Building Stream Ciphers from Block Ciphers and their Security

Hans Christoph Hudde

February 18, 2009

Seminararbeit
Ruhr-Universität Bochum

Chair for Communication Security
Prof. Dr.-Ing. Christof Paar

Contents

1 Introduction

In this chapter first the work is motivated in Section 1.1, before the outline of this Thesis is presented in Section 1.2.

1.1 Motivation

With well-established encryption algorithms like DES[1] or AES[2] at hand, one could have the impression that most of the work for building a cryptosystem - for example a suite of algorithms for the transmission of encrypted data over the internet - is already done. But the task of a cipher is very specific: to encrypt or decrypt a datablock of a specified length. Given an plaintext of arbitrary length, the most simple approach would be to break it down to blocks of the desired length and to use padding for the final block. Each block is encrypted separately with the same key, which results in identical ciphertext blocks for identical plaintext blocks. This is known as Electronic Code Book (ECB) mode of operation[3], and is not recommended in many situations because it does not hide data patterns well. Furthermore, ciphertext blocks are independent from each other, allowing an attacker to substitute, delete or replay blocks unnoticed.

FIPS PUB 81 [FIP80] specified three more modes of operation that use chaining (in CBC mode, i.e. Cipher Block Chaining) or feedback to overcome the disadvantages of ECB. The feedback modes in fact turn the block cipher into a stream cipher by using the algorithm as a keystream generator. Since every mode may yield different usage and security properties, it is necessary to analyse them in detail. For the traditional modes like Output Feedback (OFB), Cipher Feedback (CFB), Counter (CTR) and their variants, this has been done thoroughly, but there are also researches on rarely used modes like Plaintext or Key Feedback mode and combination modes like CTR-OFB and CTR-CFB.

In cases where buffering is limited or when characters must be processed as they are received (e.g. in terminals) it is useful or even mandatory to use a stream cipher for en-/decryption. Furthermore, many stream ciphers are superior to block ciphers concerning error propagation. The "Handbook of Applied Cryptography" [MvOV01] states that there are "relatively few fully-specified stream cipher algo-

[1] Data Encryption Standard, defined in [FIP93]

[2] Advanced Encryption Standard, defined in [FIP01]

[3] [Dwo01] provides a comprehensive overview for ECB and all other standard modes of operation.

rithms in the open literature", hence building stream ciphers from block ciphers can be an useful alternative to other stream ciphers. This paper aims to give an overview on these modes of operation and their security, as their understanding is imperative for any cryptosystem that is build on top of them.

1.2 Outline of this Thesis

The paper is organized as follows.

Chapter 2 gives an introduction to stream ciphers. First we discuss the basic principles and the categorization into synchronous and asynchronous stream ciphers. Then we present some of the most important attack models and notions of security.

The next chapter deals with the Cipher Feedback (CFB) mode of operation. We give a definition of the mode and discuss its properties in the first section, and then advance to an analysis of DES with a reduced round size in CFB mode.

In chapter 4 we define the Output Feedback (OFB) mode and show that in contrast to CFB mode full feedback is necessary for security. Furthermore we examine error properties and requirements on synchronization.

The counter mode (CTR) is introduced in chapter 5. Due to its deterministic input function it has raised controversial discussions that will be approached in the section on security aspects.

The last chapter deals with two rarely used modes of operation. The Key Feedback (KFB) mode employs an unusual structure but turns out to be a promising option, whereas the Plaintext Feedback (PFB) mode suffers from insignificance and vulnerabilities.

The following abbreviations will be used frequently:

- $enc_K(p)$ denotes the encryption of a plaintext p under the key K
  $dec_K(c)$ denotes the decryption of a ciphertext c under the key K
- MSB_x denotes the x leftmost (most significant) bits
  LSB_x denotes the x rightmost (least significant) bits
- $\oplus$ denotes the Exclusive-OR (XOR) function that is an addition modulo 2

Every block cipher used as underlying one-way-function for a mode of operation is considered to be secure unless otherwise noted.

2 Introduction to Stream Ciphers

This chapter gives some basic definitions and properties of stream ciphers in section 2.1 and explains several attack models in section 2.2. Both sections are based on definitive books on cryptography such as [MvOV01] and [Sch96].

2.1 Design Criteria

A stream cipher is defined as an algorithm for encryption and decryption of individual plaintext digits (e.g single bits or bytes), as opposed to block ciphers that encrypt blocks of a fixed bitlength. Usually they employ a symmetric key for generating a pseudo random cipher bit stream (keystream), which then is combined with the plaintext, typically using an XOR-operation[1]. Therefore basic stream ciphers provide only confidentiality without integrity protection. In the case of messages with a high amount of redundancy (like in natural language or in many data formats), error propagation may be sufficient to detect modifications to the message, but in general an additional cryptographic operation is needed to guarantee the integrity of a message.

Stream ciphers are especially important where data must be processed in real-time or where data comes in quantities of unknown length and padding or buffering must be avoided. The distinction between block and stream ciphers is not always clear, as block ciphers provide stream cipher properties in certain modes of operation, e.g. cipher feedback (CFB) mode, output feedback (OFB) mode and counter (CTR) mode. Furthermore a block cipher could be understood as a stream cipher with large characters, e.g. 64 bit instead of one bit or byte. Stream ciphers tend to be faster and easier to implement in hardware than block ciphers are. Even stream ciphers build from block ciphers can often be implemented more efficient than the block cipher on its own, because the block cipher decryption is not needed in most cases.

Stream ciphers are related to the one-time pad (OTP) that uses a key of the same length as the plaintext and has been proven to be theoretically secure. Stream ciphers deduce a keystream (endless and never repeating in the ideal case, but at least with a period that is much longer than the number of encrypted bits) from a shorter key to overcome the disadvantage of requiring such a long key. The output of the keystream generator at a specific time is determined by

[1] Stream ciphers that use the Exclusive-OR as output function are called binary additive stream cipher

an internal state. Both state and key must be impossible to recover by looking at the keystream, moreover it should be indistinguishable from random noise. From a theoretical point of view, analysing the security of a stream cipher can be narrowed down completely to the analysis of the pseudo randomness of the keystream.

