Two attacks against the HBB Stream Cipher

Antoine Joux[1] and Frédéric Muller[2]

[1] DGA and Univ. Versailles St-Quentin
Antoine.Joux@m4x.org
[2] DCSSI Crypto Lab
Frederic.Muller@sgdn.pm.gouv.fr

Abstract. Hiji-Bij-Bij (HBB) is a new stream cipher proposed by Sarkar at Indocrypt'03. In this algorithm, classical LFSRs are replaced by cellular automata (CA). This idea of using CAs in such constructions was initially proposed by Sarkar at Crypto'02, in order to instantiate its new Filter-Combiner model.
In this paper, we show two attacks against HBB. First we apply differential cryptanalysis to the self-synchronizing mode. The resulting attack is very efficient since it recovers the secret key by processing a chosen message of length only 2 Kbytes. Then we describe an algebraic attack against the basic mode of HBB. This attack is much faster than exhaustive search for secret keys of length 256 bits.

1 Introduction

Stream ciphers are an important class of secret key cryptosystems. Unlike block ciphers which view the plaintext as blocks of bits (typically 64 bits for the DES or 128 bits for the AES), stream ciphers handle each bit of plaintext separately. Basically, a stream cipher generates a long pseudo-random sequence (or keystream) from a seed (usually the secret key). This sequence is XORed with the plaintext to produce the ciphertext. It is widely believed that secure stream ciphers can be much faster than block ciphers.

Yet, over the last years, few stream cipher proposals have resisted cryptanalysis efforts. For instance, none of the stream ciphers candidate for the NESSIE project has been selected in the final portfolio [24], since all schemes revealed various degrees of weakness. Many of these attacks originate from the mathematical structure of Linear Feedback Shift Registers (LFSR) [5, 8], which are used as a building block by many stream ciphers. To avoid these security concerns, alternative solutions have been recently proposed. For instance, Klimov and Shamir have suggested to replace LFSRs by software-efficient nonlinear mappings based on T-functions [19].

Another contribution came from Sarkar at Crypto'02 [26]. He showed that some classical models for LFSR-based stream ciphers (Nonlinear Filter and Nonlinear Combiner) do not provide optimal security against Correlation Attacks [30]. He proposed to mix these two concepts, using a new paradigm, the Filter-Combiner Model. Unfortunately, he also showed that such a construction cannot be instantiated with LFSRs since they do not fulfill some of the basic requirements. Instead, the author showed that cellular automata are good candidates to replace LFSRs in this model. Moreover, they seem to improve the resistance against some classical attacks such as Inversion Attacks [15, 16] or the

Anderson Leakage [1]. However it was recently shown that this new construction did not necessarily increase the level of security [17].

In this paper we focus on the HBB stream cipher. This new algorithm [27] was proposed at Indocrypt'03 by Sarkar. HBB is not exactly an instantiation of the Filter-Combiner model (since the non-linear component has a memory) although its linear map is based on cellular automata. The outputs of the cellular automata are combined with a nonlinear map achieved using some of the primitives from Rijndael [12]. In addition, HBB has the particularity of offering a Self-Synchronizing (SS) mode of operation, in addition to the basic (B) mode of operation. Self-synchronizing stream ciphers are a rare primitive which can be useful in specific contexts [21]. However few dedicated designs have been proposed and many published proposals (such as [11]) did not resist cryptanalysis [18]. In fact, it is an open problem to design a secure dedicated self-synchronizing stream cipher.

In this paper, we show that both modes of operation of HBB suffer from important flaws. Against the SS mode, we use a differential attack which recovers the secret key by processing 2^{14} bits of chosen ciphertext. We also describe an algebraic attack against the B mode of operation, faster than exhaustive search for key size of 256 bits. In a first section, we give a brief overview of cellular automata and of the HBB cipher. Then, we describe our attack against the Self-Synchronizing mode of operation. Finally, we focus on the Basic mode of operation and apply algebraic cryptanalysis.

2 Stream Ciphers based on Cellular Automata

2.1 Cellular Automata Preliminaries

In general, an automaton consists in a set of l memory cells, represented at time t by $S^{(t)} = (s_1^{(t)}, \dots, s_l^{(t)})$, with a rule of evolution for each cell depending on the content of neighboring cells. Details of the theory of cellular automata are not relevant here, refer to [27] for more information. Basically, the only property we really take advantage of is their linear behavior. More precisely, a cellular automaton can be associated with a matrix M that characterizes its evolution. $S^{(t+1)}$ can then be computed by multiplying $S^{(t)}$ with M. This matrix has the additional properties of being tridiagonal and having a primitive characteristic polynomial. This guarantees that the linear recurrence has maximal period $2^l - 1$.

