Stream cipher designs: a review

Lin JIAO, Yonglin HAO and Dengguo FENG

Citation: SCIENCE CHINA Information Sciences **63**, 131101 (2020); doi: 10.1007/s11432-018-9929-x

View online: http://engine.scichina.com/doi/10.1007/s11432-018-9929-x

View Table of Contents: http://engine.scichina.com/publisher/scp/journal/SCIS/63/3

Published by the Science China Press

SCIENCE CHINA
Information Sciences

• REVIEW •

March 2020, Vol. 63 131101:1–131101:25
https://doi.org/10.1007/s11432-018-9929-x

Stream cipher designs: a review

Lin JIAO[1*], Yonglin HAO[1] & Dengguo FENG[1,2*]

[1]*State Key Laboratory of Cryptology, Beijing* 100878*, China;*
[2]*State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing* 100190*, China*

Received 13 August 2018/Accepted 30 June 2019/Published online 10 February 2020

Abstract Stream cipher is an important branch of symmetric cryptosystems, which takes obvious advantages in speed and scale of hardware implementation. It is suitable for using in the cases of massive data transfer or resource constraints, and has always been a hot and central research topic in cryptography. With the rapid development of network and communication technology, cipher algorithms play more and more crucial role in information security. Simultaneously, the application environment of cipher algorithms is increasingly complex, which challenges the existing cipher algorithms and calls for novel suitable designs. To accommodate new strict requirements and provide systematic scientific basis for future designs, this paper reviews the development history of stream ciphers, classifies and summarizes the design principles of typical stream ciphers in groups, briefly discusses the advantages and weakness of various stream ciphers in terms of security and implementation. Finally, it tries to foresee the prospective design directions of stream ciphers.

Keywords stream cipher, survey, lightweight, authenticated encryption, homomorphic encryption

Citation Jiao L, Hao Y L, Feng D G. Stream cipher designs: a review. Sci China Inf Sci, 2020, 63(3): 131101, https://doi.org/10.1007/s11432-018-9929-x

1 Introduction

The widely applied e-commerce, e-government, along with the fast developing cloud computing, big data, have triggered high demands in both efficiency and security of information processing. As the core of information security, cryptography is facing new opportunities and challenges. Higher speed and higher security level are both crucial, while lower implementation cost also becomes important in order to cater for the resource constraint scenarios such as wireless networks and radio frequency identification devices (RFID) tags. Among all kinds of cryptographic primitives, stream ciphers stand out play an important role in protecting the confidentiality of information.

Nowadays, the stream ciphers are featured with easy hardware and software implementation, fast encryption and decryption, no or only limited error propagation, thus they maintain the predominance in applications of special and confidential organizations, like diplomacy communication, government and military affairs. Many international standards and civil communication systems are also adopting stream cipher algorithms as their main components for protecting confidential information. Such examples are the RC4 [1] stream cipher in IEEE 802.11 wireless network standard, the A5/1 [2] stream cipher for the European GSM system, the SNOW 3G [3] and ZUC [4] stream ciphers for 3GPP standard, E0 stream cipher [5] for bluetooth system. Governments all over the world are also devoting significantly to the development of stream ciphers. In 2000, European government launched the NESSIE plan aiming at new signature, integrity, and encryption scheme. The program lasted until 2003, but unfortunately, of all the submitted stream ciphers, including the relatively famous SNOW 1.0 [6,7], LILI-128 [8], none were finally

* Corresponding author (email: jiaolin_jl@126.com, fengdg@263.net)

selected due to security flaws [9–11]. In the same year, Japanese government proposed the CRYPTREC project [12], to collect and evaluate cryptographic algorithms for applications in government and industry. Some nicely designed primitives like MUGI [13] have been selected. In 2004, Europe launched the ECRYPT projcect [14], which contains a special plan, eSTREAM [15], dedicated to stream ciphers. It collected 34 algorithms of diverse designs in total and 7 of them are finally selected. In particular, the 7 selected eSTREAM ciphers can be divided into algorithms for constrained hardware environment (portfolio 2), such as Grain-v1 [16], Trivium [17], Mickey-v2 [18], which are generally bit-oriented, and algorithms for high-speed software applications (portfolio 1), such as Salsa20/12 [19], SOSEMANUK [20], Rabbit [21], HC-128 [22], which are mostly word-oriented. The eSTREAM program has significantly affected the development of stream ciphers, especially the design and analysis on the stream ciphers with maximum internal state and high round initialization. Besides the projects dedicated designs, other kinds of cryptographic primitives can also be adapted to stream ciphers. The winner of NIST's SHA-3 hash function competition, Keccak, has an additional stream cipher mode [23]. Many authenticated encryption schemes submitted to the CAESAR campaign are adopting stream ciphers as their subroutines [24, 25].

As can be seen, the systematic study of stream ciphers is drawing more and more attentions, becoming one of the central topics of cryptography. In fact, the study of stream ciphers dates back to early 20th century. In the 1940's, Shannon theoretically proved that the one-time pad (OTP) cipher is fully secure in the ciphertext-only scenario [26]. OTP uses a completely random keystream whose digits are combined with the plaintext digits at a time to form the ciphertext. However, the keystream must be generated completely at random with at least the same length as the plaintext and cannot be used more than once, making the whole system quite cumbersome for practical use. Stream cipher borrows the idea of OTP by replacing the completely random keystream to a pseudo-random one generated from a secret key with much smaller length. In comparison with OTP, stream ciphers provide easier key generation, distribution and management which are crucial in practical applications. Meanwhile, well-designed stream ciphers will guarantee high secure margin as well, since it is computationally infeasible to distinguish the pseudo-random keystream from a complete random one, or to recover the secret key from the whole key space. Generally stream ciphers can be divided into synchronous and self-synchronous stream ciphers. In synchronous stream cipher, the generated keystream is independent of the sending, and the internal state only depends on the preceding internal state, that has nothing to do with the plaintext [27]. The advantage of synchronous stream ciphers lies in limited error propagation. Inversely, the keystream generated by self-synchronous stream cipher depends on the previous plaintext or ciphertext information making the security analysis hard to be performed [28]. Therefore, most modern stream ciphers are synchronous stream ciphers in current whose resistance against known attacks can be theoretically evaluated.

The execution of modern stream ciphers usually consist of two stages: the initialization stage and the keystream generation stage. In the initialization phase, the stream cipher first loads the secret key and initialization vector (IV) into an internal state and then updates the internal state by iteratively calling an updating function for sufficiently many rounds so that the secret key and IV are thoroughly mixed. In the keystream generation phase, the stream cipher generates the pseudo-random keystream digits from the highly diffused internal state by calling an output function while further updates the internal state with the updating function. The designs of the cipher structures and the updating, output functions are crucial to generate highly qualified pseudo-random keystream in an efficient manner. After decades of development, many well-designed stream ciphers have been proposed. Each of them has its unique feature catering for specific applications. The structures of stream ciphers are now highly diverse. In late 20th century, many stream ciphers are using completely linear updating functions so that the internal states can be regarded as concatenations of linear feedback shift registers (LFSR). Due to the LFSR has no immunity against cryptanalysis as Berlekamp-Massey algorithm [29] for its linear nature, it needs to adopt various nonlinear components to enhance its nonlinearity, where common methods include nonlinear combination, nonlinear filtering, irregular clock control, and combinations with memories. A typical LFSR based stream cipher is E0 [5] but it is soon realized that such LFSR based stream ciphers are vulnerable to various cryptanalysis techniques [30, 31]. Therefore, in eSTREAM, many of the candidate algorithms adopt more new structures such as nonlinear feedback shift register (NFSR). Among the 7

eSTREAM winners, Trivium [17] can be regarded as a typical NFSR-based stream cipher consisting of 3 NFSRs. Grain-v1 [16] is also an NFSR-based primitive concatenating an NFSR with an LFSR. In order to cater for resource-constrained applications (such as Internet of things (IoT)), small-state lightweight stream ciphers are proposed aiming at resisting the time-memory-data tradeoff (TMDTO) attacks while maintaining a small internal state. Typical examples are Sprout [32], Fruit [33, 34], and Plantlet [35], but some of them have been frequently challenged the security [36–38]. To meet the requirements for the application environment of homomorphic encryption, a new stream cipher structure was proposed in Eurocrypt 2016, called substitution filter generation, which can be regarded as a variant of filter generator [39]. Such a structure has been proved secure theoretically but some concrete instances of this stream cipher have been attacked [40]. There are also several provably secure structures of stream ciphers designed to resist side channel attacks, such as the leakage-resilient stream ciphers proposed at some international conferences such as CCS 2010 [41], CHES 2012 [42], CT-RSA 2013 [43].

