowed with varying authorizations towards the bank (e.g., those of the issuer or of the messenger). Communication between the intermediary and the credit institution is always encrypted.

###### Encryption and Signing

Transactions are encrypted either on the FinTS protocol layer using a key stored on a smart card (banking signature cards using the SECCOS operating system of the *Deut- sche Kreditwirtschaft*) or in a ﬁle secured by a password. Alternatively, they are encryp- ted on the TSL protocol (underlying the HTTPS-Protocol) using PIN/TAN with indexed (iTAN) and mobile (mTAN) transaction numbers, i.e., one-time passwords. *N.B.* iTAN was abandoned in 2019 by the EU payment regulations directive PSD2.

PIN (Personal Identi- ﬁcation Number)

The PIN is a personal

code used for authorization in mobile phones, online banking, access panels, and card payments.

TAN (Transaction Authentication Num-

ber) The TAN is a type of one-time-password used in online bank- ing to authorize ﬁnancial transac-

tions.

Using a smart card is the most secure option because all cryptographic operations are achieved without the secret key ever leaving the smart card securely stored on it. The PIN is entered by the card reader keyboard. If a ﬁle is used, then the key must be encrypted by a password chosen by the user and only accessible after manual entry. The PIN/TAN method is more convenient, e.g., while travelling, because it does not require a card reader. Using PIN / TAN access authorization is done with a PIN of around four digits, while transaction authorization requires an additional TAN. TAN pro- vides increased security by acting as a kind of two-factor-authentication. If an attacker obtains a document or device containing a TAN, this is unusable without knowing the credentials: the PIN. However, if they obtain the login details with the PIN, they cannot make any transaction without a valid TAN (Kubicek & Diederich, 2015).

The HBCI standard covers the entire client and bank procedure. Operations are carried out using a special chip card with a 128-bit key. For this, a special software and a chip card reader are required. For the PIN / TAN method, a bank-issued identiﬁcation num- ber and transaction number are required. Each TAN is only valid for one transaction, which makes this method secure. First, the bank creates a list of unique TANs for a user. The user needs the username and password to log in. This allows them to access the account information, but it is not sufﬁcient for making transactions. The user has to conﬁrm transactions by entering an unused TAN from their list. The bank veriﬁes whether the TAN belongs to the list that has been created for the user and whether it has not already been used. If so, the bank allows the transaction, otherwise it will be rejected. If the TAN list is revealed or stolen, the user can invalidate it by notifying the bank. This system is vulnerable to phishing attacks when an attacker obtains the pass- word and some TANs from the list. It also does not provide protection against man-in- the-middle (MITM) attacks. For example, an attacker intercepts the TAN during trans- mission and can use it for a fraudulent transaction later.

Applications

An iTAN (Indexed TAN) reduces the risk of phishing. When authorizing a transaction, the user must enter a predeﬁned TAN from the list. The TANs position in the list is ran- domly selected by the bank, which makes a randomly obtained TAN practically worth- less for an attacker. While iTANs are still vulnerable to MITM attacks, mTAN (Mobile TAN) could help. When using mTAN, the bank generates a TAN and sends it to the logged in user as an SMS. This SMS usually also contains transaction data, which allows the client to check that the transaction data was not changed when sent to the bank. The security of this method depends on the security of the mobile network used for sending the SMS.

### Blockchain

The blockchain is a chain of blocks. There is an initial block, the Genesis block, and blocks that point to their predecessors. The “arrow” pointing to the predecessor is a hash, an identiﬁcation of the entire contents of the previous block. It is an address that appears in the block header. The trunk of the block consists of an average of 2000 transactions between Bitcoin users (Wang et al., 2018).

###### Hashes

Since the hash of the block depends on the entire contents of the block, the simple change of a bit changes its hash, i.e., invalidates the arrow of the successor. Thus, for the blocks to continue to form a chain, we need to change this arrow. Thus, the succes- sor hash changes. For blocks to continue to form a chain, it is necessary to change the arrow of the successor, and so on, creating a chain reaction. If we change some detail, e.g., a transaction in the ﬁrst block, all the arrows (hashes) that follow must be recalcu- lated.

###### Mining

Mining makes this change very difﬁcult because only blocks whose hashes are small, i.e., start with many zeros, are accepted by the network (i.e., by the nodes that check). It is not enough to change a transaction and calculate the new hashes of all successors. It is necessary to make this change such that all hashes are small! It is highly difﬁcult to ﬁnd such a change. Currently, the search for such a block requires a billion years on a regular computer, but only ten minutes on the mining network. While the perpetrator started searching for acceptable blocks, the network has already created several others, invalidating this work.

###### Transactions

All existing bitcoins were generated by mining, i.e., they were given (by the coinbase transaction) as a reward to the one who created a block (with a small hash). All other transactions have an input and output. On the entry a sufﬁcient amount to pay what will be spent is added. You have to spend it all! Usually a transaction includes the sender as the recipient, the change. What you don’t expect will win the minor will include this transaction in your package as a prize (currently about $40). The recipient is designated by their public key, and only at the time they spend the coins do they need to demonstrate their possession of the corresponding private key, with their sig- nature of the transaction.

