Introduction to Cryptanalysis: Attacking Stream Ciphers

Roel Verdult

Institute for Computing and Information Sciences
Radboud University Nijmegen, The Netherlands.
rverdult@cs.ru.nl

Abstract. This article contains an elementary introduction to the cryptanalysis of stream ciphers. Initially, a few historical examples are given to explain the core aspects of cryptography and the various properties of stream ciphers. We define the meaning of cryptographic strength and show how to identify weaknesses in a cryptosystem. Then, we show how these cryptographic weaknesses can be exploited and attacked by a number of cryptanalytic techniques. The academic literature includes many articles there use complex mathematical notations to represent trivial cryptanalytic techniques. Contrarily, this paper tries to put forward the most commonly used cryptanalytic techniques by using only simplistic and comprehensible examples. The addressed techniques are used in several scientific articles to mount practical attacks on real cryptosystems.

1 Introduction

This manuscript is an excerpt of the third chapter in the doctoral dissertation of Roel Verdult [120]. The first chapters of this book contains a theoretical background with an elementary description of security protocols, cryptographic primitives and cryptanalytic attacks.

1.1 Cryptographic strength

The strength of a cryptographic algorithm is expressed in the total amount of computations an adversary needs to perform to recover the secret key. It is often referred to as the computational complexity of the cipher.

For a perfectly secure cipher, the computational complexity is the same as the key space, sometimes referred to as the *entropy* of the key. The key space refers to the set of all possible keys and represents the total number of combinations using all secret key bits. The size of the key (key-size) is the amount of bits n which define the size of the complete key space 2^n.

The naive method to recover the secret key is to try simply all combinations. Such methodology is often referred to as an exhaustive search or a brute-force attack. It would most likely require almost 2^n computations to determine the secret key. To be precise, on average an adversary finds the secret key halfway. When lucky, the key can be determined at an early stage, but it might just as well be one of the last tries. Interestingly, this property already shows that the number of computations required to determine the key is by definition lower than the actual key space.

A cipher that is perfect should be the base for a secure cryptosystem. In practice however, most ciphers are (slightly) weaker than the full entropy of their secret key. With clever optimizations it is often possible to find the secret key by doing far fewer computations than the actual entropy would require. This is called the actual attack complexity of a cipher.

1.2 Stream ciphers

A stream cipher performs an encryption which is similar to the One-time Pad (OTP) encryption technique. It produces a large chunk of secret, random looking data and combines it with the plaintext to produce ciphertext. Without the exact same data chunk, the plaintext cannot be uncovered from the ciphertext. The random data represents a stream of bits which is derived from the secret key and is commonly referred to as *keystream*. A stream cipher contains some persistent memory, called the internal cipher state, which is initialized by the secret key and propagates to a successor state after each encryption step. The output of a strong stream cipher is comparable to (and should be indistinguishable from) a contiguous bit stream produced by a Pseudo Random Number Generator (PRNG).

To be more precise, we embed the remarks made in [108] and define a stream cipher as follows: an encryption function which operates on individual plaintext digits (usually bits) where its internal state is initialized with the secret key prior to encryption. The keystream varies, depending on the initialized secret key and the moment of encryption with respect to the propagation of the internal state. Encryption of plaintext and decryption of ciphertext are both performed by the exclusive-or (XOR) operation, which is denoted by a $\oplus$ symbol and represents a bit-wise addition modulo two. A useful mathematical property of this operator is that it can be inverted. Therefore, it can be applied for encryption as well as decryption.

There are two types of stream ciphers, synchronous and self-synchronizing. In a synchronous stream cipher, the encryption bits are computed independently from the plaintext. Such ciphers are useful in situations when a communication channel is more prone to error. It might happen that just one badly transmitted bit is wrongly interpreted, which however does not directly affect the other bits that were transferred in a correct manner. Therefore, stream ciphers are very useful to encrypt streaming media where the speed of data-traffic is more important than the completeness and integrity of the data. Contrarily, a self-synchronizing stream cipher computes the successor of its internal state with a function over the previous state and the ciphertext. The internal state diverts from its original propagation path when a transmission error occurs. This dissertation focusses itself on the most widely used and best studied of the two, the synchronous stream ciphers. Therefore, a general reference to a stream cipher refers to a synchronous stream cipher.

An important objective of a stream cipher is to avoid a direct relation between the input (secret key) and output (keystream) of the cipher. Because the entropy of a stream cipher is limited to the size of the internal state, the produced keystream will eventually repeat itself. Note, that this is not a property of a regular One-time Pad (OTP).

Pure One-time Pad encryption can provide *perfect secrecy* when the keystream is truly random and uniquely generated for each message that is transmitted [109]. In such a setting, the keystream should consist of a unique bit string that contains uniformly distributed random bits. However, in practice it is difficult to generate truly random data. Alternative methods, like using the complete contents of a random book, drastically limit the number of possible keystreams. Moreover, reuse of the same keystream is very insecure. With access to previous plaintext and ciphertext, an adversary would be able to extract the keystream. If the same keystream is used in a second transmission, the adversary can use the recovered keystream and reveal the second plaintext. Exactly for this particular reason, the encryption technique is called One-time Pad. The keystream that represents the secret key should only be used once.

Keystream can be seen as an unique set of bits which must be as long as the plaintext. However, continuing distribution of fresh keystream for long data sequences is undesirable. With an increase in electronic transmissions of large transcripts in the 20th century, the need for alternative solutions grew. In response several stream cipher encryption techniques were introduced. For instance, in the 1930s stream ciphers were mainly used in the form of physical rotor machines which operated mostly mechanically. A well-known example of such a rotor machine is the Enigma, which is illustrated in Figure 1.1. A few decades later, the introduction of large scale computer networks increased the demand for more hard- and software oriented stream ciphers which supported automated communication.

Fig. 1.1: Enigma machine[1]

[1] http://www.securityninja.co.uk/wp-content/uploads/2010/09/4-rotor-enigma-open-copy-BW.jpg

One of the most popular techniques in the sixties and seventies was the non-linear binary sequence stream cipher [52, 71, 72, 85, 105]. It produces a binary keystream that allows a regular One-time Pad encryption without the requirement of a very large secret key. A typical cryptosystem based on a non-linear stream cipher is illustrated in Figure 1.2. This type of ciphers was very popular because of their small hardware footprint.

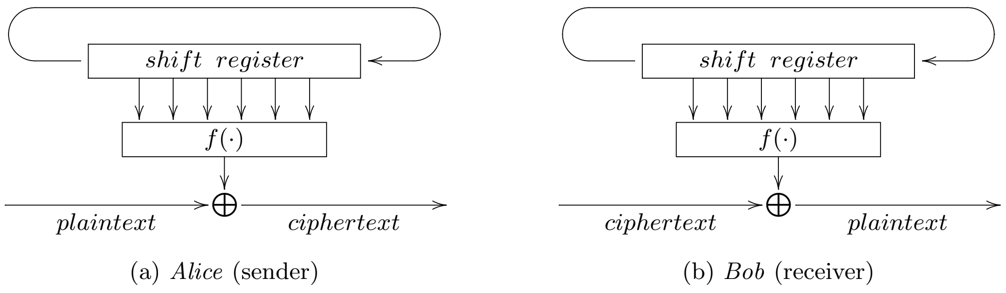


Fig. 1.2: Typical non-linear stream cipher system

The cryptographic algorithm illustrated in Figure 1.2 embeds a rotating shift register, which represents the internal state of the cipher. After the computation of a new keystream bit, the successor function updates the internal state by a linear function to preserve as much entropy to the cipher. Then, the output component applies a non-linear filter function $f(\cdot)$ to compute the next keystream bit. The keystream bits are used by the sender (Figure 1.2a) to encrypt the plaintext bits by combining both bit strings with the exclusive-or (XOR) operation. The resulting ciphertext is transmitted over an insecure channel. The receiver (Figure 1.2b) performs the exact same computations and applies another XOR operation, this time on the ciphertext bits in combination with the keystream bits. The keystream bits, already embedded in the ciphertext, are cancelled out and the original plaintext is revealed to the receiver.

The sender and the receiver use the non-linear stream cipher to compute exactly the same keystream. Then, the sender combines the keystream with the plaintext to produce the ciphertext by using the XOR operation. The receiver performs the same technique on the ciphertext together with the keystream to reconstruct and reveal the plaintext.

In most cryptosystems it is important to link multiple encrypted messages in one cryptographic session, this is called chaining of encryption. Stream ciphers inherently provide this feature since their ciphertext is produced incrementally. It uses the previous internal state and a successor function to step forward.

Besides these historical stream cipher designs there are several new proposals in the literature [4, 47, 57, 76]. Despite their advantages in flexibility and speed, stream ciphers are currently scarcely used in secure systems that provide strong cryptographic security. Typical stream cipher attacks aim to separate the plaintext from the encryption bits. For instance, a malleability attack exploits a general and unavoidable weakness in traditional stream ciphers where the keystream is generated independently from the plaintext. Small alterations (bit-flips) to the ciphertext might be sufficient to perform the attack without actually recovering the secret key. More details are given in Section 2.1.

The security that is provided by the underlying building blocks of stream ciphers are well-studied. However, the security implications of these separate components may not hold when they are combined and used together in one cryptographic algorithm. Instead, the comprehensive security implications of block ciphers are better understood [19, 25].

2 Cryptographic attacks

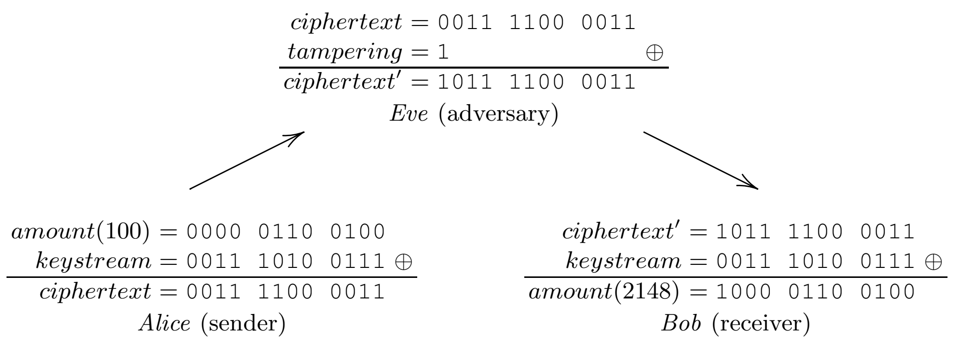
This section introduces seven fundamental cryptanalytic techniques which are used in cryptographic attacks, also referred to as cryptanalysis. There are many more advanced and complex cryptographic attack methodologies and techniques proposed in the literature [18, 22, 24, 26, 44, 45, 54, 84, 125]. However, to maintain readability, only very rudimentary versions of the fundamental techniques are introduced. The type of cryptosystem that is mainly used to demonstrate the attack techniques is a variant of the non-linear stream cipher system, introduced in Section 1.2.