In this paper, we try to classify highly diverse designs of stream ciphers according to their structures. The categories are LFSR based stream cipher, NFSR based stream cipher, shift register with carry feedback (FCSR) based stream cipher, PANAMA-like stream cipher, random shuffled stream cipher, ARX (addition, rotation, XOR) based stream cipher, Sponge structure stream cipher, block cipher based stream cipher, block cipher work mode based stream cipher, and other stream ciphers including fully homomorphic encryption systems used stream cipher, provable secure stream cipher. We also analyze the advantages and weakness of these stream ciphers in terms of security and implementation. Finally, we try to predict the future research directions of stream cipher designs.

2 Basic component

Cryptographic function. In symmetric cipher system, ones seek for Boolean functions and vectorial functions with various cryptographic properties to construct cryptographic algorithms in resistance of different kinds of attacks appeared by years (differential attack, linear attack, correlation attack, and algebraic attack, etc.). Especially, searching for or constructing cryptographic functions with a series of more optimal properties (balance, higher algebra degree, high order correlation immune, larger nonlinearity, and higher algebraic immunity, etc.) to resist various attacks at the same time, is a challenging topic in the study of the current cryptography and information security. Boolean function is mainly used for the design of stream cipher such as nonlinear combination, nonlinear filtering, and irregular clock control. Vectorial function is mainly used for the design of block cipher, such as S-box which is a nonlinear vectorial function providing confusion for cryptographic algorithms, and P-permutation which is a linear vectorial function that provides the diffusion for cryptographic algorithms. Research in this field mainly concerns on the construction, count and equivalence of perfect nonlinear (PN) and almost perfect nonlinear (APN) function, Bent and almost Bent function, and maximum algebraic immunity (MAI) function. Great contributions have been made by scientists [44–46] in the constructions and analysis of Boolean functions. For example, Qi et al. [47] has solved the construction of the symmetric Boolean functions with optimal algebraic immunity on odd number of variables, and Kan et al. [45] has solved the case of the symmetric Boolean functions with optimal algebraic immunity on even number of variables as well as decided all of their cryptographic properties. These results play a positive role in promoting the research of stream cipher designs.

Feedback shift register. Feedback shift register is one most common component of stream ciphers, divided into LFSR and NFSR according to whether its feedback function is linear, and further divided into Fibonacci pattern and Galois pattern (in Figure 1) depending on the presentation mode, between which exist certain conversion relations. Moreover there is another FCSR.

Basic operations. For hardware and software acceleration, basic operations are usually used in stream ciphers design. The three operations of module addition ($\boxplus$), interword rotation ($\lll$) and XOR ($\oplus$) qualify the completeness.

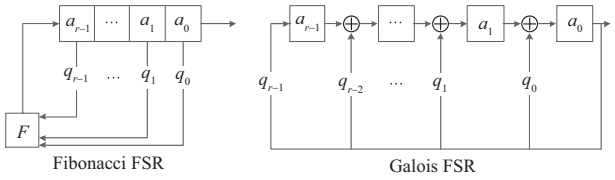

Figure 1 Types of feedback shift registers.

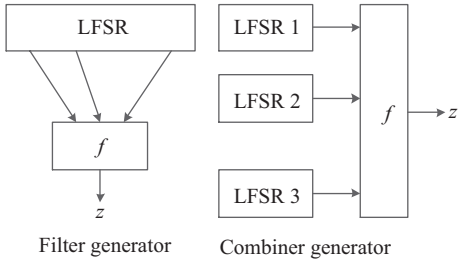

Figure 2 LFSR based generators.

3 LFSR based stream cipher

Rueppel [48] divided the design of stream ciphers into two parts: the driving part and nonlinear part, where the driving part is used to generate the basic source sequences with good properties, such as the m-sequence, and the nonlinear part is mainly used to cover the linear properties of source sequences to generate keystreams with high nonlinearity.

3.1 Bit-oriented

The most common structure of keystream generators in the 60's of last century is driven by one or more bit-unit LFSRs for a large cycle and good statistical properties, and usually combined with means of filter, combiner, or clock control to realize the nonlinearly scrambling.

3.1.1 *Filter generator*

The keystream of nonlinear filter generator is generated directly by a nonlinear function acting on some internal states of one LFSR (in Figure 2).

3.1.2 *Combiner generator*

Nonlinear combiner generator (in Figure 2) is composed of several LFSRs and a nonlinear combining function, with a balance of properties on nonlinearity, correlation immunity, algebraic degree and algebraic immunity. This structure is threatened by correlation attacks in the way of divide and conquer.

There are conversion relations between filter generators and combiner generators. These two kinds of generators have mature theoretical analysis results, especially on linear complexity, and high efficiency in both software and hardware. However, they are both vulnerable against algebraic attack series for the constant algebraic degree, and lose their allure in current stream cipher designs.

3.1.3 *Combiner generator with memory*

Nonlinear combiner generator with memory is a generalization of combiner generators, which possesses good cryptology properties and is widely used in theory and practice. One instance is Bluetooth encryption algorithm E0, which is used for point-to-point communication in wireless and Bluetooth networks.

E0. The core of E0 [5] is a typical nonlinear combiner generator, which consists of four regularly-clocked LFSRs of lengths 25, 31, 33 and 39 bits, and 4 bit memory, with the structure shown in Figure 3. The memory bits $(c_{t-1}^1, c_{t-1}^0, c_t^1, c_t^0)$ are updated by $(c_t^1, c_t^0, c_{t+1}^1, c_{t+1}^0)$, where $c_{t+1}^1 = s_{t+1}^1 \oplus c_t^1 \oplus c_{t-1}^0$, $c_{t+1}^0 = s_{t+1}^0 \oplus c_t^0 \oplus c_{t-1}^1 \oplus c_{t-1}^0$, and $(s_{t+1}^1, s_{t+1}^0) = \lfloor (b_t^1 + b_t^2 + b_t^3 + b_t^4 + 2c_t^1 + c_t^0)/2 \rfloor$. The initialization of the two-level E0 scheme is shown in Figure 3, exploiting 128-bit secret key and 74-bit IV, where G_1, G_2 and G_3 are public affine transformations. The keystream generated for each frame is only 2745 bits. In Crypto 2013, a real time attack against two-level E0 by the new technique of condition masking and correlation attacks was presented.

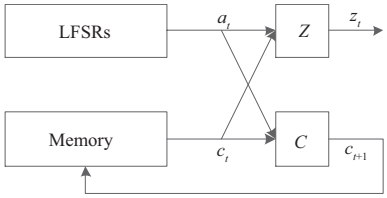

Figure 3 Combiner generator with memory.

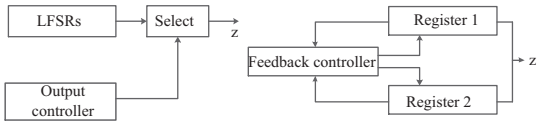

Figure 4 Clock controlled generator.

3.1.4 *Clock controlled generator*

Another approach to introduce nonlinearity is to have the LFSR clocked irregularly shown in Figure 4. The disadvantage is that irregular clock control may reduce the output efficiency. Such generators include the following models.

Stop-and-go generator. The stop-and-go generator consists of two LFSRs. One LFSR is clocked if the output of the second is a 1, otherwise it repeats its previous output.

Alternating step generator. The alternating step generator comprises three LFSRs, and the output of one of the registers decides which of the other two is to be used. The keystream is the exclusive OR (XOR) of the last bits produced by these two LFSRs.

Shrinking generator and self-shrinking generator. The shrinking generator takes two LFSRs, both clocked regularly. If the output of the first LFSR is 1, the output of the second LFSR becomes the keystream and otherwise, the output of the second is discarded, and no bit is output by the generator. The self-shrinking generator takes only one LFSR, which works as clocking the LFSR twice to obtain a pair of bits, and if the pair is 10 outputs a zero, if the pair is 11 outputs a one, otherwise, outputs nothing as the keystream.

All of the above clock controlled generators have been analyzed thoroughly, and out of use.

LILI-128. LILI-128 [49] is designed for NESSIE project, while it has been broken by algebraic attack, correlation attack, and time memory trade off (TMTO) attack. Its structure consists of two subsystems for clock control and data generation, supported by two binary LFSRs of 39-bit and 89-bit length each, and two filter functions. In the initialization, the 128-bit key is used directly to form the initial values of these two LFSRs. The LFSR for the clock-control is regularly clocked and the output of this subsystem is an integer sequence which controls the clocking of the LFSR for the data-generation, where $z(t) = f_d(\mathrm{LFSR}_d^{k(t)})$ and $k(t) \leftarrow k(t-1) + c(t)$.