###### Elliptic Curves

Bitcoin uses encryption by ﬁnite elliptic curves to sign transactions. Difﬁe-Hellman uses ﬁnite rings of number such as {0, 1, …, m} (e.g., m = 12 for the clock, and a prime number such as p = 2255 – 19 with 100 digits in Bitcoin). The concept used by Bitcoin resembles that of Difﬁe-Hellman, only instead numbers 0, 1, 2 … uses pairs of numbers (x, y), i.e., points of a curve called elliptic. The beauty of this curve is that one can add points on it. p + q + r = 0 if these three points p, r, and q lie on the same line. Instead of multiplying the same number several times, we add the same point. It is easy to add points with this formula on these ﬁnite rings, but it is very difﬁcult to know how many times a point was added to itself to get the resulting point. The ciphering corresponds to the addition, the decryption to the knowledge of this number of times.

Finally, the signature scheme is a variation of ElGamal’s signature. The signature shows that the owner of the private key was able to solve a difﬁcult equation; it is practically impossible to solve it without this private key that provides a shortcut (Puthal et al., 2018).

### Voting

For a vote, like any sensitive transaction, the voter must be authenticated, and their vote must be not only with integrity, but also kept conﬁdential. Consequently, the ballot must be anonymous as well, i.e., the receiver cannot know the sender, and there is no link between the voter and their ballot. The secrecy of the ballot guarantees that only the voters themselves, and no one else, knows of their choice.

###### Anonymity versus Integrity

Integrity is about

Applications

* whether a vote is cast as intended (which is obvious in a paper ballot) and
* whether a vote is registered as cast and whether all the votes are tallied as regis- tered (which, for paper ballots, relies on trusting the election ofﬁcials).

Integrity can only be achieved at the expense of anonymity, and precautions must be taken to preserve this as much as possible. In an end-to-end auditable (or voter veriﬁ- able) voting system, each voter receives an encrypted ID, which they can use to check on a public list whether their choice was likely cast, registered, and tallied as intended. The more voter checks and errors are committed, the more probable that one of them is detected (Smart, 2012).

However, the integrity check demands a database that assigns each vote of each ballot to a candidate and is owned by the election authorities at some point. While integrity can be likely achieved, to break the secrecy of the ballot, it sufﬁces

* to identify the ballot by its ID and the ﬁlled votes,
* to identify the candidate that corresponds to a vote by accessding the assignment database, and
* to identify the voter of a ballot, e.g., their ﬁngerprint on the ballot and access to a database of ﬁngerprints.

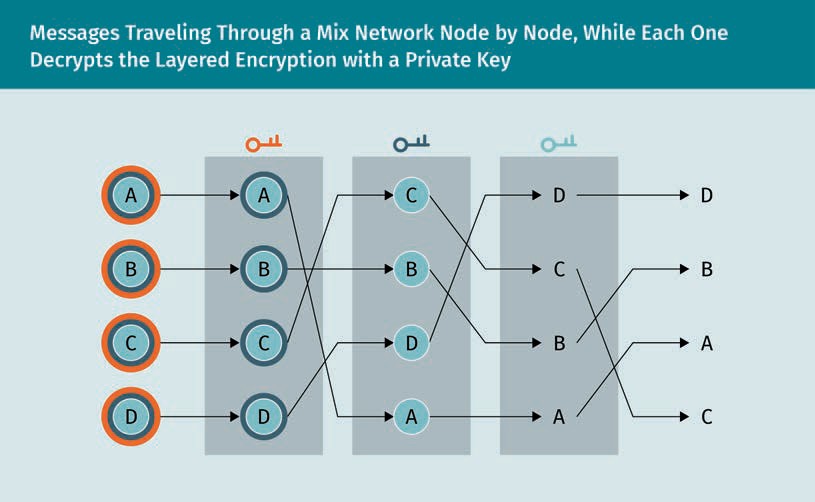
To ensure that this assignment between the votes of the ballots and candidates is left unaltered during the casting of the ballots, the election authorities commit to it before the election by cryptographically hashing the information of each ballot (such as its ID and the vote numbers) and the assignment database. Afterward, these hashes are pub- licized.

###### Integrity versus Traceability

In many countries, the voting process that underlies a democracy should be under- standable by everybody (instead of having to trust the computer), thus ruling out the use of cryptography. For example, in its decision on 3 March 2009, the Federal Constitu- tional Court declared the use of the voting computers in the election to the 16th Ger- man Bundestag (and European Parlament) unconstitutional due to insufﬁcient public traceability. The system used for this was closed source, an audit of the integrity of the source code was not allowed to the interested public. However the essential steps of the voting process and the determination of results must be veriﬁable by the citizen reliably and without special expertise by Article 38 with Article 20 (1) and (2) of the Grundgesetz (Basic Law) which require that all essential steps of the election are sub- ject to public scrutiny (Bundesverfassungsgericht 2020). In another example, Namibia was the ﬁrst country in Africa to use voting machines for 2014 national elections. The use of the machines without a veriﬁcation printout was declared unconstitutional by Namibia’s Supreme Court in February 2020 (Ndeunyema, 2020). Instead, most countries opt for paper ballots that are hand-counted. Therefore, instead of using voting machines to cast, register, and tally votes, voting procedures, such as Scantegrity, are explored that use pen and paper, but cryptographically identify the ballot with the vot- er’s choice to ensure authenticity while preserving anonymity.