Stream ciphers can be synchronous (also called key auto-key, KAK) or asynchronous (also called ciphertext autokey, CTAK).

In synchronous ciphers, the keystream is independent from plaintext and ciphertext, and therefore can be precomputed and will be unaffected by transmission errors. Bitflips in ciphertext affect only a single bit in the corresponding plaintext, which can be useful if the transmission error rate is high; besides, this property allows many error correcting codes to function normally even when applied before encryption. Measures need to be taken to allow sender and receiver to regain synchronisation after insertion or deletion of bits caused by transmission errors or an active attacker, for example utilisation of markers in the ciphertext at regular intervals, or by systematically trying offsets until messages decrypt correctly again. It is important to regain synchronisation without using part of the keystream twice.

By contrast, asynchronous stream ciphers are self-synchronizing, because they use n ciphertext characters to generate the keystream. Accordingly sender and receiver will be automatically synchronised after n ciphertext characters, and bit error propagation is limited to n plaintext characters. Since each character affects the entire following ciphertext, the statistical properties of the plaintext (e.g. frequency of letters in natural language) are disseminated through the ciphertext. Hence, asynchronous stream ciphers may be more resistant against attacks based on plaintext redundancy.

Most ciphers need an initialization vector (IV) that must be known to each involved party, for example as a replacement for a feedback value in the first round. In most cases an IV needs not to be secret, but may be subject to other requirements, such as unpredictability. One method to generate an unpredictable IV avoiding pseudo random number generators would be to generate a unique nonce (which may be a counter) and to encrypt it under the same key that is used for the encryption of the plaintext. Generation of IVs is discussed in [Dwo01] in greater detail.

2.2 Attacks

There are several attacks models for cryptanalysis, which can be applied to stream ciphers as well as block ciphers in stream cipher mode of operations. An attacker is assumed to have knowledge or control of plaintext or ciphertext and tries to deduce information about the key or plaintext. Other attacks aim to modify or replay messages without knowledge of the key.

Ciphertext-only attack

In general, ciphertext-only attacks belong to the hardest attacks for cryptanalyst, since they have no other information that the ciphertext. However, using a stream cipher key more than once together with the same initial state causes an identical keystream to be applied to different plaintexts. An attacker who applies the XOR operation to the resulting ciphertexts obtains the XOR of the corresponding plaintexts, which may allow him to find out the individual plaintexts and the keystream, especially if the plaintext messages are written in an natural language or if one of the streams consists entirely of null characters. The latter can occur in scenarios where continuous data streams are used to defeat traffic analysis, for example for military communication.

Another kind of an ciphertext-only attack is a bruteforce attack, where the attacker simply tries all possible keys. In most cases it is unproblematic to assume that he is able to distinguish a valid plaintext from random noise.

Known-plaintext attack

In many cases, the attacker knows part of the plaintext and the corresponding ciphertext, because of known structures like file headers that are encrypted together with the unknown parts of the plaintext. In the case of stream ciphers, the XOR operation of plaintext and ciphertext discloses the keystream to an attacker. If the same keystream is reused, an attacker is able to decrypt the unknown plaintext. Apart from that, he may be able to gain information about further parts of the keystream, if the keystream generator has weaknesses.

Insertion attack

Another example for an attack that exploits a reused keystream would be an insertion attack on an synchronous cipher. An attacker who knows a ciphertext and has the capability to make the sender encrypt the same plaintext with the same keystream again, but with one inserted bit *p'* that is known to the attacker, can decrypt the whole plaintext after the inserted bit. The keystream bit at position i after the insertion can be computed as $k_i = c_i \oplus p_{i-1}$ and the keystream bit at the position of the inserted bit is $k_0 = c_0 \oplus p'$.

Replay attack

As the keystream depends solely on the previous ciphertext bits in asynchronous ciphers, it is possible to perform replay attacks. After the first n replayed bits the receiver is synchronised to the attacker and can decrypt the old ciphertext without noticing that it is not fresh.

Substitution attack

An attacker can change the ciphertext in a way that may result in predictable changes in the plaintext. The ciphertext c_i is the XOR of plaintext p_i and keystream k_i. Consequently an attacker can construct some x_i and XOR it with c_i, which results in $enc_{k_i}(p_i) \oplus x_i = p_i \oplus k_i \oplus x_i = enc_{k_i}(p_i \oplus x_i)$.

Randomness and distinguishability

The capability to distinguish ciphertext from a truly random sequence does not give an attacker any knowledge about the plaintext or key and therefore cannot be called an attack, but it can be useful as a security criterion. Based on the work of Goldwasser and Micali[2], M. Bellare et al. presented four notions of security in [BDJR97] that deal with randomness and distinguishability:

- Real-or-Random (RoR) security: An encryption scheme is considered to be "good" if an adversary cannot distinguish between random messages and encrypted messages.
- Left-or-Right (LoR) security: An adversary sends two messages to an encryption oracle, that responds always with the left or always with the right message. The encryption scheme is "good" if the adversary is not able to determine whether the oracle encrypts the left or right message.
- Find-Then-Guess (FtG) security: An encryption scheme is "good" if after receiving a pair of equal-length plaintext messages and some state information, an adversary given the ciphertext of one of the plaintexts cannot determine which plaintext it belongs to.
- Semantic (SEM) security: An encryption scheme is semantically secure if whatever can be efficiently computed about the plaintext given the ciphertext can also be computed in the absence of the ciphertext.

