LEARNING AND EXPERIENCING CRYPTOGRAPHY WITH CRYPTOOL AND SAGEMATH

Bernhard Esslinger

Chapter 1

Learning and Experiencing Cryptography with CrypTool and SageMath

For a listing of recent titles in the *Artech Computer Security Library*,
turn to the back of this book.

Learning and Experiencing Cryptography with CrypTool and SageMath

Bernhard Esslinger

ARTECH HOUSE
BOSTON | LONDON
artechhouse.com

Library of Congress Cataloging-in-Publication Data
A catalog record of this book is available from the U.S. Library of Congress.

British Library Cataloguing in Publication Data
A catalog record for this book is available from the British British Library.

ISBN 978-1-68569-017-5

Cover design by Joi Garron

Accompanying software for this book can be found at:
https://www.cryptool.org/en/documentation/ctbook.

685 Canton Street
Norwood, MA 02062

10 9 8 7 6 5 4 3 2 1

Contents

CHAPTER 1

Ciphers and Attacks Against Them

For centuries, plaintext messages were encrypted by the military, by diplomats, and by alchemists, and much less frequently by businesses and the general population. The goal of cryptography was to protect the privacy between sender and receiver. Since the 1970s, further goals have been added to achieve integrity, authenticity, and non-repudiation, and also to compute on encrypted data in the cloud or to achieve quantum-computer resistance.

The science that deals with encryption is called *cryptology*—divided into the branches of cryptography (designing secure encryption procedures) and cryptanalysis (breaking encryption procedures). In reality, however, these branches are closely interrelated and the terms cryptography and cryptology are often used interchangeably. Therefore, cryptology is currently subdivided into fields like symmetric cryptography, public-key cryptography, hardware and embedded systems cryptography, theoretical cryptology, and real-world crypto [1].

The importance of cryptology continues to grow as our society becomes more and more dependent on information technology. Although cryptology and information security are interdisciplinary fields of research, mathematics now plays the largest role in cryptology. Finally, learning about cryptology can also be fun and entertaining.

The special thing about this book is that you can always try out the procedures right away—by using the links (in the footnotes) to the programs from the CrypTool project, from OpenSSL, or from SageMath. All these programs are open-source.

In this book, the basics are covered in great detail, then from the very extensive field of cryptology certain (current) topics are selected (like RSA, ECC, or lattices). This makes this book accessible to a wide audience, not just only for those interested in the natural sciences.

This chapter introduces the topic in a more descriptive way without using mathematics. To do so, it uses modern methods (RSA, AES) as examples. Then we dive deepen, for example, the property, how many possible keys (key space) different methods have (Section 1.6) and what are the best attacks against known methods (Section 1.7). Recommended books are presented in Section 1.10. In Section 1.11 you will find screenshots of how to use AES in various programs. Classic methods are presented in Chapters 2 and 3.

The purpose of encryption is to change data (plaintext messages) in such a way that only an authorized recipient is able to reconstruct the plaintext. This allows us to transmit encrypted data without worrying about it getting into unauthorized hands. Authorized recipients possess a secret information—called the key—which allows them to decrypt the data while it remains hidden from everyone else. An attacker cannot only try to break a cipher: She still can disturb the connection (e.g., denial-of-service attack) or tap metadata (who is communicating when with whom).

Plaintext is the data processed as input by the encryption method. This data can be text, but also binary data such as an image or an executable file. The encryption method is called a cipher. The output is called ciphertext. With modern ciphers the output is always binary data. Figure 1.1 shows this notation graphically.

1.1 Importance of Cryptology

With the use of the internet and wireless communication, encryption technologies are used (mostly transparently) by everyone. Cryptographic algorithms secure ATMs and the privacy of messengers, allow anonymity for voters, but also help criminals. Cryptography is dual-use, as are many human innovations.

However, cryptography is not only used today, but has been for centuries by governments, the military, and diplomats. The side with a better command of these technologies could exert more influence on politics and war with the help of secret services. This book touches on history only twice: when introducing the earlier cipher methods for didactical reasons in Chapter 2, and in Chapter 3 when explaining the real application of earlier methods. You can gain an understanding of how important cryptology was and still is by considering the following two examples: the BBC documentary film *War of the Letters* [2] and the debates around the so-called crypto wars.

The next two sections discuss the differences between symmetric (see Section 1.2) and asymmetric (see Section 1.3) methods for encryption.

1.2 Symmetric Encryption

For *symmetric* encryption, both the sender and recipient must be in possession of a common (secret) key that they have exchanged before actually starting to communicate (over another channel, out of the band). The sender uses this key

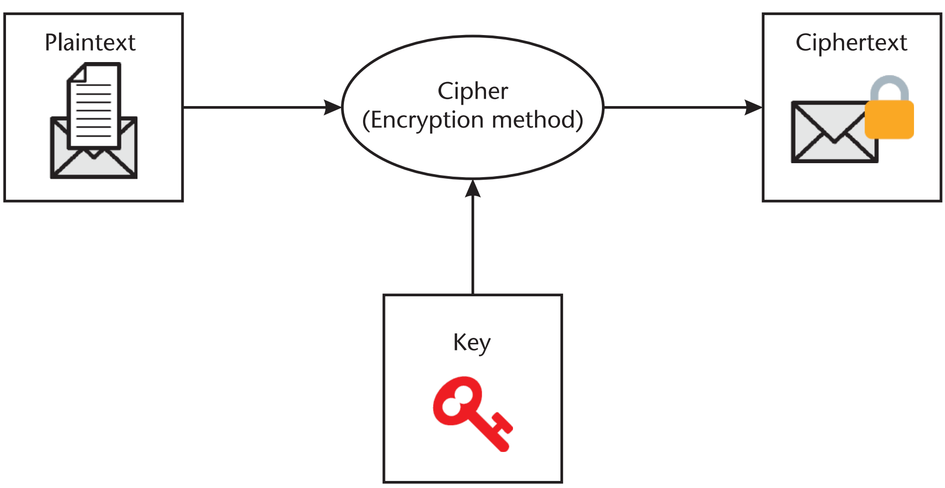

Figure 1.1 Common notations when using ciphers.

to encrypt the message and the recipient uses it to decrypt it. This is shown in Figure 1.2.

All classical ciphers are of the symmetric type. Examples can be found within the CT programs, in Chapter 2 of this book, or in [3]. In this section, however, we want to consider only modern symmetric mechanisms.

The main advantage of symmetric algorithms is the high speed with which data can be encrypted and decrypted. The main disadvantage is the high effort needed for key distribution. In order to communicate with one another confidentially, the sender and recipient must have exchanged a key using a secure channel before actually starting to communicate. Spontaneous communication between individuals who have never met therefore seems virtually impossible. If everyone wants to communicate with everyone else spontaneously at any time in a network of n subscribers, each subscriber must have previously exchanged a key with each of the other $n-1$ subscribers. A total of $n(n-1)/2$ keys must therefore be exchanged.

The current standard for modern symmetric ciphers is the Advanced Encryption Standard (AES).

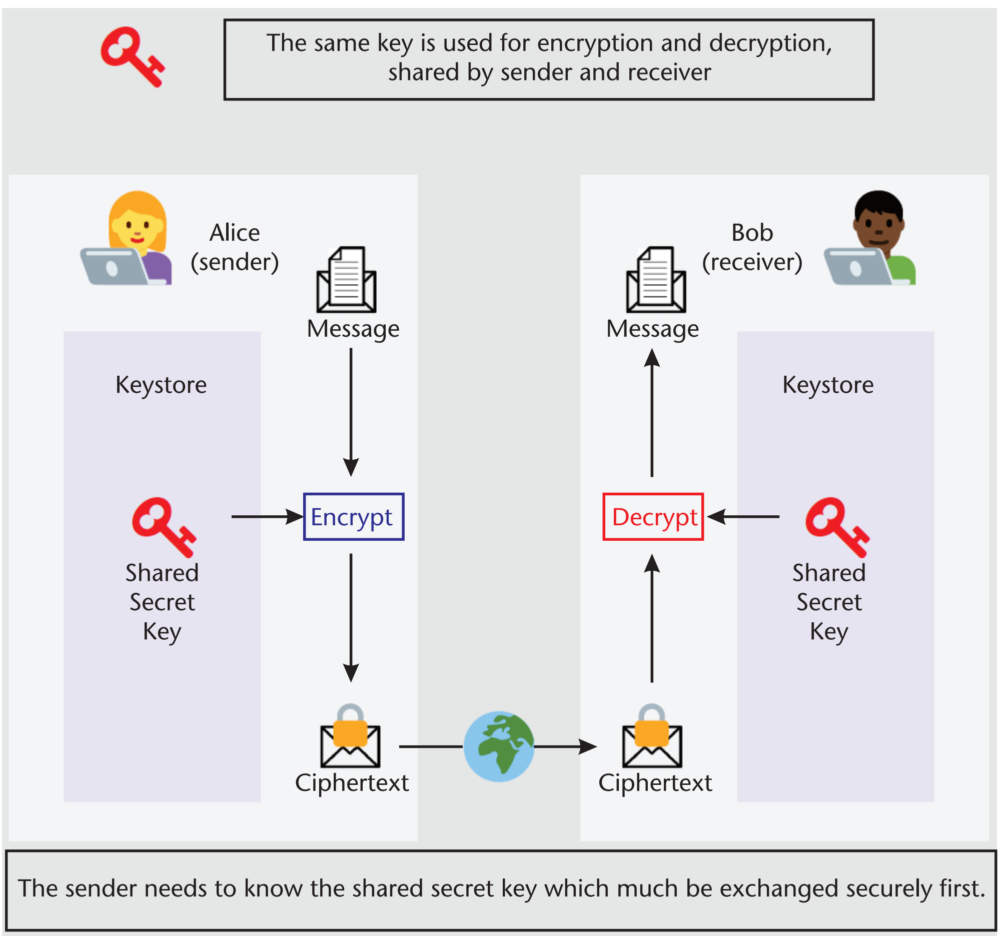

Figure 1.2 Symmetric or secret-key encryption.

1.2.1 AES[1]

Before AES, the most well-known modern symmetric encryption procedure was the Data Encryption Standard (DES). The DES algorithm was developed by IBM in collaboration with the National Security Agency (NSA), and was published as a standard in 1975. Despite the fact that the procedure is relatively old, no effective attack on it has yet been detected (what "effective" exactly means depends on the security definition—see Section 1.8). The most effective way of attacking DES consists of testing (almost) all possible keys until the right one is found (*brute-force attack*). Due to the relatively short key length of effectively 56 bit (64 bits, which however include 8 parity bits),[2] numerous messages encrypted using DES have in the past been broken. Therefore, the procedure cannot be considered secure any longer. Alternatives to the DES procedure include Triple-DES (TDES, 3DES) and especially AES.

The standard among symmetric methods today is AES. The associated Rijndael algorithm was declared the winner of the AES competition on October 2nd, 2000, and thus succeeds the DES procedure. Since then, the AES has been subjected to extensive research and has so far resisted all practical attempts at attack.

Further information about AES can be found in Section 9.2.7. Section 1.11 presents how the AES is animated in CTO, and how the AES is executed in CT2 and with OpenSSL.

1.2.2 Current Status of Brute-Force Attacks on Symmetric Algorithms

The current status of brute-force attacks on symmetric encryption algorithms can be explained with the attack on the block cipher RC5-64. A key length of 64 bit means at most $2^{64} = 18{,}446{,}744{,}073{,}709{,}551{,}616$ or about 18 quintillion (U.S.) ($= 18 \cdot 10^{18}$) keys to check.

Brute-force (exhaustive search, trial-and-error) means to completely examine all keys of the key space, which means no special analysis methods have to be used. The attacker knows only the ciphertext, and so he performs a ciphertext-only attack that requires the weakest knowledge prerequiste of all attacks. Therefore, the ciphertext is decrypted with all possible keys[3] and for each resulting text it is checked to determine whether this is a meaningful plaintext.[4] (See Section 1.6.)

1. - Using CTO in the browser, AES can be seen in two plugins: as "AES Animation" https://www.cryptool.org/en/cto/aes-animation and via "AES (step-by-step)" https://www.cryptool.org/en/cto/aes-step-by-step.
   - Using CT1 Indiv. Procedures ▹ Visualization of Algorithms ▹ AES you can find three visualizations for this cipher.
   - Using the search string `AES` in CT2 Startcenter ▹ Templates you can find a plugin performing AES step by step.
2. As a unit in formulas, we write "bit" in lower case and without the plural "s." See Section B.2.
3. - Using CT1 Analysis ▹ Symmetric Encryption (modern) you can perform brute-force attacks of modern symmetric algorithms.
   - Using CT2 Templates ▹ Cryptanalysis ▹ Modern you also can perform brute-force attacks. The KeySearcher is a highly powerful component used within these templates, which can distribute the calculations to many different computers.
4. If the plaintext is written in a natural language and at least 100 bytes long, this check also can be performed automatically. To achieve a result in an appropriate time with a single PC you should mark only at bits of the key as unknown. On a current PC in 2022, CT1 tries for AES 24 bit in about 20 seconds, but with 32 bit it takes 1:45 h. Compare screenshots in Section 1.6.

Companies like RSA Security provided so-called cipher challenges in order to quantify the security offered by well-known symmetric ciphers such as DES, 3DES, or RC5 [4, 5]. They offered prizes for those who managed to decipher ciphertexts, encrypted with different algorithms and different key lengths, and to unveil the symmetric key (under controlled conditions).[5]

It is well-known that the old standard algorithm DES with a fixed key length of 56 bit is no longer secure: This was already demonstrated in January 1999 by the Electronic Frontier Foundation (EFF). With their specialized computer Deep Crack they cracked a DES-encrypted message within less than a day.

The currently known record for strong symmetric algorithms unveiled a key that was 64-bit long. The algorithm used was RC5, a block cipher with variable key size.

The RC5-64 challenge was solved in July 2002 by the distributed.net team after 5 years [6]. In total 331,252 individuals cooperated over the internet to find the key. More than 15 quintillion ($= 15 \cdot 10^{18}$) keys were checked until the right key was found. This was about 85% of the whole search space.

Therefore, symmetric algorithms using keys of size 64 bit are (even if they have no cryptographic weakness) no longer appropriate to keep sensitive data private.

The BSI requires a security level of 120 bits for modern symmetric ciphers that will be used after 2022 (see [7], page 17f). Not only is AES-128 recommended, but details like suitable block modes and padding methods are also specified.

1.3 Asymmetric Encryption

In the case of *asymmetric* encryption (also called public-key encryption), each participant has their own pair of keys consisting of a *secret* key (called *private* key) and a *public* key. The public key, as its name implies, is made public—for example, within a certificate (see Section 7.5.2) or in a key directory on the internet (this type of billboard is also called a directory or sometimes public-key ring).

Figure 1.3 shows the process of asymmetric encryption and decryption.

If Alice[6] wants to communicate with Bob, she looks for Bob's public key and uses it to encrypt her message (plaintext) for him. She then sends this ciphertext to Bob, who is able to decrypt it again using his private key. As only Bob knows his private key, only he can decrypt messages addressed to him. Even Alice who sends the message cannot restore the plaintext from the (encrypted) message she has sent. In reality, asymmetric methods are not used to encrypt the whole message but only a session key (see Section 1.4). Asymmetric ciphers are designed in a way that the public key cannot be used to derive the private key from it.