To mount a cryptographic attack, it sometimes requires significant computational effort to recover the secret key. The computing power that is required reflects the attack complexity, see Section 1.1. It is possible to generalize the computations that are required for a cryptographic attack in such a way that they can be (partially) pre-computed. Such technique is commonly referred to as Time-Memory Trade-Off (TMTO). The general idea is to split a cryptographic attack into two phases, a pre-computation phase (off-line) and active attack phase (on-line). For more information on this topic, please refer to the TMTO methods and efficient search techniques proposed in the literature [2, 10–12, 20, 21, 27, 61, 77, 80, 102, 115, 117].

2.1 Malleability attack

The specifics of a synchronous stream cipher that produces a non-linear binary sequence are explained in Section 1.2. It shows an illustration of a typical cryptosystem that uses such a cipher. The generated binary sequence serves as the keystream and is combined with the plaintext by applying the exclusive-or (XOR) operator. Such a cryptosystem could in principle provide a secure channel which protects the confidentiality of the data transmission. However, without further protection, the integrity of the data is not guaranteed. Additional countermeasures, such as a Message Authentication Code (MAC), can protect the authenticity of the data. Without supplementary cryptographic techniques a stream cipher system is vulnerable to a malleability attack.

During a malleability attack the ciphertext is transformed in such a way that it still decrypts to bona fide plaintext, yet satisfies the attacker's purpose. Note, that the goal of a malleability attack is not to recover the secret key. In fact, it tries to undermine the security of the cryptosystem without having any knowledge of the secret key. Figure 2.1 demonstrates a data transmission tampering of a banking application that is vulnerable to a malleability attack. A fairly small money transfer is altered by a single bit-flip and suddenly represents a very large money transfer. Even if the keystream is produced by an extremely secure stream cipher, other components of the cryptosystem might still be vulnerable. It is the strength of the entire cryptosystem that defines the actual security.



ciphertext = 0011 1100 0011
tampering = 1 ⊕
ciphertext′ = 1011 1100 0011
Eve (adversary)
amount(100) = 0000 0110 0100
keystream = 0011 1010 0111 ⊕
ciphertext = 0011 1100 0011
Alice (sender)
ciphertext′ = 1011 1100 0011
keystream = 0011 1010 0111 ⊕
amount(2148) = 1000 0110 0100
Bob (receiver)

Fig. 2.1: Malleability attack alters the value of a money transfer

Although a malleability attack appears to be powerful, it is not that straightforward to mount. The adversary needs to know exactly where and when the amount is transmitted to be able to tamper with it. A bit-flip at an incorrect position is meaningless and will most likely result in a corrupted transmission. Knowledge about the keystream allows an adversary to prepare more carefully crafted tampering. However, this involves reuse of the keystream, a situation which a stream cipher should avoid at all costs. As Section 1.2 states, it is important that the binary sequence is used only once. To enforce this, the internal state should be initialized with a unique value each time the stream cipher is used. For instance, such a value can be derived from a secret key in combination with fresh random challenges.

A secure exchange of fresh random challenges and a protected Initialization Vector (IV) of the cipher is not a trivial task [15]. In fact, several proprietary stream ciphers allow an adversary to influence the initialization of the internal state in such a way that exactly the same keystream is produced as it was generated in a previous session. A comprehensive, yet practical, example of a stream cipher malleability attack is given in [48]. It describes the first practical attack on the MIFARE cryptosystem. The memory contents of a MIFARE Classic smartcard can be revealed even without any knowledge of the secret key. The MIFARE cryptosystem is further analysed and a number of attacks appeared in the literature [34, 37, 41, 62, 62, 64, 67, 81, 95, 100, 101, 118, 119, 121, 122, 126].

2.2 Divide-and-conquer attack

A very powerful way to reduce the complexity of a computation is to divide one big problem into two separate small problems, often referred to as divide-and-conquer. It is used in various fields within computer science to optimize the solving of hard computational problems [9, 13, 14, 56]. Its main purpose to reduce computational complexity and generic applicability makes this technique ideally suitable to attack certain types of cryptosystems [46].

As mentioned in Section 1.1, a cryptographically secure cipher with a secret key of significant length, for instance 80 bits, takes a lot of time and resources to solve. Dividing the exhaustive search to find the secret key for such a cryptosystem would not reduce the complexity itself. It only allows the adversary to parallelize two searches, each of exactly half the complexity of the main problem 2^{79}. However, when the adversary finds a way to divide the big problem into two smaller problems, the complexity is significantly reduced.

The cipher schematic, illustrated in Figure 2.2, is used to demonstrate how to mount a divide-and-conquer an attack. The 8-bit secret key $k = k_0 \dots k_7 \in \mathbb{F}_2^8$ is loaded into the internal state during initialization. Each encryption round the internal state rotates one step to the left after producing a keystream bit as output. The key space of the cipher that consists of only $2^8 = 256$ keys and keystream repeats itself after eight rotations. Therefore, this algorithm should be considered extremely insecure. However, its simplicity suits its purpose to demonstrate a divide-and-conquer attack.

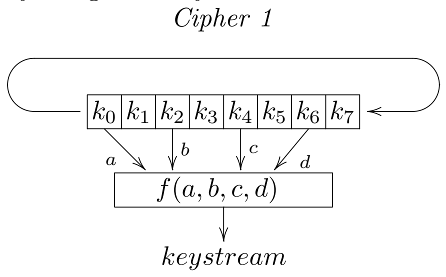
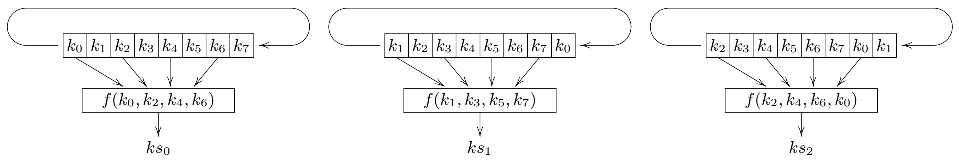


Fig. 2.2: Divide-and-conquer cipher

The eight squares in Figure 2.2 represent the rotate register of the internal state. It is loaded with the ones and zeros of the secret key during initialization. The $f(\cdot)$ component is a non-linear function which uses 4 input bits (arrows from above) and produces 1 output bit, which is referred to as keystream ks. The keystream bits are defined by ks_i where i represents the number of performed cipher rotations.

Consider the first three encryption rounds as illustrated in Figure 2.3. The input to the $f(\cdot)$ function is different for the first three keystream bits ks_0, ks_1 and ks_2. However, the secret key bits required to produce ks_0 and ks_2 are the same, although they differ in order. To be precise, the first keystream bit is given by $ks_0 = f(k_0, k_2, k_4, k_6)$ and the third keystream bit by $ks_2 = f(k_2, k_4, k_6, k_0)$.

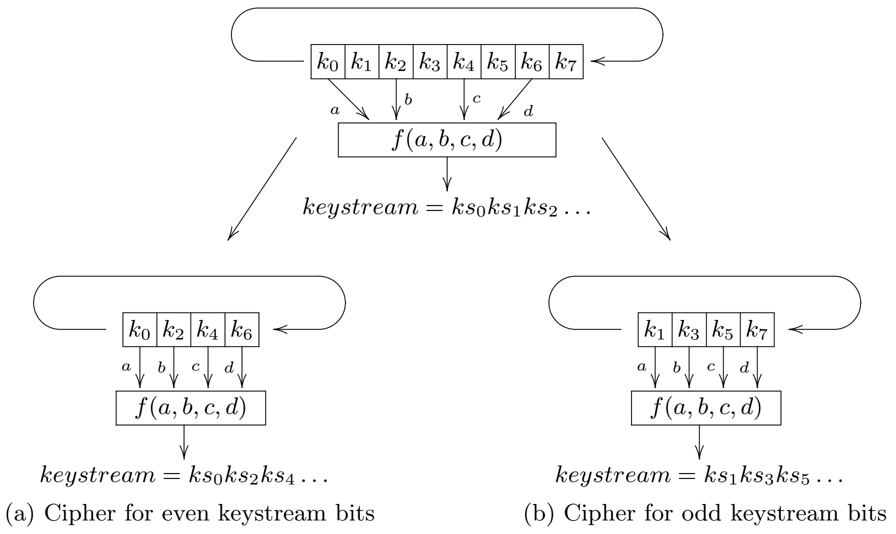


k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7
$f(k_0, k_2, k_4, k_6)$
ks_0
k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_0
$f(k_1, k_3, k_5, k_7)$
ks_1
k_2 k_3 k_4 k_5 k_6 k_7 k_0 k_1
$f(k_2, k_4, k_6, k_0)$
ks_2

Fig. 2.3: Computation of the first three encryption rounds and the derivation of ks_0, ks_1 and ks_2

The odd and even patterns in the keystream are properties of this particular divide-and-conquer cipher. Only the even key bits k_0, k_2, k_4 and k_6 are required to compute the even keystream bits ks_x where $x \in 2\mathbb{N}$ defines the even positions, while only the odd key bits k_1, k_3, k_5 and k_7 are used to compute the odd keystream bits ks_y where $y \in 2\mathbb{N} + 1$ defines the odd positions.

Cipher 1 was designed as one big algorithm, but it can easily be divided into smaller algorithms. Figure 2.4 demonstrates how to divide the cipher, both having a key-size of exactly half compared to the original cipher.



k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7
a b c d
$f(a, b, c, d)$
$keystream = ks_0 ks_1 ks_2 \dots$
k_0 k_2 k_4 k_6
a b c d
$f(a, b, c, d)$
$keystream = ks_0 ks_2 ks_4 \dots$
(a) Cipher for even keystream bits
k_1 k_3 k_5 k_7
a b c d
$f(a, b, c, d)$
$keystream = ks_1 ks_3 ks_5 \dots$
(b) Cipher for odd keystream bits

Fig. 2.4: Divide-and-conquer attack by dividing odd and even keystream bits

With only half of the internal cipher state, the computational complexity for the odd and even cipher drops to 2^4. Consequently, the total complexity for both together is reduced to only $2 \times 2^4 = 2^5 = 32$, which is considerably less than the initial computational complexity of $2^8 = 256$. Note, that a divide-and-conquer attack does not just halve the entropy of the key space. Such division would only yield a relative small complexity reduction of $\frac{2^8}{2} = 2^7 = 128$. It is far more powerful to separate the cipher into two smaller algorithms which rely on two independent secret keys, where the two keys represent both halves of the original secret key.

