Differential Attacks and Stream Ciphers

Frédéric Muller

DCSSI Crypto Lab 51, Boulevard de Latour-Maubourg
75700 Paris 07 SP France
`Frederic.Muller@sgdn.pm.gouv.fr`

Abstract. Differential Cryptanalysis (DC) is a famous technique, initially introduced by Biham and Shamir in 1990 to attack DES-like algorithms. Since then, it was successfully used against many cryptographic primitives and the underlying mechanisms are now well understood.
In the case of stream ciphers, DC has received less attention while other cryptanalysis techniques, for instance correlation attacks or algebraic attacks, have been analyzed in detail. Still, techniques based on DC have recently proved useful to break various stream ciphers. In this paper, we propose a survey of situations where DC may apply to stream ciphers. Many recent examples are used to illustrate these attacks and we discuss how to protect future designs.

1 Introduction

Symmetric key cryptosystems are generally separated into two main families of primitives :

– **Block Ciphers** handle the plaintext by blocks of data of fixed length (typically 64 bits or 128 bits for the NIST standards DES [37] or AES [38]). A mode of operation must then be specified in order to encrypt messages of arbitrary length.
– **Stream Ciphers** handle each bit of the plaintext separately. Generally an arbitrarily long, pseudo-random sequence of bits (called keystream) is generated from a short secret key (of typical length 128 bits). This sequence is then added bitwise to the plaintext in order to form the ciphertext. Therefore the keystream must be of the same length as the message to encrypt. This general structure is represented in Figure 1. Other constructions exist but most algorithms follow the basic framework.

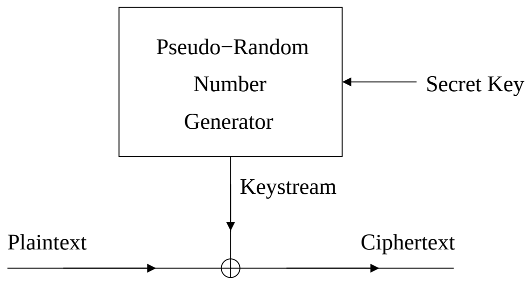

Fig. 1. Usual structure of a stream cipher

While the security of block ciphers is well understood (modern designs usually resist all known attacks), many problems subsist in the field of stream ciphers. Indeed most recent proposals have been, at least theoretically, broken quickly after they were published. An illustration was given when the European NESSIE project [35] chose not to select any stream cipher in its final portfolio of primitives [36]. Although all published attacks are not practical, the techniques used are generally not new, therefore these recent results cast a shadow on the security of stream ciphers. We believe recent attacks should be analyzed carefully, in order to soundly protect future designs against all known attacks.

Unfortunately, the panel of attacks against stream ciphers is quite wide. On the one hand, some techniques are specific for a certain class of algorithms. For instance, correlation attacks [41] have been historically developed to target LFSR-based constructions. One can also mention irregularly-clocked algorithms [6, 23], for which specific attacks exist [14, 20, 22]. On the other hand, some techniques are fairly general (*e.g.* guess-and-determine attacks, time-memory trade-off attacks [4], algebraic attacks [7], etc ...). An interesting observation is that many of these "generic" techniques have been developed after initial progress in the field of block ciphers cryptanalysis [8, 25].

In this paper, we focus our attention on Differential Cryptanalysis (DC). This technique was initially proposed against block ciphers [2, 3], but its applicability to stream ciphers has been demonstrated by several recent results [17, 28, 27, 34, 45]. The general idea of DC is to observe how the cipher behaves regarding a chosen difference on its inputs (in general, this input is the plaintext). Classically, DC was not considered a threat for stream ciphers. Indeed, in constructions using a keystream generator[1] (see Figure 1), flipping one plaintext bit results in flipping the corresponding ciphertext bit, independently of the key. Hence the output difference does not leak any secret information. Unfortunately PRNG have many practical limitations, therefore new designs often propose additional features (support for initialization vectors, self-synchronizing mode, authenticated encryption mode, etc ...). These features open a door for more sophisticated attacks.

In the next section, we remind the general principles of DC. Then we explore the situations where DC can be applied against stream ciphers. Various examples are given to illustrate this point. Finally, we propose some guidelines to protecte future designs.

2 Differential Cryptanalysis against Block Ciphers

DC was introduced in 1990 by Biham and Shamir [2] to attack DES-like block ciphers. Quickly after, this attack was extended to the first cryptanalysis of DES faster than exhaustive search [3]. Since then, DC was used to attack a large panel of block ciphers. However, the underlying principles are now well understood and modern designs have been protected against DC by construction, either using strong heuristic arguments [12] or formal proofs of security [44].

The idea behind DC is to use a "dual" point of view compared to classical cryptanalysis techniques. Instead of looking at the real values manipulated by the cipher, one focuses on the difference[2] between internal values. In general, it is assumed that an attacker can obtain the encryption of two messages with a chosen input difference. His hope is that the resulting output difference will reveal some information about the secret key.