Such a procedure can be demonstrated using a series of thief-proof letter boxes. If I have composed a message, I then look for the letter box of the recipient and post

5. Unfortunately, in May 2007 RSA Inc. announced that they will not confirm the correctness of the not-yet-solved RC5-72 challenge. Alternatively, a wide spectrum of both simple and complex, and both symmetric and asymmetric crypto riddles are included in the international cipher contest *MysteryTwister*: https://www.mysterytwister.org.

6. In order to describe cryptographic protocols, participants are often named Alice, Bob, ... (see [8, p. 23]). Alice and Bob perform all 2-person-protocols where Alice will initiate the protocol and Bob answers. The attackers are named Eve (eavesdropper) and Mallory (malicious active attacker).

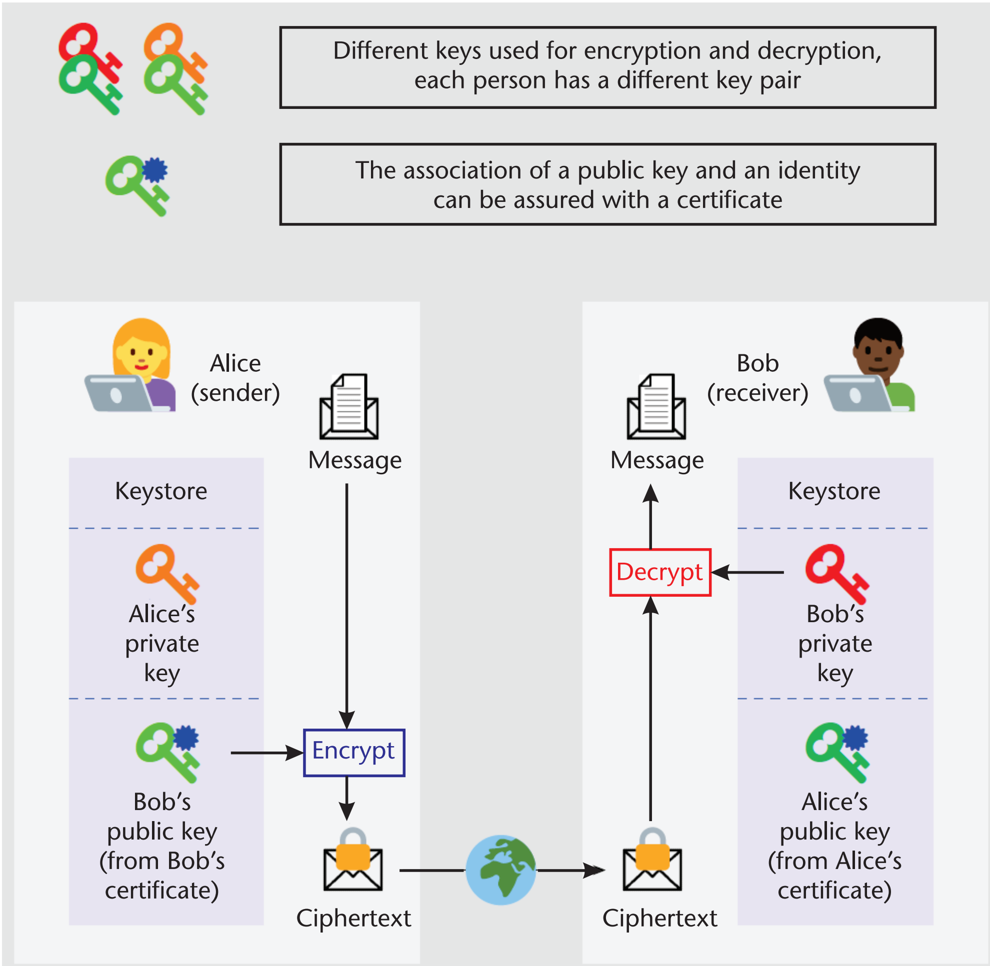

Figure 1.3 Asymmetric or public-key encryption.

the letter through it. After that, I can no longer read or change the message myself, because only the legitimate recipient has the key for the letter box.

The advantage of asymmetric procedures is the easier key management. Let's look again at a network with n subscribers. In order to ensure that each participant can establish an encrypted connection to each participant, each participant must possess a pair of keys. We therefore need $2n$ keys or n pairs of keys. Furthermore, no secure channel is needed before messages are transmitted, because all the information required in order to communicate confidentially can be sent openly. In this case, you simply have to pay attention to the accuracy (integrity and authenticity) of the public key. Nevertheless, the requirements for the key generation are not trivial. What could go wrong is explained, for example, in Section 5.12.5.4. Besides that, nowadays also (public-key) infrastructures themselves are targets of cyberattacks. A disadvantage of pure asymmetric procedures is that they take a lot longer to perform than symmetric ones (see Section 1.4).

The most well-known asymmetric procedure is the RSA algorithm,[7] named after its developers Ronald Rivest, Adi Shamir, and Leonard Adleman. The RSA algorithm was published in 1978. The concept of asymmetric encryption was first introduced by Whitfield Diffie and Martin Hellman in 1976. It is worth noting that the concept was known at the secret services Government Communications Headquarters (GCHQ) and National Security Agency (NSA) several years prior to its independent rediscovery by Diffie and Hellman. Today, the ElGamal procedures also play a decisive role, particularly the Schnorr variant in the Digital Signature Algorithm.

The German Federal Office for Information Security (BSI) requires a security level of 120 bit for processes used beyond 2022. Applied to RSA, the corresponding technical guideline recommends a key length of 3,000 bit (see [7], page 18, comment on Table 1.2).

1.4 Hybrid Procedures[8]

In order to benefit from the advantages of symmetric and asymmetric techniques together, hybrid procedures are usually used (for encryption) in practice.

In this case the bulk data is encrypted using symmetric procedures. The key used for this is a secret session key generated by the sender randomly that is only used for this message. This session key is then encrypted using the asymmetric procedure and transmitted to the recipient together with the message. Recipients can determine the session key using their private keys and then use the session key to decrypt the message.

In this way, we can benefit from the feasible key management of asymmetric procedures (using public/private keys) and we benefit from the efficiency of symmetric procedures to encrypt large quantities of data (using secret keys).

1.5 Kerckhoffs' Principle

In 1883, the Dutch cryptographer Auguste Kerckhoffs formulated six principles for the construction of secure military encryption procedures. The second one, Kerckhoffs' principle or Kerckhoffs' maxim, is now regarded as the principle of modern cryptography. It states that an encryption scheme should be secure even if everything about the scheme is known except the key used. Kerckhoffs' principle is often contrasted with "security through obscurity," in which the encryption algorithm must also be kept secret.

7. The RSA algorithm is extensively described within this book in Section 5.10. The topical research results concerning RSA are described in Section 5.12. In Section 6.5 the RSA algorithm is more deeply reasoned from number theory: The RSA plane is a model to illustrate the processes in this algorithm using pictures of rectangles.

8. - Using CT1 Encrypt/Decrypt ▹ Hybrid you can follow the single steps and its dependencies with concrete numbers. The variant with RSA as the asymmetric algorithm is graphically visualized; the variant with ECC uses the standard dialogs. In both hybrid cases AES is used as the symmetric algorithm.
   - Using JCT Algorithm Perspective ▹ Hybrid Ciphers also offers hybrid methods like ECIES.

Kerckhoffs' principle was reinterpreted several times. For example, Claude Shannon formulated that one should design encryption systems under the assumption that an enemy knows the system exactly from the very beginning (Shannon's maxim).

1.6 Key Spaces: A Theoretical and Practical View

For good encryption procedures used today, the time needed to break an encryption is so long that it is almost impossible to do so. Such procedures are considered (practically) secure—from an algorithm's point of view. After the knowledge gathered by Edward Snowden, there were many discussions debating whether encryption is secure. In [9] is the result of an evaluation, which cryptographic algorithms can be relied on—but only according to current knowledge. The article investigates: Which cryptosystems can—despite the reveal of the NSA/GCHQ attacks—still be considered as secure? Where have systems been intentionally weakened? How can we create a secure cryptographic future? What is the difference between math and implementation?

The key space of a cipher is an important indicator for the security of a cipher. In a monoalphabetic substitution (MASC; also called simple substitution) for instance, using an alphabet of length of k, the key space is $k!$. For AES-128 it is 2^{128}.

A (sufficiently) large key space (approx. 2^{100}) is a necessary prerequisite for a secure cipher, but not a sufficient condition: The MASC has a large key space (with an alphabet of 26 characters approx. $2^{88.4}$ that corresponds to the number of possible ciphertext alphabets), but it has been cracked with frequency analysis for centuries.

The key space is used to calculate the effort required for a brute-force (BF) attack (i.e., for the systematic testing of all possible keys). If the key space is so small that an attacker can carry out a complete BF attack, the procedure is broken—not only theoretically but also practically.

In the case of a BF attack, the attacker decrypts the ciphertext (or parts of it) with every possible key (see Section 1.2.2). Then the found plaintext is evaluated. How surprisingly well fitness algorithms can recognize correct natural texts can be seen in Figures 1.4[9] and 1.5.[10] CT1 uses similar fitness functions as the solvers and analyzers in CT2.

Whether an attacker really has to try the maximal, theoretical key space is questionable, at least with the older ciphers. For this reason, the *practical key space* introduced by Ralph Simpson for historic cipher devices and the *work factor*, which is also known as *attack time*, are considered.

1.6.1 Key Spaces of Historic Cipher Devices

Key spaces of historic cipher devices are often reported in the popular press as a gargantuan number designed to impress the reader about the incredible strength of the encryption. This is often a lead-in to the story of the amazing ingenuity of

9. CT1 Analysis ▷ Symmetric Encryption (modern) ▷ AES (CBC).
10. CT2 Templates ▷ Cryptanalysis ▷ Modern ▷ AES Known-Plaintext Analysis (2).

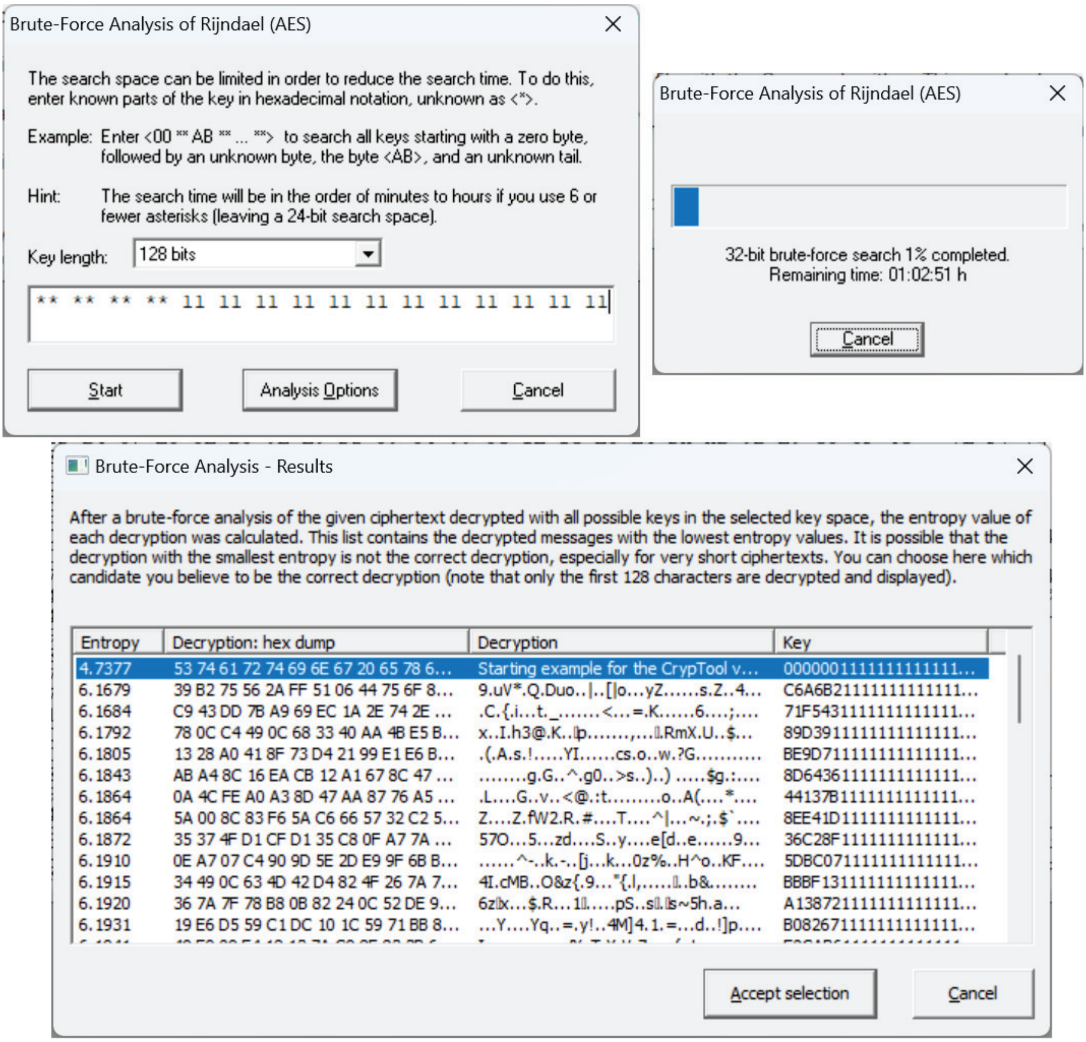

Figure 1.4 Brute-force analysis of AES in CT1 with partly known key.

the codebreakers who broke that encryption. Of course, they were all eventually broken.

For instance, the key space for the infamous Enigma I machine is larger than the number of atoms in the universe. According to Table 1.1, the theoretical key space of the Enigma is around $3 \cdot 10^{114}$, while the number of atoms in the universe is around 10^{77} (according to Table 4.13).

There are two main problems with key spaces of historic cipher devices. The first problem is that key space can be a misleading measure for the strength of the encryption. The reason for the confusion on this point arises because the key space of a modern symmetric cipher system, in contrast, usually provides a good measure for the strength of the encryption. But historic devices are mechanical or electromechanical, which results in limitations on the randomness of the encryptions. This means that methods can be developed to break that encryption without the need for brute force. Remember, key space is only a measure of the brute force

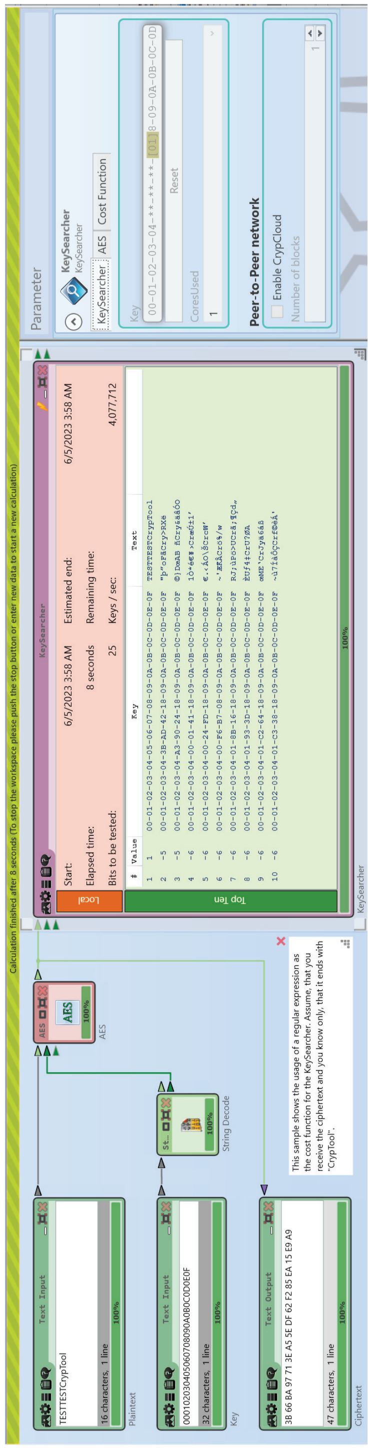

Figure 1.5 Brute-force analysis of AES in CT2 with partly-known plaintext.

required to break an encryption, without taking into account any methods used by cryptanalysts to shortcut (many) parts of that key space.