A5/1. A5/1 [2] is an encryption algorithm used by about 130 million GSM customers in Europe to protect the over-the-air privacy of their cellular voice and data communication. For each frame, the 64-bit session key is mixed with a 22-bit publicly known frame counter to generate the initial state of the generator, which just produces 228 pseudo random bits. It is a good example for avoiding the reduction of output efficiency caused by irregular clock control, in the use of a majority function of the clocking taps, and only those registers whose clocking taps agree with the majority bit are actually clocked, and at each clock cycle, the output bit is produced as an XOR of the msb's of these three registers. However, it suffers the attacks of correlation, TMTO, guess-and-determine, initialization analysis, for its minor 64-bit internal state, which only costs several seconds to break it with three GPUs.

MICKEY 2.0. MICKEY 2.0 [18] is a hardware oriented selection in eSTREAM finalists. Unlike the above clock controlled generator, it controls the feedback of the registers instead of the output. It maps an 80-bit secret key and a variable length IV (0 to 80 bits) to a keystream with maximum length of 2^{40} bits. The generator is built from two registers, each of 100-bit length. The clock control bit for each register is derived from an XOR of different pair of state bits from both registers. The update mechanisms for these two registers are linear and nonlinear respectively, in the charge of their input bits and control bits. In the keystream generation stage, each register updates itself with its own internal state bits independently in hand of its control bit, and it takes an XOR of the msb's from these two registers as the keystream bit. In the initialization stage, registers are set to all zero, and then the IV, key are successively used as the input bits to update the state by executing the keystream generation routine, where one state bit of

the nonlinearly update register always participates the update of the other. The structure of MICKEY 2.0 is totally different from the previous clock controlled generators. It adopts the work mode that these two registers control and influence each other, thus the analysis of MICKEY 2.0 is still few. There is also a version named MICKEY-128 that supports an extended key length of 128 bits.

3.2 Word-oriented

Compared with bit-oriented stream ciphers, word-oriented stream ciphers usually perform better in software implementations. A word-oriented stream cipher usually works on and outputs words of certain size, like 32, 16, 8 bits. The general structure of this kind of LFSR-based stream ciphers basically consists of the linear driver in finite field extension and a nonlinear finite state machine (FSM), which can be regarded as a generalization of filter generators, shown in Figure 5. Many stream ciphers designed with such structure have been used widely, such as ISO/IEC international standard encryption algorithm SNOW 2.0, 3GPP LTE international encryption standard algorithm SNOW 3G and ZUC.

SNOW series. SNOW 1.0 is a submission for NESSIE project, however it has a security issue. SNOW 2.0 [6] is a new version of SNOW series under the control of a 128-bit key and a 128-bit IV. The word size is 32 bits, and the LFSR is of length 16 with a feedback polynomial in $\mathbb{F}_{2^{32}}[x]$. The nonlinear part takes an FSM with two 32-bit memory units, and a transform using the S-box and mixcolumn in advanced encryption standard (AES), whose structure is similar to the diffusion and confusion form of block ciphers and has high security. The two-word input of the FSM is taken from the LFSR, and the keystream word is formed by an XOR of the FSM output and the last stage of the LFSR. The LSFR is initialized with the key and IV, while the memory units of FSM are set to zero. Then the cipher is clocked 32 times incorporated the output of the FSM in the feedback loop, and once more blank clock before the first keystream word producing. The analysis difficulty of this kind of structure is that if we transform the LFSR in the finite field extension into a binary representation, it will appear a very complex form. To solve this problem, in Crypto 2015, a fast correlation attack over extension fields was proposed, and reduced the analysis complexity of SNOW 2.0 to the best known result of $2^{164.15}$. SNOW 3G [3] forms the heart of the 3GPP confidentiality algorithm UEA2 and the 3GPP integrity algorithm UIA2. Its only difference from SNOW 2.0 is that the FSM has three 32-bit memory units, and two different S-box layers are used to respectively update the last two units. A new version of SNOW family called SNOW-V is recently designed for 5G application [50].

ZUC series. ZUC-128 [4] is the core of the standardized 3GPP confidentiality and integrity algorithms 128-EEA3 & 128-EIA3, which takes a 128-bit key and a 128-bit IV, and outputs a keystream of 32-bit words. There are 3 parts involved in ZUC-128: an LFSR defined over the field $\mathrm{GF}(2^{31}-1)$, consisting of sixteen 31-bit cells; a bit reorganization layer, which extracts the content of the LFSR to form four 32-bit words, used in the following FSM; an FSM with two 32-bit memory units, which executes the operations of linear transform and S-box parallelly. The initialization stage of ZUC is similar with that of SNOW series. Recently, a new version that supports 256-bit key is published, with only differences in the initialization and message authentication codes (MAC) generation phases, designed for the upcoming applications in 5G with high compatibility of ZUC-128.

Sosemanuk. Sosemanuk [20] is a winner of eSTREAM project, accommodates a 128-bit key and 128-bit IV. It is a 32-bit word-oriented stream cipher, composed of three parts: an LFSR containing 10 elements of $\mathbb{F}_{2^{32}}$, an FSM with a transition function and a chosen function, a round function from block cipher SERPENT, aiming at improving SNOW 2.0 both from the security and efficiency. The special point of this design is that the output transformation derived from the S-box of SERPENT mixes four successive outputs of the FSM in the bit-slice mode. Its initialization process is fully influenced by Serpent, which contains a key setup that corresponds to the Serpent24 key schedule, an IV injection that uses Serpent24 block cipher with the preceding generated subkeys and an input of the IV, and finally the Sosemanuk internal state is initialized with the outputs of the block cipher. The analysis of Sosemanuk is about correlation attacks and guess-and-determine attacks, while it still unbroken.

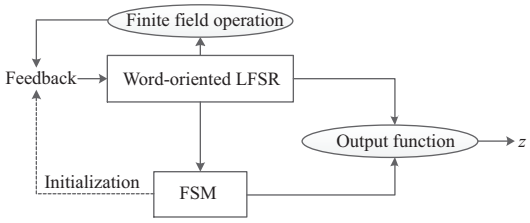

Figure 5 Schematic of SNOW-like cipher.

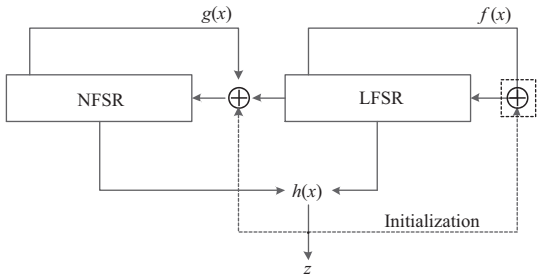

Figure 6 Schematic of Grain-like cipher.

4 NFSR based stream cipher

After eSTREAM project, many new design ideas and methods emerge endlessly. Among them a typical kind of stream ciphers take NFSR as the basic components. It usually utilizes both the nonlinear feedback and the nonlinear output to provide good sequence properties and security. There has not been yet unified mode or route to analyze the security of stream ciphers based on NFSR, but it can be expected as a mainstream of design scheme in quite a long time.

4.1 Grain series

Grain family. Grain family is a series of algorithms that combine the LFSR with the NFSR, and its feature is using the LFSR to provide good statistical characteristics and using the NFSR to add nonlinear disturbance, finally exploiting a filter function to mix the state from both two registers to further enhance good cryptographic properties as high nonlinearity, good correlation degree and statistical property. Its initialization is processed by loading the key and IV into the internal state, and clocking iterations of the state length, incorporated the output both to the LFSR and NFSR in the feedback loop shown in Figure 6. There are three stream ciphers, Grain-128a, Grain-128, and Grain-v1, in the Grain family, where Grain-v1 is in the eSTREAM portfolio and Grain-128a is standardized by ISO/IEC. In Crypto 2018, Grain family has been broken all (except the authenticated encryption mode) by a new fast correlation attack based on finite field extension and multiple linear approximations.

Grain-v1 [16] is tweaked to remove the vulnerability of Grain-v0 by increasing taps of filter function from the NFSR. It takes an 80-bit key and an 80-bit IV, using the LFSR and NFSR each of 80-bit length, and a filter function of 5 variables and algebraic degree 3. The near collision attack is the most important previous attack against Grain-v1. Grain-128 [51] supports a 128-bit key and a 96-bit IV, using the LFSR and NFSR each of 128-bit length, and a filter function of 9 variables and algebraic degree 3. Unfortunately, the low degree feedback of NFSR (a quadratic function) causes vulnerability against the dynamic cube attack. Grain-128a [52] has two operation modes, stream cipher mode and authenticated encryption mode, with higher degree of 6 for the nonlinear feedback polynomial, designed to avoid the attack against Grain-128. Its authenticated routine is similar with ZUC using the universal hash function.