###### Mix Networks

Mix networks are intended to preserve the anonymity of a vote. Invented by David Chaum in 1981, the idea of mix network operation is based on changing the order of sent messages in combination with public-key cryptography. The network consists of nodes whose task is to cache, permutate, decrypt, and send incoming packages. All par- ties in the communication (mixer, sender, and receiver) have a public and private key pair. It is assumed that every node knows the other node's public key. A message received in one node is reordered and returned in random order to the next node. Lis- tening to end-to-end-communication is quite hard because of the missing link between source and destination of requests. A node only knows their predecessor node, from which they receive messages, and their successor node, to which they send messages.



Each message travels encrypted from node to node using asymmetric cryptography. The resulting cipher is layered with the message at the innermost layer. Each node decrypts the encrypted message from its own layer to reveal where to send the message next.

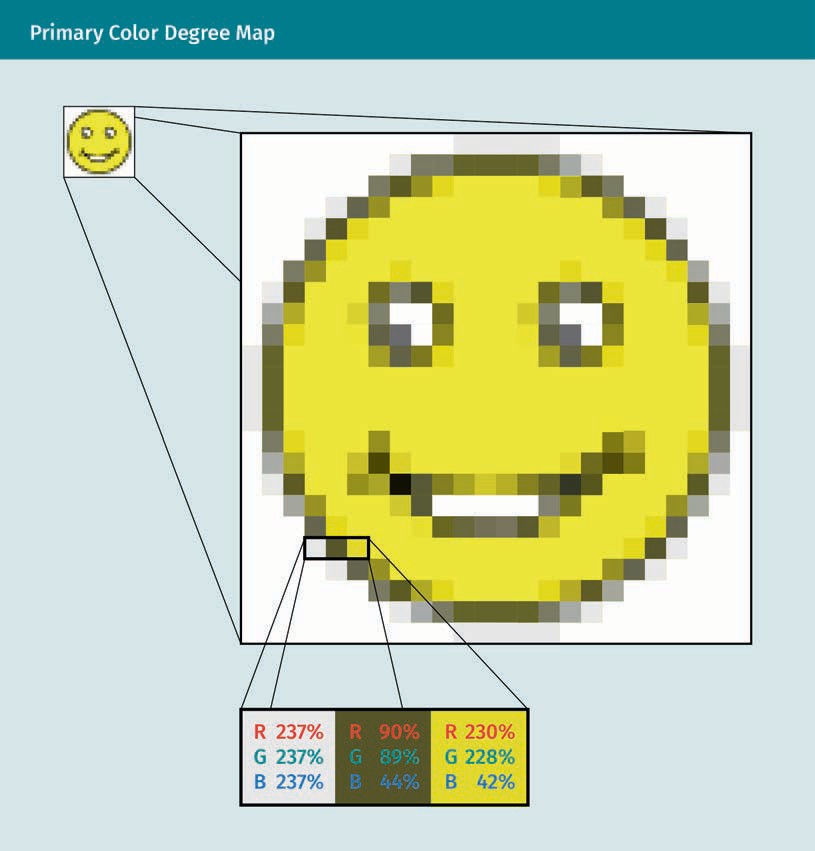
### Steganography

Steganography makes public that presumably important information is deliberately being hidden. This draws attention to this fact and possibly pitts interested cryptana- lysts against it. An example for steganography is given in the history of Herodotus, where Histaeus shaved the head of his most trusted slave and punctured a message on his head which became hidden as soon as his hair had grown back (Watkins, 2001).

Applications

###### Implementation

In computing, the message is typically hidden in a picture, but an audio ﬁle is also pos- sible. This can be used, e.g., to hide in the picture itself a copyright notice of the owner of a picture (Coded Anti-Piracy). Because a typical picture ﬁle has hundreds of kilo- bytes, a few bytes can be changed to convey a secret message without noticeable change. The secret information is usually stored in the marginal parts of the image e.g., in a bitmap.



The hidden message, as a stream of bits, can then be stored, e.g., in the least-signiﬁ- cant-bit of the eight bits for each primary color of each pixel. For each byte, as a num- ber b between 0 and 255, the hidden bit is 1 if and only if b is odd.

Steganography From the Greek for “covered,” it is the art of concealing a message by embed- ding it within

another message, so that nobody but the intended recipient knows of the exis- tence of the mes- sage.

###### Steganalysis

To read the hidden content in a medium, it must be detected and extracted. Steganaly- sis is the art of detecting and extracting the hidden content in a medium without knowledge where the hidden content is nor which algorithm was used.