The second problem with key spaces of historic devices is due to the wild variations often reported for the very same device. This variation is usually due to differences in base assumptions, but those assumptions are not always stated.

Another thing to consider about key spaces is that cryptanalysis methods for some historic devices were not developed for many decades or even centuries after their invention. As with all things crypto-related, cryptanalysis methods are not necessarily made public. As an example, the Vigenère disk, which was invented in 1466, was reported by *Scientific American* magazine to be unbreakable in 1917. This article was published the same year that Joseph Mauborgne, U.S. Army Chief Signal Officer, boasted that his cryptographers could decrypt the Vigenère disk faster than the enemy could decrypt their own messages.

Despite the problems highlighted, a study of the key spaces of historic cipher devices is a useful tool to better understand the mind of the cipher inventor, user, and codebreaker. So with modern methods, we can discount and malign the value of key spaces of historic devices, but that alone would miss the point of understanding why historical decisions were made based on the strength of the encryption implied by these large key spaces.

1.6.2 Which Key Space Assumptions Should Be Used

After selecting a common set of assumptions, the key spaces of historic devices need to be calculated so they can be compared. Since the key space quoted most often originated from the NSA document [10] about the Enigma, that set of assumptions was used to develop the chart of historic key spaces (Table 1.1). The NSA document was written by Ray Miller and first published in 1995. In this document, Miller describes a maximum and a practical key space, but unfortunately he did not explicitly define the used assumptions.

1.6.2.1 Maximum Key Space vs Practical Key Space vs Work Factor

Miller used the term *maximum key space* for the theoretical maximum number of settings that would need to be tested for a brute-force attack. He assumed that the enemy captured the device, as per Kerckhoffs' principle, but any field-replaceable parts are unknown or could be changed, such as the rotors and reflectors. So all possible wirings of rotors and reflectors would have to be cryptanalyzed and any number of possible plugboard cables could be used.

The *practical key space* is also a theoretical number of settings but assumes that the captured machine and all field-replaceable parts are known and being used. This means that the wiring of the rotors and reflector are known but the rotors selected to be inserted into the machine and the order of those rotors are not known. This also means the reflector adds no cryptographic strength at all, since its wiring is known. Also, any user-imposed limitations are known and exploited, such as the Germans in WW2 mostly used 10 plugboard cables. These factors all help to reduce the practical key space compared to the maximum key space.

Another term (not used by Miller, but closely related to the key space) is *work factor*. This is the amount of work effort really required to break an encryption.

This number is usually smaller than the practical key space because any known cryptanalysis techniques are used as shortcuts. For the Enigma, this means that Rejewski's method of separating the cryptanalysis of the plugboard from the rotors and reflector greatly reduced the total number of settings that needed to be tested. Some of these cryptanalysis techniques were not known at the time of use or were not known by the users of these cipher devices.

Work factor is a concept more commonly used for the modern cipher systems. For the historical devices, there is very little available on work factors. It depends on the size of the message or number of messages captured. And it depends on the state of the cryptanalytic techniques that could be applied. For example: Although the Enigma machine has a huge theoretical key space, the Turing-Welchman Bombe only had to check about 422,000 settings in order to break the Enigma.[11] This work factor is what is called "attack time" when comparing the best attacks against modern ciphers in Table 1.3. For DES the work factor is drastically smaller (2^{43}) than the practical key space, and for AES it is around 2 bits smaller ($2^{254.4}$).

1.6.2.2 Key Space Assumptions Defined

The objective is to have one common set of assumptions to compare all the historic cipher devices and to use the assumptions that seem to have the most popular acceptance. Since Miller did not explicitely state his assumptions, they had to be reverse-engineered. A careful reading of the NSA document yields the following assumptions.

The maximum key space, as calculated by Miller, has three assumptions:

1. The base machine is captured and known to the enemy (per Kerckhoffs' principle);
2. Field-replaceable parts can be changed, so are not known (e.g., rotor and reflector wiring);
3. A "message setting" will be sent with each message, separate from the fixed machine setting.

The practical key space, as calculated by Miller, has four assumptions:

1. The base machine is captured and known to the enemy (per Kerckhoffs' principle);
2. Field-replaceable parts are also captured and known;
3. User-imposed limitations are known (e.g., always using 10 plugboard cables);

11. Why only 422,000? The British Bombe only tested for rotor order and rotor settings; ring settings and plugboard settings were then manually determined. With three rotors chosen from five, there are $5 \cdot 4 \cdot 3 = 60$ possible rotor orders. German procedures, however, did not allow any three rotor order to be repeated in the same month, which reduced the 60 possible orders at the beginning of the month to 30 by the end of the month. In addition, the Germans did not permit any individual rotor to be in the same position on the following day, reducing the 60 possible rotor orders to 32. Combined, these two rules reduced the possible orders to 32 at the beginning of the month, declining to 16 at the end of the month, or on average 24 rotor orders. This average rotor order multiplied with the 26^3 rotor settings yielded to $24 \cdot 17{,}576 = 421{,}824$ settings tested by the Bombe for a full run.

4. A "message setting" will be sent with each message, separate from the fixed machine setting.

1.6.2.3 Explanation of the NSA Key Space Assumptions

These assumptions detailed above seem reasonable and straightforward, except for possibly the last assumption of both the maximum and practical key spaces: A "message setting" will be sent with the message, separate from the fixed machine setting. The meaning and effect of this assumption requires further explanation.

For the Enigma, all possible wirings of the rotors are included in the maximum key space. Also, Miller includes the rotational starting positions of the rotors. Including the rotor starting position in the key space—besides it is already accounted for in all possible wirings—can be considered as redundant.

For instance, if a rotor is in position "A" and a particular wiring scheme is determined to be correct, that same wiring scheme could be advanced one position and now this new wiring scheme works when the rotor is moved to position "B." So all wiring schemes should yield 26 correct solutions as you rotate the rotor through the 26 positions. It seems you should just ignore the rotor starting position for the three rotors, which accounts for a contribution to the key space of 26^3. For this reason, many others have reported the Enigma key space without this factor.

We don't go deeper here into Enigma. There are many books and articles about this rotor machine and its history. A good summary of its design (flaws) and another approach calculating its relevant key space can be found in [11].

By including the rotor setting in the key space, Miller was allowing for a slightly larger key space that would break all daily messages after cryptanalysis of the first message. All subsequent messages using the same machine setting could then be decrypted in real time, just as the enemy would decrypt their own message.

Miller's rationale of the rotor position applies to all the rotor-based historic cipher devices, including the mechanical devices, like the Hagelin M-209. For this machine, all possible pin settings on each rotor are analyzed and included in the key space. So knowledge of the rotor rotational position is not necessary to break a message. The pin settings are part of the machine setting and fixed for the day, and the rotor setting is part of the message setting, which changes with every message. Again, just like in the case of the Enigma, the rotor positions must be known to break all daily messages in real time.

1.6.3 Conclusion of Key Spaces of Historic Cipher Devices

Having a clearly defined set of assumptions for key spaces, the key spaces could be calculated accordingly.

Table 1.1 lists 34 historic and 4 modern cipher systems, showing the maximum and practical key spaces for each one, using that same set of assumptions. This table was first presented to the International Conference on Cryptographic History (ICCH) group [12] by Ralph Simpson in Decmber 2022. The key spaces for some of these devices have not been previously reported, such as the Hebern, Japanese Purple machine, NEMA, KL-7, Transvertex HC-9, Russian VIC, and Hagelin CD-57. Most of the other historic cipher devices required new calculations to match the maximum and practical assumptions listed above.

Table 1.1 Key Space Sizes for 34 Historic and 4 Modern Cipher Systems

Year	*Cipher*	*Maximum Key Space*		*Practical Key Space*	
600 BCE	Monoalphabetic substitution	$4.03 \cdot 10^{26}$	2^{88}	$4.03 \cdot 10^{26}$	2^{88}
50 BCE	Caesar	$2.50 \cdot 10^{1}$	2^{5}	$2.50 \cdot 10^{1}$	2^{5}
1466	Vigenère (repeating keyword – 15 char.)	$1.68 \cdot 10^{21}$	2^{71}	$1.68 \cdot 10^{21}$	2^{71}
1586	Vigenère (autokey – 314 char. message)	$2.00 \cdot 10^{444}$	2^{1476}	$2.00 \cdot 10^{444}$	2^{1476}
1854	Playfair	$6.20 \cdot 10^{23}$	2^{79}	$6.20 \cdot 10^{23}$	2^{79}
1860s	Wheatstone Cryptograph	$4.03 \cdot 10^{26}$	2^{88}	$4.03 \cdot 10^{26}$	2^{88}
1912	Lugagne Transpositeur	$1.30 \cdot 10^{532}$	2^{1768}	$1.32 \cdot 10^{13}$	2^{44}
1912	M-94 cylinder cipher	$3.45 \cdot 10^{666}$	2^{2214}	$3.88 \cdot 10^{26}$	2^{88}
1916	M-138A strip cipher	$3.69 \cdot 10^{799}$	2^{2656}	$1.95 \cdot 10^{59}$	2^{197}
1918	ADFGX	$4.19 \cdot 10^{47}$	2^{158}	$4.19 \cdot 10^{47}$	2^{158}
1918	ADFGVX	$1.01 \cdot 10^{64}$	2^{213}	$1.01 \cdot 10^{64}$	2^{213}
1922	Hebern 5-rotor	$1.27 \cdot 10^{140}$	2^{466}	$4.56 \cdot 10^{10}$	2^{35}
1924	Kryha	$2.02 \cdot 10^{53}$	2^{177}	$1.78 \cdot 10^{29}$	2^{97}
1926	Enigma Swiss K	$1.60 \cdot 10^{101}$	2^{336}	$1.85 \cdot 10^{9}$	2^{31}
1930	Lugagne Le Sphinx	$1.30 \cdot 10^{532}$	2^{1768}	$2.43 \cdot 10^{24}$	2^{81}
1931	Abwehr Enigma G	$7.17 \cdot 10^{121}$	2^{405}	$4.82 \cdot 10^{10}$	2^{35}
1932	Enigma I	$3.28 \cdot 10^{114}$	2^{380}	$4.31 \cdot 10^{22}$	2^{75}
1937	SIGABA	$1.82 \cdot 10^{285}$	2^{941}	$5.95 \cdot 10^{28}$	2^{96}
1939	Japanese Purple	$3.81 \cdot 10^{59}$	2^{198}	$1.45 \cdot 10^{31}$	2^{104}
1939	Japanese JN-25 codebook (100 words)	$1.00 \cdot 10^{12}$	2^{40}	$8.25 \cdot 10^{10}$	2^{36}
1941	Lorenz SZ40/SZ42	$1.05 \cdot 10^{170}$	2^{565}	$1.05 \cdot 10^{170}$	2^{565}
1941	SG-41 "Hitler Mill"	$4.24 \cdot 10^{51}$	2^{171}	$4.24 \cdot 10^{51}$	2^{171}
1942	M-209 pin & lug	$6.16 \cdot 10^{60}$	2^{202}	$6.02 \cdot 10^{58}$	2^{195}
1942	Enigma M4	$2.33 \cdot 10^{145}$	2^{483}	$3.13 \cdot 10^{25}$	2^{85}
1942	T-52d Geheimschreiber	$7.23 \cdot 10^{213}$	2^{710}	$8.11 \cdot 10^{23}$	2^{79}
1943	Typex Mark 22	$1.82 \cdot 10^{195}$	2^{649}	$5.51 \cdot 10^{54}$	2^{182}
1947	NEMA	$5.99 \cdot 10^{164}$	2^{551}	$1.83 \cdot 10^{19}$	2^{64}
1952	Hagelin C-52	$1.68 \cdot 10^{117}$	2^{389}	$7.17 \cdot 10^{57}$	2^{192}
1952	Hagelin CX-52	$1.17 \cdot 10^{123}$	2^{409}	$1.10 \cdot 10^{104}$	2^{346}
1952	KL-7	$5.87 \cdot 10^{431}$	2^{1434}	$1.70 \cdot 10^{34}$	2^{114}
1950s	Transvertex HC-9	$2.96 \cdot 10^{71}$	2^{237}	$4.39 \cdot 10^{69}$	2^{231}
1953	VIC paper & pencil	$9.09 \cdot 10^{40}$	2^{136}	$1.00 \cdot 10^{27}$	2^{90}
1956	Fialka	$2.82 \cdot 10^{458}$	2^{1523}	$6.24 \cdot 10^{77}$	2^{258}
1957	Hagelin CD-57	$1.52 \cdot 10^{103}$	2^{343}	$1.49 \cdot 10^{60}$	2^{200}
1976	DES (56 bit)	$7.21 \cdot 10^{16}$	2^{56}	$7.21 \cdot 10^{16}$	2^{56}
1977	RSA-4096	$2.22 \cdot 10^{1225}$	2^{4071}	$2.22 \cdot 10^{1225}$	2^{4071}
1992	AT&T TSD 3600-E Clipper chip	$1.21 \cdot 10^{24}$	2^{80}	$1.21 \cdot 10^{24}$	2^{80}
2001	AES-256	$1.16 \cdot 10^{77}$	2^{256}	$1.16 \cdot 10^{77}$	2^{256}

Courtesy of Ralph Simpson.

It is important to remember that these key spaces are still not a good sole indicator of the cryptographic strength of the encryption method—examples for these criticisms are monoalphabetic substitution (2^{88}), Enigma I (2^{75}), and Playfair (2^{79}). But using a common set of assumptions will at least add a level of consistency among all these disparate devices.

1.7 Best Known Attacks on Given Ciphers

Tables 1.2 and 1.3 contain the best attacks known today for well-known classical and modern ciphers. For modern procedures, the effort (number of steps or attack time) is also given in Table 1.3. To our knowledge, this is the first time such a complete table is created.

For symmetric ciphers, the key space derived from the key length is an important indicator (see Section 1.6). It is used to calculate the effort required for a BF attack, the maximum effort that an attacker can have.

The following applies to AES-128 (see Table 1.3): The key length is 128 bits. The key space is 2^{128} and so is the theoretical attack time. The best known attack (biclique attack) reduces this maximum effort to $2^{126.1}$ steps. This difference of around 2 in the exponent means that the attack is about 4 times faster than a BF attack on average. This shows that AES is vulnerable in principle, but this attack is not at all relevant to practical security.