2.2 Overview of the HBB Cipher

HBB is a classical keystream generator, which contains a linear finite state machine and a nonlinear part. It is not a basic instantiation of the Filter-Combiner model [26] since its nonlinear part has memory (128 bits of internal state), however it belongs to the same family. According to [27], the use of cellular automata should improve the security of the cipher against some attacks using the specific properties of LFSRs.

General Structure of the Cipher An overview of HBB is given in Figure 1. LC represents the Linear Component (which contains 512 bits of internal state), and NLC represents the Non Linear Component (which contains 128 bits of internal state).

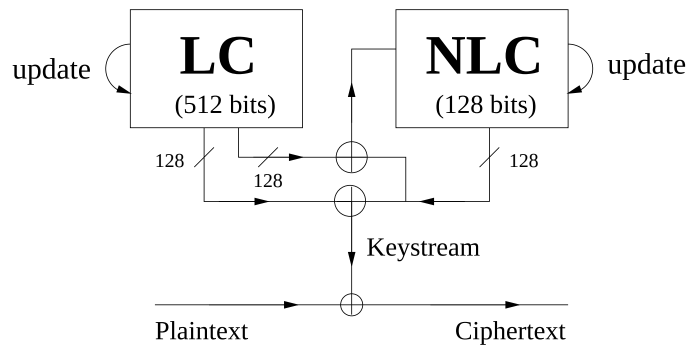

Fig. 1. One round of Hiji-Bij-Bij

Both LC and NLC have an update function which is applied at each round to their internal state. Then, 128 bits of keystream are extracted using a linear transform. To summarize, one round of encryption can be expressed as

1. Update the internal state of NLC
2. Update the internal state of LC
3. Extract 128 bits from LC and XOR it with NLC. The result is the keystream.
4. Extract 128 other bits from LC and XOR it with NLC. The result is the next state of NLC.

In general, this technique for producing keystream bits with a whitening layer after the nonlinear operations is called "linear masking" (see [5]). The initial states of LC and NLC are derived linearly from the secret key K. Then 16 rounds are applied for divergence, without using the output keystream for encryption. However the last 512 bits of keystream are XORed to the internal state of LC just before the beginning of encryption. Two key sizes are suggested for HBB : 128 and 256 bits.

The Round Function The updating function of LC is based on the cellular automata theory. Details can easily be obtained from [27]. In fact, there are two cellular automata of size 256 bits each. Both matrices contains entries of the form $(c_1, \ldots, c_{256})$ on the main diagonal, and all entries equal to 1 on the upper and lower subdiagonals.

The updating function of NLC can be seen as one round of a block cipher (see Figure 2). It consists of three consecutive layers. The first and third layers apply the Rijndael S-box [12] to each byte of NLC. The intermediate layer is a simple linear application over $\mathbb{F}_2$. The author of HBB shows that the global function has full diffusion, thus any output bit depends on all input bits. Moreover, linear approximation have been analyzed and it is shown in [27] that none can have a bias better than 2^{-12} (actually, it says 2^{-13}, but bias are represented as $0.5 \pm \varepsilon$, while we prefer the convention $0.5(1 \pm \varepsilon)$). Thus, according to the author, the cipher should resist attacks based on linear approximations, such as linear cryptanalysis [20] and correlation attacks [30]. Finally, the previous elements give an implicit description of the Basic (B) mode of operation of HBB.

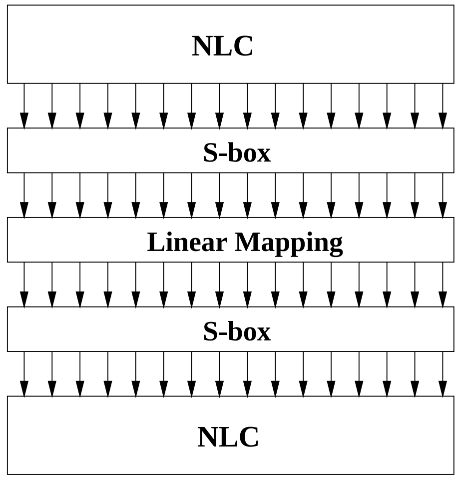

Fig. 2. The NLC updating function

The Self-Synchronizing (SS) Mode of Operation A Self-Synchronizing mode can be easily derived from the above description by making the keystream dependent on the previous bits of ciphertext. To do this, one additional step is added to the previous description.