2.3 Correlation attack

The cipher weakness that allows a correlation attack is a statically biased encryption output that is highly-influenced by certain internal state bits which are used as input. Therefore, a guess for these input bits would most likely directly influence the output bits. With the use of statistical analysis the adversary can learn information about these internal state bits.

The advantage of a correlation attack is that the adversary can significantly shrink the set of likely secret key candidates. The stronger the bias, the more information is leaked when analyzing the input and output relations. Stream ciphers in particular are often susceptible to correlation attacks. There have been numerous proposals in the literature to define fast and optimized stream cipher correlation attacks [3, 28, 32, 33, 35, 39, 82, 83, 93, 94, 97, 104].

In the case of a stream cipher, a correlation attack technique is often applicable on multiple sequential encryption outputs. After a successor state is reached within a stream cipher, several previously chosen bits might overlap with the input to the next encryption. Furthermore, the bias is verifiable on every produced output, which allows an adversary to correlate previous and successor states. The combination of multiple biased encryption outputs leads to more information of the complete internal state. The set with the most probable candidates is likely to contain the internal state that is derived from the correct secret key.

To explain the basic correlation attack [110, 111], a vulnerable stream cipher is introduced in Figure 2.5. *Cipher 2* is a simple rotating stream cipher with an output component that relies on the non-linear filter function $f(\cdot)$. To increase readability the function definition is also given in the corresponding component of the figure.

Cipher 2

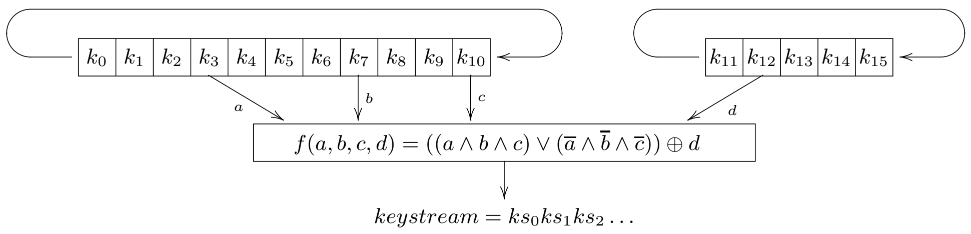

k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}
k_{11} k_{12} k_{13} k_{14} k_{15}
a
b
c
d
$f(a,b,c,d) = ((a \wedge b \wedge c) \vee (\overline{a} \wedge \overline{b} \wedge \overline{c})) \oplus d$
$keystream = ks_0 ks_1 ks_2 \ldots$

Fig. 2.5: Stream cipher with correlation weakness, initialized by secret key k

The cipher illustrated in Figure 2.5 is similar to the cipher introduced in Section 2.2 in terms of computing its successor and the way it produces encryption bits. However, it consists of two separate rotating registers and has a larger internal state that is initialized by 16 secret key bits. Furthermore, it does not suffer from the same weakness that allows separation between odd and even bits output. Instead, it employs a non-linear filter function which generates output that is statistically biased in correlation to the input. The function $f(\cdot)$ is given by Definition 1.

Definition 1. *Define the statistically biased non-linear filter function* $f\colon \mathbb{F}_2^4 \to \mathbb{F}_2$ *as*

$$f(a,b,c,d) = ((a \wedge b \wedge c) \vee (\overline{a} \wedge \overline{b} \wedge \overline{c})) \oplus d$$

The filter function takes four input bits (a, b, c and d) and produces one output bit by evaluating the specified non-linear equation. The boolean table which represents the input-ouput relation of $f(\cdot)$ is illustrated in Figure 2.6.

$abcd$	$f(a,b,c,d)$
0000	1
0001	0
0010	0
0011	1

$abcd$	$f(a,b,c,d)$
0100	0
0101	1
0110	0
0111	1

$abcd$	$f(a,b,c,d)$
1000	0
1001	1
1010	0
1011	1

$abcd$	$f(a,b,c,d)$
1100	0
1101	1
1110	1
1111	0

Fig. 2.6: Boolean input-ouput table that corresponds to Definition 1

The bias of the function is easily determined by looking at its input-output relation showed in Figure 2.6. Although $f(\cdot)$ produces a balanced output, which means that half of the produced output bits are zero and half of them are one, input bit d has significantly more influence on the output than the input bits a, b and c. Note the four marked rows in the input-output table illustrated in Figure 2.6 which point out where the input bits abc are equal to each other.

In 12 out of the 16 inputs for $abcd$, the output bit ks is exactly the same as the input bit d. Therefore, it is valid to conclude that on average with randomly distributed input bits, in $\frac{3}{4}$ of the cases $ks = d$, while only in $\frac{1}{4}$ of the cases $ks \neq d$. This property makes the second rotation register much more influential to the value of the produced keystream bits.

Assume an adversary recovers the first keystream bit ks_0. She knows that ks_0 was computed by the function $f(\cdot)$ with the arguments $f(k_3, k_7, k_{10}, k_{12})$. Therefore, she can determine with probability $\frac{3}{4}$ the correct value for input d, which is for ks_0 the thirteenth secret key bit k_{12}. Although this reduces the set of likely candidate keys significantly, the power of a correlation attack is best demonstrated when it is applied multiple consecutive times.

Subsequently, consider a set of 16 contiguous keystream bits $ks_0 ks_1 \ldots ks_{15}$. The secret key bits $k_3 k_7 k_{10} k_{12}$ are used to compute ks_0. Likewise, after five rotations the second register is completely rotated and the secret key bits $k_8 k_1 k_4 k_{12}$ are used for ks_5. According to Definition 1, the fourth bit d is the most influential input bit. For both cases, ks_1 and ks_5, the fourth input bit to $f(\cdot)$ is the secret key bit k_{12}.

Let us combine the probability that $\frac{1}{4}$ of the time $k_{12} \neq ks_0$ and that also $\frac{1}{4}$ of the time $k_{12} \neq ks_5$. If $ks_0 = ks_5$, the adversary can assume that the average probability of $ks_0 = ks_5 \neq k_{12}$ significantly drops to $\frac{1}{4} \times \frac{1}{4} = \frac{1}{16}$. This is because it is highly unlikely that in both cases the (independent) input bits a, b and c are equal to each other. A correlation attack that focusses on a combination of keystream bits provides a much greater advantage. The probability of a correct key-bit guess can be derived from the product of all probabilities that correspond to an independent verifiable statistical bias in the cipher.

The correlation attack, mounted on the output ks_0 and ks_5, is also valid for many more bits from the recovered keystream set. For every pair of keystream bits which are five positions apart ($ks_i = ks_{i+5}$ where $i \in \{0, \ldots, 10\}$), the average probability that they are exactly the same as the corresponding bit from the secret key is $\frac{15}{16}$. The properties of this cipher allow an even better (but a more complex) correlation attack. However, the purpose of the previous description is to present a simple and explanatory example.

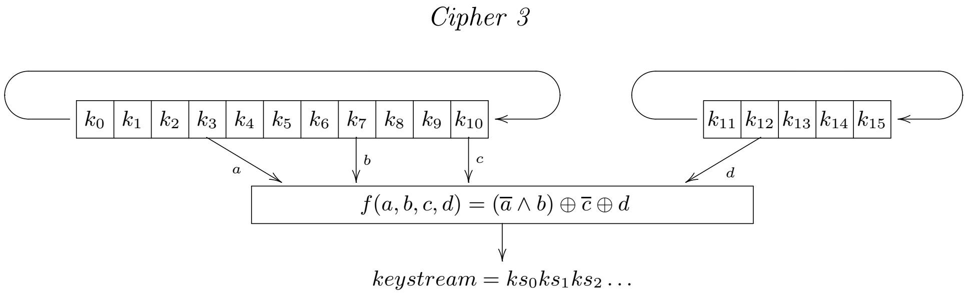
The scientific article [68] introduces a widely deployed cipher that is vulnerable to a correlation attack. It shows how to recover the secret key using a statistical bias in the output bits selection function that is part of the cryptographic algorithm. Although it is slightly different from the example presented here, the attack also exploits a correlation weakness to learn information about the internal state bits.

2.4 Guess-and-determine attack

Despite several well-known historical recommendations in the literature [5,70,86], many proprietary stream ciphers do not use their complete internal state to compute a keystream bit. Such cipher design allows an adversary to mount a guess-and-determine attack. This attack abuses the fact that only a few internal state bits are defined as input to the filter function. Therefore, only these bits determine the value of the computed keystream bit.

To mount such an attack, an adversary only guesses *used* bits, computes the output and evaluates it against the corresponding keystream bit that was recovered from an eavesdropped trace. The evaluation immediately leads to a contradiction for many of the guessed candidates. After evaluation, the internal state is updated accordingly and only additional required bits are guessed.

There are several impressive, yet slightly complex, examples in the literature [23,60,69,74, 75,103,129] that show the feasibility of guess-and-determine attacks on various cryptographic algorithms. To avoid the details of complex and optimized techniques another elementary stream cipher is introduced, see illustration in Figure 2.7.



Cipher 3
k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10}
k_{11} k_{12} k_{13} k_{14} k_{15}
a
b
c
d
$f(a,b,c,d) = (\overline{a} \wedge b) \oplus \overline{c} \oplus d$
$keystream = ks_0 ks_1 ks_2 \dots$

Fig. 2.7: Non-linear stream cipher, initialized by secret key k

Cipher 3, illustrated in Figure 2.7, is a clear example of a stream cipher that is vulnerable to a guess-and-determine attack. The output component embeds a balanced non-linear filter function which is given by Definition 2.