1.7.1 Best Known Attacks Against Classical Ciphers

The historical ciphers shown in Table 1.2 represent different periods in the history of cryptography, ranging from simple Caesar ciphers to more complex machine-assisted systems like Enigma. These selections are based on their historical significance. The attack types and methods shown in the table are the currently best known computerized methods for attacking these ciphers. All of the hand ciphers are vulnerable to simulated annealing and hill climbing. Composed ciphers, in our example here ADFGVX, need more sophisticated methods. With ADFGVX, a divide-and-conquer attack can be used to break substitution and transposition independently. Also noteworthy is SIGABA, since it can be attacked with a meet-in-the-middle attack. Additionally, all shown hand ciphers (substitution, transposition, and composed ciphers) can today be attacked in a pure ciphertext-only scenario. An exception are nomenclature ciphers, since the nomenclature elements (code words) can often only be decrypted when having either the original key or enough context to deduce them. Also, the chances of successfully attacking cipher machines, such as the Enigma and Typex, are enhanced when a crib (a partially known plaintext) is available. Only attacks on SIGABA still require the complete plaintext to be successful.

1.7.2 Best Known Attacks Against Modern Ciphers

Table 1.3 presents a selection of modern ciphers and the best attacks against them. The table includes historically significant ciphers such as DES and FEAL, ISO standards like AES, Camellia, and SNOW 2, national standards like GOST and SM4, and ciphers that were actively used in industrial solutions such as KeeLoq and A5.1. Cipher names typically encompass a family of encryption methods rather than referring to a single algorithm. These algorithms usually differ in the size of the key used and, in the case of block ciphers, the size of the data block. It is important to note that the best attacks against various versions of a cipher may differ. For the sake of brevity, we provide a single example from each cipher family and present the most successful attack against it.

In the right-most column of Table 1.3, the term “attack time” is used. “Time” is an established term used in modern cryptography. In order to understand what the attack time—as a measure for the resistability of a cipher—means, see Section 1.8 which introduces attack costs and different attack types.

Table 1.2 Best Known Attacks Against 17 Historical Ciphers

Cipher	*Attack Requirements*	*(Best) Cryptanalysis Methods*	*References*
Substitution ciphers			
Caesar	PCO	Brute force, frequency analysis	[13]
Monoalphabetic substitution	PCO	Hill climbing, frequency analysis	[13]
Homophonic substitution	PCO	Hill climbing / simulated annealing	[14]
Nomenclatures	PCO	Manual (deduced by context; or nomenclature available)	[15, 16]
Polyalphabetic substitution	PCO	Hill climbing / simulated annealing / (Friedman + Kasiski)	[13]
Playfair	PCO; crib	Simulated annealing	[17, 18]
Code books	PCO; crib/KP	Manual (deduced by context; availability of similar code book)	[19]
Chaocipher	PCO	Hill climbing / simulated annealing	[20]
Transposition ciphers			
Scytale	PCO	Brute force	[13]
Columnar transposition	PCO	Brute force (short keys) / hill climbing / simulated annealing	[21]
Double columnar transposition	PCO	Hill climbing / simulated annealing; IDP attack	[22]
Composed			
ADFGVX	PCO	DAC + hill climbing / simulated annealing	[23]
Machines			
Enigma	PCO, crib	DAC; hill climbing / simulated annealing; Turing Bombe	[24, 25, 26]
Typex	PCO, crib	DAC; hill climbing / simulated annealing; Turing Bombe	[24, 25, 26]
SZ42	PCO, crib	Testery methods and hill climbing	[27]
M209	PCO, crib	Simulated annealing / hill climbing	[28, 29]
SIGABA	KP	Meet in the middle; hill climbing / simulated annealing	[30, 31]

PCO = pure ciphertext-only, KP = known-plaintext, DAC = divide and conquer.

1.8 Attack Types and Security Definitions

If you are interested in the definitions used in modern cryptography, this section explains them with the fewest amount of mathematics as possible. Also, the relationship between the various definitions is declared—something which often falls short in courses. We believe that only understanding the differences between the various concepts enables learners to grasp the idea and apply it correctly later.

1.8.1 Attack Parameters

In cryptography, a security parameter is a way of measuring of how hard it is for an adversary to break a cryptographic scheme. Attack parameters describe the conditions available for the attacker.

Table 1.3 Best Known Attacks Against 36 Modern Ciphers

Cipher	*Attack Types*	*(Best) Cryptanalysis Methods*	*Attack Time*
Block ciphers			
DES	Single key. KPA. Full	Linear [32, 33]	2^{43}
3DES (TDEA). 3-key version [34]	Single key. KPA. Full	Meet-in-the-middle [34]	2^{112}
AES-128 (Rijndael) [35]	Single key. CCA. Full	Biclique [36]	$2^{126.1}$
Camellia-128 [37]	Single key. CPA. 11/18 rounds	Truncated differential [38]	$2^{121.3}$
MISTY1 [39]	Single key. CPA. Full	Integral [40, 41]	$2^{107.9}$
KASUMI [42]	Related-key. CCA. Full	Boomerang [43]	2^{32}
HIGHT [44]	Single key. CCA. Full	Biclique [45]	$2^{126.4}$
CAST-128 [46]	Single key. CPA. 9/16 rounds	Differential [47]	2^{73}
SEED-128 [48]	Single key. CPA. 8/16 rounds	Differential [49]	2^{122}
PRESENT [50]	Single key. CPA. 26/31 rounds	Truncated differential [51]	2^{70}
CLEFIA-128 [52]	Single key. CPA. 14/18 rounds	Truncated differential [38]	2^{108}
LEA-128 [53]	Single key. CPA. 13/24 rounds	Differential [54]	2^{127}
SM4 [55]	Single key. KPA. 24/32 rounds	Linear [56]	$2^{126.6}$
GOST 28147-89 [57] (Magma)	Single key. CPA. Full	Guess then truncated differential [58]	2^{179}
GOST R 34.12-2015 (Kuznechik) [59]	Single key. CCA. 5/10 rounds	Meet-in-the-middle [60]	2^{140}
KeeLoq [61]	Single key. KPA. Full	Slide and meet-in-the-middle [62]	$2^{44.5}$
Simon64/128 [63]	Single key. KPA. 31/44 rounds	Multidimensional linear [64]	2^{120}
Speck64/128 [63]	Single key. CPA. 20/27 rounds	Differential [65]	$2^{93.56}$
FEAL-32 [66]	Single key. CPA. 31/32 rounds	Differential [67]	2^{63}
Twofish-128 [68]	Single key. CPA. 7/16 rounds	Saturation [69]	2^{126}
Stream ciphers			
RC4	Variable-key. Plaintext recovery. COA	Statistical [70]	2^{31}
A5/1 [71]	Single key. KPA. Full	Time-memory-data trade-off [72]	2^{24}
A5/2 [73]	Single key. KPA. Full	Time-memory-data trade-off [72]	2^{16}
Chacha [74]	Single key. KPA. Chosen IV. 7/20 rounds	Differential [75]	2^{255}
Salsa20 [76]	Single key. KPA. Chosen IV. 8/20 rounds	Differential [75]	2^{255}
Crypto-1 [77]	Single key. KPA. Full	Algebraic [78]	2^{32}
Grain-128 [79]	Single key. KPA. Chosen IV. Full	Dynamic cube attack [80]	2^{74}
Trivium [81]	Single key. KPA. Chosen IV. 799/1152 rounds	Dynamic cube attack [82], see also footnote 1	2^{62}
Rabbit [83]	Not known	See also footnote 2	
Enocoro 128v2 [84]	Distinguishing. KPA. Chosen IV. 22/96 rounds	Higher order differential [85]	2^{16}
SNOW 2-128 [86]	Single key. KPA. Chosen IV. 14/32 rounds	Cube [87]	$2^{162.86}$
MUGI [88]	Distinguishing. KPA. Chosen IV. 21/32 rounds	Differential [89]	$2^{61.59}$
ZUC 1.6 [90]	Not known	See also footnote 3	

Table 1.3 *Continued*

Cipher	*Attack Types*	*(Best) Cryptanalysis Methods*	*Attack Time*
Public-key encryption			
RSA [91]	Single key. COA. For RSA-250 (829-bit number)	Number field sieve [92, 93], see also footnote 4	$2^{68.5}$
ElGamal [94]	Single key. CCA	Trivial algebraic	Instant
NTRUEncrypt [95]	Single key. COA	Hybrid [96] (Lattice reduction and combinatorial search)	PB, see also foot-note 5

PB = parameter-based.

1. Another attack claiming to break 855 rounds [97] of Trivium has been questioned in [98].
2. We are not aware of any attacks faster than brute force. Rabbit has four initialization rounds. The values within the cipher become balanced after two rounds [83], hence there is a trivial distinguishing attack against at least one round of the cipher.
3. There exist attacks against earlier versions of the cipher. The cryptanalysis of the final version made by the designers is secret to the best of our knowledge.
4. Our upper-bound estimation: In [93], the attack time is given as 2, 700 core years of computations using Intel Xeon Gold 6130 CPU (each 2.1 GHz). To convert this attack time to the RSA-250 encryptions, we would need to know how much time is required on average to apply one encryption on the mentioned processor. For a rough estimate, we assume that one encryption requires less time than one integer operation as tested in [99].
5. The actual attack time depends on the specific parameter choices. See [100] for more details.

Attack definition. Before proceeding to the discussion about various attack types (see Section 1.8.1), it's essential to clarify the concept of an attack against a modern cipher. We start this explanation with Kerckhoffs's principle (see Section 1.5). This principle emphasizes that a cryptosystem should be secure even if all the system details, excluding the secret key, are known to the attacker.

However, the principle brings up the term "secure." To formulate the definition of security, we use ideas about the *infeasibility of distinguishing*—see Sections 1.8.2 and 1.8.3. In a nutshell, a cryptographic attack is an algorithm that aims to demonstrate the lack of security in a given cryptosystem.

Attack costs. When analyzing how difficult it is to apply a cryptographic attack, the *computational complexity* of the corresponding algorithm is evaluated. The computational complexity is the amount of resources needed to run the algorithm. There are typically three main resources considered: time, memory, and data.

- Time complexity of the attack, or just attack time, is an estimated upper limit of the number of operations required to successfully break a cipher. Time is the primary resource taken into account. If "computational complexity" is mentioned without further specification, it typically refers to "time complexity."
- Memory complexity is the storage space needed to execute the attack.
- The data complexity refers to the amount of data (plaintext, ciphertext, or both) that the attacker needs access to in order to carry out the attack.

Attack time. The *attack time* is generally expressed in the number of a particular cipher's encryptions. This is done in order to demonstrate by which factor the corresponding attack is faster than the brute-force attack. As discussed in Section 1.2.2, the key-space size has a direct relation to the attack time of the brute force. Testing each of the keys requires the corresponding encryption algorithm to run once completely. So if the key (in binary representation) length is L, and all possible variants of the key lead to different ciphertexts, then the key space size is 2^L. It means that in order to certainly break a cipher, 2^L encryptions are always enough. This determines the attack time of the exhaustive search.

Different attacks may not require running the encryption algorithm itself, but to perform other computational operations. In this case, an estimation is done on how many of such operations require the time equivalent to the time of one encryption. Then the whole number of operations needed to apply the attack is divided over the number of operations equating to a single encryption. This results in the time complexity for the current attack measured in encryptions.

Security parameter. A cryptographic attack is considered to be successful if it requires less costs than defined by the security parameter set by the designers of a cryptosystem. A security parameter measures the level of difficulty for an adversary to break a cryptographic scheme. It is often expressed in bits. For example, one can say that a certain scheme offers κ-bit security if the attack time is of $O(2^\kappa)$ encryptions. The $O()$ notation (also called big O notation or Bachmann–Landau notation or asymptotic notation) describes an upper bound on the time complexity of an algorithm. Essentially, it gives the worst-case scenario for how the run time grows as the input size increases. Here we don't need the big O notation, which is used to describe the limiting behavior of a function when the argument tends towards a particular value. But here in the table, we use the concrete versions of the ciphers and provide the complexities with a constant argument.

In the context of symmetric encryption schemes, the security parameter is typically equal to the key size. This is because the brute-force attack sets the minimum limit for the security parameter. However, the security parameter can be lower than the key size if an attack faster than the brute force is known at the stage of the design of a cipher. This is a common situation for public-key encryption schemes.

Goal. In modern cryptology, different classifications of cryptanalytical attacks exist. By the goal of the attacker we differentiate between *key-recovery attacks* and *distinguishing attacks*. The key-recovery attacks aim to obtain the actual encryption or decryption key, compromising the security of the cryptographic system completely. On the other hand, distinguishing attacks focus on the ability to differentiate encrypted data from truly random data, indicating deviations or weaknesses in the cipher that may lead to key-recovery attacks.

Single/multiple keys. Cryptanalytic attacks also vary based on the attacker's ability to observe different numbers of encryption instances related to distinct keys. *Single-key attacks* assume access to the ciphertexts encrypted under the same key. *Variable-key attacks* assume access to ciphertexts encrypted under multiple unknown keys. This often mirrors real-world situations where a cipher's user must change the key after a certain number of encryptions. If an attacker gains access to several corresponding ciphertexts, he can use this information as an advantage in attempting to break any of the corresponding encryptions. *Related-key attacks* assume that an attacker has knowledge of a certain mathematical relationship that exists between different secret keys and that she can observe the corresponding ciphertexts. Although at first glance, such a scenario can be seen as too unrealistic, several cryptosystems were broken using related-key attacks in the real world (e.g., [43]).

Access to data (ciphertext-plaintext pairs). The cryptographic attacks can be divided into the following four main categories based on the type of access to the ciphertext and plaintext (assuming the key is always unknown):

- Ciphertext-only attacks (COA) assume access only to ciphertexts without knowledge of corresponding plaintexts;
- Known-plaintext attacks (KPA) involve pairs of known plaintext and their corresponding ciphertext, aiming to recover the secret key;
- Chosen-plaintext attacks (CPA) allow the attacker to choose arbitrary plaintexts and obtain their ciphertexts, providing flexibility in analyzing the encryption algorithm;
- Chosen-ciphertext attacks (CCA) enable the attacker to choose arbitrary ciphertexts and obtain their plaintexts, possessing the power to manipulate ciphertexts during decryption.

Additionally, attacks differ based on specific mathematical methods, such as differential cryptanalysis (analyzing how differences between inputs of the ciphers affect resultant differences between outputs), linear cryptanalysis (exploiting linear relationships in the encryption process), meet-in-the-middle, biclique, integral, boomerang, cube, and other attacks. All these methods are unique, so we refer to the provided references for a comprehensive explanation.

1.8.2 Indistinguishability Security Definitions

The attack types CPA and CCA have a direct relationship with the cryptographic security definitions IND-CPA, IND-CCA1, and IND-CCA2. These definitions play a crucial role in the provable security branch of cryptography. This field focuses on proving mathematically the security of the cryptographic schemes. This is achieved by demonstrating that breaking a certain scheme would require solving a problem that is widely known to be difficult, such as factoring large numbers or computing discrete logarithms.