A crucial observation is that linear components have a simple differential behavior. For instance, when a linear function $\mathcal{L}$ is applied to some input X by $Y = \mathcal{L}(X)$, an input difference Δ will correspond to the output difference Δ', such that :

$$\Delta' = \mathcal{L}(X \oplus \Delta) \oplus \mathcal{L}(X) = \mathcal{L}(\Delta)$$

So the relation between Δ and Δ' is linear and it is independent of the input X. In particular, subkey additions have the nice property to keep the differences unchanged.

[1] also called Pseudo-Random Number Generator (PRNG)

[2] In general we consider XOR-differences, *i.e.* the difference between two variables a and b is $a \oplus b$. However in some particular situations we might consider "real" integer differences (of the form $a - b$)

Unfortunately things are more complicated for non-linear components. For instance, the differential behavior of S-boxes can generally be predicted only through statistical approximations. One can express pairs of input/output difference $(\delta_{in}, \delta_{out})$ with their corresponding probability p (also see Figure 2). This is generally referred to as a **differential characteristic**.

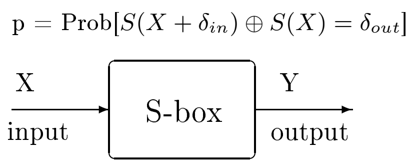

Fig. 2. Differential approximation of a S-box

The goal of the cryptanalyst is to combine such approximations over several rounds. Ultimately one hopes to find the best possible **characteristic for the full cipher**. In the case of DES, there are differential characteristics with probability up to $\frac{1}{4}$ for one round (not taking into account the trivial $\delta_{in} = 0 \rightarrow \delta_{out} = 0$ with probability 1). The resulting attack against the full DES has been proposed in 1992 [3]. Its complexity is about 2^{47} chosen messages.

3 Situations where DC can be applied to stream ciphers

In the case of stream ciphers, DC is not a traditional topic of research. The main reason is that for "classical" designs of stream ciphers (with an autonomous PRNG), the difference between two ciphertexts reveals nothing about the secret key (it is exactly the difference between the two plaintexts). But PRNG suffer from practical limitations, so we need to slightly change this point of view.

3.1 Limitations of Classical Constructions

The main limitations encountered with basic PRNG's are :

– **Encrypting several messages.** It is insecure to re-use the same key for several messages. Indeed, the PRNG is a deterministic mechanism, thus the same keystream sequence is always produced for a given key. Thus an attacker can learn the XOR between two plaintexts encrypted with the same key. A classical solution is to randomize the encryption, using an Initialization Vector for instance (see Section 3.2).
– **Synchronization Problems.** It is important that the sender and the receiver always stay synchronized in order for the encryption/decryption mechanisms to work fine. In practice, this problem of synchronization is not always easy to solve and some weaknesses can even raise during the resynchronization process [11].
– **Authenticated Encryption** In most applications, one would like to guarantee the integrity of the communication, in addition to protecting its confidentiality. This generally requires additional mechanisms.

Therefore new stream ciphers often propose one or several additional features to avoid these limitations. New attack scenarios result from this evolution.

3.2 Initialization Vectors

Initialization Vectors (IV) are used to allow encryption of several messages with a single key. An IV can be seen as a "salt" used to randomize the output of the PRNG (see Figure 3). The security requirement resulting from this additional input has been long overlooked. Security flaws resulting from improper use of IV's have been frequently observed, even when the underlying algorithm was still considered robust (see the example of RC4 [17, 31]) and the practical consequences have often been devastating [1, 43].

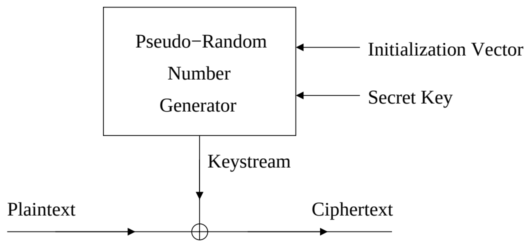

Fig. 3. Usual structure of a stream cipher with IV support

The IV is generally sent into the clear right before the ciphertext. Thus an attacker might learn the keystream generated with several known IV's. Then he might exploit the cipher behavior regarding IV difference to learn secret information. Such an attack scenario is referred to as **Known IV Differential Cryptanalysis**.

Moreover one can even envisage that the IV is chosen by the attacker, in particular situations. Then we speak of **Chosen IV Differential Cryptanalysis**. This is more difficult to mount in practice than a known IV attack, however reasons why this threat should be taken into account are :

– Implementation of IV's is not always well specified. They may be generated by an unspecified source of randomness, or come from a counter. Hence it is possible that an attacker could control, at least partially, the generation of IV's.
– In a man-in-the-middle attack, the attacker could intercept the communications and modify the IV on the fly. For instance if a given message is sent to two different users, the attacker could change the IV sent to the first user, and then compare the two decryptions of the message.