Definition 2. *Redefine the filter function* $f\colon \mathbb{F}_2^4 \to \mathbb{F}_2$ *as*

$$f(a,b,c,d) = (\overline{a} \wedge b) \oplus \overline{c} \oplus d$$

The boolean table that represents the input-ouput relation of $f(\cdot)$ from Definition 2 is shown in Figure 2.8. Note, that c and d have always influence on the computed keystream bit, while the influence of a and b depend on one each other.

abcd	$f(a,b,c,d)$
0000	1
0001	0
0010	0
0011	1

abcd	$f(a,b,c,d)$
0100	0
0101	1
0110	1
0111	0

abcd	$f(a,b,c,d)$
1000	1
1001	0
1010	0
1011	1

abcd	$f(a,b,c,d)$
1100	1
1101	0
1110	0
1111	1

Fig. 2.8: Boolean input-ouput table that corresponds to Definition 2

The adversary searches for a weakness in the cipher state transitions that allows a guess-and-determine attack. She starts by examining the corresponding cipher structure and the specified input bits which are required to compute the non-linear function. As mentioned in Section 2.3, the cipher illustrated in Figure 2.5 enables the adversary to guess only a few bits of secret key per computed output.

The cipher requires only four inputs bits at a time to compute one keystream bit. The 16 bit long secret key initializes an internal state. Therefore, the internal state has an entropy of 16 bits. Nevertheless, if only four bits of the internal state are used each time to compute a keystream bit, it makes no sense to increase complexity and guess all the bits at once. Especially, when the guessed values are not used to compute the next output bit. The power of a guess-and-determine attack rests itself in guessing each time only the bits that are actually used and make sure their output do not contradict with the keystream.

An adversary can easily perform an exhaustive-search over all 16 different input bits *abcd* which are shown in Figure 2.8. Next, she computes the result for each $f(a,b,c,d)$ and tests the output against one of the recovered keystream bits. This would allow her to distinguish good and bad candidates.

Function $f(\cdot)$ is balanced, so it evaluates for all 16 different values of *abcd* to eight times to a zero and eight times to a one. Therefore, when the keystream bit is a zero, the eight *abcd* candidates, which lead to a one, are disqualified (also referred to as eliminated).

After computing $f(\cdot)$ and performing one rotation on both registers, the cipher successor state is reached. The next keystream bit is computed with four different secret key bits. The adversary again uses the same technique and eliminates incorrect *abcd* candidates for the second keystream bit. The computational complexity drops significantly when it is possible to evaluate the input for each consecutive keystream bit independently from the others.

Assume an adversary recovered 16 consecutive keystream bits $ks_0 ks_1 \ldots ks_{15}$, let us examine how she can mount a guess-and-determine attack on the cipher illustrated in Figure 2.5. To clarify the secret key bits which are used to compute a keystream bit, the first 5 evaluations of the filter function $f(\cdot)$ are specified in Figure 2.9.

$$\begin{aligned} ks_0 &= f(k_3, k_7, k_{10}, k_{12}) \\ ks_1 &= f(k_4, k_8, k_0, k_{13}) \\ ks_2 &= f(k_5, k_9, k_1, k_{14}) \\ ks_3 &= f(k_6, k_{10}, k_2, k_{15}) \\ ks_4 &= f(k_7, k_0, k_3, k_{11}) \end{aligned}$$

Fig. 2.9: First five evaluations of $f(\cdot)$

The four secret key bits $k_3 k_7 k_{10} k_{12}$ are used to produce the first keystream bit ks_0. The adversary needs to guess all $2^4 = 16$ candidates for the 4 secret key bits $k_3 k_7 k_{10} k_{12}$. After elimination, only half of the candidate set survives the test. So, each guess of 4 bits allows the adversary to eliminate 1 bit of entropy. This leaves the adversary after 2^4 computations with a smaller set of $\frac{2^4}{2} = 2^3$ possible candidates.

Evaluation of the second keystream bit ks_1 requires the adversary to guess another 4 bits of the secret key, this time for $k_4 k_8 k_0 k_{13}$. Again, only half of the candidate set survives. These four secret key bits are completely different from those used to compute ks_0. In fact, the adversary guessed now 8 independent secret key bits, namely $k_0 k_3 k_4 k_7 k_8 k_{10} k_{12} k_{13}$. Even so, she only has $2^3 \times 2^4 = 2^7$ candidates after guessing the bits that compute ks_1. In contrast to the regular 2^8 candidates one should consider when 8 bits are unknown. It becomes even better, after computing the keystream bit for all 2^7 candidates, half of them are once again eliminated. It leaves a set of only 2^6 possible candidates with values that represent 8 bits of the secret.

The adversary applies the same technique on the third keystream bit ks_2. She guesses 12 secret key bits and after 2^{10} computations only 2^9 possible candidates survive. The fourth keystream bit ks_3 is computed with the secret key bits $k_6 k_{10} k_2 k_{15}$. Note that the secret key bit k_{10} was already guessed to compute the first keystream bit ks_0. It introduces two (partially) independent problems, a property which is similar to the divide-and-conquer attack described in Section 2.2.

Evaluating of ks_3 is rather special, this time the adversary only has to guess three bits instead of four. After guessing the bits and before evaluating the candidate set against the keystream, the size of the candidate set is 2^{12}. Since a computation has to be performed for each of the candidates in this set, the total attack complexity grows to 2^{12} computations. Nevertheless, after evaluating ks_3, the adversary guessed 15 secret key bits and compiled a set of only 2^{11} candidates.

The last secret key bit k_{11}, which is still not considered, is required to compute the fifth keystream bit ks_4. Guessing k_{11} only mildly increases the computational complexity of the attack. The candidate set is first doubled (again) to 2^{12} and after 2^{12} computations immediately halved again to 2^{11} candidates. Testing the candidates requires in total two times the largest computation of 2^{12} plus some insignificant and negligible smaller computations which were performed on the smaller candidate sets. This results in a total attack complexity of $(2 \times 2^{12}) + 2^{10} + 2^7 + 2^4 = 9360 \approx 2^{13}$ instead of the supposed $2^{16} = 65536$ computations.

Although the computational complexity is reduced significantly in this example, further (more complex) optimizations of the guess-and-determine strategy could improve this attack even more. Such techniques are not further explored in this section, although they are further addressed in the cryptographic attacks described in [62, 95, 123, 124].

2.5 Differential cryptanalysis

The Data Encryption Standard (DES), introduced in 1977, was considered theoretically secure for many years. This confidence lasted until 1991 when Biham and Shamir showed in [16] that it is possible to cryptographically attack a cipher that is similar to DES. They used a new attack technique which they introduced as *differential cryptanalysis*.

The company IBM, responsible for the initial design of DES, claimed in [36] that they were already familiar with this particular attack technique for more than 15 years. Furthermore, they explained to have used their knowledge to prevent differential cryptanalysis on DES as much as possible.

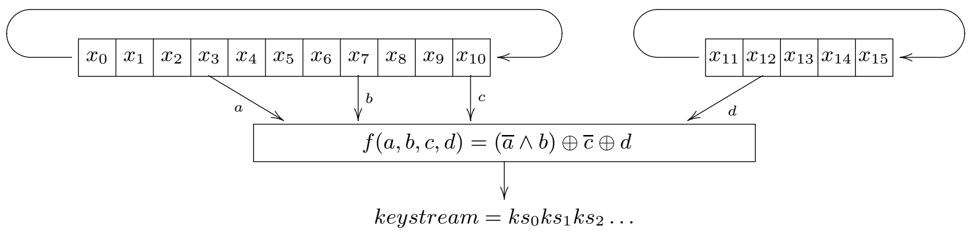
Nonetheless, two years later, Biham and Shamir published another article [17] in which the authors specifically attacked the cryptographic algorithm of DES. The attack requires an enormous amount of gathered data and is therefore considered purely theoretical. Their publications inspired many fellow academics to further explore and optimize the differential attack technique both for block ciphers [1,31,87] and for stream ciphers [53,99,128].

The applicability of differential cryptanalysis highly depends on the possibility to gather a set of similar encryptions which differ only to a certain extent. A straightforward approach would be to find a way that directly influences and only slightly changes the internal state of the cipher. To apply such a technique in practice, often additional components of the cryptosystems are used to intentionally create the desired difference in the internal state. Examples of such components are the internal state initialization procedure, key diversification schemes and random number generators. With control over these components an adversary can often predict and pre-compute the desired changes.

The initialization procedure of the cipher might allow an attacker to specifically change one internal state bit at a certain position. Such a minor change could lead directly to a different output which indicates the changed bit is a significant input to the filter function. Likewise, when the change does not influence the corresponding keystream bit, it reveals that the bit is an insignificant input to filter function.

Consider the non-linear stream *Cipher 4*, illustrated in Figure 2.10. It uses the non-linear filter function from Definition 2. Although the workings of the cipher are similar to that of *Cipher 3*, the internal state is initialized in a different way.

Cipher 4



x_0 x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}
x_{11} x_{12} x_{13} x_{14} x_{15}
a
b
c
d
$f(a,b,c,d) = (\overline{a} \wedge b) \oplus \overline{c} \oplus d$
$keystream = ks_0 ks_1 ks_2 \ldots$

Fig. 2.10: Non-linear stream cipher that initializes a fresh internal state with the key k and nonce n by $x_i \leftarrow k_i \oplus n_i$ where $i \in \{0,\ldots,15\}$

Many cryptosystems use random challenges to add freshness to the encryption. In this example the challenges consist of a 16-bit random value which is called the nonce n. Furthermore, the bits of k are not just placed into the internal state of the cipher like *Cipher 3*. Instead, they are loaded with the exclusive-or (XOR) of the secret key k and the nonce n. To be precise, the initialization is performed by $x_i \leftarrow k_i \oplus n_i$ where $i \in \{0,\ldots,15\}$.

Assume two additional attack vectors on the cryptosystem. The adversary has complete control over the nonce n and secondly, she can invoke and observe plain keystream from an authentication attempt as often as she likes. This allows her to perform differential cryptanalysis in the following way. First, she authenticates using a nonce n and stores the first computed keystream bit ks_0. Next, she flips the fourth bit n_3 from the nonce n which gives her an alternative nonce n'. Then, she re-authenticates with the nonce n' and compares the computed bit ks'_0 against the previously stored keystream bit ks_0.

After the two authentications, the adversary did not recover the actual value of x_3. However, she does know that the difference between the internal states of the two authentication sessions is a negation of the bit x_3. Furthermore, the bit x_3 represents input a to the filter function, which in turn is used to compute the first keystream bit for both authentications ks_0 and ks'_0. Note, that input b, represented by x_7, was not altered by the nonce since $n_7 = n'_7$. The same holds for input c and d, since the fourth bit of the nonce was changed.