[2]Probabilistic Encryption (1984)

3 Cipher Feedback Mode

This chapter deals with the Cipher Feedback Mode[1], giving its definition in section 3.1 and discussing its security in the case of DES-CFB with reduced round size in 3.2.

3.1 Definition and properties

In CFB, a block cipher is used to build a self-synchronizing stream cipher that provides confidentiality. It utilises a character size $r < n$, where n is the block size of the block cipher. For r-bit-CFB encryption, r bit of the ciphertext of the previous cycle are fed back as input for block cipher encryption through an n-bit shift register. The plaintext is XORed with r bits of the block cipher output, while the remaining n-r bits are discarded. Decryption works similar; note that for both encryption and decryption, the block cipher is used in encryption mode[2].

For the first input block, CFB mode requires an IV, which does not need to be secret, but must be unpredictable.[3] From the second input block on, the input block I_i is the concatenation of the r-bit leftshifted previous input block and the previous ciphertext. The output block is the block cipher encryption of the input block, and the ciphertext is computed as XOR of Plaintext P_i and the r most significant output block bits. For decryption, the ciphertext is XORed with the output block again to cancel out the keystream and produce the original plaintext.

Definition 3.1.1 *Cipher Feedback Mode*

$I_i = LSB_{n-r}(I_{i-1})|C_{i-1} \quad (I_1 = IV)$
$O_i = enc_K(I_i)$
$C_i = P_i \oplus MSB_r(O_i)$
$P_i = C_i \oplus MSB_r(O_i)$

As each input block depends on the previous ciphertext, CFB encryption cannot be parallelised, except when interleaving[4] is used. It is possible to perform

[1] Based on the description in [Dwo01]

[2] Asymmetric cryptography uses a public encryption key, hence it cannot be used for CFB mode, but it would not be interesting due to efficiency reasons anyway.

[3] However, in [VSK83] Voydock and Kent put higher demands on the IV, in particular they claim that it must be protected from disclosure.

[4] Interleaving uses multiple encryption chains that can be executed in parallel to increase speed.

CFB decryption in parallel if the input blocks are assembled first (in series) from the ciphertext and the IV.

Changes to order or value of a ciphertext affects decryption due to chaining dependencies until the erroneous r-bit block has shifted entirely out of the input block register. This self-synchronization occurs after $\lceil \frac{n}{r} \rceil$ blocks. The decrypted error block will differ from the original plaintext exactly in those bit positions where the error occurred; the following affected blocks will typically be random.

Identical plaintexts are encrypted into identical ciphertexts if the same key and IV are used.

CFB mode discards $n-r$ bits of each block cipher encryption, thereby decreasing throughput by a factor $\frac{n}{r}$ compared to ECB or CBC mode.

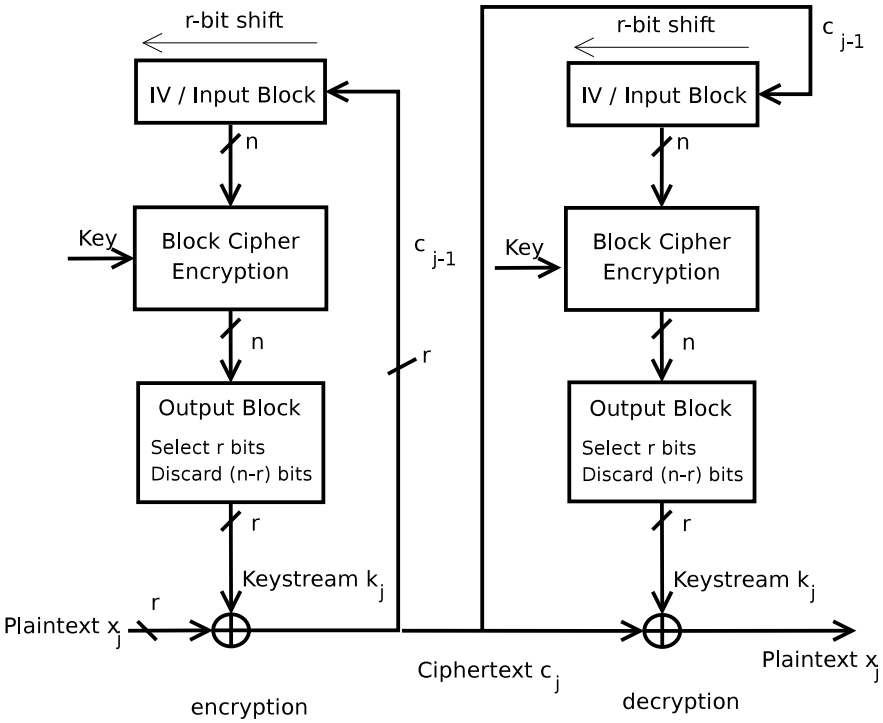

Figure 3.1: Cipher Feedback Mode

3.2 Security aspects

One approach to improve the efficiency of CFB mode with a small symbol size (e.g. 1-bit-CFB) could be to reduce the number of rounds of the underlying block cipher. Preneel et al. demonstrate in [PNB94] that reducing the round size of

For example, one chain could encrypt every second bit, while the other chain encrypts every other bit. Both could use the same key, but need a different IV.

DES in CFB mode allows an attacker to gain knowledge of several key bits via differential or linear attacks.

- Standard DES[5] has 16 rounds (denoted by N in the following), a key size of 56, a block length of 64, and uses S-Boxes for diffusion of the effect of the key bits. In DES with less than 5 rounds, due to the selection of the leftmost bits in r-bit-CFB with r less than 64, the number of key bits that influence the ciphertext can be considerably lower than 56. This is not only caused by the S-Boxes, but also by the initial (IP) and final (IP^{-1}) permutation in DES, that normally has no cryptographic meaning and was probably introduced to facilitate hardware implementation.

- Chaum and Evertse presented a meet in the middle attack on DES with a reduced number of rounds in [CE85] that is faster than exhaustive search for $N < 6$. Preneel et al. show that this attack is significantly harder for r-bit-CFB with r less than 64.