Indistinguishability under chosen-plaintext attack (IND-CPA). In this model, an attacker is allowed to choose arbitrary plaintexts and obtain the corresponding ciphertexts from the encryption oracle as many times as he needs. Then the adversary chooses two distinct challenge messages and sends them to the encryption oracle, which returns a ciphertext of just one of them called challenge ciphertext. After that, the attacker is allowed to perform any number of additional computations and encryptions. An encryption scheme is considered secure if the attacker can't guess to which plaintext the challenge refers to with the probability higher than $|1/2 + \eta|$ where η is negligible. Clearly, the attacker cannot choose the same messages for the challenge for which he gets the ciphertexts from the oracle. This security definition can be applied to both symmetric and asymmetric encryption schemes, although formally they are described differently [101]. However, in case of deterministic asymmetric encryption schemes, an attacker has access to the public key, which means that he can easily distinguish which ciphertext was produced by which message by encrypting the messages by himself. Therefore, the definition is only applied to probabilistic public-key encryption schemes where randomness is used in the encryption process. This implies that the same message encrypted several times under the probabilistic encryption scheme results in different ciphertexts.

Indistinguishability under chosen-ciphertext attack, also known as nonadaptive or lunchtime attack (IND-CCA1). This security definition imposes a higher level of security than IND-CPA. In this model, an attacker can choose both the plaintexts and obtain their corresponding ciphertexts from the oracle, and also decrypt arbitrary ciphertexts and get the corresponding plaintexts. The further procedure is similar to the IND-CPA case. However, in the case of IND-CCA1 after the adversary gets the challenge the decryption oracle becomes unavailable.

Indistinguishability under adaptive chosen-ciphertext attack (IND-CCA2). This is the strongest definition providing the highest level of security. It allows the attacker to continue to interact with the decryption oracle even after the challenge ciphertext is received.

When considering modern cryptographic encryption primitives, selecting the best attack is not a straightforward task. In Table 1.3, we have kept the information concise and prioritized key-recovery attacks requiring minimal computation and being faster than brute-force, which is a universal attack method against any encryption algorithm. By this prioritizing, we have left out other complexities such as data and memory costs (e.g., number of required plaintext-ciphertext pairs).

Single-key scenarios are typically targeted, except for two exceptions in our table: the related-key attack against Kasumi cipher and the variable-key attack against RC4. If the full cipher is not compromised, we aim to select attacks that break as many rounds as possible. We only refer to distinguishing attacks against MUGI and Enocoro as we are not aware of any published key-recovery attacks.

1.8.3 Security Definitions

Modern cryptography is heavily based on mathematical theory and computer science practice. Cryptographic algorithms are designed around computational hardness assumptions, making such algorithms hard to break in practice by any adversary.

There are different approaches (categories) to define the security of cryptosystems.

Most commonly, two fundamental approaches are used for formally defining the security of an encryption scheme [102]:

- The first one is semantic security, which implies that it is infeasible for an attacker to learn any information about the plaintext from the ciphertext;

- The second definition determines security as the infeasibility of distinguishing between encryptions of two given messages.

In both definitions of security, the term "infeasible" rather than "impossible" is used. This is because generic attacks exist against almost every known encryption scheme (with the exception of the one-time-pad). One such universal attack, namely a brute force, was discussed in Section 1.2.2. Brute-force attacks can be extended to time-memory trade-off (TMTO) attacks, a broader class of attacks, which in certain cases allow to reduce the key-recovery time by increasing the memory cost. See Table 1.3 for an in-depth discussion of different attack types.

Another main category in literature defines security depending on the adversary's capabilities (e.g., *Cryptography 101* [103, Chap. 1.2.2]):

Computational, conditional, or practical security. A cipher is *computationally* secure if it is theoretically possible to break such a system, but it is infeasible to do so by any known practical means. Theoretical advances (e.g., improvements in integer factorization algorithms) and faster computing technology require these solutions to be continually adapted.

Even using the best known algorithm for breaking it will require so many resources (e.g., 1,000,000 years) that essentially the cryptosystem is secure.

So this concept is based on assumptions of the adversary's limited computing power and the current state of science.

A typical example of a pragmatically secure procedure is AES: No practicable attack is known on it. Even so, AES is theoretically broken, which just means it can be broken with less effort than a brute-force attack. This effort is still unrealistically high. See Section 1.7.

Information-theoretical or unconditional security. A cipher is considered *unconditionally* secure if its security is guaranteed no matter how many resources (time, space) the attacker has. Even if the adversary has unlimited resources he is unable to gain any meaningful data from a ciphertext.

The only information-theoretically secure schemes that provably cannot be broken even with unlimited computing power are the *one-time pad* (OTP) or variants of it.

Figure 1.6 shows that it may be impossible to determine the correct plaintext from a OTP (if the OTP method has been applied correctly and if all keys have the same likelihood). The example in this figure uses an 8-character long given ciphertext: `11 1B 1E 18 00 04 0A 15`. The hex values correspond to the ASCII values of the letters: For example, the letter `C` has the numerical value `67` (decimal), which is `43` in hex representation.

There are many meaningful words with eight letters and for each there is a correct key. So an attacker cannot determine alone from the ciphertext which is the correct key and which is the correct plaintext word. In other words, with different keys the same ciphertext can lead to different meaningful plaintexts and so, in this case, it cannot be distinguished which plaintext is the correct one.[12]

12. The OTP procedure is discussed in more detail in Section 2.2.4 in item "One-time pad."
Also see Figure 9.12, where a corresponding example with text strings is built with SageMath, and the XOR method is explained.

Key (hex)	Revealed plaintext (hex)	Revealed plaintext (txt)
52 49 47 48 54 4B 45 59	43 52 59 50 54 4F 4F 4C	CRYPTOOL
41 5A 50 57 52 45 47 54	50 41 4E 4F 52 41 4D 41	PANORAMA
54 77 7B 68 68 65 64 61	45 4C 45 50 48 41 4E 54	ELEPHANT
50 55 5B 55 4F 4A 4F 46	41 4E 45 4D 4F 4E 45 53	ANEMONES
52 53 5F 55 50 4D 45 5B	43 48 41 4D 50 49 4F 4E	CHAMPION
42 58 5B 56 41 56 43 5A	53 43 45 4E 41 52 49 4F	SCENARIO

Figure 1.6 Illustration of the information-theoretically secure OTP scheme.[13]

As the OTP is information-theoretically secure it derives its security solely from information theory and is secure even with unlimited computing power at the adversary's disposal. However, OTP has several practical disadvantages (the key must be used only once, must be randomly selected, and must be at least as long as the message being protected), which means that it is hardly used except in closed environments such as for the hot wire between Moscow and Washington.

Two more security concepts are sometimes used:

- *Provable security.* This means that breaking such a cryptographic system is as difficult as solving some supposedly difficult problem, such as discrete logarithm computation, discrete square root computation, or very large integer factorization.

  Example: Currently we know that RSA is at most as difficult as factorization, but we cannot prove that it's exactly as difficult as factorization. So RSA has no proven minimum security. Or in other words, we cannot prove that if RSA (the cryptosystem) is broken, then factorization (the hard mathematical problem) can be solved.

  The Rabin cryptosystem was the first cryptosystem that could be proven to be computationally equivalent to a hard problem (integer factorization).

- *Ad-hoc security.* A cryptographic system has this security feature if it is not worth trying to break the system because the effort to do so is more expensive than the value of the data that would be obtained by doing so. Or an attack can't be done in sufficiently short time (see [104]).

  Example: This may apply if a message relevant to the stock market will be published tomorrow, and you would need a year to break it.

13. Source of the four photos: https://pixabay.com/.

1.9 Algorithm Types and Self-Made Ciphers

Here, two aspects of crypto procedures are mentioned briefly, which are often not discussed early enough: types of algorithms and the thinking up of new algorithms.

1.9.1 Types of Algorithms

Algorithms can be categorized as follows:

- *Random-based.* Algorithms can be divided up into *deterministic and heuristic* methods. Often students only become aware of deterministic methods, where the output is uniquely determined by the input. On the other hand, heuristic methods make decisions using random values and the results are only correct with a certain probability. One can differentiate even more precisely between randomized algorithms, and probabilistic and heuristic methods, but these subtleties are not important for understanding the contrast to deterministic methods.

  Random looms large in cryptographic methods. Keys have to be selected randomly, which means that at least for the key generation "random" is necessary. In addition, some methods, especially from cryptanalysis, are heuristic.
- *Constant-based.* Many modern methods (especially hash methods and symmetric encryption) use numeric constants. Their values should be plausible, and they shouldn't contain back doors. Numbers fulfilling this requirement are called nothing-up-my-sleeve numbers.

1.9.2 New Algorithms

It happens again and again that someone without deeper knowledge of adequate design concepts comes up with a "new" encryption procedure. However, reality shows that this is not a good idea. That's why people usually learn early not to design their own cryptosystem if they hope that the fact that it is not known will protect them. There are many reasons for this, including that it only takes one disgruntled employee or any other malicious actor to reveal the secrets that make the scheme secure. Designing secure cryptographic schemes is extremely difficult. It is incredibly easy to create something that looks secure, but actually leaks information.

Offering prize money and just single ciphertexts is unprofessional—serious researchers have little time and will not spend any effort on it (perhaps they give it to students as an exercise for didactic reasons). Modern best practice is that if you want to create a new encryption scheme, first publish it with a detailed explanation of how it works, its advantages, and any evidence of its security. Then you can see if anyone can find any weaknesses. This is not a quick process—you should expect it to take years.

1.10 Further References and Recommended Resources

Here are some good cryptography books that can serve as useful background on various topics in order from beginners (history) to intermediate (applied) to advanced (theory-focused):

- David Kahn: *The Codebreakers*, 1995.
- Elonka Dunin and Klaus Schmeh: *Codebreaking: A Practical Guide*, 2nd ed, 2023.
- Simon Singh: *The Code Book*, 2000 [105].
- Bruce Schneier, *Applied Cryptography, Protocols, Algorithms, and Source Code in C*, 2nd ed, 1996 [8].
- Christof Paar and Jan Pelzl: *Understanding Cryptography*, 2009 [106].
- David Wong: *Real-World Cryptography*, 2020 [107] (our favorite).
- Jean-Philippe Aumasson: *Serious Cryptography*, 2017 [108].
- Mike Rosulek: *The Joy of Cryptography*, 2021.
- Niels Ferguson, Bruce Schneier, and Tadayoshi Kohno: *Cryptography Engineering*, 2010.
- Dan Boneh and Victor Shoup: *A Graduate Course in Applied Cryptography*, v0.6, 2023.
- Mark Stamp and Richard M. Low: *Applied Cryptanalysis: Breaking Ciphers in the Real World*, 2007 [109].
- Rolf Oppliger, *Cryptography 101*, 2021 [103].
- Jonathan Katz and Yehuda Lindell: *Introduction to Modern Cryptography*, 3rd ed, 2020.
- Douglas R. Stinson: *Cryptography – Theory and Practice*, 3rd ed, 2006 [110].

Besides the information in these books and in the following chapters, there is also a good number of websites and the online help of all CrypTool variants that contain many details about encryption methods.

The book by Bruce Schneier [8] offers an easy overview of the different encryption algorithms. For a more in-depth introduction, in addition to the book by Rolf Oppliger [103], we also recommend the books by David Wong [107], Jean-Philippe Aumasson [108], and Douglas R. Stinson [110].

1.11 AES Visualizations/Implementations

AES is now probably the most widely used modern encryption algorithm worldwide. AES is a secure, standardized, symmetrical process that encrypts data, for example, in Wi-Fi and browser connections. The AES-192 and AES-256 variants are approved for top-class government documents in the United States.

In the following sections, first an AES animation is presented in CTO; and then AES is executed directly—once in CT2 and twice with OpenSSL (once on the command line of the operating system and once in the OpenSSL WebAssembly plugin in CTO).

1.11.1 AES Animation in CTO[14]

Figure 1.7 shows that the modern encryption algorithm receives both inputs (the key and the plaintext) in binary form and creates the output in binary form. Like most modern (block) ciphers, the algorithm contains a key scheduling part where from the given key (also called session key, master key, or cipher key) the round keys are generated, and another part where then the actual encryption is carried out using the generated round keys.

Figures 1.7 to 1.8 are taken from the AES animation in CrypTool-Online (CTO). Figure 1.9 is from CT1, but the image is also part of the CTO animation.

1.11.2 AES in CT2

After these visualizations, we want—in a concrete example—to encrypt a plaintext of length 128 bits (one block) with a 128-bit key with AES in CBC mode. From the

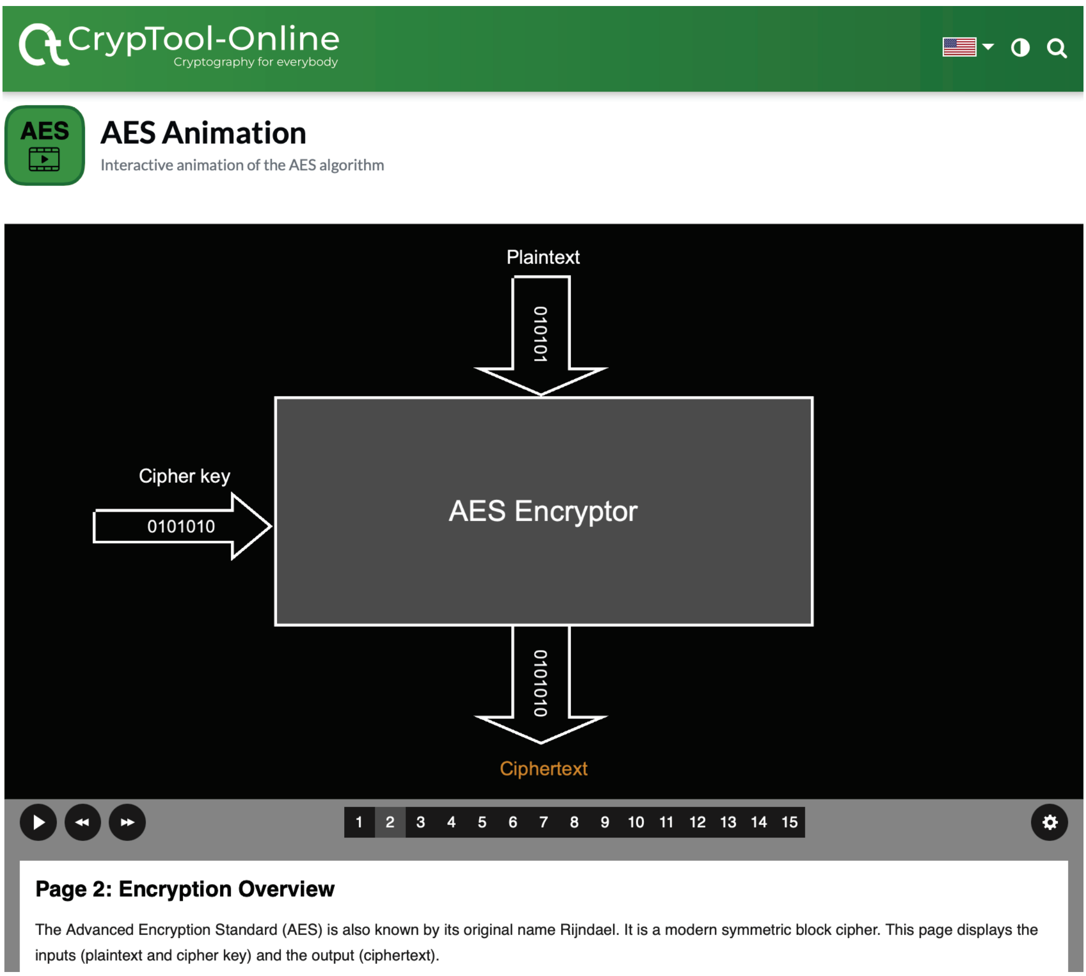

Figure 1.7 AES visualization from CTO (part 1).