Small state stream ciphers. Currently RFID technology has been widely used in access control, high speed charge, identification, cargo tracking, and other fields. Internet of things with a combination of RFID technology, which is a typical example of the new informatization tide generation, is gradually deep into people's lives in various aspects. The reliable information transmission for Internet of things must base on cryptographic algorithms to provide security services, and as a cryptographic algorithm using environment, it mainly has the following characteristics: application components are generally embedded processors of less computing power; the storage available for computing is often small due to constrained resource; the power consumption must be controlled within a certain range considering the functional requirements. Therefore, the traditional cryptographic algorithms cannot adapt well to this restricted environment, which makes the study of lightweight cryptographic algorithms a hot issue. An important principle in the design of traditional stream ciphers is that the size of the internal state is at least 2 times of the security level in order to resist TMDTO attacks, which makes the design of lightweight stream

ciphers a difficulty in terms of hardware implementation. To solve this difficulty, several novel lightweight stream cipher designs are proposed based on the Grain family's structure.

(1) Method one. Allowing for the key is fixed stored in some equipment, and the area to store a fixed value is much smaller than a same length register, in FSE 2015, Armknecht put forward a design method that to divide the internal state of the stream cipher into two parts, one is a variable state changed with time, the other is the fixed, which is generally realized by reusing the key for feasibility. Thus this kind of stream ciphers are also called CKU (continuous key-use) stream cipher, shown in Figure 7. The feature of such stream ciphers is that the key does not only participate in the initialization, but also work for the state update in the key generation. Its representative instances are Sprout, Fruit, Plantlet stream ciphers, in which the key section at the top is fixed, while the NFSR and LFSR section are mutable. The variable states in these three ciphers are 80, 80, and 101 bits respectively, but the goals are all to provide 80-bit security, which equal the key length.

In Sprout cipher [32], the key bit linearly participates or even does not participate in the state update of the NFSR, with the feedback function $n_{t+40} = g(N_t) + k_t^* + l_t + c_t^4$, round key function $k_t^* = k_{(t \bmod 80)} \cdot (l_4 + l_{21} + l_{37} + n_9 + n_{20} + n_{29})$ and 9-bit counter. It is the design difference from Grain family, and also the main revealed weakness. In addition that the register size is too small, several TMTO attacks and guess-and-determine attacks were presented and broke the cipher. These results prompt designers to carefully construct the key round function. The initialization of Sprout is processed by loading the 70-bit IV into the internal state and clocking 320 above iterations, XORed the output both to the LFSR and NFSR in the feedback loop.

For Fruit [33], there is an original version, which is broken by correlation attack and divide-and-conquer attack for the nonlinearity of its filter function is low. Then the designer modified the algorithm and presented Fruit-v2. The internal state consists of a 43-bit LFSR, a 37-bit NFSR, a 7-bit counter and an 8-bit counter. In Fruit, the key nonlinearly participate in the state update of the NFSR controlled by the different bit combinations of the counter. The initialization phase is a little complicated, by expanding the 70-bit IV to 130 bits, loading the key into the internal state, clocking 130 iterations XORed the output bit and expanded IV bit to the NFSR and LFSR in the feedback loop, setting the 7-bit counter, and clocking 80 blank iterations more without the feedback to the LFSR and NFSR.

Plantlet [35] takes much bigger internal state of a 40-bit NFSR, a 61-bit LFSR and a 9-bit counter, and a 90-bit IV. In Plantlet, the key is directly injected into the update function of the NFSR, where $k_t = k_{(t \bmod 80)}$, which makes the cipher much safer. The initialization phase of Plantlet is similar with that of Sprout. Plantlet takes 2 different phase-dependent LFSRs: a 60-bit LFSR for the initialization and a 61-bit LFSR for the keystream generation, while to save area, both LFSRs use the same register and logic, which is called double-layer LFSR. Moreover, a paper was published in eprint in 2017, which proposed a construction method of distinguishing attacks based on TMTO technique against such kind of CKU stream ciphers, and pointed out a solution that the IV should also be involved in the updates.

(2) Method two. The state update function of most stream ciphers either in the keystream generation or the state initialization is efficiently invertible. As a consequence, if an attacker manages to recover any internal state in keystream generation, he will also be able to recover the corresponding initial state, and by inverting the state initialization, the secret key, which is the key recovery process of TMDTO attacks. To break this basis for TMDTO attacks, the state initialization algorithm of Lizard cipher [53] represents the first instantiation of the FP(1)-mode [54], which works in the way:

- Loading : $S_{\text{load}} = (\text{IV}, K)$;
- Mixing : $S_{\text{load}} \rightarrow S_{\text{mixed}}$;
- Hardening : $S_{\text{init}} = S_{\text{mixed}} \oplus K$.

The structure of Lizard is similar to the Grain family, while its 121-bit internal state is distributed over two interconnected NFSRs. It takes 120-bit key and 64-bit IV, in order to achieve 80-bit security. We can see that computing the initial state from any of the later internal states is also possible for Lizard, but the secret key cannot be computed efficiently from the initial state due to the FP(1)-mode. It was proved that by using the FP(1)-mode, one can reach $\frac{2}{3}n$-bit security against TMDTO attacks aiming at key recovery for a keystream generator with an internal state of length n, instead of $\frac{1}{2}n$-bit security.

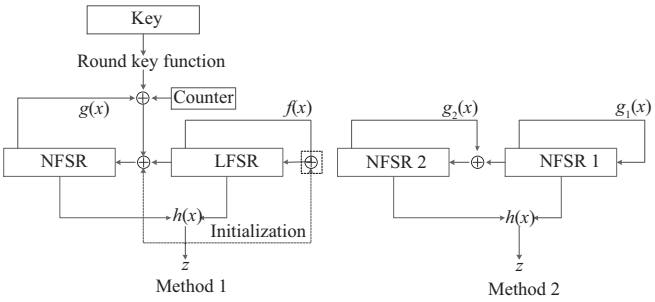

Figure 7 Schematic of small state stream cipher.

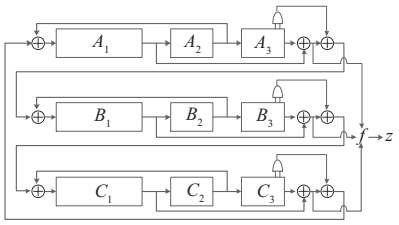

Figure 8 Schematic of Trivium-like cipher.

4.2 Trivium series

Trivium series are hardware oriented stream ciphers with extremely simple design, which take the structure of three interconnected NFSRs with low degree feedback functions and linear or quadratic filter functions, shown in Figure 8.

Trivium. Trivium [17] belongs to the eSTREAM final portfolio, which takes an 80-bit key and an 80-bit IV, generates up to 2^{64} keystream bits. It contains a 288-bit internal state. At each step, 15 specific state bits are extracted in three quadratic feedback functions to update 3 bits of the internal state in the initialization and keystream generation, and 6 of them are linearly XORed to compute the keystream bit. After loading the key and IV, the state is rotated over 1152 rounds for the initialization without outputs. Although its simple design is likely vulnerable to possibly devastating attacks, it survives a long period of public scrutiny despite. Its main attacks are cube attack against reduced-round Trivium, and the best known result is 855 rounds, while yet no attack has been discovered on its full version, which inspires more confidence in such simple design.

TriviA-SC. Trivia-SC [24] is the based stream cipher modified from Trivium with much larger internal state, for the authenticated encryption algorithm TriviA-ck (v2), which is a submission to CAESAR competition. Trivia-SC is loaded with 128-bit key and 128-bit IV, and uses three NFSRs of size 132, 105, and 147 bits, respectively. It also extracts 15 specific state bits in three quadratic feedback functions to update 3 bits of the internal state at each step, while a quadratic term of another two state bits is added into the linear sum to compute the keystream bit. Its initialization is similar with Trivium of 1152 rounds. Trivia-SC is parallelizable up to 64 bits. Its main attacks are also cube attacks against the reduced-round version, and the best known result just exceeds the 1000-round barricade.