Hidden content can be detected statistically by looking for signiﬁcant deviations to similar reference data without hidden content. For these deviations to be statistically less signiﬁcant, the hidden content itself should be statistically random and pattern- free. This is achieved through encryption.

For example, the least-signiﬁcant-bit substitution can be detected by a histogram, a diagram of bars, one for each color in the picture whose height is proportional to the number of pixels of that color. Because bytes that were originally odd can only be decremented by one, and those that are even can only be increased by one, the heights of each pair of neighboring bars, i.e., the frequency of pixels of neighboring col- ors, will be signiﬁcantly closer to their mean than in the original picture. The least-sig- niﬁcant-bit substitution can thus be reliably detected when the hidden information makes up less than one percent of the whole picture.

To counter this detection method, least signiﬁcant bit matching adds or subtracts one to the byte randomly. If the bit of the hidden message is “1,” then a coin ﬂip decides whether the byte is incremented or decremented (instead of decrementing the byte if it is odd and decrementing it if it is even). Then, as in least-signiﬁcant-bit substitution, for each byte, as with a number *b* between 0 and 255, the hidden bit is 1 if and only if b is odd. However, the heights of each pair of neighboring bars in the histogram are no longer signiﬁcantly closer to their mean than statistically expected. The problem of detecting Kerckhoff’s principle, that an attacker who knows of the stenographic algo- rithm used to store the hidden information cannot detect it, remains more difﬁcult to achieve in steganography than in cryptography (Watkins, 2001).

###### Application

The command-line tool Tomb by Roio (2020) (aka Jaromil), is a Linux shell script that encrypt folders by dm-crypt, which is part of the kernel. The encrypted folders are called Tombs, whereas dm-crypt calls them containers. These can be created and inte- grated into the running system with a few commands on the command line, which, however, need partial administrative rights. For the highest security, tomb and its key should not be stored on the same device. If the tomb is on a PC or notebook, the tomb could be stored on a USB stick. If, however, the key must be stored on the same device as the tomb, then Tomb can hide the key in a JPEG picture using steganography. This hides the key from unauthorized eyes and helps remember the key’s location (Roio, 2020).

Applications

Tomb can mount folders that other applications need at runtime, e.g., the mailbox of an email client. To this end, it needs the package steghide. The key can be hidden in a small JPEG picture by tomb bury and extracted by tomb exhume.

|  |  |
| --- | --- |
| To hide a key in a picture | tomb bury secret.tomb -k picture.jpg |
| To retrieve a key from a picture | tomb exhume picture.jpg -k secret.key |
| To open a tomb with a steganography key | tomb open secret.tomb -k picture.jpg |

### The Tor Project

Tor is an implementation of second-generation onion routing to guarantee anonymity on the internet. Although it was originally sponsored by the US Naval Research Labora- tory, the Electronic Frontier Foundation (EFF) sponsored it between late 2004 and 2005 (Tor Project, n. d.).

###### Connection to Network

To connect to the network, each client fetches a list of Tor nodes from a server, auto- matically chooses a random path (that may change after a while), and builds a circuit in which each node knows only its predecessor and successor.

The ﬁrst node in the circuit knows the requested IP address. But from the second node on, the negotiation is done through the already built partial circuit, so that the second node, e.g., will only know the IP address of the ﬁrst node (and eventually of the third node). The packets to be routed are identiﬁed by a code (chosen at the time the circuit is built) of the owner of the circuit. Each node of the circuit receives its own private (asymmetric) key encrypted by the public (asymmetric) key dedicated to that node.

Packet Exchange between client and server

Before dispatching a TCP packet to the server, the client encrypts it as many times as there are nodes (a) with the public key corresponding to the last node, numbered n; (b) with the public key of the penultimate node, numbered n – 1; and (c) with the key of n

* 2. The last time, it encrypts with that of the ﬁrst node.

At this point, all layers of the onion enclose the TCP packet. The onion is peeled when the client dispatches it to the circuit they have built. First, the ﬁrst server in the circuit decrypts the packet with key number 1 and sends it to the second server. Next, the sec- ond server decrypts this packet with key number 2 and sends it to the third server. At the end, the last server decrypts this packet with its own private key number n and receives the original plaintext packet.

Onion Routing

In this chain, every node knows only its immediate prede- cessor and succes- sor, and all trafﬁc between both end- points is indecipher- able to every node but the endpoints.

Proxy

A user on the Tor network can set up their web browser to use a personal proxy server to access Tor (such as Privoxy), e.g., to connect to IUBH.de. The steps are

* + 1. Their web browser sends the HTTP request to Privoxy.
    2. Privoxy removes the non-anonymous information and passes the information via SOCKS to the Tor client.
    3. The Tor client builds a circuit (if it hasn’t already done so), encrypts the data to be sent, and passes it to the ﬁrst node.
    4. The ﬁrst node decrypts part of the envelope and forwards the data to the exit node.
    5. This exit node sends the request to IUBH.de.

For the IUBH website to connect to the user, the steps are carried out in inverse order.