At the end of round i, the last four blocks of ciphertext (128 bits each), represented as C_i, C_{i-1}, C_{i-2} and C_{i-3}, are XORed together with the secret key K (if K is 256 bits long, both halves of K are first XORed together). The resulting value is the new internal state of NLC :

$$NLC = Fold(K) \oplus C_i \oplus C_{i-1} \oplus C_{i-2} \oplus C_{i-3}$$

On the linear side of the generator, the secret key K is linearly expanded into a 512 bits value, then XORed with the concatenation of the last four ciphertext blocks, to produce the new value of LC :

$$LC = Expand(K) \oplus (C_i || C_{i-1} || C_{i-2} || C_{i-3})$$

Therefore, in the SS mode, the internal state of the cipher at the beginning of each round depends only on the secret key and the last 512 bits of ciphertext, in a linear manner.

3 Cryptanalysis of the SS mode of operation

In this section, we focus on the SS mode of HBB with secret key K of 128 bits long. Each round of encryption simply consists in a fixed function applied to K and the last 4 blocks of ciphertext. We show that this construction actually does not resist to a chosen-ciphertext differential attack.

3.1 Background on Self-Synchronizing Stream Ciphers

Generally a Self-Synchronizing Stream Cipher (SSSC) is one in which the keystream bit is a function of the key and a fixed number m of previous ciphertext bits. This parameter m is called the *memory* of the cipher.

In order to describe formally a SSSC, let x_t denote plaintext bit number t, y_t the corresponding ciphertext bit and z_t the corresponding keystream bit. The encryption is generally described

$$y_t = x_t \oplus z_t$$

where the keystream bit is computed as

$$z_t = F(y_{t-1}, \ldots, y_{t-m}, K)$$

F denotes the keystream function and K the secret key. The basic idea is to encrypt each plaintext bit with a function depending only on the secret key and the previous m ciphertext bits. Therefore each ciphertext bit can be correctly deciphered as long as the previous m ciphertext bits have been successfully received.

General properties and design criteria for SSSCs have been studied by Maurer [21]. It pointed out that a self-synchronization mechanism has several advantages from an engineering point of view. For instance, it may be helpful in contexts where no lower layer of protocol is present to assure error-correction. In particular, it prevents long bursts of error when a bit insertion or deletion occurs during the transmission of the ciphertext.

However in terms of security the analysis of SSSCs requires a completely different approach from conventional stream ciphers. Indeed, since the pseudo-random generator does not behave autonomously, the cipher might be subject to chosen message attacks. Therefore it is not straightforward to turn a conventional stream cipher into a SSSC and few dedicated designs have been proposed. Using a block cipher in 1-bit Cipher FeedBack (CFB) mode [14] is usually quite inefficient in terms of encryption speed, and reduced-round optimizations may be subject to attacks [25]. KNOT, one of the few dedicated designs [11] was broken at FSE'03 [18]. An improved version of KNOT, described in Daemen's thesis [10], appears to resist this attack, but currently there is no other secure dedicated SSSC in the literature.

3.2 The attack against HBB in SS mode

This section describes a differential attack using chosen ciphertexts. Accordingly, we suppose that an attacker gains access to a decryption oracle and introduces chosen blocks of ciphertext.

The goal of the attacker is to obtain two inputs of the NLC round function that differ only on one byte, with difference δ. This can be achieved by introducing the blocks of ciphertext C_1, C_2, C_3, C_4 and then $C_1, C_2, C_3, C_4 \oplus \delta$. Let j denote the position of this one byte difference and K_j the corresponding byte of secret key.

At first sight, it seems that the attacker has no access to the output difference of the NLC updating function because of the linear masking. However at the beginning of each round, LC is set to the value (see Section 2.2)

$$LC = Expand(K) \oplus (C_i || C_{i-1} || C_{i-2} || C_{i-3})$$

thus the difference of linear masking is a purely linear function of the introduced ciphertexts only. Thus an attacker able to observe the keystreams can cancel out the difference of linear masking and observe directly the output difference of the NLC updating function.