- As only a part of the output is known in the CFB mode, the conventional differential attack on DES fails, but the authors propose an extended differential attack that discloses part of the key to an attacker. The exact number of disclosed bits varies for different N and r, e.g. 3 bit for $r = 8$ and 6 bit for $r = 16$ for $N = 4$ rounds. Again, this attack is influenced by the final permutation: without IP^{-1} only information on the output of S-boxes of the last round would be available and the attack would fail.

- The linear attack discovered by Mitsuru Matsui in 1993 using known plaintexts can be applied to CFB without much modifications and reveals 7 bits of the key for DES with 8 rounds.

Moreover the authors state that the most powerful attack on CFB mode is a chosen ciphertext attack. For r-bit CFB, known and chosen plaintext attacks are equivalent as in both cases the attacker has no control over the input of the block cipher; however, a chosen ciphertext attack on CFB corresponds to a chosen plaintext attack in ECB mode, because the ciphertext is fed back into the block cipher. Nevertheless, a chosen ciphertext attack on CFB is harder, because the cryptanalyst can observe only r bits of the block cipher output.

[5]Defined in [FIP93]

4 Output Feedback Mode

In this chapter we define the Output Feedback Mode[1] in section 4.1 and present the results of several analyses in section 4.2.

4.1 Definition and properties

OFB is a confidentiality mode that is build very similar to CFB, but differs in properties like error propagation. The reason is a different feedback structure: As the name implies, OFB uses the previous output of the block cipher instead of the previous ciphertext. That makes the ciphertext independent of plaintext and ciphertext, and therefore identifies OFB as a synchronous stream cipher.

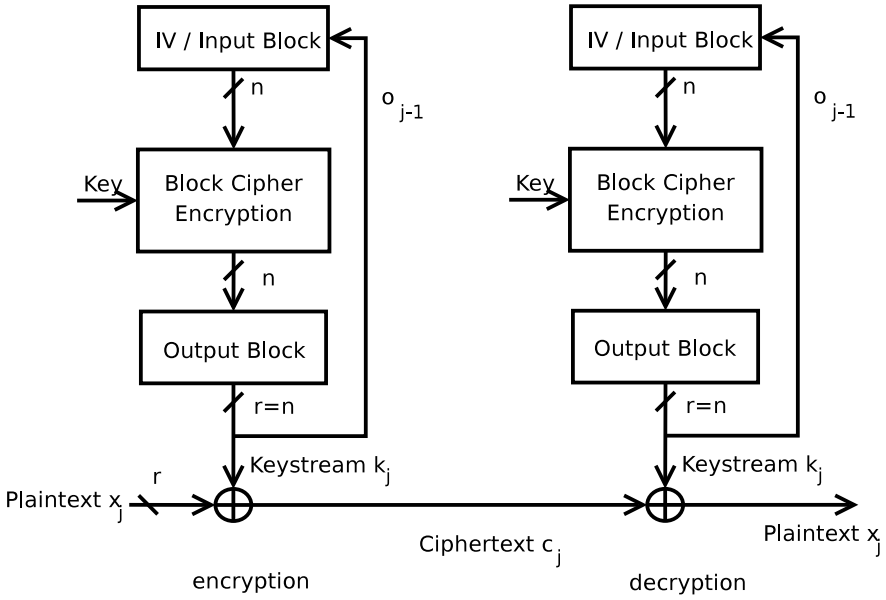

Figure 4.1: Output Feedback Mode

Definition 4.1.1 *Output Feedback Mode*
$I_i = O_{i-1} \quad (I_1 = IV)$
$O_i = enc_K(I_i)$

[1]Based on the description in [Dwo01]

$C_i = P_i \oplus O_i$
$P_i = C_i \oplus O_i$

In the first cycle, an IV is encrypted by the block cipher and the resulting output block is XORed with the plaintext to produce the ciphertext. For the subsequent cycles, the IV is replaced by the previous output block. For an incomplete final block the remaining bits are discarded. As OFB is a synchronous stream cipher, keystream generation is exactly the same for encryption and decryption and can be pre-computed prior to the availability of the plaintext or ciphertext data if the IV is known. Multiple cycles cannot be performed in parallel, because each block cipher encryption requires the previous block cipher output as input. Interestingly it is possible to obtain an OFB mode keystream by using CBC mode with a constant string of zeroes as input.

Like in CFB, OFB requires a unique IV for every message, because the confidentiality of a message may be compromised if a combination of a previously used key and IV is reused. Encrypting identical plaintexts with different IVs results in different ciphertexts.

In contrast to CFB, single bit errors in the plaintext complement only the corresponding bits in the ciphertext and do not affect further blocks. This property, together with the possibility to make OFB operate at very high speeds makes OFB particularly well suited for the encryption of data streams like voice and video, especially on noisy channels where error propagation could easily render encrypted transmission nearly impossible.

OFB requires both parties to be synchronous, thererfore loss or insertion of bits destroys the alignment until explicit re-synchronization is enforced. Jueneman [Jue82] names multiple solutions for this problem, sending synchronisation signals at predetermined intervals being the most prominent.

CBC mode and OFB with full feedback both operate on the same block size, for example 64 bit for DES. Still, OFB is considered to be a stream cipher mode of operation, while CBC is considered to be a block cipher. It is possible to understand CBC as a stream cipher with very large character size, but unlike OFB, CBC always has to wait for completion of a whole block before it can be applied. However, for OFB the keystream - although composed of blocks of the same size as in CBC - can be used for single bits without the need for an plaintext buffer.

4.2 Security aspects

This section deals with several security-relevant properties of OFB mode.