Implementations

The most accessible use of the Tor network without advanced computer skills is the Tor Browser. This web browser, available for Linux, Microsoft Windows, and Mac, tunes Mozilla Firefox to leave as few traces as possible on the network and the computer. The effects include

* + browser trafﬁc is by default redirected through the tor instance started at initializa- tion,
  + lack of browsing history,
  + DuckDuckGo as the default search engine, and
  + the NoScript and HTTPS-Everywhere extensions enabled by default.

Tails (The Amnesic Incognito Live System) is an operating system that uses the Tor net- work by default and designed to leave no trace on the computer being used. It is built to run on removable media (such as USB drives) and is based on the Linux distribution Debian.

Summary

The online banking protocol FinTS standardizes data exchange between the cus- tomer and their bank with the XML format, and ensures best cryptographic practi- ces such as storage of the secret keys on a chip card that never leaves it, but is only used to respond to challenges based on knowledge of the key, permanent encryp- tion and authentication of all sensitive data, and the use of proven encryption algorithms.

Blockchain replaces a third trusted party by a database of entries that successively point to each other. This pointer is a hash of the whole prior entry, ensuring the database’s integrity. Thus, a change of the entry entails a change of the hash, inva- lidating the pointer and the whole chain. Because the hashes must have many leading zeros, ﬁnding valid entries, or mining, demands considerable computa- tional power and becomes practically impossible.

Applications

Veriﬁability of the tallying in an election with cryptography places additional requirements on its implementation because the casting of votes must be anony- mous, and the tallying must be comprehensible to the average voter. While cryptog- raphy hides a message by reversibly scrambling it, steganography hides it by embedding in a plaintext message. Statistical analysis gives away the existence of concealed information; however, if done with care, it is rarely discovered. Finally, the Tor network achieves anonymous data transfer on the internet through onion rout- ing, where trafﬁc passes through a chain of nodes in which every node only knows its immediate predecessor and successor, and all trafﬁc between both endpoints is indecipherable to every node but the endpoints.



# Appendix 1

## List of References

List of References

Abdalla, M., Benhamouda, F., & Pointcheval, D. (2016). Public-key encryption indistin- guishable under plaintext-checkable attacks. *IET Information Security, 10*(6), 288—303.

Acdx. (2008). Digital signature diagram. https://commons.wikimedia.org/w/index.php? title=File:Digital\_Signature\_diagram.svg&oldid=362770490

Ajtai, M., & Dwork, C. (1997). A public-key cryptosystem with worst-case/average-case equivalence. In *STOC ’97 (El Paso, TX)* (pp. 284—293). ACM.

Aspect, A., Grangier, P., & Roger, G. (1982). Experimental realization of Einstein-Podolsky- Rosen-Bohm Gedankenexperiment: A new violation of Bell’s inequalities. *Physical Review Letters*, *49*(2), 91—94. https://doi.org/10.1103/PhysRevLett.49.91

Bauer, C. P. (2016). *Secret history: The story of cryptology*. CRC Press.

Bauer, F. L. (2000). *Entzifferte Geheimnisse: Methoden und Maximen der Kryptologie*

[Deciphered secrets: Methods and maxims of cryptology]. Springer.

Bell, J. S. (1964). On the Einstein Podolsky Rosen paradox. *Physics Physique Fizika*, *1*(3), 195—200. https://doi.org/10.1103/PhysicsPhysiqueFizika.1.195

Bellare, M., & Rogaway, P. (2000). Encode-Then-Encipher encryption: How to exploit non- ces or redundancy in plaintexts for efﬁcient cryptography. In T. Okamoto (Ed.), *Advances in Cryptology—ASIACRYPT 2000 (*pp. 317—330). Springer. https://doi.org/ 10.1007/3-540-44448-3\_24

Borst, J., Preneel, B., & Rijmen, V. (2001). Cryptography on smart cards. *Computer Net- works*, *36*(4), 423—435.

Braden, R. (1989). *Requirements for internet hosts—communication layers*. Internet Engi- neering Task Force. http//tools.ietf.org/html/rfc1122

Brassard, G., & Hoyer, P. (1997). An exact quantum polynomial-time algorithm for Simon’s problem. *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*, 12—23.

Brumley, D., & Boneh, D. (2005). Remote timing attacks are practical. *Computer Net- works*, *48*(5), 701—716.

Bundesverfassungsgericht. (2020). BVerfG, Judgment of the Second Senate of 03 March 2009, 2 BvC 3/07. Retrieved July 7, 2020, from https://[www.bundesverfassungsgericht.de/](http://www.bundesverfassungsgericht.de/) SharedDocs/Entscheidungen/EN/2009/03/cs20090303\_2bvc000307en.html

Buonafalce, A. (2014). Alberti cipher disk. https://commons.wikimedia.org/w/index.php? title=File:Alberti\_cipher\_disk.JPG&oldid=121453667

List of References

Chaum, D. L. (1981). Untraceable electronic mail, return addresses, and digital pseudo- nyms. *Communications of the ACM*, *24*(2), 84—90. https://doi.org/10.1145/358549.358563

Chen, L., Takabi, H., & Le-Khac, N.-A. (2019). *Security, privacy, and digital forensics in the cloud*. John Wiley & Sons.