14. https://www.cryptool.org/en/cto/aes-animation.

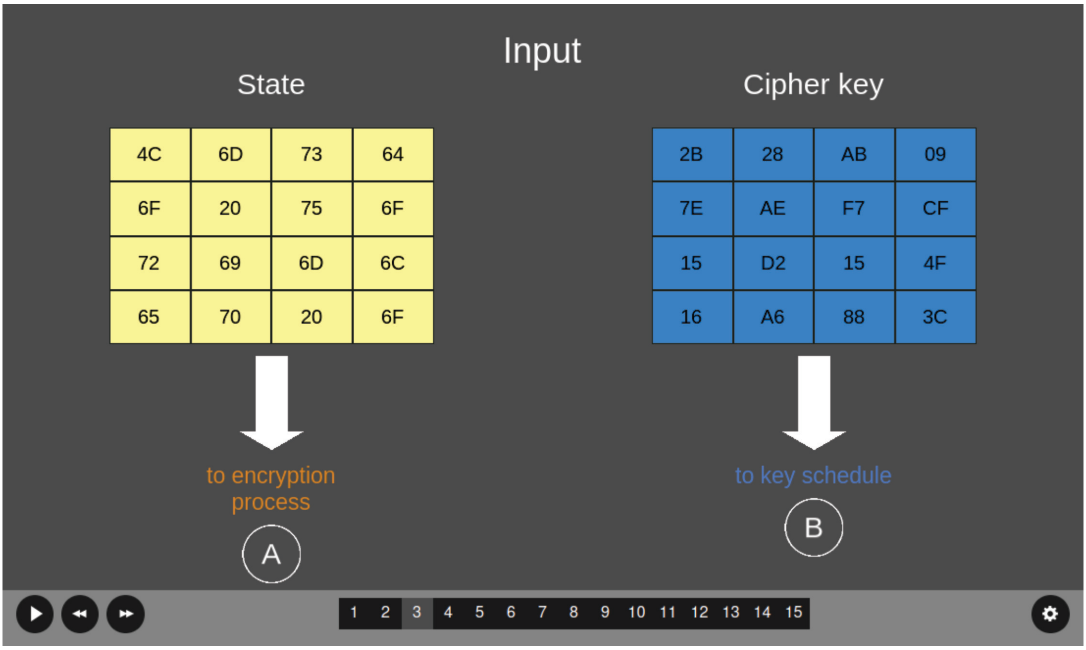

Figure 1.8 AES visualization from CTO (part 2).

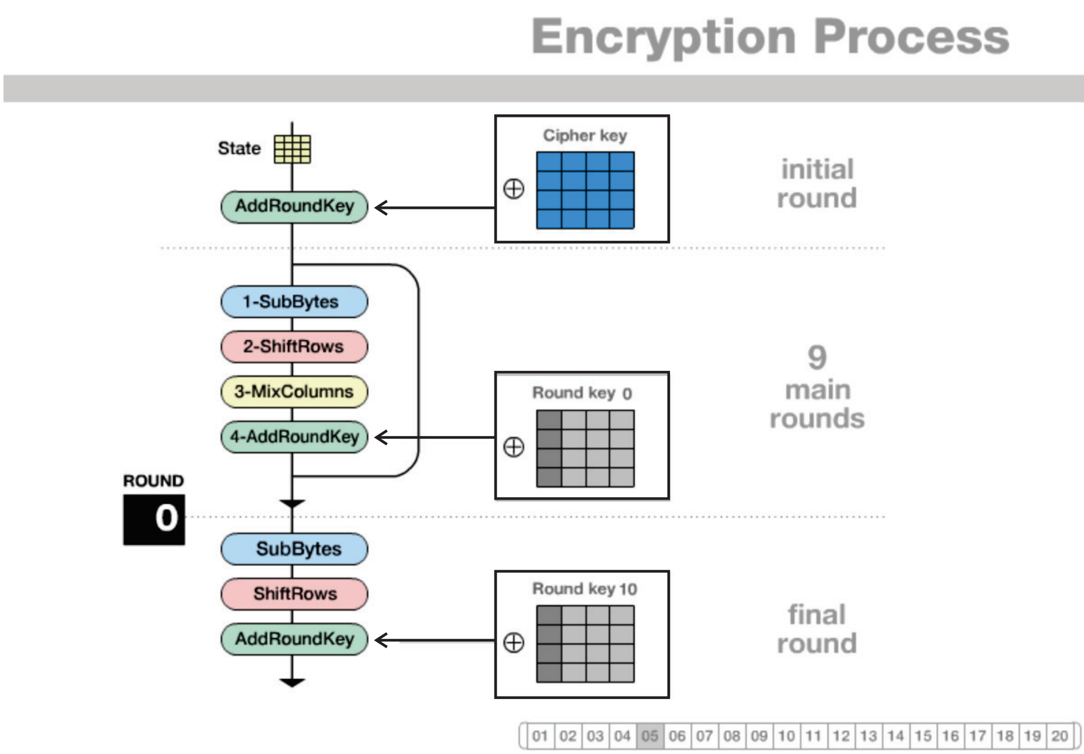

Figure 1.9 AES visualization by Enrique Zabala from CT1.

received ciphertext we are only interested in the first block (if the plaintext doesn't fill up a complete block, for the sake of simplicity, here we use zero padding).

For demonstration, we do it once with CT2 and twice with OpenSSL.[15]

The plaintext `AESTEST1USINGCT2` is converted to hex (41 45 53 54 45 53 54 31 55 53 49 4E 47 43 54 32). Using this and the key 3243F6A8885A308D313198A2E0370734 the AES component creates the ciphertext, which is in hex: B1 13 D6 47 DB 75 C6 D8 47 FD 8B 92 9A 29 DE 08.

Figure 1.10 shows the encryption of one block in `CT2`.[16]

1.11.3 AES with OpenSSL at the Command Line of the Operating System

OpenSSL Example 1.1 achieves the same result as CT2 with `OpenSSL` from the (Windows) command line.

OpenSSL Example 1.1: AES Encryption (Of Exactly One Block and Without Padding)

```
>openssl enc -e -aes-128-cbc -K 3243F6A8885A308D313198A2E0370734 -iv 00▸
 ▸000000000000000000000000000000 -in klartext-1.hex  -out klartext-1.▸
 ▸hex.enc
>dir
06.07.2016  12:43              16 key.hex
20.07.2016  20:19              16 klartext-1.hex
20.07.2016  20:37              32 klartext-1.hex.enc
```

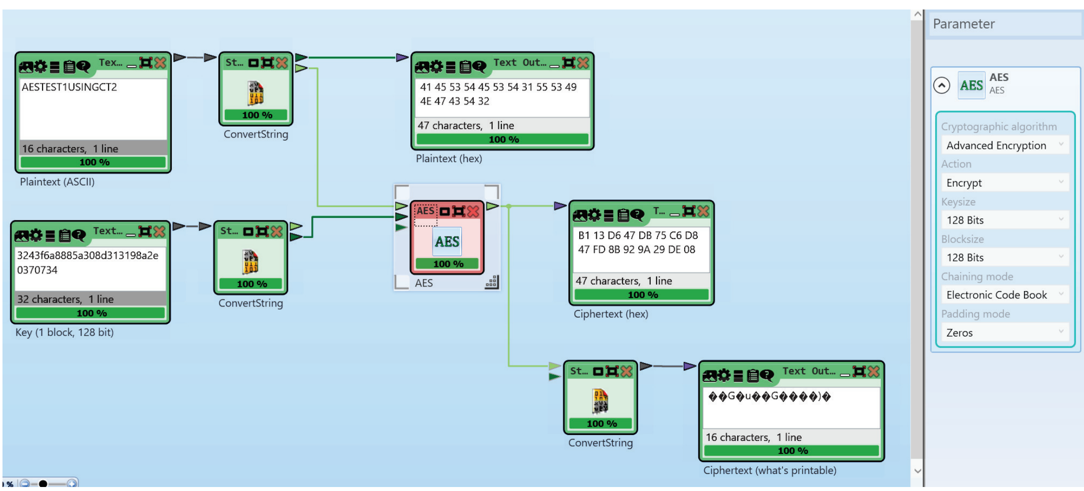

Figure 1.10 AES encryption (here exactly 1 block and without padding) in CT2.

15. OpenSSL is a widespread free open-source crypto library that contains the command line tool `openssl`.
Using OpenSSL you can try out the functionality on many operating systems.
You can find an introduction into the CLI `openssl` (e.g. at https://www.cryptool.org/en/documentation/ctbook/).

16. This is similar to the following template: CT2 Templates ▹ Cryptography ▹ Modern ▹ Symmetric ▹ AES Cipher (Text Input).

Note: As OpenSSL Example 1.2 shows, with a little effort, pipes, and the tool xxd, this can be achieved also in a Bash shell and without using temporary files:[17]

OpenSSL Example 1.2: AES Encryption (Without Temporary Files) With Bash

```
$ echo 0: 41 45 53 54 45 53 54 31 55 53 49 4E 47 43 54 32 | xxd -r | ▸
  ▸openssl enc -e -aes-128-cbc -nopad -K 3243F6A8885A308D313198A2E03707▸
  ▸34 -iv 00000000000000000000000000000000  | xxd -p
b113d647db75c6d847fd8b929a29de08

$ echo -n AESTEST1USINGCT2 | openssl enc -e -aes-128-cbc -nopad -K 3243▸
  ▸F6A8885A308D313198A2E0370734 -iv 00000000000000000000000000000000  |▸
  ▸ xxd -p
b113d647db75c6d847fd8b929a29de08
```

1.11.4 AES with OpenSSL within CTO[18]

As CTO has integrated a WebAssembly-based version of OpenSSL, this also can be done locally in your browser without the need to install OpenSSL. While Linux systems mostly have OpenSSL on board, Windows systems or smart phones don't. For such systems this plugin is helpful.

For the example in Figure 1.11 we store the message AESTEST1USINGCT2 in a file called "klartext-1.hex." Then we upload this file from the file system of the operating system into a virtual file system in the browser: This upload is done in the tab "Files" of the OpenSSL plugin. Then in the OpenSSL plugin the same `openssl` command is executed as before in the terminal (see Section 1.11.3). And if you download the resulting file klartext-1.hex.enc and compare it with the result from the terminal, you see both are identical.

1.12 Educational Examples for Symmetric Ciphers Using SageMath

Section 1.12.1 shows the SageMath implementation of a cipher (called Mini-AES) stripped for didactic purposes. Further publications with ciphers reduced for didactic reasons are listed in Section 1.12.2.

1.12.1 Mini-AES

The SageMath module `crypto/block_cipher/miniaes.py` supports Mini-AES to allow students to explore the inner working of a modern block cipher.

Mini-AES, originally described in [111], is a simplified variant of AES to be used for cryptography education.

Here is a short list about how Mini-AES was simplified compared to AES:

17. xxd creates a hex dump of a given file or of standard input. With the option "-r" it converts hex dump back to its original binary form.
18. `https://www.cryptool.org/en/cto/openssl`.

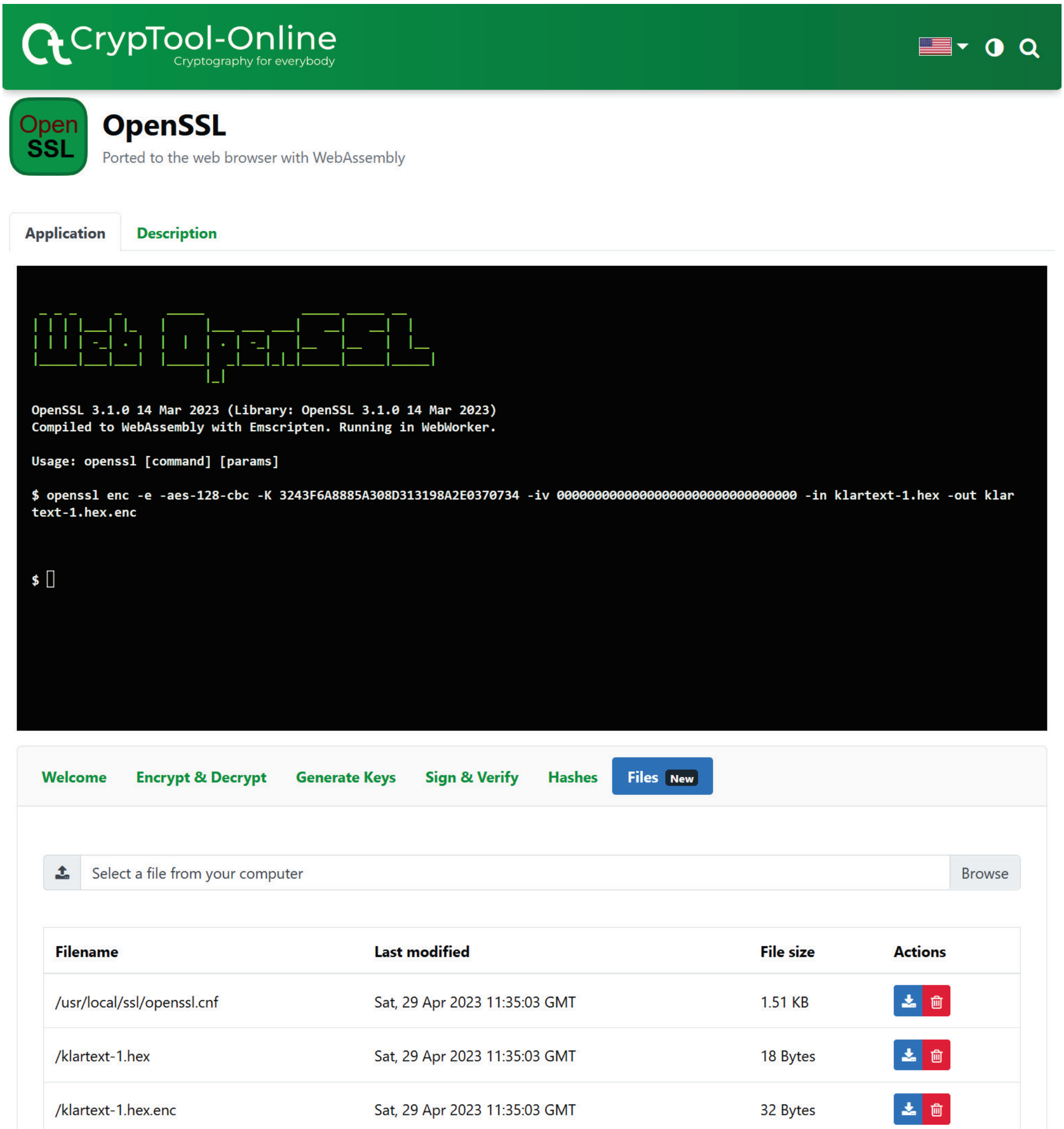

Figure 1.11 AES encryption using OpenSSL in the browser.

- The AES has a block size of 128 bits, and supports key sizes of 128, 192, and 256 bits. The number of rounds is 10, 12, or 14 for the three different key sizes, respectively.