In general, it is widely considered that new algorithms should be resistant against chosen IV differential attacks (which further implies resistance against known IV attacks).

3.3 Self-Synchronizing Algorithms

To prevent synchronization problems, a possible solution is to use a Self Synchronizing Stream Cipher (SSSC). This primitive is usually obtained by adding a ciphertext-feedback to the PRNG. For instance the Ciphertext FeedBack (CFB) [16] mode of operation for block ciphers, with 1-bit feedback is an example of SSSC. Unfortunately it is not very efficient in terms of encryption speed, since one full block encryption is required to produce one keystream bit, so dedicated mechanisms are often preferred [32]. Without loss of generality, a SSSC can be represented as in Figure 4.

The function F applies only to the last m ciphertext bits (which are stored in a register) and to the secret key (possibly with an initialization vector). Therefore, as long as

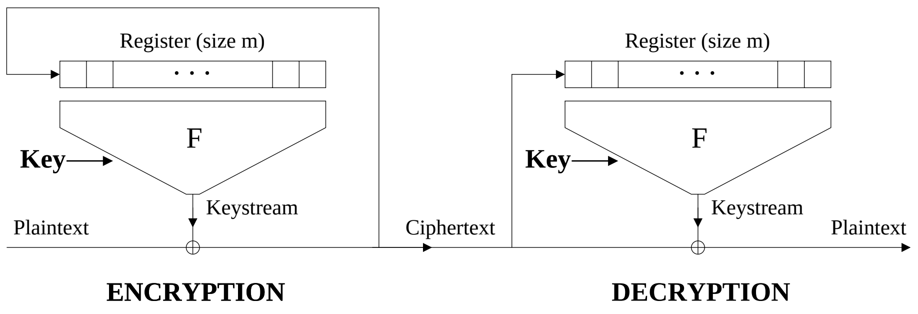

Fig. 4. General Structure of a SSSC

m consecutive ciphertext bits are transmitted correctly, the decryption works fine, despite any eventual error occurring previously in the communication channel. Thus the **re-synchronization is automatic** after m advances at most. This feature can be useful in many contexts, especially for error-prone channels [32]. However there are also some natural weaknesses to the construction (for instance, portions of ciphertext can be replayed). Therefore the use of SSSCs should generally be considered with caution.

Because of the ciphertext feedback, an attacker potentially gains some control on the PRNG. For instance we can consider **Chosen Ciphertext Differential Cryptanalysis**. In this case, the content of the register is chosen by the attacker. But observing the output of the PRNG should not help him to retrieve secret information. Resistance against chosen-ciphertext attacks is a rather strong requirement in cryptography.

However we can envisage relaxed scenarios here. For example, a man-in-the-middle attacker could introduce errors in the communication channel and simply observe how long the resynchronization takes. Besides, chosen-ciphertext attacks can sometimes be turned into chosen-plaintext attacks : the difficulty is that the attacker needs to predict the upcoming keystream bits. This can be done if several copies of the cipher (running with the same key) are available, or if the attack has the ability to reinitialize the cipher.

3.4 Authenticated Encryption Modes

AEM modes are used to satisfy simultaneously integrity and confidentiality. Such modes have been proposed for block ciphers, by combining an encryption mode and a MAC (see [19, 29] among others). For stream ciphers, some modes have also been proposed [21], and the solution is generally to add a plaintext feedback to the PRNG (see Figure 5).

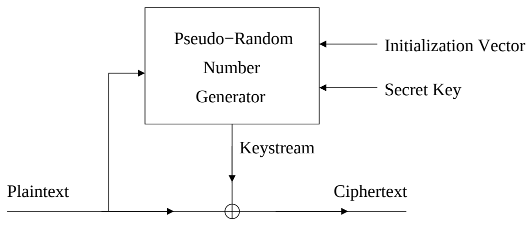

Fig. 5. Asynchronous stream cipher with plaintext-feedback

Here the internal state of the PRNG depends on the message. At the end of encryption, some function is computed from this state, with the expectation to guarantee the message integrity. A classical mistake is to encrypt a short padding sequence (for instance 128 zeroes) at the end of the message and use the resulting ciphertext as MAC. This is not secure when messages of variable length are allowed (an attacker can use the encryption of a message to predict the MAC of a truncation of the same message). In fact it is not trivial to find a satisfying function. Recent stream ciphers frequently propose AEM modes, but their security only relies on heuristic arguments.

In practice, a new concern is that an attacker can gain some control on the internal state of the cipher, because of the plaintext feedback. In the light of recent results [34, 45], two scenarios of differential attacks are possible here. First, if two plaintexts are encrypted with the same key material, there is a threat that the corresponding ciphertext difference reveal information about the key or the internal state. Secondly, two different plaintexts should yield the same internal state only with negligible probability, otherwise the same MAC will be computed and a MAC forgery is possible. Both attacks belong to the DC family. AEM modes should be protected against both of them.