Initially, the difference δ is confined to one byte. In addition, after the first layer of S-box in this computation, the difference still only concerns the byte number j in the internal state. This difference is of the form

$$\delta' = S(K_j \oplus x) \oplus S(K_j \oplus x \oplus \delta) \tag{1}$$

where x is a known byte depending on the introduced ciphertext. If δ' has hamming weight equal to 1, it is true by construction that the hamming weight of the difference after the linear layer will be exactly 3 (see [27]). Thus, only 3 bytes will differ after the last layer of S-boxes. This difference trail on NLC is described in Figure 3 where dashed areas represent the differences.

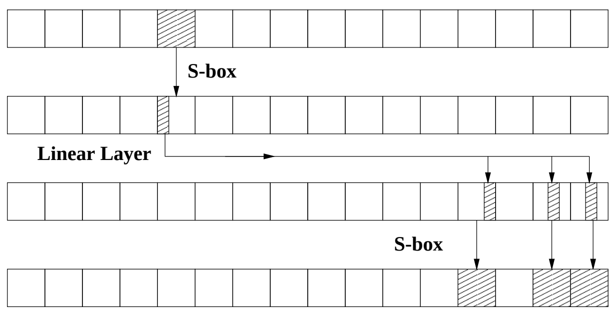

Fig. 3. An illustration of the Differential Trail on NLC

We observe that, when $\texttt{hamming}(\delta') = 1$, the difference on the output of the NLC updating function is null for 13 bytes of 16. Otherwise this property is very unlikely. In average the event that the hamming weight of δ' is 1 happens with probability $\frac{8}{256} = 2^{-5}$. Thus, according to the birthday paradox, testing 2^3 values of δ should be sufficient to detect a collision on 13 out of 16 output bytes. This event provides a condition on K_j using relation (1), of the form

$$\texttt{hamming}(S(K_j \oplus x) \oplus S(K_j \oplus x \oplus \delta)) = 1$$

In general, only a few K_j values will verify it. Furthermore it is straightforward to eliminate false candidates for K_j with a few extra decryption queries. To summarize, one byte of secret key can be recovered by processing about $2^3 = 8$ blocks of keystream in average. Repeating it 16 times, the full secret key can be easily obtained.

Moreover this attack can be mounted by processing just a single message. Indeed an attacker can just concatenate all the chosen ciphertexts he needs for successive applications of the attack and submit the resulting message to the decryption oracle. In practice this can be done by just adapting the choice of ciphertext block number i to the previous blocks of ciphertext. Therefore, to process $2^3 \times 16$ blocks of keystream, a message of length $3 + (2^3 \times 16)$ blocks is sufficient. This corresponds to about 2 Kbytes of chosen ciphertext. Besides, when δ' has hamming weight equal to $2, 3$ or 4, we also obtain detectable collisions on the keystream. So, we think it is even possible to lower a little our data requirement, using a precise analysis of all these events. We have implemented our attack using only the case of hamming weight 1 and were able to recover successfully the bytes of secret key as expected, with about 8 blocks/byte. An attack against the 256 bits secret keys would work by first recovering the 128 bits used in NLC, and then obtaining the remaining key bits by other means.

This attack represents a real threat in practical applications where the SS mode of the HBB cipher is used. Indeed, an active attacker can easily introduce a chosen ciphertext sequence of a few Kbytes in the communication layer. Since self-synchronizing modes of operation are usually implemented for fast streaming communications on faulty channels, it is even likely that the error caused by the action of the attacker would go unnoticed. Then, if the attacker is able to observe the resulting decrypted plaintext, our attack applies and he would recover the complete secret key with little offline processing. Thus, we believe the SS mode of operation of HBB is weak and should not be used as proposed. More generally, resistance of self-synchronizing stream ciphers against differential attacks should always be investigated, as in the case of block ciphers (see [22] for more details).

4 Algebraic attack against the B mode of operation

In the previous section, we have described a very efficient differential attack against the SS mode of HBB. Obviously, this attack does not apply to the Basic mode of operation since the attacker cannot choose the inputs of NLC (or LC) at each round. However, other cryptanalysis techniques may be envisaged here. In particular we propose an algebraic attack against HBB which is faster than exhaustive search for a key size of 256 bits.