Corbellini, A. (2015a). Elliptic curve cryptography: A gentle introduction. https:// andrea.corbellini.name/2015/05/17/elliptic-curve-cryptography-a-gentle-introduction/

Corbellini, A. (2015b). Implementation of ECDHE in ‘Python‘. Retrieved July 7, 2020, from https://github.com/andreacorbellini/ecc/blob/master/scripts/ecdhe.py

Crypto, M. (2005). DES-f-function. https://commons.wikimedia.org/w/index.php? title=File:DES-f-function.png&oldid=141155336

CrypTool-Projekt. (2008). Skytale3d de. https://commons.wikimedia.org/w/index.php? title=File:Skytale3d\_de.png&oldid=154927861

Daemen, J., & Rijmen, V. (1999). AES proposal: Rijndael. <http://www.cs.miami.edu/home/> burt/learning/Csc688.012/rijndael/rijndael\_doc\_V2.pdf

Daemen, J., & Rijmen, V. (2002). *The design of Rijndael*. Springer. https://doi.org/ 10.1007/978-3-662-04722-4

Datenschutz. (2020). [Data protection]. https://datenschutz.bund.de/

Davies, D. (1997). A brief history of cryptography. *Information Security Technical Report*, *2*(2), 14—17. https://doi.org/10.1016/S1363-4127(97)81323-4

de Leeuw, K. M. M., & Bergstra, J. (2007). *The History of information security: A compre- hensive handbook*. Elsevier.

Deutsche Kreditwirtschaft. (2020). FinTS. Retrieved July 7, 2020, from https://www.hbci- zka.de/

Di, T., Hillery, M., & Zubairy, M. S. (2004). Cavity QED-based quantum walk. *Physical Review A*, *70*(3), 032304. https://doi.org/10.1103/PhysRevA.70.032304

IBM. (n. d.). Authentication. In *Dictionary of IBM & Computing Terminology*. Retrieved July 7, 2020 from https://[www.ibm.com/ibm/history/documents/pdf/glossary.pdf?cv=1](http://www.ibm.com/ibm/history/documents/pdf/glossary.pdf?cv=1)

Difﬁe, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644—654.

DNSCrypt. (n. d.). Ofﬁcial website. https://dnscrypt.info/

Dooley, J. F. (2013). *A brief history of cryptology and cryptographic algorithms.* Springer International.

Driscoll, M. (2019). *Illustrated TLS connection 1.3 with each download explained*. https// tls13.ulfheim.net/

Ducklin, P. (2013). *Android random number ﬂaw implicated in bitcoin thefts*. Sophos Ltd. https.//nakedsecurity.sophos.com/2013/08/12/android-random-number-ﬂaw-implica- ted-in-bitcoin-thefts/

Dukhovni, V. (2014). *Opportunistic security: Some protection most of the time*. Internet Engineering Task Force. http//tools.ietf.org/html/rfc7435

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. In *Advances in Cryptology (Santa Barbara, Calif., 1984)* (Vol. 196, pp. 10—18). Springer. https://doi.org/10.1007/3-540-39568-7\_2

FIDO2. (2020). https://ﬁdoalliance.org/ﬁdo2/ﬁdo2-web-authentication-webauthn/

Forma Estudio. (2007). Rijndael animation v.4. Retrieved July 7, 2020 from https:// formaestudio.com/portfolio/aes-animation/

Friedl, S. J. (2005). An illustrated guide to IPsec. Retrieved July 7, 2020, from http// [www.unixwiz.net/techtips/iguide-ipsec.html](http://www.unixwiz.net/techtips/iguide-ipsec.html)

Goldreich, O., Goldwasser, S., & Halevi, S. (1997). Public-key cryptosystems from lattice reduction problems. In B. S. Kaliski (Ed.), *Advances in Cryptology—CRYPTO ’97* (pp. 112— 131). Springer.

Goldwasser, S., Micali, S., & Rackoff, C. (1985). The knowledge complexity of interactive proof systems. *STOC85: Annual ACM Conference on Theory of Computing*, 291—304.

Goldwasser, S., Micali, S., & Tong, P. (1982). Why and how to establish a private code on a public network. *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, 134—44.

Gordon, D. M. (1993). Discrete logarithms in gf(p) using the number ﬁeld sieve. *SIAM Journal on Discrete Mathematics*, *6*(1), 124—138. https://doi.org/10.1137/0406010

Grau, S. (2018a). Elliptic curve plotter. <http://www.graui.de/code/elliptic2/> Grau, S. (2018b). Modular function plotter. <http://graui.de/code/ffplot/>

Gringer. (2011, October 26). Rgb-raster-image. https://commons.wikimedia.org/wiki/ File:Rgb-raster-image.svg

Heartbleed. (2020). In *Wikipedia*. https://en.wikipedia.org/wiki/Heartbleed

Heer, T. (2008). Introduction to network security. In A. Gurtov (Ed.), *Host Identity Protocol (HIP)* (pp. 11—42). Wiley & Sons. https://doi.org/10.1002/9780470772898.ch2

List of References

Heys, H. M. (2002). A tutorial on linear and differential cryptanalysis. *Cryptologia*, *26*(3), 189—221.