3.5 Summary

We have described several possible DC scenarios against stream ciphers. Surely, this is not exhaustive, however most recent examples (see Section 4) can be ranged in one of the categories of Table 1.

Limitation	**Solution**	**New Threat**
Encrypting multiple messages	IV	→ Known IV DC → Chosen IV DC
Synchronization	SSSC	→ Chosen Ciphertext DC
Integrity	AEM	→ Chosen Plaintext DC → MAC forgery

Table 1. Summary of all envisaged DC against stream ciphers

The term "cryptanalysis" of a stream cipher is somewhat ambiguous. Various types of attack can actually be thought of : distinguishing attacks, keystream prediction attacks, key recovery attacks, ... The last mentioned is of course the strongest one, although distinguishing attacks were considered sufficient to reject NESSIE candidates [36], which raised some controversy [39]. Given the rather strong assumptions behind DC (it often requires chosen messages and total or partial control of IV's), we focus mostly on Key Recovery Attacks in the following (except for MAC forgeries).

4 Some Recent Examples

4.1 HELIX

Helix is a high-speed stream cipher proposed in 2003 [15]. It was designed in order to be twice faster than AES in software, and it also includes a mechanism for authentication. A support for IV's[3] is also included in the design. Therefore Helix might be susceptible to several of the attack scenarios described in Section 3. As a matter of fact, differential attacks against Helix have been proposed in 2004 [34]. Although not quite practical, they enlighten some weaknesses of the design.

[3] more precisely for nonces, so the value is supposed to be different for each message

Chosen Plaintext DC
A first attack against Helix takes advantage of the plaintext feedback. Indeed, the plaintext is introduced sequentially, by words of 32 bits, inside Helix encryption function (an auxiliary mechanism is used after encryption to produce a MAC, so the plaintext feedback is necessary). This structure is summarized in Figure 6.

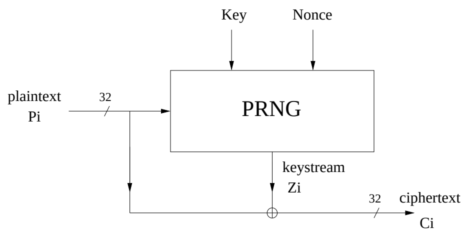

Fig. 6. General structure of HELIX

In [34], an interesting observation is made about the differential behavior of Helix. All other inputs (key, nonce) being unchanged, it is assumed an attacker modifies the plaintext word number i (referred to as P_i). An analysis shows that the resulting difference on keystream word Z_i leaks partial information about the secret key. Repeating this process about 4000 times, a key recovery attack is proposed with time complexity of 2^{88} computation steps.

The source of this problem lies in a lack of diffusion. Indeed, the plaintext introduced P_i is only modified by two XOR, two cyclic shifts, and one addition modulo 2^{32}, before being used as the next keystream word Z_i (see the specifications of Helix for more details [15]). The behavior of these functions (XOR, cyclic shift, addition) regarding differences is rather simple, so an attacker can predict the differential behavior with good probability. Moreover this behavior is correlated with several secret internal state bits. However this attack requires the re-use of nonces which should not be allowed in a practical setting. So the attack might be avoided by adding appropriate countermeasures. Still, the lack of diffusion in the plaintext introduction is an important concern. One would expect the output Z_i to look random even when the attacker chooses P_i and nonces are fixed.

Chosen IV DC
A second differential weakness of HELIX is pointed out in [34], regarding the nonce introduction. Indeed, the nonce is used in two occasions by Helix :

- to initialize the internal state
- to derive some subkeys, which are used at the i-th round (*i.e.* at the introduction of the i-th word of plaintext)

An interesting observation is that these subkeys are not derived from the full nonce. One can modify several bits of the nonce without affecting subkeys of several consecutive rounds. As a result, some particular internal collisions are preserved long enough to be detected (by searching for certain patterns in the keystream). Unfortunately, the result is only a distinguishing attack with high complexity (of the order of 2^{114} computation steps). But compared to the previous attack, it has the advantage to not require replay of nonces.

To summarize, two weaknesses of Helix have been observed quickly after the algorithm was proposed. The resulting attacks are based on techniques from DC but are not practical. However they are worrying since they reveal weaknesses in the mixing and diffusion of IV and plaintext.

4.2 RC4

RC4 is one of the most popular stream ciphers of the last decade. It was initially developed in 1987 by Ron Rivest for RSA Data Security, but kept confidential as a trade secret by the company. After being used in many (software) applications, the design was eventually leaked and cryptographers started analyzing it after 1994. Several statistical weaknesses of RC4 have been demonstrated over the years [17, 18, 31, 33], however the general mechanisms of RC4 are still considered secure in practice, provided the first output bits are discarded.