4.1 Algebraic attacks and Stream Ciphers

Algebraic attacks form a class of cryptanalysis techniques which has received a huge interest in the last years. Indeed new applications have been described in various fields including block ciphers [9, 23], stream ciphers [7, 8] and even public key cryptography [13]. Algebraic attacks exploit polynomial equations describing exactly an algorithm. There is a contrast with classical cryptanalysis techniques which are often based on approximations of the behavior of the algorithm. In the recent years many stream ciphers [2–4, 8] have been broken using algebraic attacks and it has now become important to investigate the security of new algorithms regarding these techniques.

The general idea in algebraic attacks on stream ciphers is to write keystream bits as a polynomial of low enough degree in the bits of the secret key, and then apply an appropriate algorithm for solving this polynomial system. Many strategies exist like the simple linearization attack or the refined relinearization attack [28]. In these basic attacks, all monomials are replaced by new variables, then the resulting linear system is solved by usual linear algebra. In some cases, better strategies may also apply, such as sparse linear algebra or other dedicated algorithms. For instance, Gröbner base techniques (for an illustration of these techniques, see [13]) are a well-known mathematical tool helpful in the case of algebraic attacks. Besides an alternative solution, the XL algorithm has also been proposed [29] to resolve polynomial systems.

In general, for a recent cipher, it should be impossible to write low degree equations involving the secret key bits and the plaintext bits. In the next section, we show that HBB fails to meet these requirements.

4.2 The case of HBB

It is not straightforward to express directly the keystream in function of the secret key because of the divergence steps executed before the beginning of encryption. However, an attacker can focus on recovering the initial state of LC (of length 512 bits). Then in a second phase, it might be possible to recover the key from the initial state, if necessary. Therefore we first focus on writing polynomial equations between the keystream and the initial state.

In Figure 1, we can see that the internal state of NLC at the beginning of any round i is a linear function of the initial state of LC and keystream bits. Looking at two consecutive states of NLC, we can express both the input and the output of the NLC updating function as linear functions of the initial state and the keystream bits.

Besides, it is easy to write equations of degree 7 relating inputs and outputs of the NLC updating function (referred to as Φ). Let $(a_i)_{0\leq i\leq 127}$ and $(b_i)_{0\leq i\leq 127}$ respectively denote the input and output bits of Φ. Notations c_i and d_i are used for the intermediate states after the first and second layer of Φ. It is well known that the Rijndael S-box (and its inverse) has algebraic degree 7, i.e. the output bits of the S-box (and its inverse) can be expressed as degree 7 polynomials in its input bits. Consequently,

$$b_i = P_i(a_0, \ldots, a_{127})$$

where P_i is a polynomial of degree 7. Similarly,

$$c_i = Q_i(d_0, \ldots, d_{127})$$

where Q_i is a polynomial of degree 7. Furthermore, b_i and c_i are related by a linear transform. Thus, 128 relations of degree 7 of the form

$$\sum_{i=0}^{127} \lambda_i P_i(a_0, \ldots, a_{127}) \oplus \sum_{i=0}^{127} \mu_i Q_i(d_0, \ldots, d_{127}) = 0$$

can be written. As we argued previously the a_i's and d_i's depend linearly on the initial state and the keystream, thus the previous relation can be rewritten as

$$R_t(v_0, \ldots, v_{511}, Z_t) = 0$$

where R_t is a degree 7 polynomial, the v_i's are the bits of initial state and Z_t is the keystream block number t (which is known).

The number of monomial of degree 7 on 512 unknowns is

$$\binom{512}{7} \simeq 2^{50.6}$$

which is not sufficient to provide a security level of 256 bits. Indeed, a simple linearization attack would proceed by linear algebra on all monomials of degree 7. Using Gaussian elimination, the corresponding complexity would be about

$$\binom{512}{7}^3 \simeq 2^{151.8}$$

basic binary instructions for inverting the matrix ($2^{146.8}$ on a 32 bit processor). Besides, about $2^{50.6}$ bits of known plaintext would be needed. The solution of this system reveals the initial state of the cipher from which it might be possible to retrieve the actual secret key. This algebraic attack is faster than exhaustive search only for keys of 256 bits.