  Mini-AES has a 16-bit block size, a 16-bit key size, and 2 rounds.
- The 128-bit block of the AES is expressed as a matrix of 4×4 bytes, in contrast to Mini-AES expressing its 16-bit block as a matrix of 2×2 nibbles (half-bytes).
- The AES key schedule takes the 128-bit secret key and expresses it as a group of four 32-bit words.

> The Mini-AES key schedule takes the 16-bit secret key and expresses it as a group of four nibbles (4-bit words).

How to use Mini-AES is exhaustively described at this SageMath reference page: https://doc.sagemath.org/html/en/reference/cryptography/sage/crypto/block_cipher/miniaes.html.

SageMath Example 1.1 was originally taken from the release tour of SageMath 4.1[19] and calls the implementation of the Mini-AES.

SageMath Example 1.1: Encryption and Decryption with Mini-AES

```
print("\n# CHAP01 -- Sage-Script-SAMPLE 010: =========")

# (1) Encrypting a plaintext using Mini-AES
from sage.crypto.block_cipher.miniaes import MiniAES
maes = MiniAES()
K = FiniteField(16, "x")
MS = MatrixSpace(K, 2, 2)

P = MS([K("x^3 + x"), K("x^2 + 1"), K("x^2 + x"), K("x^3 + x^2")]); ▸
  ▸print("(1) P:\n",P, sep="")

key = MS([K("x^3 + x^2"), K("x^3 + x"), K("x^3 + x^2 + x"), K("x^2 + x ▸
  ▸+ 1")]); print("key:\n",key, sep="")

C = maes.encrypt(P, key); print("C:\n",C, sep="")

# decryption process
plaintxt = maes.decrypt(C, key)
print(plaintxt == P)

# (2) Working directly with binary strings
maes = MiniAES()
bin = BinaryStrings()
key = bin.encoding("KE"); print("\n(2) key:\n",key, sep="")

P = bin.encoding("Encrypt this secret message!"); print("P:\n",P,sep▸
  ▸="")
C = maes(P, key, algorithm="encrypt"); print("C:\n",C,sep="")
plaintxt = maes(C, key, algorithm="decrypt")
print(plaintxt == P)

# 3) Or working with integers n such that 0 <= n <= 15:
maes = MiniAES()
P = [n for n in range(16)]; print("\n(3) P:\n",P, sep="")
key = [2, 3, 11, 0]; print("key:\n",key, sep="")

P = maes.integer_to_binary(P)
key = maes.integer_to_binary(key)
C = maes(P, key, algorithm="encrypt"); print("C:\n",C, sep="")
plaintxt = maes(C, key, algorithm="decrypt")
print(plaintxt == P)
```

19. See `https://mvngu.wordpress.com/2009/07/12/sage-4-1-released/`.
Further example code for Mini-AES can be found in [112].

Further details concerning cryptosystems within SageMath (e.g., about the Simplified Data Encryption Standard (SDES)) can be found in the thesis of Minh Van Nguyen [113].

1.12.2 Symmetric Ciphers for Educational Purposes

Compared to public-key ciphers based on mathematics, the structure of AES and most other modern symmetric ciphers (like DES, IDEA, or Present), is very complex and cannot be explained as easily as RSA.

So, simplified variants of modern symmetric ciphers were developed for educational purposes in order to allow beginners to perform encryption and decryption by hand and gain a better understanding of how the algorithms work in detail. These simplified variants also help to understand and apply the corresponding cryptanalysis methods.[20]

The most well-known of these variants are SDES[21] and Simplified-AES (S-AES)[22] by Ed Schaefer and his students [115], and Mini-AES (see Section 1.12.1):

- Edward F. Schaefer: *A Simplified Data Encryption Standard Algorithm* [116].
- Raphael Chung-Wei Phan: *Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students* [111].
- Raphael Chung-Wei Phan: *Impossible Differential Cryptanalysis of Mini-AES* [117].
- Mohammad A. Musa, Edward F. Schaefer, Stephen Wedig: *A Simplified AES Algorithm and Its Linear and Differential Cryptanalyses* [118].
- Nick Hoffman: *A Simplified Idea Algorithm* [119].
- S. Davod. Mansoori, H. Khaleghei Bizaki: *On the Vulnerability of Simplified AES Algorithm Against Linear Cryptanalysis* [120].

References

[1] International Association for Cryptologic Research (IACR), *IACR*, https://www.iacr.org/.

[2] *War of the Letters*, BBC documentary film (in German, *Krieg der Buchstaben*), 1994, https://www.youtube.com/watch?v=yfw1JLI8aWk.

[3] Nichols, R. K., *Classical Cryptography Course*, Volumes 1 and 2, Tech. Rep., 12 lessons, Aegean ParkPress 1996, https://www.cryptogram.org/resource-area/crypto-lessons-tutorials-lanaki/.

20. A very good starting point to learn cryptanalysis is the book from Mark Stamp [109]. Also good, but very high-level and concentrating on analyzing symmetric block ciphers only, is the article from Bruce Schneier [114].

    Several of the cipher challenges at MysteryTwister (`https://www.mysterytwister.org`) are also well suited for educational purposes.

21. If you double-click on the title of the icon of the SDES component in CT2 you can see a visualization of the SDES algorithm, showing how the bits of the given data flow through the whole algorithm. A corresponding screenshot: https://www.facebook.com/CrypTool2/photos/a.505204806238612.1073741827.243959195696509/597354423690316.

22. See the template: `CT2 Templates ▷ Cryptography ▷ Modern ▷ Symmetric ▷ S-AES`.

[4] *The RSA Secret Key Challenge*, RSA Labs (formerly RSA Security), https://web.archive.org/web/20170417095446/http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-laboratories-secret-key-challenge.htm.

[5] *DES Challenge*, RSA Labs (former RSA Security), https://web.archive.org/web/20061210141223/http://www.rsasecurity.com/rsalabs/node.asp?id=2108.

[6] *Press and Articles Related to Project RC5-64*, https://www.distributed.net/Pressroom_press-rc5-64.

[7] BSI, *Technical Guideline TR-02102-1, Cryptographic Mechanisms: Recommendations and Key Lengths (Version 2022-01)*, Tech. Rep. 2022, https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/TechGuidelines/TG02102/BSI-TR-02102-1.pdf.

[8] Schneier, B., *Applied Cryptography, Protocols, Algorithms, and Source Code in C*, Second Edition, Wiley, 1996.

[9] Esslinger, B., J. Schneider, and V. Simon. "Krypto + NSA = ? – Kryptografische Folgerungen aus der NSA-Affäre," in *KES Zeitschrift für Informationssicherheit*, March 2014, pp. 70–77, https://www.cryptool.org/assets/ctp/documents/krypto_nsa.pdf.

[10] Miller, A. R., *The Cryptographic Mathematics of Enigma*, Revised Edition, 2019, Center for Cryptologic History, National Security Agency, 1995, https://www.nsa.gov/portals/75/documents/about/cryptologic-heritage/historical-figures-publications/publications/wwii/CryptoMathEnigma_Miller.pdf.

[11] Ostwald, O., *Cryptographic Design Flaws of Early Enigma*, 2023, https://cryptocellar.org/enigma/files/enigma-design-flaws.pdf.

[12] International Conference on Cryptologic History (ICCH), https://www.cryptologichistory.org/.

[13] Kopal, N, "Solving Classical Ciphers with CrypTool 2," in *Proceedings of the 1st International Conference on Historical Cryptology*, 2018, pp. 29–38.

[14] Kopal, N, "Cryptanalysis of Homophonic Substitution Ciphers Using Simulated Annealing with Fixed Temperature," in *Proceedings of the 2nd International Conference on Historical Cryptology*, 2019, pp. 107–116.

[15] Lasry, G., N. Biermann, and S. Tomokiyo, "Deciphering Mary Stuart's Lost Letters from 1578–1584," *Cryptologia*, Vol. 47, No. 2, 2023, pp. 101–202.

[16] Lasry, G., B. Megyesi, and N. Kopal, "Deciphering Papal Ciphers from the 16th to the 18th Century," *Cryptologia*, Vol. 45, No. 6, 2021, pp. 479–540, https://www.tandfonline.com/doi/full/10.1080/01611194.2020.1755915.

[17] Dunin, E., et al., "How We Set New World Records in Breaking Playfair Ciphertexts," *Cryptologia*, Vol. 46, No. 4, 2022, pp. 302–322.

[18] Lasry, G., "Solving a 40-Letter Playfair Challenge with CrypTool 2," in *Proceedings of the 2nd International Conference on Historical Cryptology*, 2019, pp. 23–26.

[19] Lasry, G., "Deciphering German Diplomatic and Naval Attaché Messages from 1914–1915," in *Proceedings of the 1st International Conference on Historical Cryptology*, 2018, pp. 55–64.

[20] Lasry, G., et al., "Cryptanalysis of Chaocipher and Solution of Exhibit 6," *Cryptologia*, Vol. 40, No. 6, 2016, pp. 487–514.

[21] Lasry, G., N. Kopal, and A. Wacker, "Cryptanalysis of Columnar Transposition Cipher with Long Keys," *Cryptologia*, Vol. 40, No. 4, 2016, pp. 374–398.

[22] Lasry, G., N. Kopal, and A. Wacker, "Solving the Double Transposition Challenge with a Divide-and-Conquer Approach," *Cryptologia*, Vol. 38, No. 3, 2014, pp. 197–214.

[23] Lasry, G., et al., "Deciphering ADFGVX Messages from the Eastern Front of World War I," *Cryptologia*, Vol. 41, No. 2, 2017, pp. 101–136.

[24] Gillogly, J. J., "Ciphertext-Only Cryptanalysis of Enigma," *Cryptologia*, Vol. 19, No. 4, 1995, pp. 405–413.

[25] Lasry, G., N. Kopal, and A. Wacker, "Cryptanalysis of Enigma Double Indicators with Hill Climbing," *Cryptologia*, Vol. 43, No. 4, 2019, pp. 267–292.

[26] Ostwald, O., and F. Weierud, "Modern Breaking of Enigma Ciphertexts," *Cryptologia*, Vol. 41, No. 5, 2017, pp. 395–421.

[27] Lasry, G., "Solving a Tunny Challenge with Computerized 'Testery' Methods," in *Proceedings of the 3rd International Conference on Historical Cryptology*, 2020.

[28] Lasry, G., N. Kopal, and A. Wacker, "Automated Known-Plaintext Cryptanalysis of Short Hagelin M-209 Messages," *Cryptologia*, Vol. 40, No. 1, 2016, pp. 49–69.

[29] Lasry, G., N. Kopal, and A. Wacker, "Ciphertext-Only Cryptanalysis of Short Hagelin M-209 Ciphertexts," *Cryptologia*, Vol. 42, No. 6, 2018, pp. 485–513.

[30] Lasry, G., "A Practical Meet-in-the-Middle Attack on SIGABA," in *Proceedings of the 2nd International Conference on Historical Cryptology*, 2019, pp. 23–26.

[31] Lasry, G., "Cracking SIGABA in Less than 24 Hours on a Consumer PC," *Cryptologia*, Vol. 47, No. 1, 2023, pp. 1–37.

[32] Matsui, M., "Linear Cryptanalysis Method for DES Cipher," in *Advances in Cryptology—EUROCRYPT'93: Workshop on the Theory and Application of Cryptographic Techniques*, Lofthus, Norway, May 23–27, 1993, Springer, pp. 386–397.

[33] Junod, P., "On the Complexity of Matsui's Attack," in *Selected Areas in Cryptography: 8th Annual International Workshop, SAC 2001*, Toronto, Ontario, Canada, August 16–17, 2001, Springer, pp. 199–211.

[34] Merkle, R. C., and M. E. Hellman, "On the Security of Multiple Encryption," *Communications of the ACM*, Vol. 24, No. 7, 1981, pp. 465–467.

[35] Daemen, J., and V. Rijmen, *The Design of Rijndael*, Volume 2, Springer, 2002.

[36] Bogdanov, A., D. Khovratovich, and C. Rechberger, "Biclique Cryptanalysis of the Full AES," in *Advances in Cryptology–ASIACRYPT 2011: 17th International Conference on the Theory andApplication of Cryptology and Information Security*, Seoul, South Korea, December 4–8, 2011, Springer, pp. 344–371.

[37] Aoki, K., et al., "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms—Design and Analysis," in *Selected Areas in Cryptography: 7th Annual International Workshop, SAC 2000*, Waterloo, Ontario, Canada, August 14–15, 2000, Springer, pp. 39–56.

[38] Li, L., et al., "Meet-in-the-Middle Technique for Truncated Differential and Its Applications to CLEFIA and Camellia," in *Fast Software Encryption: 22nd International Workshop, FSE 2015*, Istanbul, Turkey, March 8–11, 2015, Revised Selected Papers, Springer, pp. 48–70.

[39] Matsui, M., "New Block Encryption Algorithm MISTY," in *Fast Software Encryption: 4th International Workshop, FSE'97*, Haifa, Israel, Springer, pp. 54–68.

[40] Bar-On, A., and N. Keller, "A 2^{70}Attack on the Full MISTY1," in *Advances in Cryptology–CRYPTO 2016: 36th Annual International Cryptology Conference*, Springer: Santa Barbara, California, August 14–18, 2016, pp. 435–456.

[41] Todo, Y., "Integral Cryptanalysis on Full MISTY1," *Journal of Cryptology*, Vol. 30, No. 3, 2017, pp. 920–959.

[42] ETSI (2014-10), *Universal Mobile Telecommunications System (UMTS); LTE; 3G Security; Specification of the 3GPP Confidentiality and Integrity Algorithms; Document 2: Kasumi Specification* (3GPP TS 35.202 version 12.0.0 Release 12), https://www.etsi.org/deliver/etsi_ts/135200_135299/135202/07.00.00_60/ts_135202v070000p.pdf.

[43] Dunkelman, O., N. Keller, and A. Shamir, "A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony," in *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference*, Santa Barbara, California, August 15–19, 2010, Springer, pp. 393–410.

[44] Hong, D., et al., "HIGHT: A New Block Cipher Suitable for Low-Resource Device," in *Cryptographic Hardware and Embedded Systems—CHES 2006: 8th International Workshop*, Yokohama, Japan, 2006, Springer, pp. 46–59.

[45] Hong, D., B. Koo, and D. Kwon, "Biclique Attack on the Full HIGHT," in *Information Security and Cryptology-ICISC 2011: 14th International Conference*, Seoul, Korea, November 30–December 2, 2011, Revised Selected Papers 14, Springer, pp. 365–374.

[46] Adams, C., *RFC2144: The CAST-128 Encryption Algorithm*, 1997, https://www.rfc-editor.org/rfc/r fc2144.

[47] Wang, S., T. Cui, and M. Wang, "Improved Differential Cryptanalysis of CAST-128 and CAST-256," in *Information Security and Cryptology: 12th International Conference, Inscrypt 2016*, Beijing, China, November 4–6, 2016, Springer, pp. 18–32.

[48] Lee, H. J., et al., *RFC4009: The SEED Encryption Algorithm*, 2005, https://www.rfc-editor.org/rfc/rfc4269.

[49] Sung, J., "Differential Cryptanalysis of Eight-Round SEED," *Information Processing Letters*, Vol. 111, No. 10, 2011, pp. 474–478.