Average cycle length for r-bit-OFB

For the reasons given in 2.2, the security of stream ciphers depends on the unpredictability of the keystream. In the case of OFB with DES (which operates on blocks of 64 bit) the keystream generator is a finite state machine with 2^{64} states, hence it must repeat after 2^{64} states or less. In [DP82], Davies and Parkin analyse the conditions under which the maximum average cycle length is can be achieved.

In OFB with full feedback (as shown in fig. 4.1), the keystream is generated by the repeated application of the block cipher encryption, which is effectively a random choice among all $2^{64}!$ permutations. The actual randomness of the keystream is in fact less important than the cycle size; and as the encryption function has an unique inverse, namely decryption, for each key all states out of the possible 2^{64} states must be member of one single disjoint cycle. The authors consider the resulting average cycle length of approximately 2^{63} to be "large enough for all practical purposes", as it is very near to the obtainable maximum. Smaller cycles are possible but improbable; for example the probability of cycles of 10^6 states or less is 2^{-44}.

If OFB is used with feedback limited to $r < n$ bits (where n is the block size of the block cipher), the rightmost $n - r$ bits are discarded and the remaining bits become the next keystream bits and also get placed into the shift register that serves as input block for r-bit-OFB. In this case, a mathematical model of the keystream generator is necessary to estimate the average cycle length, but the requisite assumptions would have been difficult to justify on their own. Therefore Davies and Parkin carried out an experiment that reduced DES to a randomly chosen permutation of 256 states of an 8 bit register. For each value of r, they performed a test with 10000 permutations and calculated the cycle length distribution.

The experiment confirmed their theoretical results, showing that OFB with any other value than $r = 64$ is of greatly inferior security, since the average cycle length is reduced by a factor of 2^{32} or more. Davies and Parkin propose that $r = n$ should be the only recognised OFB mode, as opposed to the specification in [FIP80]. There is no advantage in using OFB with other values; furthermore, r-bit-OFB decreases throughput as per CFB mode.

Error detection and integrity protection

As OFB is a synchronous stream cipher, changes to individual bits in the ciphertext do not affect the encryption of other parts of the ciphertext. This lack of error propagation is regarded as a disadvantage in some cases, since it allows an adversary to make predictable changes to the deciphered plaintext and "thus hampers development of mechanisms that detect message-stream modification

attacks."[2] It is important to understand that the authors refer to error detection codes that were designed to detect random bit errors caused by the transmission channel instead of changes made by an adversary with intent to obtain a message that generates the same error code value. Error detection codes may be sufficient to realise integrity protection in modes like CFB or CBC which both have a large error propagation, but are not adequate for synchronous stream ciphers. Voydock and Kent conclude that therefore "the OFB mode is not suitable as a basis for communication security for most applications", regardless of the purpose of OFB that is only confidentiality.

Jueneman proposes in [Jue82] the usage of message segment numbers and a checksum or hash function to protect messages against modification, but points out that the Modification Detection Code (MDC) proposed in the then-current drafts of the Federal Standard 1025 and 1026 are an example for a failed protection. He suggests using a message authentication code (MAC) for maximum security and shows that it is necessary to compute MAC and encryption with two separate encryption operations and different keys or at least different IVs. To avoid this effort, he introduces a quadratic residue checksum that involves the value of every bit of a message as well as its position and includes a secret, random seed. However, in many cases relevant to OFB, the type of traffic and the physical characteristics of the transmission medium may already provide an adequate integrity protection.

Resynchronisation and key change schedule

A cycle produced by OFB from an IV and a key shares no points with any other cycle that could be produced with that key due to the existence of an unique inverse to the encryption function. Randomly choosing a new IV in order to avoid the possibility of short cycles or to restart or resynchronise a communication increases the risk of using an IV that overlaps with a previously used cycle. Choosing an IV unique instead of randomly, for example by using an incrementing counter, would guarantee that no two cycles will be generated that begin at the same point, but they still could partly overlap. Based on these considerations, Jueneman calculates the probability of repetition under various conditions and presents the results in [Jue82]. According to his calculations, the IV can be changed randomly up to 10000 times, before the probability of repetition increases rapidly. Hence, data encryption keys for OFB should be changed after 10000 reinitializations, four billion DES cycles, or one day, whichever comes first. If data encryption keys are derived from a master key, it should be changed weekly to monthly.

If resynchronisation does not involve a newly generated IV, care must be taken that the keystream is not applied twice to different plaintexts.

[2] Voydock and Kent in [VSK83]

5 Counter Mode

Section 5.1 gives the definition of Counter Mode[1] and section 5.2 discusses its practical usability.

5.1 Definition and properties

CTR mode is a confidentiality mode similar to OFB mode that features a random-access property, since it utilises a counter instead of the previous ciphertext or output block for updating the block cipher input block. It is not necessary to use a literal counter: any sequence that deterministically generates unique values for a long period starting from an appropriate initial value can be used, for example an Pseudo Random Number Generator (PRNG). As using identical counter values twice with the same key on different messages compromises the confidentiality of the corresponding plaintext blocks, the counter blocks must be unique, i.e. the incrementing function should have a big cycle length.[2]

For the same reasons as in the previously discussed modes, CTR mode needs an IV that is for example XORed with the counter as shown in figure 5.1 or used as a part of the initial counter value. The result is referred to as *ctr* and is used as input block for the block cipher. Lipmaa et al. give several other options for forming *ctr* in [LRW00], e.g. deriving it from other protocol elements or even maintaining a separate secure channel to synchronize on *ctr*, and state that a well-designed standard for CTR mode should describe a small number of recommended ways, because there is no single method that is best in all situations.