However, with better linear algebra such as algorithms[1] with exponent $w = \log_2(7)$, the complexity can be lowered to $2^{142.2}$ binary instructions, and we even expect that better techniques exist, for instance if we exploit the sparsity of the system. According to Courtois [7], this kind of attacks can also be further improved by using optimized resolution algorithms and a precomputation step. On the whole, it is likely that an improved version of this attack could break the 128 bit version of HBB.

This attack illustrates the fact that linear masking techniques and cellular automata by themselves do not provide protection against the class of algebraic attacks. Basically there is too much linearity in this algorithm. This type of problem is also encountered when using LFSRs as a building block. Sound countermeasures are irregularly clocked stream ciphers, like the Shrinking Generator [6] or the use of nonlinear mappings as a building block instead of LFSRs or cellular automata [19].

4.3 Recovering the key from the initial state

After the previous attack, we know the initial state of the linear component LC. This is not fully satisfying in practice. It is often expected to go further and recover the secret key. We refer to the initial state of LC as $\mathcal{S}$. Our goal is to derive from $\mathcal{S}$ the secret key K. We consider first the key size of 256 bits, since the previous attack has a complexity larger than 2^{128}.

By construction, $\mathcal{S}$ is a linear function of K and 4 keystream blocks (of 128 bits each) produced during the key schedule and discarded immediately after. We call these blocks $T_0, \ldots, T_3$ as in [27]. The first block of keystream (called Z_0) can be expressed by (see also Figure 1) :

$$Z_0 = \lambda_1(\mathcal{S}) \oplus \phi(T_3 \oplus \lambda_2(K)) \tag{2}$$

where λ_1, λ_2 are linear functions and ϕ is the NLC updating function. From (2), we retrieve $T_3 \oplus \lambda_2(K)$, since ϕ is invertible. With this additional equation, we get a total of

$$512 + 128 = 640$$

binary linear equations involving

$$256 + 4 \times 128 = 768$$

unknowns (corresponding to K, T_0, T_1, T_2 and T_3). In average this linear system should contain 2^{128} solutions, one of them corresponding to the correct K. These solutions can be obtained with the usual Gaussian reduction algorithm. The complexity required here is about 2^{128} steps of computation. Besides, for keys of 128 bits, there are 128 unknowns less in the system, hence only one solution is expected.

---

[1] We did not consider other algorithms with smaller exponent such as the Coppersmith-Winograd algorithm, because they are not practical enough

To summarize, the extra work required to find K from the initial state of LC depends on the key size. For keys of length 256 bits, this complexity is about 2^{128}, and for keys of 128 bits, the complexity is negligble. These results allow to complete the attacks against HBB without increasing the overall complexity.

5 Conclusion

In this paper, we have described two attacks against the new HBB stream cipher. First a very efficient differential attack breaks the self-synchronizing (SS) mode of operation by processing a message of length about 2 Kbytes. Then we described an algebraic attack which breaks HBB in B mode with workload of about 2^{142} steps of computation. This is faster than exhaustive search for secret keys of length 256 bits and we believe optimized versions could threaten 128 bit keys as well.

These attacks enlighten some important weaknesses in the design of the new HBB stream cipher. In addition, by breaking the SS mode, we have also shown that the challenge of designing a secure dedicated self-synchronizing stream cipher, initially proposed by [21] is still an open problem.

References

1. R. Anderson. Searching for the Optimum Correlation Attack. In B. Preneel, editor, *Fast Software Encryption – 1994*, volume 1008 of *Lectures Notes in Computer Science*, pages 137–143. Springer, 1995.
2. F. Armknecht and M. Krause. Algebraic Attacks on Combiners with Memory. In D. Boneh, editor, *Advances in Cryptology – Crypto'03*, volume 2729 of *Lectures Notes in Computer Science*, pages 162–175. Springer, 2003.
3. 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.
4. J.Y. Cho and J. Pieprzyk. Algebraic Attacks on SOBER-t32 and SOBER-t16 without Stuttering. In B. Roy and W. Meier, editors, *Fast Software Encryption – 2004*, volume 3017 of *Lectures Notes in Computer Science*, pages 49–64. Springer, 2004.
5. D. Coppersmith, S. Halevi, and C. Jutla. Cryptanalysis of Stream Ciphers with Linear Masking. In M. Yung, editor, *Advances in Cryptology – Crypto'02*, volume 2442 of *Lectures Notes in Computer Science*, pages 515–532. Springer, 2002.
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. Fast Algebraic Attacks on Stream Ciphers with Linear Feedback. In D. Boneh, editor, *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lectures Notes in Computer Science*, pages 176–194. Springer, 2003.
8. 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.
9. 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.
10. J. Daemen. *Cipher and hash function design. Strategies based on linear and differential cryptanalysis.* PhD thesis, march 1995. Chapter 9.