A closer look at the input-output relation of the filter function (see Figure 2.8) reveals that the value of input a only matters if input b is *true*. The first keystream bit is computed with x_3 as input a and x_7 as input b. Consequently, x_3 only influences the keystream bit ks_0 when $x_7 = 1$. Therefore, if $ks_0 = ks'_0$ the adversary can conclude that $x_7 = 0$. Likewise, the contrapositive is valid as well, if $ks_0 \neq ks'_0$ she can conclude that $x_7 = 1$.

In conclusion, a differential attack can be mounted on this cipher simply by flipping the fourth bit of the nonce and uses it to re-authenticate. The observed differences between the first computed keystream bit in both authentications reveals the value of x_7. With the knowledge of nonce bit n_7 and the determination of x_7, the secret key bit k_7 can be simply recovered by computing $k_7 = x_7 \oplus n_7$.

This particular example shows how to recover one bit of the secret key by performing only two authentications and with negligible computational complexity. Since the stream cipher is rotating regularly, the same technique could be applied to recover more bits of the secret key. Likewise, flipping nonce bit n_4 and observation of a difference in ks_1 reveals x_8.

Differential cryptanalysis is a powerful technique to recover a secret key. Some carelessly designed cryptosystems are vulnerable to such an extent that it is even possible to mount a differential attack in practice. For instance, the MIFARE Classic and iClass ciphers, presented in [67,95] and [63,65,66], are vulnerable to a practical differential attack. Both ciphers allow manipulation of the input in such a way that the computed output leaks information about the secret key.

A differential attack often requires specific input-output differences. The initialization of a cryptosystem might not always allow an adversary to enforce such differences in the internal state. Nonetheless, there is always the possibility to gather many traces and filter out the ones that satisfy the predetermined conditions. Moreover, according to the birthday paradox, the number of traces the adversary needs to gather is much smaller than the number usually suggested by intuition, see [112] for more details.

2.6 Algebraic attacks

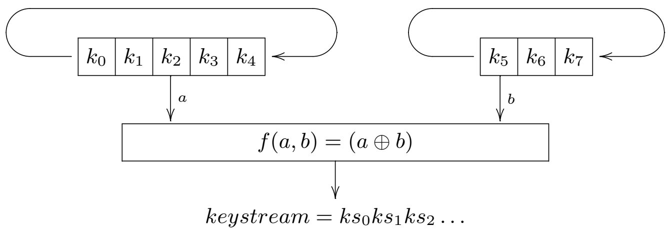
Several cryptographic attack methodologies aim to reduce the computational complexity of a stream cipher by attacking the non-linear function. However, some cipher designs consist of one or more linear components. Such ciphers are vulnerable to algebraic attacks. The literature includes several historical contributions concerning algebraic attacks [78, 79, 106, 107]. Throughout the last decade, several attack generalizations and optimizations were proposed [7,8,30,38,40,42,43,59,92]. This section, however, only defines the basic principles of algebraic attacks. Note that algebraic attacks should not be confused with linear cryptanalysis, which was introduced in [90] to attack the block cipher DES.

A property of a linear boolean function is the possibility to postpone an evaluation. Computational problems which are formalized during a cryptographic attack can be written as a system of boolean equations [114]. Instead of computing the outcome directly, a combination of these equations can be solved by well-known techniques such as Gaussian elimination [78,89,98,107,113,127].

In boolean algebra, a function f is $\mathbb{F}_2$-linear if it satisfies $f(x \oplus y) = f(x) \oplus f(y)$ for any pair (x, y) of elements in its domain. Moreover, a boolean linear function defines that the input variables either always or never influence the output. Consequently, an observation of influenced output bits, that were computed by a linear function, leak a deterministic linear relation about the corresponding changes of the input bits. When a sequence of influenced output bits exceeds the number of input bits, it is just a matter of solving the system of equations without the need for expensive computations. Contrarily, a non-linear function makes the inversion process more difficult, since the influence of the individual input bits depends on the actual value of these bits. In contrast to a linear function, it is much more difficult to generalize an equation for non-linear functions which holds for every input.

Consider the cipher design with *linear* output as illustrated in Figure 2.11. It consist of two very small bitwise rotating registers which both deliver one input bit to the *linear* filter function $f(\cdot)$ in order to produce the next keystream bit.

Cipher 5



k_0 k_1 k_2 k_3 k_4
k_5 k_6 k_7
a
b
$f(a, b) = (a \oplus b)$
$keystream = ks_0ks_1ks_2\ldots$

Fig. 2.11: A linear stream cipher, initialized by secret key k

The linear filter function $f(\cdot)$ consists of only one exclusive-or (XOR) operation and is redefined in Definition 3. The XOR operation, denoted by the symbol $\oplus$, represents a mathematical addition modulo 2.

Definition 3. *Redefine the filter function* $f\colon \mathbb{F}_2^2 \to \mathbb{F}_2$ *as*

$$f(a, b) = (a \oplus b)$$

The XOR operator is a bitwise linear operation and allows an adversary to compile a system of equations. Consider the following keystream bits $ks_0ks_1\ldots ks_7 = 10110101$. Figure 2.12 consists of four columns, each of which represents a write-up or evaluation step. After three steps, eight equalities are formed which correspond to the considered keystream. A combination of those equalities determines two equations which should be satisfied when guessing the actual secret key bits.

Given the equalities denoted in Figure 2.12d it is feasible to deduce two equations, which are given below in Equation (1) and Equation (2). The equations specify that there are two sets of secret key bits. All elements within one set carry the same value (either zero or one). However, both sets represent the opposite value to each other, which is confirmed by the fact that k_6 is an element of the first set and its complement $\overline{k_6}$ is an element of the second set.

$$k_6 = k_3 = k_1 = k_5 = k_7 \tag{1}$$

$$\overline{k_6} = k_4 = k_2 = k_0 \tag{2}$$

According to Equation (1) and Equation (2) there could be only two solutions, either $k_0k_1\ldots k_7 = 10101000$ or its bitwise complement $k_0k_1\ldots k_7 = 01010111$. The computational complexity to evaluate the equation is negligible. However, the actual work that is required also includes the evaluation of the keystream bits, deducing the equalities and formulating

$$
\begin{array}{cccc}
ks_0 = k_2 \oplus k_5 & k_5 = k_2 \oplus ks_0 & k_5 = k_2 \oplus 1 & k_5 = \overline{k_2} \\
ks_1 = k_3 \oplus k_6 & k_6 = k_3 \oplus ks_1 & k_6 = k_3 \oplus 0 & k_6 = k_3 \\
ks_2 = k_4 \oplus k_7 & k_7 = k_4 \oplus ks_2 & k_7 = k_4 \oplus 1 & k_7 = \overline{k_4} \\
ks_3 = k_0 \oplus k_5 & k_5 = k_0 \oplus ks_3 & k_5 = k_0 \oplus 1 & k_5 = \overline{k_0} \\
ks_4 = k_1 \oplus k_6 & k_6 = k_1 \oplus ks_4 & k_6 = k_1 \oplus 0 & k_6 = k_1 \\
ks_5 = k_2 \oplus k_7 & k_7 = k_2 \oplus ks_5 & k_7 = k_2 \oplus 1 & k_7 = \overline{k_2} \\
ks_6 = k_3 \oplus k_5 & k_5 = k_3 \oplus ks_6 & k_5 = k_3 \oplus 0 & k_5 = k_3 \\
ks_7 = k_4 \oplus k_6 & k_6 = k_4 \oplus ks_7 & k_6 = k_4 \oplus 1 & k_6 = \overline{k_4} \\
\text{(a)} & \text{(b)} & \text{(c)} & \text{(d)}
\end{array}
$$

Fig. 2.12: Evaluation of $ks_0ks_1 \ldots ks_7 = 10110101$ leads to the equalities of (d)

the final equation. It is much harder to generalize such tasks in terms of computational complexity, especially since several tricks to efficiently optimize such effort were proposed in the literature [7, 29, 88, 92].

Techniques that find their origin in the previously explained algebraic attacks are applied in practice to the proprietary Hitag2 cipher described in [123]. At first it considers two linearly combined inputs as one element of the initialization. This allows an adversary to drastically reduce the list of possible internal states by avoiding independent evaluation of both inputs.

2.7 Meet-in-the-middle attack

It is not trivial to increase the computational complexity of a cryptosystem. General reasoning could steer a designer to apply inefficient enhancements [58, 96]. For instance, it is misleading to think that applying a cipher twice, using a completely different secret key, would double the total computational complexity of a cryptographic algorithm. At first sight it suggests that an adversary needs to perform an exhaustive search twice, to independently determine the first and second secret key. In practice however, it is unlikely that an adversary would use such a method to attack a cryptosystem that applies multiple encryptions. This section introduces a powerful cryptanalytic technique that attacks such a cryptosystem from two sides, the input and the output, and tries to correlate the results in the middle. The technique is commonly referred to as a meet-in-the-middle attack. It can be seen as a special form a divide-and-conquer attack, since it generally tries to solve two independent problems concurrently.

Doubling the computational complexity means that the entropy of both secret keys are multiplied with each other, resulting in the product of both secret keys. However, a scheme where two independent ciphers are applied in a consecutive manner is vulnerable to a meet-in-the-middle attack. Such an attack reduces the computational complexity to the sum of both entropies instead of to the product.

A meet-in-the-middle attack requires knowledge of at least one plaintext-ciphertext pair. With such a pair, the adversary can attack both independent ciphers at the same time. The plaintext is used as input to the first cipher, while the ciphertext is considered as output of the second cipher. The idea is to find a list of candidate outputs (ciphertexts) of the first cipher and a list of candidate inputs (plaintexts) for the second cipher. The intersection of both lists reveals a relation between the two independent secret keys. A plaintext-ciphertext pair that consists of a long enough bitstring singles out only one possible candidate. When multiple candidates are found, more plaintext-ciphertext pairs can be utilized as test vectors to recover the valid secret keys.

For instance, Figure 2.13 shows a cryptosystem that utilizes two different stream ciphers with completely independent secret keys. The ciphers are applied consecutively after each other to encrypt the plaintext. After encryption with the first cipher, the plaintext is transformed into an intermediate state which represents the output of the first cipher as well as the input to the second cipher.

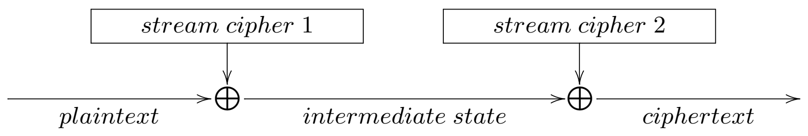


Fig. 2.13: cryptosystem which is vulnerable to a meet-in-the-middle attack

Assume that both stream ciphers in Figure 2.13 are not susceptible to a known cryptographic weakness. This leaves the adversary with no other option than to perform an exhaustive search over the complete key space of both ciphers. Instead of trying for each first cipher key guess all possible second cipher keys, the adversary performs the following attack steps.