Kreyvium. Kreyvium [55] is oriented to efficient homomorphic-ciphertext compression, which consists of five registers. While the modification from Trivium is the 128-bit top register and the 128-bit bottom registers, the three middle registers of length 93, 84, 111 bits are corresponding to Trivium. Kreyvium claims 128-bit security and accepts 128-bit key and 128-bit IV. Its initial loading is with a difference from the case of Trivium, that is, the IV and the key are also loaded to the top and bottom register respectively. Besides, the initialization is similar with Trivium of 1152 rounds. At each step, the key and IV both linearly participate in the state update, and the key also linearly participates in the keystream generation. The best known distinguishing attack against the reduce-round Kreyvium is 872.

4.3 Others

ACORN. ACORN [25] is a bit-oriented authenticated encryption algorithm in the finalistss of CAESAR, with the advantage of lightweight hardware realization. It adapts to 128-bit key, 128-bit nonce and less than or equal to 128-bit tag length, contains a state of 293 bits which is concatenated by six small LFSRs (in Figure 9). The function to compute the overall feedback bit and the function to filter the keystream from the state are both quadratic based on the majority function and chosen function. ACORN includes four stages of a non-invertible initialization of 1792 steps, an associated data processing of at least 256 steps, a plaintext encryption of at least 256 steps, an authentication tag generation of

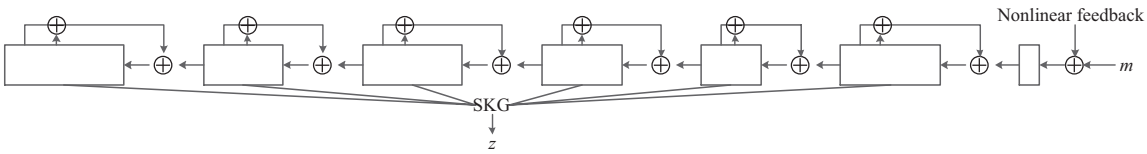

Figure 9 Schematic of ACORN.

768 steps, which presents better protect of the secret key in case of nonce reuse. It provides different parameters to distinguish different stages for resistance to sliding attacks. ACORN works without the need of the plaintext padding, the information of the plaintext length, and the steps for the finalization is fixed. ACORN can be implemented in parallel of 32 steps and as a result, its software implementation performance is also good. Its design is simple, and remains secure for the moment.

5 FCSR based stream cipher

There is another shift register, known as the FCSR, provided by rational 2-adic numbers. The sequence obtained from an FCSR has the same suitable statistical properties with the LFSR sequence, such as the known period, balancedness, equal distribution of patterns, moreover it has intrinsic nonlinearity inherited from its quadratic transition function added with carries. Thus a linear filter can be used to extract the keystream from the internal state, since FCSR sequences are predictable by synthesizing algorithms and therefore not suitable to be directly output. Even though the algorithm F-FCSR designed based on FCSR, which was submitted to eSTREAM project, was eventually found not secure, it cannot rule out the possibility of new safe stream cipher instances based on FCSR structure. That is, this kind of shift register is still a potential alternative source sequence.

There also exist two presentation forms for FCSR: the Galois setup and Fibonacci setup. Here we focus on the Galois setup, which is an automaton that computes the binary expansion of the quotient of an integer p by a fixed odd integer q following the ascending order of the powers of 2. Let $d = (1-q)/2$ and put $d = \sum_{i=0}^{n-1} d_i 2^i$, with $d_i \in \{0,1\}$, $d_{n-1} = 1$. The FCSR automaton is composed of a main register M and a carry register C. The main register M is composed of n binary cells, where each bit is denoted by $m_i(t)$ $(0 \leqslant i \leqslant n-1)$. The carry register C consists of l cells, where $l+1$ is the Hamming weight of the binary expansion of d. $c_i(t)$ denotes the carry bit, and it is present only if $d_i = 1$, $i < n-1$. For FCSRs generators, it requires that q is a prime of bit size $n+1$, and the order of 2 modulo q is $|q|-1$, while $T = (|q|-1)/2$ is also prime and the Hamming weight of the binary expansion of $d = (1+|q|)/2$ is about $n/2$ or slightly greater. The n bit word d can define a filter which outputs a keystream word of bitsize s as follows: bit j of output word $= \sum_{i=0}^{n/s-1} d_{si+j} m_{si+j}$. The total structure of the keystream generator based on an FCSR is showed in Figure 10.

F-FCSR. F-FCSR [56] contains two designs: F-FCSR-H, which takes an 80-bit key and an 80-bit IV, and F-FCSR-16 which takes a 128-bit key and a 128-bit IV. The initialization phase of F-FCSR consists of three steps: K/IV loading to the internal state, re-initializing the main register with its 20/16 clocks outputs, clocking $n+2$ times more before the keystream output. It chose $q = -1993524591318275015328041611344215036460140087963$ and $q = -1839714408456194711298691618093441316582983176559231357530171284621556187150 19$ for F-FCSR-H and F-FCSR-16, respectively. F-FCSR was broken by Hell et al. [57], mainly using its linear translation features, i.e., if all the carry registers and feedback bits value 0, the register values will be linear changed in the flip-flop. The attacker just used about 2^{26} bytes of keystream to list enough linear equations in order to break F-FCSR.

6 PANAMA-like stream cipher

PANAMA structure designed by Daemen et al. [58] has not yet been found any security problem as a keystream generator. A PANAMA-like keystream generator consists of two internal states: state a and

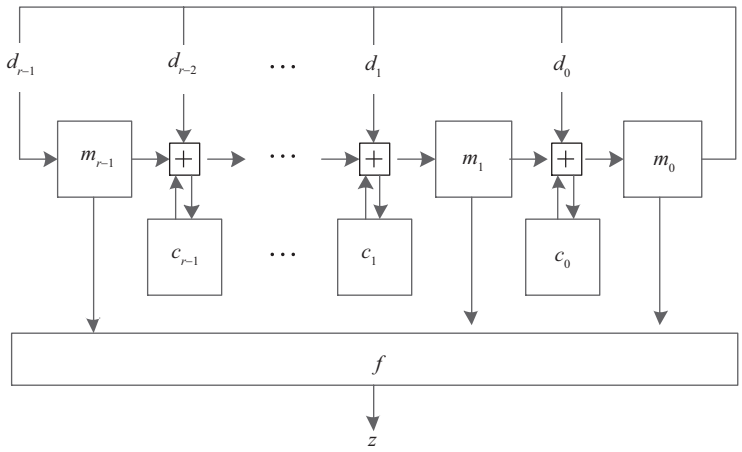

Figure 10 Keystream generator based on FCSR.

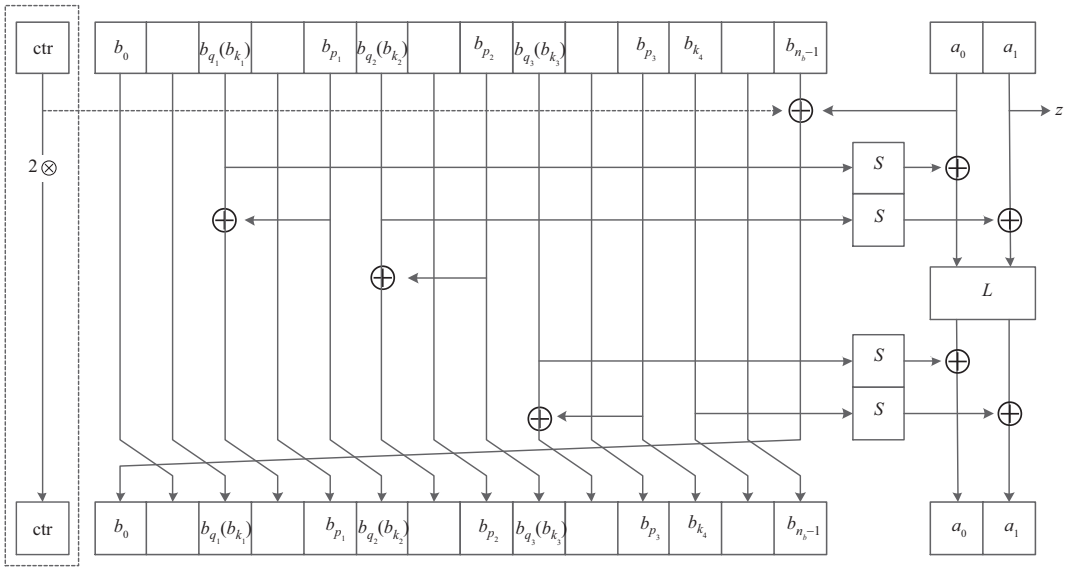

Figure 11 Initialization (in grid)/keystream generation of Enocoro.

buffer b, their update functions ρ and λ, and an output function f, where ρ includes a substitution-permutation network (SPN) transformation that uses parts of buffer b as a parameter $a(t+1) = \rho(a(t), b(t))$, λ is a linear transformation that uses a part of state a as a parameter $b(t+1) = \lambda(b(t), a(t))$, f outputs a part of state a which is typically no more than 1/2.