A major practical problem with RC4 regards the security protocols of the 802.11 standard for wireless networks [26]. In this context IV's are required since many frames must be encrypted with the same key. RC4 does not offer a proper support for IV's, however it allows variable key size. The chosen solution was thus to prepend the IV to the actual secret key. For instance, in the standard, two key sizes are possible : 128 bits or 64 bits, but in both cases the first 24 "key" bits are actually the IV. Unfortunately, in 2001, it was shown [17] that the first keystream words generated by RC4 were strongly correlated to some bits of the key. This weakness can be further exploited to a key recovery attack provided several IV's of the right form are used.

This attack was implemented in practice against wireless networks [43] and has been a major security problem since then, although better security mechanisms are on the verge of replacing the old ones in the standard. This popular attack is closely related to **Chosen IV DC**. Indeed the principle of the attack is to use several IV's with particular differences between them (only one byte should differ between these IV's). As pointed out in [17, 43], **Known IV DC** can even be sufficient in practice, depending on the implementation.

4.3 SOBER-128

SOBER-128 is a recent algorithm, designed by Hawkes and Rose [24]. It is a software-efficient stream cipher based on a linear feedback register over GF(2^{32}). In addition to the conventional encryption mode, it offers a MAC functionality. To that purpose a Plaintext Feedback Function (PFF) is added when authentication and encryption are required (see Figure 7).

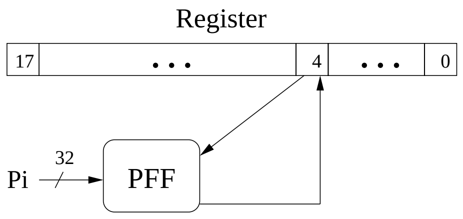

Fig. 7. General structure of SOBER-128

In [45], it was shown that the PFF function has undesirable properties regarding differential attacks. Accordingly, for well chosen differences in the plaintext word number i (called P_i), the resulting difference in the cell R_4 of the LFSR is predictable with high probability. Furthermore, these differences in the LFSR can later be eliminated using the linearity of the register. The resulting attack allows us to replace a message M and its MAC m computed with some secret key K, by another message M', such that M and M' give the same MAC under K, with probability $\frac{1}{16}$. Of course this attack is an important concern, and the MAC support for SOBER-128 has been removed since then [42]. We can range this result in the category of **Chosen Plaintext MAC Forgery**.

4.4 KNOT

KNOT is an algorithm proposed by Daemen *et.al.* in 1992 [10]. It is an instantiation of a general method to build dedicated SSSCs proposed initially by Maurer [32] and further investigated in [10]. Among design criteria of KNOT was the resistance of the Keystream Generation Function (denoted by F in Figure 4) against Differential Cryptanalysis. The proposal even contained an analysis of the differential properties of F.

However some problems subsisted in the design and have been identified a few years later [27]. In KNOT, the $m = 96$ more recent ciphertext bits are not stored in a register like in Figure 4, but are used to derive the content of a 128 bits non-linear register NL. Using the differential properties of NL, Joux and Muller proposed a chosen-ciphertext attack based on detecting collisions in the internal state of NL. This attack recovers the secret key with complexity of 2^{62}. More formally this attacks takes advantage of differential characteristics of the form $\delta \longrightarrow 0$, which had not been taken into account by the designers.

4.5 TURING

TURING is a software-oriented stream ciphers proposed in [40]. The general structure of this algorithm is quite similar to SOBER-128 : a linear register over GF(2^{32}) and a plaintext extraction function. Besides, TURING uses a technique called linear masking (some bits extracted from the register are XORed to the output of the PRNG). The general structure is represented in Figure 8.

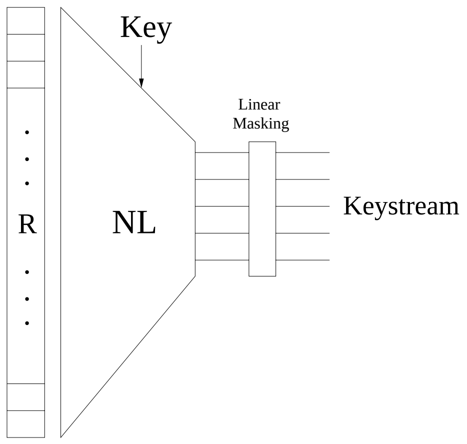

Fig. 8. General structure of TURING

In the case of TURING, no weakness has been identified yet for the keystream generator itself (although algebraic attacks with high complexity might be possible for 256 bit keys). However, some weaknesses in the key scheduling have been identified quickly after TURING was published [28]. This attack belongs to the family of Chosen IV DC. Basically the initial state of Turing is derived from the key K and the IV in two steps

1. Initialize the register with K and IV
2. Apply a linear transform (the Pseudo-Hadamard Transform - PHT). Basically the PHT consists in summing up all cells of the register. The result T is then added individually to each cell.

The attacks described in [28] exploit the fact that a collision is possible in the intermediate value T for two different IVs. Indeed T is an object of 32 bits. Therefore, using the birthday paradox, one expects a collision on T after 2^{16} pairs of chosen IV's. This event can be detected by looking at the first bits of keystream generated by TURING. Furthermore, when a collision is detected, an attacker learns some information about the key. The resulting key recovery attack has complexity 2^{72}.