[50] Bogdanov, A., et al., "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems-CHES 2007: 9th International Workshop*, Vienna, Austria, September 10–13, 2007, Springer, pp. 450–466.

[51] Blondeau, C., and K. Nyberg. "Links between Truncated Differential and Multidimensional Linear Properties of Block Ciphers and Underlying Attack Complexities," *Eurocrypt*, Vol. 14, 2014, pp. 165–182.

[52] Shirai, T., et al.,"The 128-Bit Blockcipher CLEFIA,"in *Fast Software Encryption*, *14th International Workshop, FSE 2007, LNCS 4593*, 2007, pp. 181–195. 2007, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.111.9703&rep=rep1&type=pdf#page=191.

[53] Hong, D., et al. "LEA: A 128-Bit Block Cipher for Fast Encryption on Common Processors," in *Information Security Applications: 14th International Workshop, WISA 2013*, Jeju Island, Korea, August 19–21, 2013, pp. 3–27.

[54] Dwivedi, A. D., and G. Srivastava, "Differential Cryptanalysis of Round-Reduced LEA," *IEEE Access*, Vol. 6, 2018, pp. 79105–79113.

[55] Diffie, W., and G. Ledin, "SMS4 Encryption Algorithm for Wireless Networks," *Cryptology ePrint Archive*, 2008.

[56] Liu, Y., et al., "New Linear Cryptanalysis of Chinese Commercial Block Cipher Standard SM4," *Security and Communication Networks*, 2017.

[57] Zabotin, I. A., G. P. Glazkov, and V. B. Isaeva, "Cryptographic Protection for Information Processing Systems, Government Standard of the USSR, GOST 28147-89," *Government Committee of the USSR for Standards*, 1989.

[58] Courtois, N. T., "An Improved Differential Attack on Full GOST," in *The New Codebreakers: Essays Dedicated to David Kahn on the Occasion of His 85th Birthday* (P. Y. A. Ryan, D. Naccache, and J.-J. Quisquater, eds.), Berlin: Springer, 2016, pp. 282–303.

[59] Federal Agency on Technical Regulation and Metrology (GOST), *GOST R 34.12-2015: Block Cipher "Kuznyechik,"* https://www.rfc-editor.org/rfc/rfc7801.

[60] AlTawy, R., and A. M Youssef, "A Meet in the Middle Attack on Reduced Round Kuznyechik," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. 98, No. 10, 2015, pp. 2194–2198.

[61] Bruwer, F. J., W. Smit, and G. J. Kuhn, *Microchips and Remote Control Devices Comprising Same*, U.S. Patent 5,517,187, May 1996.

[62] Indesteege, S., et al., "A Practical Attack on KeeLoq," in *Advances in Cryptology–EUROCRYPT 2008: 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Istanbul, Turkey, April 13–17, 2008, Springer, 2008, pp. 1–18.

[63] Beaulieu, R., et al., "The SIMON and SPECK Lightweight Block Ciphers," in *Proceedings of the 52nd Annual Design Automation Conference*, 2015, pp. 1–6.

[64] Chen, H., and X. Wang, "Improved Linear Hull Attack on Round-Reduced Simon with Dynamic Key-Guessing Techniques," in *Fast Software Encryption: 23rd International Conference, FSE 2016*, Bochum, Germany, March 20–23, 2016, Springer, pp. 428–449.

[65] Song, L., Z. Huang, and Q. Yang, "Automatic Differential Analysis of ARX Block Ciphers with Application to SPECK and LEA," in *Information Security and Privacy: 21st Australasian Conference, ACISP 2016*, Melbourne, Victoria, Australia, July 4–6, 2016, Springer, pp. 379–394.

[66] Miyaguchi, S., "The FEAL Cipher Family," in *Advances in Cryptology-CRYPTO'90: Proceedings 10*, Springer. 1991, pp. 628–638.

[67] Biham, E., and A. Shamir, "Differential Cryptanalysis of Feal and N-Hash," in *Advances in Cryptology—EUROCRYPT'91: Workshop on the Theory and Application of Cryptographic Techniques, Proceedings 10*, Brighton, UK, April 8–11, 1991, Springer, pp. 1–16.

[68] Schneier, B., et al., "Twofish: A 128-Bit Block Cipher," *NIST AES Proposal*, Vol. 15, No. 1, 1998, pp. 23–91.

[69] Lucks, S., "The Saturation Attack—A Bait for Twofish," in *Fast Software Encryption: 8th International Workshop, FSE 2001*, Yokohama, Japan, Springer, pp. 1–15.

[70] Vanhoef, M., and F. Piessens, "All Your Biases Belong to Us: Breaking RC4 in WPA-TKIP and TLS" in *24th USENIXSecurity Symposium (USENIXSecurity 15)*, 2015, pp. 97–112.

[71] Briceno, M., "A Pedagogical Implementation of A5/1," 1995, http://www. scard.org.

[72] Barkan, E., E. Biham, and N. Keller, "Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication," in *Advances in Cryptology-CRYPTO 2003: 23rd Annual International Cryptology Conference, Proceedings 23*, Santa Barbara, California, August 17–21, 2003, Springer, pp. 600–616.

[73] Briceno, M., I. Goldberg, and D. Wagner, "A Pedagogical Implementation of the GSM A5/1 and A5/2 'Voice Privacy' Encryption Algorithms," 1999, *http://www.scard.org, mirror at http://cryptome.org/gsm-a512.htm* 26.

[74] Bernstein, D. J., et al., "ChaCha, a Variant of Salsa20," *Workshop Record of SASC*, Vol. 8, Citeseer, 2008, pp. 3–5.

[75] Aumasson, J.-P., et al., "New Features of Latin Dances: Analysis of Salsa, ChaCha, and Rumba," in *Fast Software Encryption: 15th International Workshop, FSE 2008, Revised Selected Papers 15*, Lausanne, Switzerland, February 10–13, 2008, Springer, pp. 470–488.

[76] Bernstein, D. J., "The Salsa20 Family of Stream Ciphers," *New Stream Cipher Designs: the eSTREAM Finalists*, 2008, pp. 84–97.

[77] Nohl, K., "Mifare, Little Security, Despite Obscurity," in *The 24th Congress of the Chaos Computer Club in Berlin*, December 2007.

[78] Courtois, N. T., K. Nohl, and S. O'Neil, "Algebraic Attacks on the Crypto-1 Stream Cipher in MiFare Classic and Oyster Cards," *Cryptology ePrint Archive*, 2008, https://eprint.iacr.org/2008/166.pdf.

[79] Hell, M., et al., "A Stream Cipher Proposal: Grain-128," in *2006 IEEE International Symposium on Information Theory*, IEEE, 2006, pp. 1614–1618.

[80] Fu, X., et al., "Determining the Nonexistent Terms of Non-linear Multivariate Polynomials: How to Break Grain-128 More Efficiently," *IACR Cryptol ePrint Archive*, 2017, p. 412.

[81] De Canniere, C., and B. Preneel, "Trivium," *New Stream Cipher Designs: The eSTREAM Finalists*, 2008, pp. 244–266.

[82] Fouque, P.-A., and T. Vannet, "Improving Key Recovery to 784 and 799 Rounds of Trivium Using Optimized Cube Attacks," in *Fast Software Encryption: 20th International Workshop, FSE 2013*, Singapore, March 11–13, 2013, Springer, pp. 502–517.

[83] Boesgaard, M., et al. "Rabbit: A New High-Performance Stream Cipher," in *Fast Software Encryption: 10th International Workshop, FSE 2003, Revised Papers 10*, Lund, Sweden, February 24–26, 2003, Springer, pp. 307–329.

[84] Watanabe, D., et al. "Update On Enocoro Stream Cipher," in *2010 International Symposium On Information Theory & Its Applications*, IEEE, 2010, pp. 778–783.

[85] Shibayama, N., and Y. Igarashi, "A New Higher Order Differential of Enocoro-128v2," in *2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW)*, IEEE, 2021, pp. 379–384.

[86] Ekdahl, P., and T. Johansson, "A New Version of the Stream Cipher SNOW," in *Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002*, St. John's, Newfoundland, Canada, August 15–16, 2002, Springer, pp. 47–61.

[87] Funabiki, Y., et al., "Several MILP-Aided Attacks Against SNOW 2.0," in *Cryptology and Network Security: 17th International Conference, CANS 2018, Proceedings*, Naples, Italy, September 30–October 3, 2018, Springer, pp. 394–413.

[88] Watanabe, D., etal., "A New Key Stream Generator MUGI," in *Fast Software Encryption: 9th International Workshop, FSE 2002*, Leuven, Belgium, February 4–6, 2002, Springer, pp. 179–194.

[89] Watanabe, D., et al., *MUGI Psuedorandom Number Generator, Self Evaluation*, Tech. Rep., Hitachi Ltd., 2001, http://www.sdl.hitachi.co.jp/crypto/mugi/index-e.html.

[90] ETSI/SAGESpecification, *Specification of the 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3, Document 2: ZUC Specification, Version: 1.6*, 2011.

[91] Rivest, R. L., A. Shamir, and L. Adleman,"A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, 1978, pp. 120–126.

[92] Boudot, F., et al., "Comparing the Difficulty of Factorization and Discrete Logarithm: A 240-Digit Experiment," in *Advances in Cryptology–CRYPTO 2020: 40th Annual International Cryptology Conference, CRYPTO 2020, Proceedings, Part II 40*, Santa Barbara, CA, August 17–21, 2020, Springer, pp. 62–91.

[93] Boudot, F., et al., "The State of the Art in Integer Factoring and Breaking Public-Key Cryptography," *IEEE Security & Privacy*, Vol. 20, No. 2, 2022, pp. 80–86.

[94] ElGamal, T., "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," *IEEE Transactions on Information Theory*, Vol. 31, No. 4, 1985, pp. 469–472.

[95] Hoffstein, J., et al., "Practical Lattice-Based Cryptography: NTRUEncrypt and NTRUSign," in *The LLL Algorithm: Survey and Applications* (P. O. Nguyen, and V. Vallee, eds.), Berlin: Springer-Verlag, 2009, pp. 349–390.

[96] Howgrave-Graham, N., "A Hybrid Lattice-Reduction and Meet-in-the-Middle Attack against NTRU," in *Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Proceedings 27*, Santa Barbara, CA, August 19–23, 2007, Springer, pp. 150–169.

[97] Fu, X., et al., "A key-Recovery Attack on 855-Round Trivium," in *Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Proceedings, Part II 38*, Santa Barbara, CA, August 19–23, 2018, Springer, pp. 160–184.

[98] Hao, Y., et al., "Observations on the Dynamic Cube Attack of 855-Round TRIVIUM from Crypto'18," *Cryptology ePrint Archive*, 2018.

[99] PassMark Software, *CPU Benchmarks. Intel Xeon Gold 6130, 2.10GHz*, https://www.cpubenchmark.net/cpu.php?cpu=Intel+Xeon+Gold+6130+%40+2.10GHz&id=3126.

[100] Hoffstein, J., et al., "Choosing Parameters for NTRUEncrypt," in *Topics in Cryptology–CT-RSA 2017: The Cryptographers' Track at the RSA Conference 2017*, Springer: San Francisco, CA, February 14–17, 2017, pp. 3–18.

[101] Bellare, M., and P. Rogaway, *Introduction to Modern Cryptography*, University of California Davis Department of Computer Science, 2005, p. 283.

[102] Goldreich, O., *Foundations of Cryptography: Volume 2, Basic Applications*, Cambridge, UK: Cambridge University Press, 2009.

[103] Oppliger, R., *Cryptography 101: From Theory to Practice*, Norwood, MA: Artech House, 2021, https://rolf.esecurity.ch/?page_id=465.

[104] Menezes, A. J., P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, Fifth Edition, Boca Raton, FL: CRC Press, 2001, https://cacr.uwaterloo.ca/hac/.

[105] Singh, S., *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*, New York: Anchor Books, 1999.

[106] Paar, C., and J. Pelzl, *Understanding Cryptography–A Textbook for Students and Practioners*, Berlin: Springer Verlag, 2009, https://www.crypto-textbook.com/.

[107] Wong, D., *Real-World Cryptography*, Shelter Island, NY: Manning Publications, 2021, https://www.manning.com/books/real-world-cryptography.

[108] Aumasson, J.-P., *Serious Cryptography: A Practical Introduction to Modern Encryption*, San Francisco, CA: NoStarch Press, 2017, https://books.google.de/books?id=hLcrD-wAAQBAJ.

[109] Stamp, M., and R. M. Low, *Applied Cryptanalysis: Breaking Ciphers in the Real World*, Hoboken, NJ: Wiley-IEEE Press, 2007, https://www.cs.sjsu.edu/~stamp/crypto/.

[110] Stinson, D. R., *Cryptography: Theory and Practice*, Third Edition, Boca Raton, FL: Chapman & Hall/CRC Press, 2006.

[111] Raphael Chung-Wei Phan. "Mini Advanced Encryption Standard (Mini-AES): A Testbed for Cryptanalysis Students," *Cryptologia* 26.4 (2002), pp. 283–306.

[112] Nguyen, M. V., *Number Theory and the RSA Public Key Cryptosystem – An Introductory Tutorial on Using SageMath to Study Elementary Number Theory and Public Key Cryptography*, 2009, https://faculty.washington.edu/moishe/hanoiex/Number%20Theory%20Applications/numtheory-crypto.pdf.

[113] Nguyen, M.V., *Exploring Cryptography Using the Sage Computer Algebra System*, Bachelor of Science Thesis, Victoria University, Australia, 2009, www.sagemath.org/files /thesis/nguyen-thesis-2009.pdf url2: https://www.sagemath.org/library-publications.html.

[114] Schneier, B., "A Self-Study Course in Block-Cipher Cryptanalysis," *Cryptologia*, Vol. 24, 2000, pp. 18–34, https://www.schneier.com/wp-content/uploads/2015/01/paper-self-study.pdf.

[115] Schaefer, E. F., *Cryptography Research: Devising a Better Way to Teach and Learn the Advanced Encryption Standard*, Santa Clara University, 2011, https://web.archive .org/web/20110829213229/http://www.scu.edu/cas/research/cryptography.cfm.

[116] Schaefer, E. F., "A Simplified Data Encryption Standard Algorithm," *Cryptologia*, Vol. 20, No. 1, 1996, pp. 77–84.

[117] Chung-Wei Phan, R., "Impossible Differential Cryptanalysis of Mini-AES," *Cryptologia*, Vol. 27, No. 4, 2003, pp. 361–374, https://www.tandfonline.com/doi/abs/10.1080/0161-110391891964.

[118] Musa, M. A., E. F. Schaefer, and S. Wedig. "A Simplified AES Algorithm and Its Linear and Differential Cryptanalyses," *Cryptologia*, Vol. 17, No. 2, April 2003, pp. 148–177, https://www.rose-hulman.edu/~holden/Preprints/s-aes.pdf.

[119] Hoffman, N., "A Simplified IDEA Algorithm," 2006, https://www.nku.edu/~christensen/simplified%20IDEA%20algorithm.pdf.

[120] Davod Mansoori, S., and H. Khaleghei Bizaki, "On the Vulnerability of Simplified AES Algorithm Against Linear Cryptanalysis," *IJCSNS International Journal of Computer Science and Network Security*, Vol. 7, No. 7, 2007, pp. 257–263, http://paper.ijcsns.org/07_book/200707/20070735.pdf.