Hosenball, M., & Dunham, W. (2014, April 12). *White House, spy agencies deny NSA explo- ted ‘Heartbleed’ bug.* Reuters. https://[www.reuters.com/article/us-cybersecurity-inter-](http://www.reuters.com/article/us-cybersecurity-inter-) net-bug-nsa/white-house-spy-agencies-deny-nsa-exploited-heartbleed-bug-idUS- BREA3A1XD20140411

Hounsel, A., Borgolte, K., Schmitt, P., Holland, J., & Feamster, N. (2020). Comparing the effects of DNS, DoT, and DoH on web performance. *Proceedings of The Web Conference 2020 (WWW ’20)*. Association for Computing Machinery, 562—572. https://doi.org/ 10.1145/3366423.3380139

IANA. (2020). Service Names Port Numbers. https://[www.iana.org/assignments/service-](http://www.iana.org/assignments/service-) names-port-numbers/service-names-port-numbers.xhtml

ISO. (2005, June). ISO/IEC 17799:2005: Information technology—Security techniques— Code of practice for information security management. Retrieved June 6, 2020, from https://[www.iso.org/cms/render/live/en/sites/isoorg/contents/data/](http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/) standard/03/96/39612.html

Jagetiya, A., & Krishna, C. R. (2020). Evolution of information security algorithms. In D. Goyal., S. Balamurugan, S.-L. Peng, & O. P. Verma (Eds.), *Design and Analysis of Security Protocol for Communication (*pp. 29—77). Wiley Online Library.

Kku. (2019). Web of trust-en. https://commons.wikimedia.org/w/index.php? title=File:Web\_of\_Trust-en.svg&oldid=375289053

Kochis, P. C. (1996). Timing attacks on implementations of Difﬁe-Hellman, RSA, DSS, and other systems. *Annual International Cryptology Conference*, 104—113.

Krawczyk, H., Bellare, M., & Canetti, R. (1997). *RFC2104: HMAC: Keyed-hashing for message authentication*. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/ rfc2104

Kubicek, H., & Diederich, G. (2015). Sicherungsverfahren im Online-Banking im Vergleich [Comparison of security procedures in online banking]. In H. Kubicek & G. Diederich (Eds.), *Sicherheit im Online-Banking: PIN/TAN und HBCI im magischen Dreieck aus Sicherheit, Kosten und einfacher Bedienbarkeit* [Security in online banking: PIN/TAN and HBCI in the magical triangle of security, costs and ease of use] (pp. 7—19). Springer Fachmedien. https://doi.org/10.1007/978-3-658-09960-2\_2

Kumar, H., Kumar, S., Joseph, R., Kumar, D., Singh, S. K. S., Kumar, P., & Kumar, H. (2013). Rainbow table to crack password using MD5 hashing algorithm. *2013 IEEE Conference on Information & Communication Technologies*, 433—439.

Lamport, L. (1978). The implementation of reliable distributed multiprocess systems.

*Computer Networks*, *2*(2), 95—114. https://doi.org/10.1016/0376-5075(78)90045-4

L’Ecuyer, P., & Simard, R. (2007). TestU01: A C library for empirical testing of random number generators. *ACM Transactions on Mathematical Software*, *33*(4), 1—40.

Lee, H. K., Malkin, T., & Nahum, E. (2007). Cryptographic strength of ssl/tls servers: Cur- rent and recent practices. *Proceedings of the 7th ACM SIGCOMM Conference on internet Measurement*, 83—92. https://doi.org/10.1145/1298306.1298318

Lenstra, A. K. (2006). Key lengths. In H. Bidgoli (Ed.), *Handbook Of Information Security, Volume 1: Key Concepts, Infrastructure, Standards And Protocols*. Wiley.

Lenstra, A., & Verheul, E. (2001). Selecting cryptographic key sizes. *Journal of Cryptology*, *14*, 255—293

List of random number generators. (2020, July 2). In *Wikipedia*. https://en.wikipedia.org/ wiki/List\_of\_random\_number\_generators

Manualzz. (2020). Security articles from Wikipedia. https://manualzz.com/doc/ 43781390/security-articles-from-wikipedia

McEliece, R. J. (1978). A public key cryptosystem based on algebraic coding theory. *DSN Progress Report*, 42—44.

Melnikov, A. (2011). *RFC6331: Moving digest-md5 to historic*. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc6331

Menon-Sen, A., Williams, N., Melnikov, A., & Newman, C. (2010). *Salted challenge response authentication mechanism (SCRAM) SASL and GSS-API mechanisms*. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/rfc5802

Mockapetris, P. (1987). *Domain names—implementation and speciﬁcation*. Internet Engi- neering Task Force. http//tools.ietf.org/html/rfc1035

Neuman, B. C., & Ts’o, T. (1994). Kerberos: An authentication service for computer net- works. *IEEE Communications Magazine*, *32*(9), 33—38.