11. 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.
12. J. Daemen and V. Rijmen. AES Proposal: Rijndael (Version 2). 1999. NIST AES website `http://csrc.nist.gov/encryption/aes`.
13. J-C. Faugère and A. Joux. Algebraic Cryptanalysis of Hidden Field Equations (HFE) Cryptosystems Using Gröbner Bases. In D. Boneh, editor, *Advances in Cryptology – CRYPTO'03*, volume 2729 of *Lectures Notes in Computer Science*, pages 44–60. Springer, 2003.
14. FIPS PUB 81. *DES Modes of Operation*, 1980.
15. J. Golić. On the Security of Nonlinear Filter Generators. In D. Gollman, editor, *Fast Software Encryption – 1996*, volume 1039 of *Lectures Notes in Computer Science*, pages 173–188. Springer, 1996.
16. J. Golić, A. Clark, and E. Dawson. Generalized Inversion Attack on Nonlinear Filter Generators. In *IEEE Transactions on Computers 49(10)*, pages 1100–1109, 2000.
17. J. Hong, D.H. Lee, S. Chee, and P. Sarkar. Vulnerability of Nonlinear Filter Generators Based on Linear Finite State Machines. In B. Roy and W. Meier, editors, *Fast Software Encryption – 2004*, volume 3017 of *Lectures Notes in Computer Science*, pages 193–209. Springer, 2004.
18. 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.
19. 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.
20. M. Matsui. Linear Cryptanalysis Method for DES Cipher. In T. Helleseth, editor, *Advances in Cryptology – Eurocrypt'93*, volume 765 of *Lectures Notes in Computer Science*, pages 386–397. Springer, 1993.
21. 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.
22. F. Muller. Differential Attacks and Stream Ciphers. In *State of the Art in Stream Ciphers*. ECRYPT Network of Excellence in Cryptology, 2004. Workshop Record.
23. S. Murphy and M. Robshaw. Essential Algebraic Structure within the AES. In M. Yung, editor, *Advances in Cryptology - CRYPTO'02*, volume 2442 of *Lectures Notes in Computer Science*, pages 1–16. Springer, 2002.
24. NESSIE - New European Schemes for Signature, Integrity and Encryption. `http://www.cryptonessie.org`.
25. B. Preneel, M. Nuttin, R. Rijmen, and J. Buelens. Cryptanalysis of the CFB Mode of the DES with a Reduced Number of Rounds. In D.R. Stinson, editor, *Advances in Cryptology – Crypto'93*, volume 773 of *Lectures Notes in Computer Science*. Springer, 1993.
26. P. Sarkar. The Filter-Combiner Model for Memoryless Synchronous Stream Ciphers. In M. Yung, editor, *Advances in Cryptology – Crypto'02*, volume 2442 of *Lectures Notes in Computer Science*, pages 533–548. Springer, 2002.
27. P. Sarkar. Hiji-Bij-Bij : A New Stream Cipher with a Self-Synchronizing Mode of Operation. In T. Johansson and S. Maitra, editors, *Progress in Cryptology – INDOCRYPT'03*, volume 2904 of *Lectures Notes in Computer Science*, pages 36–51. Springer, 2003.
28. A. Shamir and A. Kipnis. Cryptanalysis of the HFE Public Key Cryptosystem. In M. Wiener, editor, *Advances in Cryptology – Crypto'99*, volume 1666 of *Lectures Notes in Computer Science*, pages 19–30. Springer, 1999.
29. A. Shamir, J. Patarin, N. Courtois, and A. Klimov. Efficient Algorithms for solving Overdefined Systems of Multivariate Polynomial Equations. In B. Preneel, editor,

Advances in Cryptology – Eurocrypt'00, volume 1807 of *Lectures Notes in Computer Science*, pages 392–407. Springer, 2000.

30. T. Siegenthaler. Correlation-immunity of Nonlinear Combining Functions for Cryptographic Applications. In *IEEE Transactions on Information Theory*, volume 30, pages 776–780, 1984.