Definition 5.1.1 *Counter Mode*
$ctr = Counter_i \oplus IV$ *(example)*
$O_i = enc_K(ctr)$
$C_i = P_i \oplus O_i$
$P_i = C_i \oplus O_i$

CTR mode is fully parallelizable since it does not use feedback, thus allowing the use of aggressive pipelining, multiple instruction dispatch per clock cycle and other architectural features that enhance the performance. Pre-computation of

[1] Based on the description in [Dwo01]
[2] The choice of an incrementing function and initial counter values is discussed more detailed in [Dwo01]

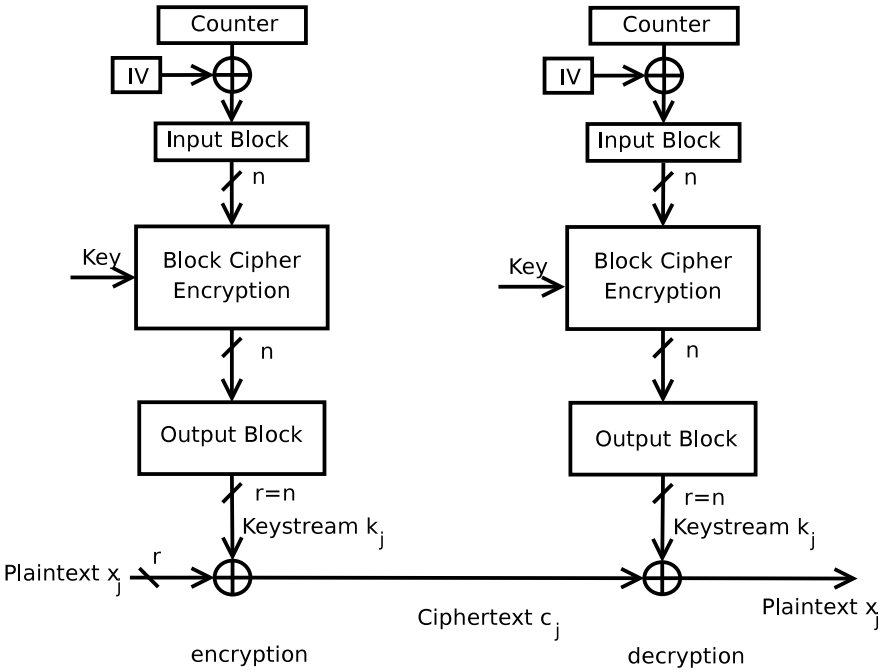

Figure 5.1: Counter Mode

the keystream prior to the availability of the plaintext or ciphertext can be used to further increase speed. To allow encryption and decryption to be done out of order, each block must be tagged with an index to obtain the needed counter value.

Error properties for CTR mode are identical to those of OFB mode, since both are synchronous stream ciphers. So it is not surprising that CTR mode raised similar discussions on possible disadvantages of the lack of error propagation as OFB did. Lipmaa et al. respond to this by pointing out that with respect to integrity protection, the error properties are "neither good nor bad, but just irrelevant". They contradict the often stated opinion that synchronous stream ciphers have stronger requirements on integrity protection or error detection than for example CBC, because protection should be "accomplished at a different architectural level" if needed. In general, other modes of operation do not provide any meaningful message integrity either.

Variations of CTR mode include the so called Dual Counter Mode (DCM) proposed by Boyle and Salter in [**?**] for high-speed encryption of IP-packages. It is described as a hybrid of ECB mode and CTR mode and was designed to cope with out-of-order packages and to allow a simple integrity check. But in [DGW01] the authors prove that both variants of DCM can neither protect secrecy nor integrity in the case of an chosen-plaintext attack. They come to the conclusion that there seems to be no simple way to solve these problems without sacrificing the performance gains.

5.2 Security aspects

This section deals with several security-relevant properties of Counter mode.

Usage of a deterministic input function

The usage of a deterministic counter as input for the block cipher encryption has raised continuing controversial discussions.

Jueneman's statement written in [Jue82] in 1982, three years after the counter mode has been proposed by Diffie and Hellman and and nearly 20 years before it was finally accepted by NIST[3], is a typical example for critique on CTR mode: "But the author's intuitive feeling is that to deliberately expose any cryptosystem to a known systematic input [...] would represent an unnecessary risk. [...] It does not seem wise to do part of the attacker's work for him." This point of view is based on the experience that predictability is a common point of failure for cryptosystems, and on the assumption that an incrementing counter as block cipher input cannot create enough entropy.

Lipmaa et al. address these critiques in their paper on counter-mode encryption [LRW00], stating that "most of the perceived disadvantages of CTR mode are not valid criticisms, but rather caused by the lack of knowledge". For example they remark that the small Hamming difference between successive input blocks in case of an incrementing counter does only facilitate a differential cryptanalysis if the underlying cipher is differentially weak. The authors advance the view that a mode of operation is not responsible to compensate such a weakness, and would most likely not be able to compensate it anyway. Furthermore, the small Hamming difference can easily be avoided by using a counter function that does not advance the counter value by incrementing.

Tirtea and Deconinck present a fault attack in [TD04] that is particularly successful under the assumption that an incrementing counter is used. A fault model assumes that due to hardware faults a specific bit is flipped randomly from time to time or permanently. Fault attacks were introduced 1996 by Boneh, Demillo and Lipton and take advantage of hardware failures in a way that reduces the confidentiality of a cipher or to affect digital signatures and identification schemes. Failures can be provoked by an adversary for example by cutting a wire or destroying a single memory cell. In the case of CTR, a failure that permanently affects the least significant counter bit would result in the violation of the uniqueness requirement for counter values, because pairs of successive ciphertext blocks are then encrypted under an identical keystream.

[3]National Institute of Standards and Technology

Provable security

In 1997, Bellare et. al published a paper [BDJR97] that extended the pioneering work of Goldwasser and Micali on concrete security notions to the aspects named in 2.2 and applied them to CBC mode and a so-called XOR scheme to establish tight bounds on the success of adversaries depending on their resources. They discuss a stateless and a stateful model of the XOR scheme, the latter of them being similar to CTR mode. An updated version of the paper from 2000 renamed the stateful XOR scheme to the usual CTR mode, giving strong evidence for the security of CTR mode.