1. She first recovers a plaintext-ciphertext pair of moderate length. It is preferably a longer bitstring than the secret keys length of both stream ciphers together. More bits, means more information and avoids multiple (false) candidates when correlating the output of the first cipher with the input of the second cipher.
2. Then, she encrypts the plaintext with the first cipher using all possible secret keys and stores the (encrypted) output in one big list. This output list represents the intermediate state candidates.
3. Next, she decrypts the ciphertext with the second cipher using all possible secret keys and looks if the (decrypted) input entry exists in the output list of the first cipher.
4. Finally, the decrypted input entry that exists in the encrypted output list is the value where both ciphers *meet in the middle*. The secret keys used to encrypt and decrypt to this value are the secret keys used by the original system to produce the plaintext-ciphertext pair.

Instead of performing a brute-force attack on the second cipher for each first cipher secret key guess, the adversary only has to perform two independent brute-force attacks and correlate the intermediate results. This makes the cryptographic problem only twice as big, meaning the sum of twice the original $(\textit{computational complexity}) \times 2$, instead of the product $(\textit{computational complexity})^2$.

The meet-in-the-middle attack is a well-known strategy to defeat weak components and schemes embodied in various cryptosystem. Several improvements and practical implementations of meet-in-the-middle attacks are given in the literature [6, 49–51, 55, 73, 116]. Likewise, the article [68] introduces a variant of the meet-in-the-middle attack that recovers the secret key during the cipher initialization phase. It first shows how to recover the internal state of the CryptoMemory cipher. Then, it starts from the default initialization and guesses the first half of the secret key while running forward and guesses the other half running backwards. During the intersection of both internal state candidate lists, the overlapping state represents the secret key.

Some cryptosystems explicitly define encryption schemes to mitigate meet-in-the-middle attacks. For instance, the Triple DES (3DES) specification defines the use of multiple consecutive utilizations of the Data Encryption Standard (DES). To mitigate a meet-in-the-middle attack, the crypto operation is performed three times instead of two. Note, that the increase in computational complexity is limited to two times the original $2^{56\times 2} = 2^{112}$ bits, instead of the intuitively expected three times. Since the achieved complexity is only two single DES keys, the 3DES scheme defines a way to securely perform three times single DES using exactly two secret keys of 2^{56} bits. It involves a sequence of encryption, followed by a decryption and encryption again.

Multiple encryption schemes are useful to increase the computational complexity of block ciphers, yet they do not provide the same security features as stream ciphers. Combination of various stream cipher outputs means that multiple keystreams are all combined together with the plaintext using an XOR operation. This inherently defines a direct linear relation between the computed output bits of each utilized stream cipher. In fact, [91] shows that the keystream produced by a combination of two additive stream ciphers is as secure as the strongest of the two. Although this feature does not increase the cryptographic strength, it could be used to spread the potential risk of cipher weaknesses over multiple independent ciphers.

3 Acronyms

3DES	Triple DES	15
DES	Data Encryption Standard	11
IV	Initialization Vector	5
MAC	Message Authentication Code	4
OTP	One-time Pad	1
PRNG	Pseudo Random Number Generator	2
TMTO	Time-Memory Trade-Off	4
XOR	exclusive-or	2

4 About the author

Roel Verdult successfully defended his doctoral dissertation titled "*The (in)security of proprietary cryptography*" at the Aula of the Radboud University, The Netherlands on 21st of April, 2015, at 14.30 exactly. He earned his doctorate at two universities and received a dual degree from the *Digital Security group, Institute for Computing and Information Sciences, Radboud University* and the *Computer Security and Industrial Cryptography, Electrical Engineering Department, KU Leuven, Belgium.*

References

1. Martin Albrecht and Carlos Cid. Algebraic techniques in differential cryptanalysis. In *Fast Software Encryption*, pages 193–208. Springer-Verlag, 2009.
2. Hamid Reza Amirazizi and Martin E Hellman. Time-memory-processor trade-offs. *IEEE Transactions on Information Theory*, 34(3):505–512, 1988.
3. Ross Anderson. Searching for the optimum correlation attack. In *2nd International Workshop on Fast Software Encryption (FSE 1994)*, volume 1008 of *Lecture Notes in Computer Science*, pages 137–143. Springer-Verlag, 1995.
4. Ross Anderson and Charalampos Manifavas. Chameleon - a new kind of stream cipher. In *4th International Workshop on Fast Software Encryption (FSE 1997)*, volume 1267 of *Lecture Notes in Computer Science*, pages 107–113. Springer-Verlag, 1997.
5. Ross J Anderson. Tree functions and cipher systems. *Cryptologia*, 15(3):194–202, 1991.
6. Kazumaro Aoki and Yu Sasaki. Meet-in-the-middle preimage attacks against reduced SHA-0 and SHA-1. In *29th International Cryptology Conference, Advances in Cryptology (CRYPTO 2009)*, volume 5677 of *Lecture Notes in Computer Science*, pages 70–89. Springer-Verlag, 2009.
7. Frederik Armknecht. Improving fast algebraic attacks. In *11th International Workshop on Fast Software Encryption (FSE 2004)*, volume 3017 of *Lecture Notes in Computer Science*, pages 65–82. Springer-Verlag, 2004.
8. Frederik Armknecht and Matthias Krause. Algebraic attacks on combiners with memory. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)*, volume 2729 of *Lecture Notes in Computer Science*, pages 162–175. Springer-Verlag, 2003.
9. Mikhail J Atallah, Richard Cole, and Michael T Goodrich. Cascading divide-and-conquer: A technique for designing parallel algorithms. *SIAM Journal on Computing*, 18(3):499–532, 1989.
10. Gildas Avoine, Pascal Junod, and Philippe Oechslin. Characterization and improvement of time-memory trade-off based on perfect tables. *ACM Transactions on Information and System Security (TISSEC 2008)*, 11(4):1–22, 2008.
11. Steve Babbage. A space/time tradeoff in exhaustive search attacks on stream ciphers. In *European Convention on Security and Detection*, volume 408 of *Conference Publications*, pages 161–166. IEEE Computer Society, 1995.
12. Elad Barkan, Eli Biham, and Adi Shamir. Rigorous bounds on cryptanalytic time/memory tradeoffs. In *26th International Cryptology Conference, Advances in Cryptology (CRYPTO 2006)*, volume 4117 of *Lecture Notes in Computer Science*, pages 1–21. Springer-Verlag, 2006.

13. Jon Louis Bentley. Multidimensional divide-and-conquer. *Communications of the ACM*, 23(4):214–229, 1980.
14. Jon Louis Bentley and Michael Ian Shamos. Divide-and-conquer in multidimensional space. In *8th ACM Symposium on Theory of Computing (STOC 2013)*, pages 220–230. ACM, 1976.
15. Côme Berbain and Henri Gilbert. On the security of IV dependent stream ciphers. In *14th International Workshop on Fast Software Encryption (FSE 2007)*, volume 4593 of *Lecture Notes in Computer Science*, pages 254–273. Springer-Verlag, 2007.
16. Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991.
17. Eli Biham and Adi Shamir. *Differential cryptanalysis of the Data Encryption Standard*, volume 28. Springer-Verlag, 1993.
18. Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *17th International Cryptology Conference, Advances in Cryptology (CRYPTO 1997)*, volume 1294 of *Lecture Notes in Computer Science*, pages 513–525. Springer-Verlag, 1997.
19. Alex Biryukov. Block ciphers and stream ciphers: The state of the art. *IACR Cryptology ePrint Archive*, 2004(94):1–22, 2004.
20. Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar. Improved time-memory trade-offs with multiple data. In *12th International Workshop on Selected Areas in Cryptography (SAC 2005)*, volume 3897 of *Lecture Notes in Computer Science*, pages 110–127. Springer-Verlag, 2006.
21. Alex Biryukov and Adi Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In *6th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2000)*, volume 1976 of *Lecture Notes in Computer Science*, pages 1–13. Springer-Verlag, 2000.
22. Alex Biryukov and David Wagner. Slide attacks. In *6th International Workshop on Fast Software Encryption (FSE 1999)*, volume 1636 of *Lecture Notes in Computer Science*, pages 245–259. Springer-Verlag, 1999.
23. Andrey Bogdanov. Attacks on the KeeLoq block cipher and authentication systems. In *3rd Conference on RFID Security (RFIDSec 2007)*, volume 2007, 2007.
24. Andrey Bogdanov, Dmitry Khovratovich, and Christian Rechberger. Biclique cryptanalysis of the full AES. In *17th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2011)*, volume 7073 of *Lecture Notes in Computer Science*, pages 344–371. Springer-Verlag, 2011.
25. Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. PRESENT: An ultra-lightweight block cipher. In *9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer-Verlag, 2007.
26. Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. In *16th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1997)*, volume 1233 of *Lecture Notes in Computer Science*, pages 37–51. Springer-Verlag, 1997.
27. Johan Borst, Bart Preneel, Joos Vandewalle, and Joos V. On the time-memory tradeoff between exhaustive key search and table precomputation. In *19th Symposium in Information Theory in the Benelux*, pages 111–118, 1998.
28. Paul Camion, Claude Carlet, Pascale Charpin, and Nicolas Sendrier. On correlation-immune functions. In *11th International Cryptology Conference, Advances in Cryptology (CRYPTO 1991)*, volume 576 of *Lecture Notes in Computer Science*, pages 86–100. Springer-Verlag, 1992.
29. John F Canny, Erich Kaltofen, and Lakshman Yagati. Solving systems of nonlinear polynomial equations faster. In *2nd International Symposium on Symbolic and Algebraic Computation (ISSAC 1989)*, pages 121–128. ACM, 1989.
30. Claude Carlet and Keqin Feng. An infinite class of balanced functions with optimal algebraic immunity, good immunity to fast algebraic attacks and good nonlinearity. In *14th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2008)*, volume 5350 of *Lecture Notes in Computer Science*, pages 425–440. Springer-Verlag, 2008.
31. Florent Chabaud and Serge Vaudenay. Links between differential and linear cryptanalysis. In *13th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1994)*, volume 950 of *Lecture Notes in Computer Science*, pages 356–365. Springer-Verlag, 1995.