Enocoro. Enocoro is a lightweight stream cipher family standardized by IEC and ISO in ISO/IEC 29192-3, which has two versions named according to their key lengths as Enocoro-80 and Enocoro-128v2 both with 64-bit IV. There has not been any attacks ever violated their full-round security. Enocoro (in Figure 11) is a byte-oriented stream cipher with excellent software performance, of the structure specified by 11 parameters $(n_b; b_{k_1}, b_{k_2}, b_{k_3}, b_{k_4}, q_1, p_1, q_2, p_2, q_3, p_3)$. Here n_b is the buffer size in byte, and the state a consists of two bytes. Parameters b_{k_1} to b_{k_4} are the inputs from the buffer to the ρ function. The ρ function works as shown in the schematic, with a linear transformation L as a 2-by-2 matrix over GF(2^8), and four S-boxes of S_8 permutation. Parameters q_1, p_1, q_2, p_2, q_3, p_3 define the linear λ function, which shuffles as shown in the schematic. The output is $a_{1,t}$. The initialization firstly sets the registers with the K/IV and constants, then updates the state by clocking 96 blank iterations with an XOR of the counter to b_{n_b-1}.

MUGI. MUGI is a typical PANAMA-like stream cipher with a 128-bit key and a 128-bit IV, suitable for both software and hardware implementations. It was selected for standardization by the Japanese

government project CRYPTREC and is also one of the proposed ISO stream cipher standards. No security flaw of the full MUGI stream cipher has been reported so far. MUGI is word-oriented with a word size of 64 bits, and its internal state consists of 19 words. The update function ρ of state is a kind of Feistel structure, where the F-function consists of a data addition from the buffer, a non-linear transformation using the S-box, a linear transformation using the maximum distance separable (MDS) matrix and a byte shuffling, which are the same as for AES. The update function λ of buffer b is a linear transformation as $b_{i,t+1} = b_{i-1,t}$ $(i \neq 0, 4, 10)$, $b_{0,t+1} = b_{15,t} \oplus a_{0,t}$, $b_{4,t+1} = b_{3,t} \oplus b_{7,t}$, $b_{10,t+1} = b_{9,t} \oplus (b_{13,t} \lll 32)$, which is similar as Enocoro. The initialization of MUGI is divided into three steps: initializes the buffer with 16 rounds of ρ function on key, initializes the state with another 16 rounds of ρ function on IV, mixes the whole internal state with 16 more rounds of the whole update function.

7 Random shuffled stream cipher

In order to achieve high efficiency of software implementation, a series of algorithms represented as RC4 cipher are proposed with an employ of randomly shuffled tables, which shuffle a list to create random permutations. Since RC4 has got a number of applications in practice, it inspired subsequent designs of this type of structure, against which the state recovery attacks often require extremely high time complexity. Now the understanding for this type of stream ciphers is still at the stage of development, and any influential analysis result on RC4 may advance the acknowledge of this type of structure with leaps.

RC4 series. RC4 [1] is a byte-oriented stream cipher, which uses a numerical table containing 0 to 255, and a permutation mode to generate replacements with two bytes index-pointers. The key length of RC4 is variable typically between 40 and 128 bits, and it does not take a separate nonce alongside the key as the modern stream cipher. The table is initialized with a mix of the key. And then in the keystream generation phase, it modifies the table and outputs a byte of the keystream at each iteration (in Figure 12). Its speed and simplicity leads to efficient implementation in software and easy development in hardware, which makes it a success over a wide range of applications until that in 2015 it was prohibited for all versions of TLS. Although the initialization phase of RC4 and the statistical properties of its first several bytes have been questioned a lot, its key generation phase is still considered secure by abandoning enough keystream bytes ahead, especially for its 128-bit security.

In view of the above weaknesses, the designer upgraded the algorithm named Spritz [59], which still adopts the similar structure of RC4, while the state update function is more complex in order to obtain better randomness. However, in FSE 2016, Banik and Isobe found that the randomness of the first two bytes of Spritz was still not ideal enough to resist to distinguish attacks [60].

HC-128. HC-128 [22] is a software-efficient cipher in eSTREAM portfolio, which maps a 128-bit key and a 128-bit IV to at most 2^{64} keystream bits. It also has an extended version of HC-256 [61]. HC-128 consists of two secret tables, each one with 512 elements of 32 bits. At each step in the keystream generation phase, one element of a table is updated with a function of the elements from its own table, meanwhile one 32-bit output is generated by a function of elements from both two tables. These four functions are nonlinear with operations of modulo-2^{32} addition, modulo-512 subtraction, XOR, shift (rotation) and table look-up. The initialization process of HC-128 consists of expanding the key and IV into these two tables, and running the cipher 1024 steps with the outputs being used to replace the elements in these two tables. That is, these two tables are interrelated controlled in the initialization like a combination of the alternating step generator with RC4 cipher, while they are self-controlled in the keystream generation. HC-128 is suitable for modern superscalar microprocessors since the dependency between operations in HC-128 is very small. HC-128 seems secure, for it is lack of effective analysis method and means, which on the other hand, limits its further application in actual.

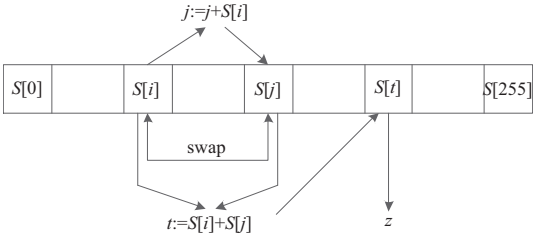

Figure 12 Keystream generation of RC4.

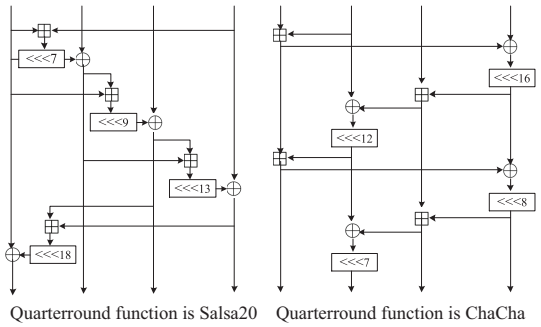

Figure 13 Round function in ARX based stream cipher.

8 ARX based stream cipher

Many modern stream ciphers are proposed as ARX based algorithms, whose round function only involves three hybrid operations of module addition ($\boxplus$), interword rotation ($\lll$) and XOR ($\oplus$), to achieve required security strength. These ARX operations are popular since they are relatively fast and cheap in software implementation, and also run in constant time which is immune to timing attacks. ARX based stream ciphers have attracted a number of scholars' interest and the security analysis method for this kind of designs have made good progress.

Salsa20 and ChaCha. Salsa20 [19] and the closely related ChaCha family [62] are both 32-bit module addition, XOR and rotation operations based core hash functions, where Salsa20 is the original cipher selected in eSTREAM software profile, and ChaCha is a modification published in 2008 using a new round function to increase diffusion. Google has selected ChaCha20 along with Bernstein's Poly1305 message authentication code as a replacement for RC4 in TLS used for Internet security. Salsa20 and ChaCha both work on an internal state of sixteen 32-bit words arranged as a 4×4 matrix and map a 256-bit key (128-bit key is also suitable for Salsa20), a 64-bit nonce, and a 64-bit counter to the keystream of 512-bit blocks. Their initial state are both made up of 8 words of key, 2 words of stream position (counter), 2 words of nonce, and 4 words of constant, while the arrangements are different. The core operation in Salsa20 is the quarterround function (4 word $\rightarrow$ 4 word), shown in Figure 13. Two consecutive rounds of columnround and rowround (16 word $\rightarrow$ 16 word) together, that applies the quarterround function to vectors of the four columns and then the four rows in the 4×4 matrix with the starting element in the diagonal, are called a doubleround. Salsa20 performs 10 doubleround on its input, and finally the mixed matrix is added word by word to the original matrix to obtain the 64-byte keystream block. Here it uses littleendian function to transform between 4-byte sequence and a word. ChaCha replaces Salsa20's quarterround, shown in Figure 13, which updates each word twice with the same number of operations in Salsa20. Rather than alternating quarterrounds down columns and across rows in Salsa20, ChaCha performs quarterrounds down columns and along diagonals in a doubleround. ChaCha20 also uses 10 iterations of the doubleround and the same output function. There have been only attacks on reduced round Salsa20/12 rather than the full round Salsa20/20 so far, and there have been attacks for 256-bit ChaCha6 with complexity 2^{139}, 256-bit ChaCha7 with complexity 2^{248}, 128-bit ChaCha6 with 2^{107}, while 128-bit ChaCha7 is still secure.