From this example, we see that the differential property of the IV introduction are an important security concern. In the case of TURING, the PHT does not provide enough diffusion in the key setup. With a reasonable probability, two different IV's result in almost identical initial states. Surprisingly, this could be avoided by replacing the PHT by several clocking of the register before encryption.

Many other examples of attacks related to DC could be given. For instance the linear introduction of IV's in the A5/1 and A5/2 algorithms of the GSM standard have been very helpful in recent attacks [1]. In the next section, we propose some guidelines to follow for future stream cipher proposals.

5 Guidelines for New Designs

Given the lack of actual standard regarding stream ciphers, one of the main goals of the new European project ECRYPT [13] is *"the development of secure and efficient stream ciphers"*. Therefore a better understanding of cryptanalytic techniques is crucial to help new designs. For that purpose, this section will be dedicated to guidelines regarding new designs, especially ways to prevent differential attacks.

5.1 Protecting All Inputs Against DC

A large number of the previous attacks exploits the differential property regarding the introduction of one of the inputs - either the IV (nonce) or the plaintext. These external inputs are not protected like a secret key and we assume they can be controlled by an attacker in various scenarios (see Section 3).

Initialization Vectors
Concerning IV's, the introduction is generally part of the key setup. So we have to consider some function ϕ applied to the key K and the IV,

$$\phi(K, IV) = \text{ Initial State}$$

This function ϕ should be strongly protected against DC regarding its second input. This problem is similar to those encountered with block ciphers, though the security requirements are weaker. Indeed the attacker has only partial access to the output of ϕ (typically he can detect total or partial collisions of initial state). However we think it is reasonable to use a robust DC protection, for instance by mixing in the function ϕ layers of non-linear operations (*e.g.* S-boxes) with linear layers of diffusion (either using a matrix or a LFSR). Of course, no one should use a purely linear key setup. Following this guideline is not as easy as it first seems : weaknesses can result from subtle properties of the mechanisms used (see the attack against TURING in Section 4.5).

Plaintext
Concerning the plaintext feedback function, the analysis depends highly on the general structure of the stream cipher. Among recent proposals using plaintext-feedback, we can identify two families :

- **Iterative ciphers**. The stream cipher uses an iterated round function, where the internal state is initialized with the key material. The typical example is Helix [15]. Of course the round function is much simpler than a block cipher since speed is an important design criteria and the plaintext is introduced inside this function. The security of the cipher relies partly on the differential properties of this introduction. Fortunately the analysis is similar to what is generally done for block ciphers. The round function must be protected against DC, by mixing appropriately operations in different group structures, and/or by combining linear and non-linear layers in a proper manner. However one should be careful in choosing these operations as demonstrated by the attacks against Helix (see Section 4.1).
- **LFSR-based ciphers**. These algorithms use an autonomous component with large period, which is often linear and built with one or several LFSR's, although non-linear constructions have been proposed [5, 30]. The Plaintext Feedback Function (PFF) can be seen as a perturbation of this component. Differential properties of the PFF should be hard to predict (for instance introducing directly the plaintext inside the LFSR is generally not a good idea). An example of bad choice of PFF is the MAC forgery attack against SOBER-128 (see Section 4.3).

Many recent breaks (see Section 4) show that non-secret inputs (like the plaintext or the IV) should be manipulated with caution when introduced inside the secret internal state of a stream cipher. In this section, we have identified several typical situations and described basic rules to follow as a countermeasure against DC.

5.2 The particular case of SSSCs

Among the various types of attacks described in Section 3, a particular case is Self-Synchronizing Stream Ciphers. As mentioned previously, such ciphers are useful in specific situations and can be built using the CFB mode of operation for block ciphers, with 1-bit feedback. However dedicated algorithms are generally more efficient, thus various dedicated SSSCs have been proposed [9, 10, 32]. These constructions are rarely built on top of a traditional stream ciphers. Indeed stronger security requirements must be fulfilled here. For instance, resistance against Chosen Ciphertext Attacks is generally expected (see Section 3.3).

As demonstrated by the example of KNOT (see Section 4.4), it is quite difficult to satisfy these constraints, even when resistance to DC is among the design criteria. Differential Attacks are a powerful tool to exploit weaknesses in the F function (see Figure 4). Techniques proposed in [9] may be a promising direction, however their efficiency is not necessarily better than AES in CFB mode. Therefore achieving a satisfying dedicated SSSC is still an open challenge. Methods to protect these designs against DC are closely related to methods used to protect block ciphers.

6 Conclusion

New stream ciphers propose an increasing number of features, including support of Initialization Vectors or Authenticated Encryption Modes. However these new mechanisms induce new directions for cryptanalysis. In particular, the resistance of stream ciphers to the family of Differential Cryptanalysis as long been overlooked and must now be considered an important threat in the light of various recent attacks.