32. Vladimir Chepyzhov and Ben Smeets. On a fast correlation attack on certain stream ciphers. In *10th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1991)*, volume 547 of *Lecture Notes in Computer Science*, pages 176–185. Springer-Verlag, 1991.
33. Vladimor V Chepyzhov, Thomas Johansson, and Ben Smeets. A simple algorithm for fast correlation attacks on stream ciphers. In *7th International Workshop on Fast Software Encryption (FSE 2000)*, volume 1978 of *Lecture Notes in Computer Science*, pages 181–195. Springer-Verlag, 2001.
34. Yi-Hao Chiu, Wei-Chih Hong, Li-Ping Chou, Jintai Ding, Bo-Yin Yang, and Chen-Mou Cheng. A practical attack on patched MIFARE Classic. In *9th International Conference on Information Security and Cryptology (INSCRYPT 2013)*, volume 8567 of *Lecture Notes in Computer Science*, pages 150–164. Springer-Verlag, 2014.
35. Andrew Clark, Jovan Dj Golić, and Ed Dawson. A comparison of fast correlation attacks. In *3rd International Workshop on Fast Software Encryption (FSE 1996)*, volume 1039 of *Lecture Notes in Computer Science*, pages 145–157. Springer-Verlag, 1996.
36. Don Coppersmith. The data encryption standard (DES) and its strength against attacks. *IBM journal of research and development*, 38(3):243–250, 1994.
37. Nicolas Courtois, Karsten Nohl, and Sean O'Neil. Algebraic attacks on the crypto-1 stream cipher in MIFARE Classic and oyster cards. *IACR Cryptology ePrint Archive*, 2008:166, 2008.
38. Nicolas T Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)*, volume 2729 of *Lecture Notes in Computer Science*, pages 176–194. Springer-Verlag, 2003.
39. Nicolas T Courtois. Higher order correlation attacks, xl algorithm and cryptanalysis of toyocrypt. In *5th International Conference on Information Security and Cryptology (ICISC 2002)*, volume 2587 of *Lecture Notes in Computer Science*, pages 182–199. Springer-Verlag, 2003.
40. Nicolas T Courtois. Algebraic attacks on combiners with memory and several outputs. In *11th International Conference on Information Security and Cryptology (ICISC 2008)*, volume 3506 of *Lecture Notes in Computer Science*, pages 3–20. Springer-Verlag, 2005.
41. Nicolas T. Courtois. The dark side of security by obscurity - and cloning MIFARE Classic rail and building passes, anywhere, anytime. In *4th International Conference on Security and Cryptography (SECRYPT 2009)*, pages 331–338. INSTICC Press, 2009.
42. Nicolas T Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *22nd International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2003)*, volume 2656 of *Lecture Notes in Computer Science*, pages 345–359. Springer-Verlag, 2003.
43. Nicolas T Courtois and Josef Pieprzyk. Cryptanalysis of block ciphers with overdefined systems of equations. In *8th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2002)*, volume 2501 of *Lecture Notes in Computer Science*, pages 267–287. Springer-Verlag, 2002.
44. Joan Daemen, Lars Knudsen, and Vincent Rijmen. The block cipher SQUARE. In *4th International Workshop on Fast Software Encryption (FSE 1997)*, volume 1267 of *Lecture Notes in Computer Science*, pages 149–165. Springer-Verlag, 1997.
45. Donald Davies and Sean Murphy. Pairs and triplets of DES S-boxes. *Journal of Cryptology*, 8(1):1–25, 1995.
46. Ed Dawson and Andrew Clark. Divide and conquer attacks on certain classes of stream ciphers. *Cryptologia*, 18(1):25–40, 1994.
47. Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In *9th International Conference on Information Security (ISC 2006)*, volume 4176 of *Lecture Notes in Computer Science*, pages 171–186. Springer-Verlag, 2006.
48. Gerhard de Koning Gans, Jaap-Henk Hoepman, and Flavio D. Garcia. A practical attack on the MIFARE Classic. In *8th Smart Card Research and Advanced Applications Conference (CARDIS 2008)*, volume 5189 of *Lecture Notes in Computer Science*, pages 267–282. Springer-Verlag, 2008.
49. Hüseyin Demirci and Ali Aydın Selçuk. A meet-in-the-middle attack on 8-round AES. In *15th International Workshop on Fast Software Encryption (FSE 2008)*, volume 5086 of *Lecture Notes in Computer Science*, pages 116–126. Springer-Verlag, 2008.
50. Hüseyin Demirci, Ali Aydin Selçuk, and Erkan Türe. A new meet-in-the-middle attack on the IDEA block cipher. In *10th International Workshop on Selected Areas in Cryptography (SAC 2003)*, volume 3006 of *Lecture Notes in Computer Science*, pages 117–129. Springer-Verlag, 2004.

51. Hüseyin Demirci, İhsan Taşkın, Mustafa Çoban, and Adnan Baysal. Improved meet-in-the-middle attacks on AES. In *10th International Conference on Cryptology in India, Progress in Cryptology (INDOCRYPT 2009)*, volume 5922 of *Lecture Notes in Computer Science*, pages 144–156. Springer-Verlag, 2009.
52. Whitfield Diffie and Martin E Hellman. Privacy and authentication: An introduction to cryptography. *Proceedings of the IEEE*, 67(3):397–427, 1979.
53. Cunsheng Ding. The differential cryptanalysis and design of natural stream ciphers. In *1st International Workshop on Fast Software Encryption (FSE 1993)*, volume 809 of *Lecture Notes in Computer Science*, pages 101–115. Springer-Verlag, 1994.
54. Itai Dinur and Adi Shamir. Cube attacks on tweakable black box polynomials. In *28th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2009)*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer-Verlag, 2009.
55. Orr Dunkelman, Gautham Sekar, and Bart Preneel. Improved meet-in-the-middle attacks on reduced-round DES. In *8th International Conference on Cryptology in India, Progress in Cryptology (INDOCRYPT 2007)*, volume 4859 of *Lecture Notes in Computer Science*, pages 86–100. Springer-Verlag, 2007.
56. Rex A Dwyer. A faster divide-and-conquer algorithm for constructing delaunay triangulations. *Algorithmica*, 2(1-4):137–151, 1987.
57. Patrik Ekdahl and Thomas Johansson. A new version of the stream cipher SNOW. In *9th International Workshop on Selected Areas in Cryptography (SAC 2002)*, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer-Verlag, 2003.
58. Shimon Even and Oded Goldreich. On the power of cascade ciphers. *ACM Transactions on Computer Systems (TOCS)*, 3(2):108–116, 1985.
59. Jean-Charles Faugere and Antoine Joux. Algebraic cryptanalysis of hidden field equation (hfe) cryptosystems using gröbner bases. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)*, volume 2729 of *Lecture Notes in Computer Science*, pages 44–60. Springer-Verlag, 2003.
60. Xiutao Feng, Jun Liu, Zhaocun Zhou, Chuankun Wu, and Dengguo Feng. A byte-based guess and determine attack on SOSEMANUK. In *16th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2010)*, volume 6477 of *Lecture Notes in Computer Science*, pages 146–157. Springer-Verlag, 2010.
61. Amos Fiat and Moni Naor. Rigorous time/space tradeoffs for inverting functions. In *Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 534–541. ACM, 1991.
62. Flavio D. Garcia, Gerhard de Koning Gans, Ruben Muijrers, Peter van Rossum, Roel Verdult, Ronny Wichers Schreur, and Bart Jacobs. Dismantling MIFARE Classic. In *13th European Symposium on Research in Computer Security (ESORICS 2008)*, volume 5283 of *Lecture Notes in Computer Science*, pages 97–114. Springer-Verlag, 2008.
63. Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult. Exposing iClass key diversification. In *5th USENIX Workshop on Offensive Technologies (WOOT 2011)*, pages 128–136. USENIX Association, 2011.
64. Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult. Tutorial: Proxmark, the swiss army knife for RFID security research. Technical report, Radboud University Nijmegen, 2012.
65. Flavio D. Garcia, Gerhard de Koning Gans, and Roel Verdult. Wirelessly lockpicking a smart card reader. *International Journal of Information Security*, 13(5):403–420, 2014.
66. Flavio D. Garcia, Gerhard de Koning Gans, Roel Verdult, and Milosch Meriac. Dismantling iClass and iClass Elite. In *17th European Symposium on Research in Computer Security (ESORICS 2012)*, volume 7459 of *Lecture Notes in Computer Science*, pages 697–715. Springer-Verlag, 2012.
67. Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Wirelessly pickpocketing a MIFARE Classic card. In *30th IEEE Symposium on Security and Privacy (S&P 2009)*, pages 3–15. IEEE Computer Society, 2009.
68. Flavio D. Garcia, Peter van Rossum, Roel Verdult, and Ronny Wichers Schreur. Dismantling SecureMemory, CryptoMemory and CryptoRF. In *17th ACM Conference on Computer and Communications Security (CCS 2010)*, pages 250–259. ACM, 2010.
69. Praveen S Gauravaram and William L Millan. Improved attack on the cellular authentication and voice encryption algorithm (CAVE). In *Cryptographic Algorithms and their Uses (CAU 2004)*, pages 1–13. Queensland University of Technology, 2004.