Rabbit. Rabbit [21] is selected in eSTREAM software Profile, using a 128-bit key and a 64-bit IV to generate up to 2^{64} blocks of keystream. The size of the internal state is 513 bits, divided into eight 32-bit state variables, eight 32-bit counters and one counter carry bit. The counter carry bit is initialized to zero. The state variables and counters are initialized by loading the expended key, followed with four iterations of the next-state function and one more XORing operation on the counters, and then the counters are XORed with the IV, followed with four more system iterations. Each state variable is updated by a coupled nonlinear consecutive iterations of a g-function, which squares a 32-bit number to produce a 64-bit number and then combines the left half and the right half of that square number with XOR to produce a 32-bit result, and the modulo-2^{32} addition and rotation operations. The counters are

updated with the dynamics carry bit, which ensures a lower bound on the period length for the state variables. Rabbit generates a 128-bit block of keystream at each iteration, which is derived by XORing the internal state bits. The cipher's strength rests on a strong mixing of its inner state. There exists a small bias in the output of Rabbit, however the best known attack against Rabbit is with complexity 2^{158}, which is still not a threat to its security.

MORUS. MORUS [63] is a family of authenticated ciphers selected in the additional finalists of CAESAR, with three parameter sets including MORUS-640-128, MORUS-1280-128, MORUS-1280-256, in which the internal state of size either 640 or 1280 bits consists of five elements of size either 128 or 256 bits, and the key size is either 128 or 256 bits. MORUS uses a 128-bit nonce that should not be reused, and generates 128-bit tag for authentication. The design of MORUS is based on ARX operations, with a similar structure to HELIX/PHELIX. It includes four stages of 16 step associated initialization, data processing, plaintext encryption, more than 8 steps tag generation with the length of the associated data and message. Each state update of MORUS contains a short critical path of 5 rounds with similar operations to update the state, where the associated data or plaintext blocks are embedded in round 2 to 5. The round function uses the basic operations of bit-wise AND (&), XOR ($\oplus$) and rotation, where the rotation involves Rotl_xxx_yy function, which divides an xxx-bit state element into 4 words of yy-bit and then performs left rotation operation for every yy-bit word, and the left rotation of the whole state element for better diffusion. Each keystream word is a nonlinear combination of the internal state passed the critical path. MORUS is software-oriented aimed at high-end CPU application, which no more depends on the AES instructions to achieve high performance. It makes full use of the advantages of AVX2 instruction set, and provides good software and hardware performance, strong adaptability and security. However, all versions of full MORUS were broken by the correlation of quadratic Boolean functions to search for linear trails in Crypto 2019.

9 Sponge structural stream cipher

Since Keccak won the SHA-3 competition, there recently emerged a series of stream ciphers designed based on permutations or Sponge structure, which outputs some internal state directly as the keystream sequence. Keccak [23] is based on a novel approach called Sponge construction, which uses a wide random permutation, and allows inputting (absorbing) and outputting (squeezing) any amount of data, which leads to great flexibility. The advantage of this structure is that it can provide authentication function naturally without additional authentication module, and provide high security as long as the round function of the hash algorithm has good security properties. It can be expected that this kind of stream cipher structure will be greatly developed in future, but its security analysis is still a problem to be solved for whether such ciphers have undiscovered weaknesses is unknown. From the implementation point of view, how many rounds should be included in the permutation is also a very important issue. The representation of Sponge structure stream ciphers includes KETJE, NORX, ICEPOLE, Ascon, APE, which are all embodied in fixed permutations, where Ascon takes SPN structure with a nonlinear layer similar to SHA3, Ketje and Keyak both directly adopt the fixed permutation of SHA3 with only difference in the state scale, and NORX uses the design method of ARX class.

Ascon. Ascon [64] is a family of authenticated encryption designs selected as the first choice for use case 1 in the finalists of CAESAR, including Ascon-128 and Ascon-128a two algorithms, both with 128-bit key, 128-bit nonce, and 128-bit tag. Ascon adopts a Duplex structure (in Figure 14), and a fixed permutation of iterative functions with SPN structure. The core permutations operate on a Sponge state of 320 bits, including a rate of r bits and a capacity of c bits. During the initialization and finalization stage, the number of iterations in Ascon-128 and Ascon-128a's permutation are both 12 rounds, while in the message processing stage, the number of iterations in Ascon-128's permutation is 6 and each time a 64-bit message block is embedded, the number of iterations in Ascon-128a's permutation is 8, and each time a 128-bit message block is embedded. According to the security proofs of Sponge structure, the design core of Ascon is the fixed permutation. The iterative function for Ascon's permutation consists of constant addition,

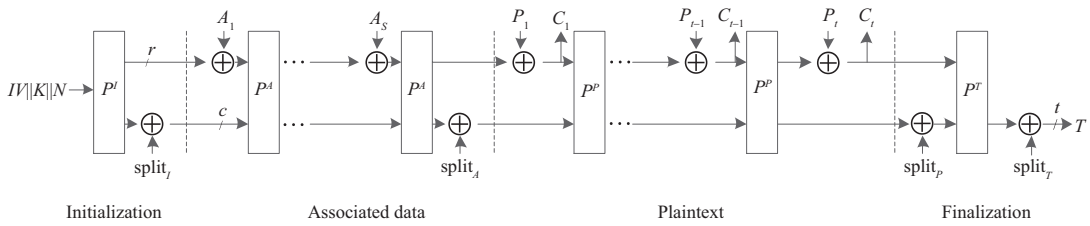

Figure 14 Schematic of Ascon.

substitution layer and linear diffusion layer. The substitution layer uses a 5×5 S-box similar to Keccak, which is directly constructed with the bit-slice technology, so it can be parallel implemented efficiently on the 64-bit processor without the need of table look-up, which resists to the cache timing attack. The linear diffusion layer is similar to SHA2's $\sum$ function, which uses five similar linear transformations, employing the operations of XOR and 64-bit word rotation for internal diffusion. Each linear transformation has a branch of 4, and the rotation parameters are carefully selected to increase the number of minimum active S-boxes in Ascon's permutation. The main feature of Ascon is lightweight, and taking into account both of hardware implementation cost and software performance. Compared with other algorithms based on the Sponge structure, Ascon uses more complex initialization and finalization, which makes it hard to restore the key or forge the message even acquired the intermediate state information. The analyses on Ascon are mainly based on cube attacks, and the best known result is a key recovery attack against 7-round Ascon.

10 Block cipher based stream cipher

Many stream ciphers are proposed with the design idea of block ciphers, such as based on the basic operation, round function or algorithm structure of a block cipher, even the entire block cipher with efficient software-oriented instruction set (such as AES) and the tweakable block cipher with work mode. This kind of stream ciphers are difficult to evaluate with the traditional analysis method against stream ciphers, since its internal state is always very large.

AEGIS. AEGIS [65] is an authenticated encryption scheme in the finalists of CAESAR, which uses the AES encryption round function as the basic module, and builds its round transform with the parallelization of several AES round functions, making full use of AES instructions and resources in modern CPU. It contains three parameter selections of AEGIS-128L, AEGIS-128, AEGIS-256 with 128/128/256-bit key, 128/128/256-bit nonce, 1024/640/768-bit state, 128/128/128-bit tag, and 8/5/6 AES round functions to process 32/16/15-byte message block in each step. Since their structures are similar, only the state update function of AEGIS-128 is shown in Figure 15, where R indicates the AES encryption round function without XORing the round key and w is a temporary 16-byte word. The initialization of AEGIS consists of loading the key and IV into the state, and running the cipher for steps with the key and IV being used as message to make the difference fully random. Next, the associated data is used to update the state. At each step of the encryption, a plaintext block is used to update the state, and also encrypted with the leakage information achieved by processing simply operations like &, $\oplus$ on some of the intermediate state. Finally, the authentication tag is generated by updating the state with the length of the associated data and message. AEGIS is fast software implemented with about half computational cost of AES, and suitable for network communication. It also offers a high security as long as the nonce is not reused, which takes full use of the analysis results of AES, while the techniques adopt in its IV and key processing, plaintext embedding, ciphertext leakage, and message length processing are still worthy of further analysis.