In this paper, we have reviewed the current advances in applying Differential Cryptanalysis to stream ciphers. We analyzed the corresponding attack scenarios and proposed some guidelines to follow for new stream cipher proposals.

References

1. E. Barkan, E. Biham, and N. Keller. Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communication. In D. Boneh, editor, *Advances in Cryptology – Crypto'03*, volume 2729 of *Lectures Notes in Computer Science*, pages 600–616. Springer, 2003.
2. E. Biham and A. Shamir. Differential Cryptanalysis of DES-like Cryptosystems. In A. Menezes and S. Vanstone, editors, *Advances in Cryptology – Crypto'90*, volume 537 of *Lectures Notes in Computer Science*, pages 2–21. Springer, 1990.
3. E. Biham and A. Shamir. Differential Cryptanalysis of the Full 16-round DES. In E.F. Brickell, editor, *Advances in Cryptology – Crypto'92*, volume 740 of *Lectures Notes in Computer Science*, pages 487–496. Springer, 1992.
4. A. Biryukov and A. Shamir. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In T. Okamoto, editor, *Advances in Cryptology – Asiacrypt'00*, volume 1976 of *Lectures Notes in Computer Science*, pages 1–13. Springer, 2000.
5. M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scaveius. Rabbit : A New High-Performance Stream Cipher. In T. Johansson, editor, *Fast Software Encryption – 2003*, volume 2887 of *Lectures Notes in Computer Science*, pages 307–329. Springer, 2003.
6. D. Coppersmith, H. Krawczyk, and Y. Mansour. The Shrinking Generator. In D. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lectures Notes in Computer Science*, pages 22–39. Springer, 1994.
7. N. Courtois and W. Meier. Algebraic Attacks on Stream Ciphers with Linear Feedback. In E. Biham, editor, *Advances in Cryptology – Eurocrypt'03*, volume 2656 of *Lectures Notes in Computer Science*, pages 345–359. Springer, 2003.
8. N. Courtois and J. Pieprzyk. Cryptanalysis of Block Ciphers with Overdefined Systems of Equations. In Y. Zheng, editor, *Advances in Cryptology – Asiacrypt'02*, volume 2501 of *Lectures Notes in Computer Science*, pages 267–287. Springer, 2002.
9. J. Daemen. *Cipher and Hash Function Design. Strategies based on Linear and Differential Cryptanalysis.* PhD thesis, march 1995. Chapter 9.
10. J. Daemen, R. Govaerts, and J. Vandewalle. A Practical Approach to the Design of High Speed Self-Synchronizing Stream Ciphers. In *Singapore ICCS/ISITA '92*, pages 279–283. IEEE, 1992.
11. J. Daemen, R. Govaerts, and J. Vandewalle. Resynchronization Weaknesses in Synchronous Stream Ciphers. In T. Helleseth, editor, *Advances in Cryptology – EUROCRYPT'93*, volume 765 of *Lectures Notes in Computer Science*, pages 159–167. Springer, 1994.
12. J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard.* Springer, 2002.
13. ECRYPT Network of Excellence in Cryptology. `http://www.ecrypt.eu.org/index.html`.
14. P. Ekdahl, W. Meier, and T. Johansson. Predicting the Shrinking Generator with Fixed Connections. In E. Biham, editor, *Advances in Cryptology – Eurocrypt'03*, volume 2656 of *Lectures Notes in Computer Science*, pages 330–344. Springer, 2003.
15. N. Ferguson, D. Whiting, B. Schneier, J. Kelsey, S. Lucks, and T. Kohno. Helix, Fast Encryption and Authentication in a Single Cryptographic Primitive. In T. Johansson, editor, *Fast Software Encryption – 2003*, volume 2887 of *Lectures Notes in Computer Science*. Springer, 2003.
16. FIPS PUB 81. *DES Modes of Operation*, 1980.
17. S. Fluhrer, I. Mantin, and A. Shamir. Weaknesses in the Key Scheduling Algorithm of RC4. In S. Vaudenay and A.M. Youssef, editors, *Selected Areas in Cryptography – 2001*, volume 2259 of *Lectures Notes in Computer Science*, pages 1–24. Springer, 2001.
18. S. Fluhrer and D. McGrew. Statistical analysis of the alleged RC4 keystram generator. In B. Schneier, editor, *Fast Software Encryption – 2000*, volume 1978 of *Lectures Notes in Computer Science*, pages 19–30. Springer, 2000.
19. V.D. Gligor and P. Donescu. Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes. In M. Matsui, editor, *Fast Software Encryption – 2001*, volume 2355 of *Lectures Notes in Computer Science*, pages 192–108. Springer, 2001.
20. J. Golić. Correlation Analysis of the Shrinking Generator. In J. Kilian, editor, *Advances in Cryptology – CRYPTO'01*, volume 2139 of *Lectures Notes in Computer Science*, pages 440–457. Springer, 2001.