Ndeunyema, N. (2020, February 17). *Vote, but you cannot verify: The Namibian supreme court’s presidential election decision*. Oxford Human Rights Hub. https:// ohrh.law.ox.ac.uk/vote-but-you-cannot-verify-the-namibian-supreme-courts-presiden- tial-election-decision/

National Institute for Standards and Technology [NIST]. (2000). *AES competition*. https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines/archived-

crypto-projects/aes-development

Pixis. (2019). Kerberos en active directory [Kerberos in action directory]. https:// beta.hackndo.com/kerberos/

List of References

Primepq. (2008, December 13). Red de mezcla. https://commons.wikimedia.org/wiki/ File:Red\_de\_mezcla.png

Project Gutenburg & Webster's Unabridged Dictionary (1913) [OPTED]. (n. d.). *The online plaintext English dictionary.*

Puthal, D., Malik, N., Mohanty, S. P., Kougianos, E., & Yang, C. (2018). The blockchain as a decentralized security framework [future directions]. *IEEE Consumer Electronics Maga- zine*, *7*(2), 18—21. https://doi.org/10.1109/MCE.2017.2776459

RadioFan. (2010). Enigma rotor wiring. https://commons.wikimedia.org/w/index.php? title=File:Enigma\_rotor\_wiring.png&oldid=260095687

Ramsdell, B. (2009). *Secure/multipurpose internet mail extensions (s/mime) version 3.1*. Internet Engineering Task Force. http//tools.ietf.org/html/rfc3851

Rao, D. S., Kour, G., & Jyoti, D. (2011). One time password security through cryptography for mobile banking. *International Journal of Computer Technology and Applications*, *2*(5), 1563—1567.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120—126.

Roio. (2020). A minimalistic commandline tool to manage encrypted volumes aka The Crypto Undertaker. Retrieved July 1, 2020 from https://github.com/dyne/Tomb

Schell, B., & Martin, C. (2006). *Webster’s new world hacker dictionary*. Wiley.

Schnorr, C.-P. (1991). Factoring integers and computing discrete logarithms via Diophan- tine approximation. In *Advances in cryptology—EUROCRYPT ’91* (Vol. 547, pp. 281—293). Springer. https://doi.org/10.1007/3-540-46416-6\_24

Schoof, R. (1995). Counting points on elliptic curves over ﬁnite ﬁelds. *Journal de Théorie Nombres de Bordeaux, 7*(1), 219—254. [http://jtnb.cedram.org/item?](http://jtnb.cedram.org/item) id=JTNB\_1995 7\_1\_219\_0

Shannon, C. E. (1949). Communication theory of secrecy systems. *Bell System Technical Journal*, *28*(4), 656—715. https://doi.org/10.1002/j.1538-7305.1949.tb00928.x

Shor, P. W. (1994). Algorithms for quantum computation: Discrete logarithms and factor- ing. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 124—134.

Smart, M. J. (2012). *Anonymity vs. traceability: Revocable anonymity in remote electronic voting protocols* [Doctoral dissertation, University of Birmingham]. https://ethe- ses.bham.ac.uk/id/eprint/3386/

Snipes, R. (2019, October 8). Thunderbird, Enigmail and OpenPGP. *The Thunderbird Blog*. https//blog.thunderbird.net/2019/10/thunderbird-enigmail-and-openpgp/

Steane, A. (1998). Quantum computing. *Reports on Progress in Physics*, *61*(2), 117—173. https://doi.org/10.1088/0034-4885/61/2/002

Theveßen, E., Müller, P. F., & Stoll, U. (2020, February 2). *Operation Rubikon* [Video]. ZDF. https://[www.zdf.de/politik/frontal-21/operation-rubikon-100.html](http://www.zdf.de/politik/frontal-21/operation-rubikon-100.html)

Tor Project. (n. d.). https://[www.torproject.org/](http://www.torproject.org/)

Tudhope, G. V. (1993). *The discovery of Francis Bacon’s cipher signatures in James Ander- son’s constitutions of the Free Masons*. Health Research Books.

van Oorschot, P. C., & Wiener, M. J. (1996). Improving implementable meet-in-the-middle attacks by orders of magnitude. In N. Koblitz (Ed.), *Advances in Cryptology—CRYPTO ’96* (pp. 229—236). Springer.

Wang, H., Zheng, Z., Xie, S., Dai, H. N., & Chen, X. (2018). Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, *14*(4), 352. https://doi.org/10.1504/IJWGS.2018.10016848

Watkins, J. (2001). Steganography-messages hidden in bits. *Multimedia Systems Course- work, Department of Electronics and Computer Science.* University of Southampton.

Zeilenga, K. D. (2008). *CRAM-MD5 to historic*. Internet Engineering Task Force (IETF). https://tools.ietf.org/html/draft-ietf-sasl-crammd5-to-historic-00