Sung et. al propose two multiple modes in [SLL+02] called CTR-OFB and CTR-CFB and claim that they retain the security properties attested by Bellare et. al, but achieve higher resistance against some practical attacks. To my knowledge, no further research has been done on this modes.

Disk encryption in CTR mode

Its random-access property makes counter mode a promising candidate for hard disk encryption, because it would significantly facilitate access to non-consecutive blocks. Nevertheless it is not suitable for hard disk encryption[4], because an adversary may very well be able to gather different encrypted versions of the same block due to to file system or caching mechanisms or because he has access to the disk before and after a change to a block. Again, the XOR of different ciphertext versions may allow him to decipher the individual plaintexts.

[4]Stated in [Fru05] without giving reasons

6 Alternatives

This chapter describes alternative modes of operation, first the Key Feedback Mode in section 6.1 and secondly the Plaintext Feedback Mode in section 6.2.

6.1 Key Feedback Mode

KFB is a rarely used confidentiality mode that uses the block cipher output block to form the block cipher key of the next round. As the keystream is independent of plaintext and ciphertext, KFB turns a block cipher into a synchronous stream cipher and therefore shares most of the error properties and some other characteristic with OFB. KFB outputs m bits (typically $m = 8$) at a time and uses a constant bitstring p as block cipher input, a key k of n bit length, and a $n \times m$ matrix with non-zero rows as IV. It is possible to reduce the size of the matrix and it does not need to be secret but must be random. If the output block size of the block cipher is not equal to the key size, a function is needed to form a valid key from the output block.

The concept of KFB has already been mentioned 1982 by Hellman and Reyneri [HR82], but only as a theoretical construct to analyse the cycle length of random functions with the purpose of comparing DES to a truly random function. In 2000, Håstad and Näslund published a paper [Fir00] that proves KFB to be secure in terms of the theorems of complexity theory introduced by Blum, Micali, Levin and Goldreich, giving a quantitative relation between the effort of distinguishing the keystream from true randomness to the effort of retrieving the secret key. The authors name several disadvantages compared to the standard modes of operation:

- The speed is a bit lower than for OFB, mainly because the key schedule has to been done for each key
- It is slightly more complex than OFB and CFB
- The average cycle length is $2^{n/2}$

Definition 6.1.1 *Key Feedback Mode*
$O_i = f(enc_{K_i}(Constant), IV)$
$K_i = g(K_{i-1}) \quad K_0 = k$
$C_i = P_i \oplus O_i$
$P_i = C_i \oplus O_i$

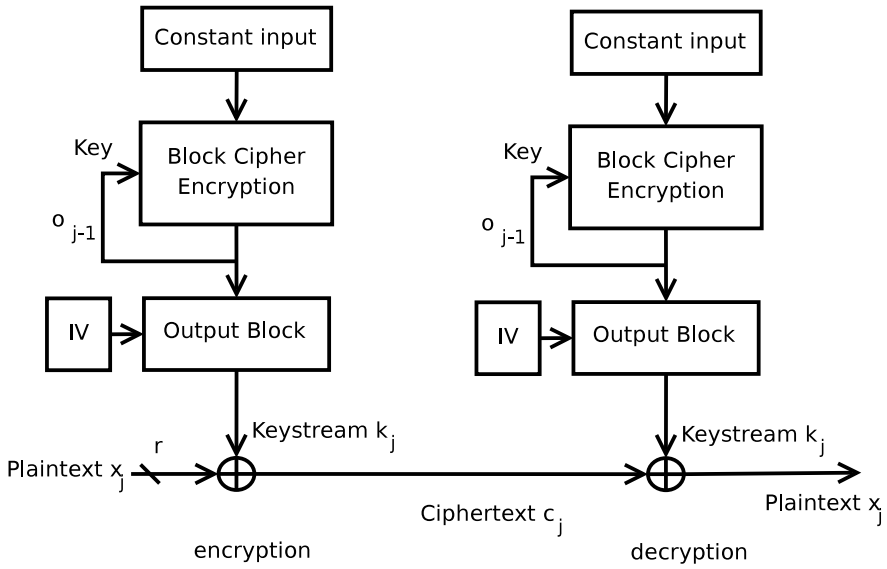

Figure 6.1: Key Feedback Mode

6.2 Plaintext Feedback mode

PFB is a rarely used confidentiality mode that works similar to CFB mode, but feeds plaintext instead of ciphertext back into the block cipher. As the keystream is a function of the previous plaintext, PFB turns a block cipher into a self-synchronizing stream cipher and therefore shares most of the error properties and some other characteristic with CFB.

Definition 6.2.1 *Plaintext Feedback Mode*
$I_i = P_{i-1} \quad (I_1 = IV)$
$O_i = enc_K(I_i)$
$C_i = P_i \oplus MSB_r(O_i)$
$P_i = C_i \oplus MSB_r(O_i)$

Schneier mentions PFB in [Sch96], but recommends not to use it because it has no significant advantages over well-researched modes. For example, it resists known-plaintext attacks, but allows chosen-plaintext attacks, because an adversary could launch a chosen-plaintext attack choosing two identical successive plaintext blocks to cause them to be enciphered by an identical keystream. To my knowledge, no in-depth literature on PFB exists but only some newsgroup contributions[1] discussing its properties.