21. J. Golić. Modes of Operation of Stream Ciphers. In D. Stinson and S. Tavares, editors, *Selected Areas in Cryptography – 2000*, volume 2012 of *Lectures Notes in Computer Science*, pages 233–247. Springer, 2001.
22. J. Golic and R. Menicocci. Edit Distance Correlation Attack on the Alternating Step Generator. In B. Kaliski, editor, *Advances in Cryptology–CRYPTO'97*, volume 1294 of *Lectures Notes in Computer Science*, pages 499–512. Springer, 1997.
23. C. Günther. Alternating Step Generators Controlled by De Bruijn Sequences. In D. Chaum and W. Price, editors, *Advances in Cryptology – Eurocrypt'87*, volume 304 of *Lectures Notes in Computer Science*, pages 5–14. Springer, 1988.
24. P. Hawkes and G. Rose. Primitive Specification for SOBER-128. Cryptology ePrint Archive, Report 2003/081, 2003. http://eprint.iacr.org/.
25. M. Hellman. A Cryptanalytic Time-Memory Tradeoff. *IEEE Transactions on Information Theory*, 26(4):401–406, July 1980.
26. IEEE 802.11 Working Group. `http://grouper.ieee.org/groups/802/11/`.
27. A. Joux and F. Muller. Loosening the KNOT. In T. Johansson, editor, *Fast Software Encryption – 2003*, volume 2887 of *Lectures Notes in Computer Science*, pages 87–99. Springer, 2003.
28. A. Joux and F. Muller. A Chosen IV Attack Against Turing. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – 2003*, volume 3006 of *Lectures Notes in Computer Science*, pages 194–207. Springer, 2004.
29. C. Jutla. Encryption Modes with Almost Free Message Integrity. In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt'01*, volume 2045 of *Lectures Notes in Computer Science*, pages 529–544. Springer, 2001.
30. A. Klimov and A. Shamir. Cryptographic Applications of T-functions. In M. Matsui and R. Zuccherato, editors, *Selected Areas in Cryptography – 2003*, volume 3006 of *Lectures Notes in Computer Science*, pages 248–261. Springer, 2004.
31. I. Mantin and A. Shamir. A Practical Attack on Broadcast RC4. In M. Matsui, editor, *Fast Software Encryption – 2001*, volume 2355 of *Lectures Notes in Computer Science*, pages 152–164. Springer, 2002.
32. U. Maurer. New Approaches to the Design of Self-Synchronizing Stream Ciphers. In D.W. Davies, editor, *Advances in Cryptology – Eurocrypt'91*, volume 547 of *Lectures Notes in Computer Science*, pages 458–471. Springer, 1991.
33. I. Mironov. (Not So) Random Shuffles of RC4. In M. Yung, editor, *Advances in Cryptology - CRYPTO'02*, volume 2442 of *Lectures Notes in Computer Science*, pages 304–319. Springer, 2002.
34. F. Muller. Differential Attacks against the Helix Stream Cipher. In B. Roy and W. Meier, editors, *Fast Software Encryption – 2004*, volume 3017 of *Lectures Notes in Computer Science*, pages 94–108. Springer, 2004.
35. NESSIE - New European Schemes for Signature, Integrity and Encryption. `http://www.cryptonessie.org`.
36. NESSIE Portfolio of recommended cryptographic primitives. Available at `http://www.cryptonessie.org`.
37. National Institute of Standards and Technology (NIST). Data Encryption Standard (DES) FIPS Publication 46-3, October 1999. Available at `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`.
38. National Institute of Standards and Technology (NIST). Advanded Encryption Standard (AES) FIPS Publication 197, November 2001. Available at `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.
39. G. Rose and P. Hawkes. On the Applicability of Distinguishing Attacks Against Stream Ciphers, 2002. Available at `http://eprint.iacr.org/2002/142.pdf`.
40. G. Rose and P. Hawkes. Turing : a Fast Stream Cipher. In T. Johansson, editor, *Fast Software Encryption – 2003*, volume 2887 of *Lectures Notes in Computer Science*. Springer, 2003.
41. T. Siegenthaler. Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications. In *IEEE Transactions on Information Theory*, volume 30, pages 776–780, 1984.
42. SOBER 128. See `http://www.qualcomm.com.au/Sober128.html`.
43. A. Stubblefield, J. Ioannidis, and A.D. Rubin. Using the Fluhrer, Mantin and Shamir Attack to Break WEP. Technical report, AT&T Labs Technical Report TD-4ZCPZZ, 2001.

44. S. Vaudenay. Provable security for block ciphers by decorrelation. In M. Morvan, C. Meinel, and D. Krob, editors, *STACS'98*, volume 1373 of *Lectures Notes in Computer Science*, pages 249–275. Springer, 1998.
45. D. Watanabe and S. Furuya. A MAC Forgery Attack on SOBER-128. In B. Roy and W. Meier, editors, *Fast Software Encryption – 2004*, volume 3017 of *Lectures Notes in Computer Science*, pages 472–482. Springer, 2004.