70. Jovan Dj Golić. On the security of nonlinear filter generators. In *3rd International Workshop on Fast Software Encryption (FSE 1996)*, volume 1039 of *Lecture Notes in Computer Science*, pages 173–188. Springer-Verlag, 1996.
71. S.W. Golomb. *Shift Register Sequences.* Holden-Day Series in Information Systems. Holden-Day, 1967.
72. El Groth. Generation of binary sequences with controllable complexity. *IEEE Transactions on Information Theory*, 17(3):288–296, 1971.
73. Jian Guo, San Ling, Christian Rechberger, and Huaxiong Wang. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2. In *16th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2010)*, volume 6477 of *Lecture Notes in Computer Science*, pages 56–75. Springer-Verlag, 2010.
74. Philip Hawkes and Gregory G Rose. Exploiting multiples of the connection polynomial in word-oriented stream ciphers. In *6th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2000)*, volume 1976 of *Lecture Notes in Computer Science*, pages 303–316. Springer-Verlag, 2000.
75. Philip Hawkes and Gregory G Rose. Guess-and-determine attacks on SNOW. In *9th International Workshop on Selected Areas in Cryptography (SAC 2002)*, volume 2595 of *Lecture Notes in Computer Science*, pages 37–46. Springer-Verlag, 2003.
76. Martin Hell, Thomas Johansson, and Willi Meier. Grain: a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing*, 2(1):86–93, 2007.
77. Martin E. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.
78. Lester S. Hill. Cryptography in an algebraic alphabet. *American Mathematical Monthly*, 36(6):306–312, 1929.
79. Lester S Hill. Concerning certain linear transformation apparatus of cryptography. *American Mathematical Monthly*, pages 135–154, 1931.
80. Jin Hong and Sunghwan Moon. A comparison of cryptanalytic tradeoff algorithms. *Journal of Cryptology*, 26(4):559–637, 2013.
81. Bart Jacobs and Ronny Wichers Schreur. Logical formalisation and analysis of the MIFARE Classic card in PVS. In *2nd International Conference on Interactive Theorem Proving*, volume 6898 of *Lecture Notes in Computer Science*, pages 3–17. Springer-Verlag, 2011.
82. Thomas Johansson and Fredrik Jönsson. Fast correlation attacks through reconstruction of linear polynomials. In *20th International Cryptology Conference, Advances in Cryptology (CRYPTO 2000)*, volume 1880 of *Lecture Notes in Computer Science*, pages 300–315. Springer-Verlag, 2000.
83. Fredrik Jönsson and Thomas Johansson. A fast correlation attack on lili-128. *Information Processing Letters*, 81(3):127–132, 2002.
84. John Kelsey, Bruce Schneier, and David Wagner. Mod n cryptanalysis, with applications against RC5P and M6. In *6th International Workshop on Fast Software Encryption (FSE 1999)*, volume 1636 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 1999.
85. Edwin Key. An analysis of the structure and complexity of nonlinear binary sequence generators. *IEEE Transactions on Information Theory*, 22(6):732–736, 1976.
86. GJ Kuhn. Algorithms for self-synchronizing ciphers. In *1st Southern African Conference on Communications and Signal Processing (COMSIG 1988)*, pages 159–164. IEEE, 1988.
87. Xuejia Lai, James L Massey, and Sean Murphy. Markov ciphers and differential cryptanalysis. In *10th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1991)*, volume 547 of *Lecture Notes in Computer Science*, pages 17–38. Springer-Verlag, 1991.
88. Chein-Shan Liu, Satya N Atluri, et al. A novel time integration method for solving a large system of non-linear algebraic equations. *Computer Modeling in Engineering & Sciences (CMES)*, 31(2):71–83, 2008.
89. Harry M Markowitz. The elimination form of the inverse and its application to linear programming. *Management Science*, 3(3):255–269, 1957.
90. Mitsuru Matsui. Linear cryptanalysis method for DES cipher. In *12th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1993)*, volume 765 of *Lecture Notes in Computer Science*, pages 386–397. Springer-Verlag, 1994.
91. Ueli M Maurer and James L Massey. Cascade ciphers: The importance of being first. *Journal of Cryptology*, 6(1):55–61, 1993.

92. Willi Meier, Enes Pasalic, and Claude Carlet. Algebraic attacks and decomposition of boolean functions. In *27th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 2004)*, volume 3027 of *Lecture Notes in Computer Science*, pages 474–491. Springer-Verlag, 2004.
93. Willi Meier and Othmar Staffelbach. Fast correlation attacks on stream ciphers. In *7th International Conference on the Theory and Application of Cryptographic Techniques, Advances in Cryptology (EUROCRYPT 1988)*, volume 330 of *Lecture Notes in Computer Science*, pages 301–314. Springer-Verlag, 1988.
94. Willi Meier and Othmar Staffelbach. Fast correlation attacks on certain stream ciphers. *Journal of Cryptology*, 1(3):159–176, 1989.
95. Carlo Meijer and Roel Verdult. Ciphertext-only cryptanalysis on hardened Mifare Classic cards. In *22nd ACM Conference on Computer and Communications Security (CCS 2015)*. ACM, 2015.
96. Ralph C Merkle and Martin E Hellman. On the security of multiple encryption. *Communications of the ACM*, 24(7):465–467, 1981.
97. William Millan, EP Dawson, and LJ O'Connor. Fast attacks on tree-structured ciphers. In *1st Workshop in Selected Areas in Cryptography (SAC 1994)*, pages 146–158, 1994.
98. David E Muller. A method for solving algebraic equations using an automatic computer. *Mathematical Tables and Other Aids to Computation*, 10(56):208–215, 1956.
99. Frédéric Muller. Differential attacks against the Helix stream cipher. In *11th International Workshop on Fast Software Encryption (FSE 2004)*, volume 3017 of *Lecture Notes in Computer Science*, pages 94–108. Springer-Verlag, 2004.
100. Karsten Nohl. Cryptanalysis of crypto-1. *Computer Science Department University of Virginia, White Paper*, 2008.
101. Karsten Nohl, David Evans, Starbug, and Henryk Plötz. Reverse engineering a cryptographic RFID tag. In *17th USENIX Security Symposium (USENIX Security 2008)*, pages 185–193. USENIX Association, 2008.
102. Philippe Oechslin. Making a faster cryptanalytic time-memory trade-off. In *23rd International Cryptology Conference, Advances in Cryptology (CRYPTO 2003)*, volume 2729 of *Lecture Notes in Computer Science*, pages 617–630. Springer-Verlag, 2003.
103. Enes Pasalic. On guess and determine cryptanalysis of LFSR-based stream ciphers. *IEEE Transactions on Information Theory*, 55(7):3398–3406, 2009.
104. Walter T Penzhorn. Correlation attacks on stream ciphers: Computing low-weight parity checks based on error-correcting codes. In *3rd International Workshop on Fast Software Encryption (FSE 1996)*, volume 1039 of *Lecture Notes in Computer Science*, pages 159–172. Springer-Verlag, 1996.
105. Vera S Pless. Encryption schemes for computer confidentiality. *IEEE Transactions on Computers*, 100(11):1133–1136, 1977.
106. James Reeds. "Cracking" a random number generator. *Cryptologia*, 1(1):20–26, 1977.
107. Frank Rubin. Decrypting a stream cipher based on J-K flip-flops. *IEEE Transactions on Computers*, 100(7):483–487, 1979.
108. Adi Shamir. Invited talk: Stream ciphers: Dead or alive? In *10th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2004)*, volume 3329 of *Lecture Notes in Computer Science*, page 78. Springer-Verlag, 2004.
109. Claude E Shannon. Communication theory of secrecy systems. *Bell system technical journal*, 28(4):656–715, 1949.
110. Thomas Siegenthaler. Correlation-immunity of nonlinear combining functions for cryptographic applications. *IEEE Transactions on Information Theory*, 30(5):776–780, 1984.
111. Thomas Siegenthaler. Decrypting a class of stream ciphers using ciphertext only. *IEEE Transactions on Computers*, 100(1):81–85, 1985.
112. William Simon. *Mathematical magic*. Courier Dover Publications, 1964.
113. Volker Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13(4):354–356, 1969.
114. Moiez A. Tapia and Jerry H. Tucker. Complete solution of boolean equations. *IEEE Transactions on Computers*, 100(7):662–665, 1980.
115. Fabian van den Broek and Erik Poll. A comparison of time-memory trade-off attacks on stream ciphers. In *6th International Conference on Cryptology in Africa, Progress in Cryptology (AFRICACRYPT 2013)*, volume 7918 of *Lecture Notes in Computer Science*, pages 406–423. Springer-Verlag, 2013.

116. Paul C Van Oorschot and Michael J Wiener. Improving implementable meet-in-the-middle attacks by orders of magnitude. In *16th International Cryptology Conference, Advances in Cryptology (CRYPTO 1996)*, volume 1109 of *Lecture Notes in Computer Science*, pages 229–236. Springer-Verlag, 1996.
117. Paul C Van Oorschot and Michael J Wiener. Parallel collision search with cryptanalytic applications. *Journal of Cryptology*, 12(1):1–28, 1999.
118. Roel Verdult. Proof of concept, cloning the OV-chip card. Technical report, Radboud University Nijmegen, 2008.
119. Roel Verdult. Security analysis of RFID tags. Master's thesis, Radboud University Nijmegen, 2008.
120. Roel Verdult. *The (in)security of proprietary cryptography.* PhD thesis, Radboud University, The Netherlands and KU Leuven, Belgium, April 2015.
121. Roel Verdult and Gerhard de Koning Gans. Proxmark.org - A Radio Frequency IDentification tool. http://www.proxmark.org, 2009.
122. Roel Verdult, Gerhard de Koning Gans, and Flavio D. Garcia. A toolbox for RFID protocol analysis. In *4th International EURASIP Workshop on RFID Technology (EURASIP RFID 2012)*, pages 27–34. IEEE Computer Society, 2012.
123. Roel Verdult, Flavio D. Garcia, and Josep Balasch. Gone in 360 seconds: Hijacking with Hitag2. In *21st USENIX Security Symposium (USENIX Security 2012)*, pages 237–252. USENIX Association, 2012.
124. Roel Verdult, Flavio D. Garcia, and Barış Ege. Dismantling megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In *22nd USENIX Security Symposium (USENIX Security 2013)*, pages 703–718. USENIX Association, 2015.
125. David Wagner. The boomerang attack. In *6th International Workshop on Fast Software Encryption (FSE 1999)*, volume 1636 of *Lecture Notes in Computer Science*, pages 156–170. Springer-Verlag, 1999.
126. Ronny Wichers Schreur, Peter van Rossum, Flavio D. Garcia, Wouter Teepe, Jaap-Henk Hoepman, Bart Jacobs, Gerhard de Koning Gans, Roel Verdult, Ruben Muijrers, Ravindra Kali, and Vinesh Kali. Security flaw in MIFARE Classic. *Press release, Digital Security group, Radboud University Nijmegen, The Netherlands*, March 2008.
127. Edward L Wilson, Klaus-Jürgen Bathe, and William P Doherty. Direct solution of large systems of linear equations. *Computers & Structures*, 4(2):363–372, 1974.
128. Hongjun Wu and Bart Preneel. Differential-linear attacks against the stream cipher Phelix. In *14th International Workshop on Fast Software Encryption (FSE 2007)*, volume 4593 of *Lecture Notes in Computer Science*, pages 87–100. Springer-Verlag, 2007.
129. Bin Zhang and Dengguo Feng. New guess-and-determine attack on the self-shrinking generator. In *12th International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology (ASIACRYPT 2006)*, volume 4284 of *Lecture Notes in Computer Science*, pages 54–68. Springer-Verlag, 2006.