[1] [Gol01]

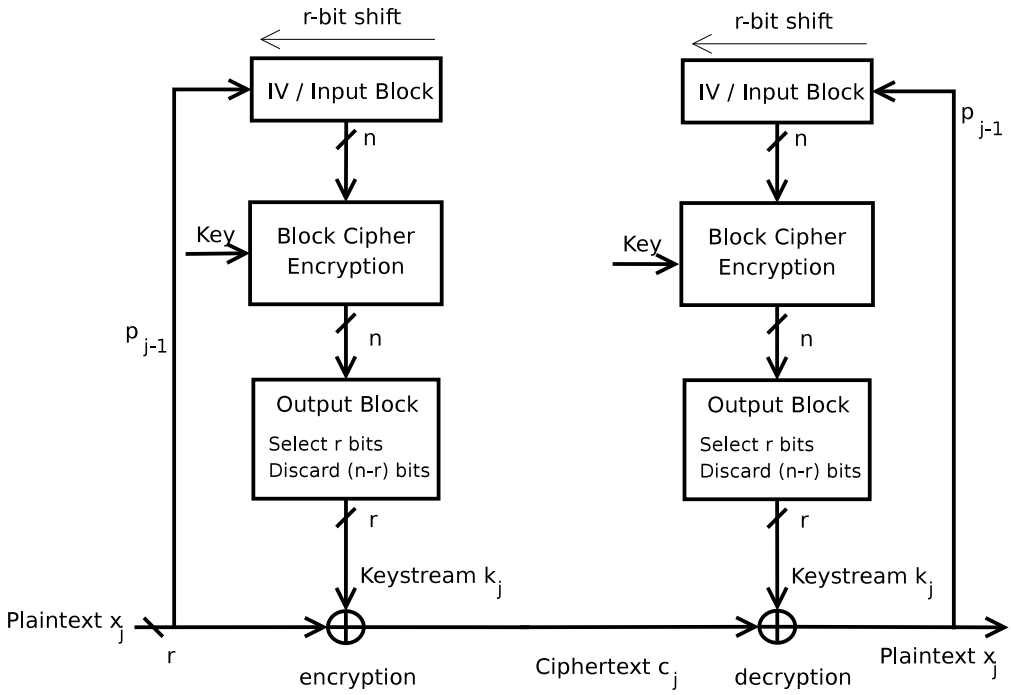

Figure 6.2: Plaintext Feedback Mode

7 Conclusion

As an overview of the state of the art of stream ciphers build from block ciphers, this paper presented the standard modes of operation along with some uncommon modes. Apart from pointing out methods of exploiting and avoiding the most common design errors, we introduced techniques such as the analysis of average cycle lengths and the security notions for the definition of security bounds.

The categorization into synchronous and asynchronous stream ciphers proved helpful as a first estimate for the properties of a stream cipher and hence can be used as a starting point for the choice of algorithms during the development of a cryptosystem.

Though it is often advisable to stick to the standard modes, the counter mode can be seen as an example for an algorithm that was sceptically received for a long time, but finally convinced most experts and made its way into the standards. For the two alternative modes examined in the last chapter this seems improbable in case of PFB mode, but cannot be ruled out in case of KFB.

Bibliography

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. A concrete security treatment of symmetric encryption. pages 394–403, 1997.

[CE85] D. Chaum and J.-H. Evertse. Cryptanalysis of DES with a reduced number of rounds. 1985.

[DGW01] Pompiliu Donescu, Virgil D. Gligor, and David Wagner. A note on NSA's dual counter mode of encryption. Technical report, 2001.

[DP82] D. W. Davies and G. I. P. Parkin. The average cycle size of the key stream in output feedback encipherment. In *Advances in Cryptology, Proceedings of CRYPTO 82*, pages 263–282, 1982.

[Dwo01] Morris Dworkin. Special publication 800-38a: Recommendation for block cipher modes of operation, December 2001.

[FIP80] FIPS. DES modes of operation. Technical report, 1980.

[FIP93] FIPS. Data encryption standard (DES). Technical report, 1993.

[FIP01] FIPS. Advanced encryption standard (AES). Technical report, 2001.

[Fir00] First Modes of Operation Workshop for Symmetric Key Block Ciphers, Baltimore, Maryland, USA. *Key feedback mode: A keystream generator with provable security*, October 2000.

[Fru05] Clemens Fruhwirth. New methods in hard disk encryption. Master's thesis, Institute for Computer Languages, Theory and Logic Group, Vienna University of Technology, 2005.

[Gol01] Benjamin Goldberg. *Cryptography-Digest 219*. April 2001. Downloaded at 2009-01-15 from http://www.mail-archive.com/cryptography-digest@senator-bedfellow.mit.edu/msg05413.html.

[HR82] Martin E. Hellman and Justin M. Reyneri. Drainage and the DES. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 129–131, 1982.

[Jue82] Robert R. Jueneman. Analysis of certain aspects of output feedback mode. In *Advances in Cryptology, Proceedings of CRYPTO 82*, pages 99–127, 1982.

[LRW00] Helger Lipmaa, Phillip Rogaway, and David Wagner. *Comments to NIST concerning AES modes of operation: CTR-mode encryption.* 2000.

[MvOV01] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 2001.

[PNB94] Bart Preneel, Marnix Nuttin, and Johan Buelens. Cryptanalysis of the cfb mode of DES with a reduced number of rounds. In *Advances in Cryptology, Proceedings of CRYPTO 93*, pages 212–223. Springer-Verlag LNCS, 1994.

[Sch96] Bruce Schneier. *Angewandte Kryptographie . Protokolle, Algorithmen und Sourcecode in C.* Addison-Wesley, May 1996.

[SLL+02] Jaechul Sung, Sangjin Lee, Jongin Lim, Wonil Lee, and Okyeon Yi. Concrete security analysis of ctr-ofb and ctr-cfb modes of operation, 2002.

[TD04] R. Tirtea and G. Deconinck. Specifications overview for counter mode of operation. security aspects in case of faults. In *Electrotechnical Conference, 2004. MELECON 2004. Proceedings of the 12th IEEE Mediterranean*, pages 769–773 Vol.2, 2004.

[VSK83] Victor L. Voydock, Stephen, and T. Kent. Security mechanisms in high-level network protocols. *ACM Computing Surveys*, 15:135–171, 1983.