Tiaoxin. Tiaoxin [66] is an authenticated encryption scheme submitted to CAESAR, which constructed by AES round function and generalized feistel structure. It takes 128-bit key and 128-bit nonce, with three substates of 3/4/6 128-bit words. Each substate update round function is independent and calls two AES round functions as the basic operations parallelly, where the former is keyless and the pos-

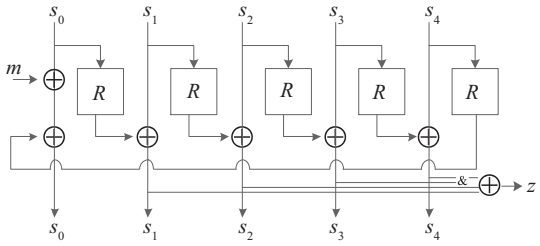

Figure 15 Schematic of AEGIS-128.

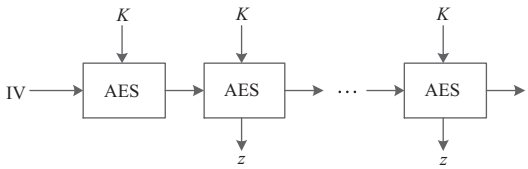

Figure 16 Schematic of LEX.

terior uses a constant as the subkey. It processes the associated data and message in blocks of 2 words, which are respectively XORed to the first and second state, and meanwhile their XOR is embedded to the third state, in order to improve the whole security and reduce the average use of AES round functions. Tiaoxin works in four phases of initialization, processing associated data, encryption, and finalization. The initialization loads the key and nonce into the three substates and conducts 15 rounds. Then the associated data and message blocks are processed, and the ciphertext blocks are generated by XOR and AND operations on the words of the state. Finally, it goes through 20 more rounds to produce the tag as an XOR of all the words of the state. Although the security of Tiaoxin is not verifiable, it is easy to provide the optimal difference path and rule out the high difference, since the substates adopt different scales without diffusions to one another and the state update process is simple for analysis. The output function also makes it impossible to get the internal state value directly. Different constants are used for initialization and finalization to improve the security against sliding attacks, cyclic difference attacks, etc. Tiaoxin needs to call 3 times of AES round functions to process 128-bit data, which improves the previous 4 times design. Thus, Tiaoxin has a good software implementation performance.

LEX. LEX [67] is a 128-bit key stream cipher, which is a submission for the eSTREAM project. The design is simple and uses AES in a natural way: at each AES round it outputs certain four bytes (different in the even and odd rounds) from the intermediate variable as the keystream (in Figure 16), where the roundkeys are generated first by a standard AES key-schedule with the secret key, and the 128-bit IV is encrypted as the input of AES. Although LEX has some security problem, it still provides a general design principle that can take an application of any other block ciphers, but a careful study is required in each particular case, in order to decide which parts of the internal state may be given as the output and at what output frequency. This mainly depends on the strength of the cipher's round function and the cipher's key-schedule. For example, ciphers with good diffusion may allow to output larger parts of the internal state at each round than ciphers with weak diffusion.

Scream. Scream [68] is a word-oriented stream cipher, aimed to be a more secure SEAL, which is fast for software implementations with a block cipher-like round function. Its half round function operates on a 64-bit block represented as a 2×4 matrix of bytes, where each byte is sent through an S-box, then the second row in the matrix is cyclic shifted, and finally each column of the matrix is mixed by multiplying an invertible matrix (in Figure 17). The round function F is a mix of a Feistel ladder and an SP-network, using two half-round functions with different S-boxes and invertible matrixes, where the S-boxes are derived from the AES S-box in a key-dependent fashion. Scream maintains a state that consists of the evolving state x and some round keys y, z, each of one 16-byte block, and a mask table W of 16 blocks. At round i, the evolving state is modified by $x = F(x \oplus y) \oplus z$, and then it outputs $x \oplus W[i \bmod 16]$ and rotates y by a few bytes after each use, which provides some protection against low-diffusion attacks and linear analysis. Both the mask table and the round keys are modified every pass of 16 rounds. Scream uses a 128-bit key and a 128-bit nonce, and the initialization is straightforward designed using the round function F to derive all the quantities needed.

Deoxys. Deoxys [69] is an authenticated encryption cipher in the finalists of CAESAR, based on a tweakable block cipher Deoxys-BC with work mode. Deoxys-BC is an AES-like design that uses rounds of SPN transformations, where only its subtweakeys production is different under a TWEAKEY framework

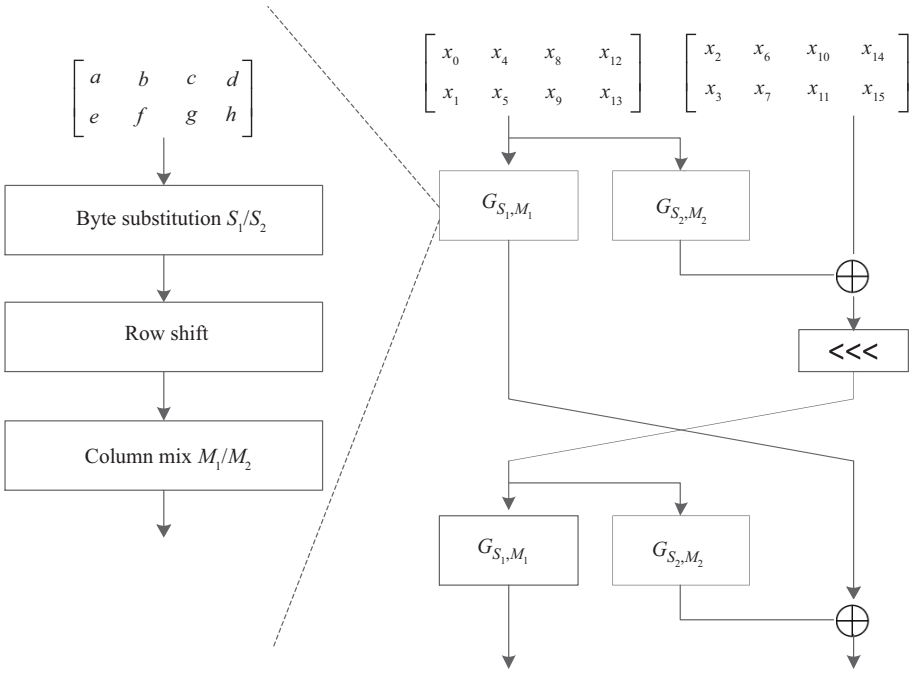

Figure 17 State update function of scream.

generally proposed to unify the key and tweak into 128-bit words, which are iteratively XORed as the subtweakey and updated by parallel byte permutation and byte shuffle (in Figure 18). The security of the tweakable block cipher designed under TWEAKEY framework is worth of further consideration. Deoxys-BC has two versions: Deoxys-BC-128 and Deoxys-BC-384 with 256/384-bit cumulative size of the key and tweak, 14/16 rounds of SPN transformations, used to build Deoxys with 128-bit and 256-bit key, respectively. Deoxys has two modes: nonce-respecting Deoxys-I and nonce misuse-resistance Deoxys-II. Several parameter sets are recommended as Deoxys-I-128-128, Deoxys-I-256-128, Deoxys-II-128-128, Deoxys-II-256-128, where the parameters are the key length and tag length. The nonce size is 64 bits for Deoxys-I and 120 bits for Deoxys-II. The handling of the associated data for Deoxys-I and Deoxys-II has the same work mode. The message processing for Deoxys-I is similar to the tweakable block cipher generalization of OCB3, and the message processing for Deoxys-II heavily relies on an inverse-free authenticated encryption mode SCT, where the main difference is that Deoxys-I applies one pass on the message blocks, while Deoxys-II performs two passes, shown in Figure 18. Here E_K^T denotes the tweakable block cipher with key K and tweak T, and N denotes the nonce. Deoxys performs well in software, particularly for small messages. The authenticated encryption algorithm based on a tweakable block cipher is clear in structure, easy for security proof, and likely to obtain birthday-bound security.

11 Block cipher work mode based stream cipher

Using the block cipher work mode, a block cipher can be converted into stream ciphers. Under the assumption that the based block cipher is safe, the work mode can provide some verifiable security for the generated stream cipher. There are several authenticated encryption algorithms designed with block cipher work mode in CAESAR, they usually call AES block cipher as a black box, and optimize the software implementation with AES instruction set.

CFB. The cipher feedback mode (CFB) transforms a block cipher into a self-synchronous stream cipher, shown in Figure 19. Its main advantage is of a limited error propagation, can be used for authentication and realize the synchronization, while the disadvantage is that the encryption efficiency is relatively low.

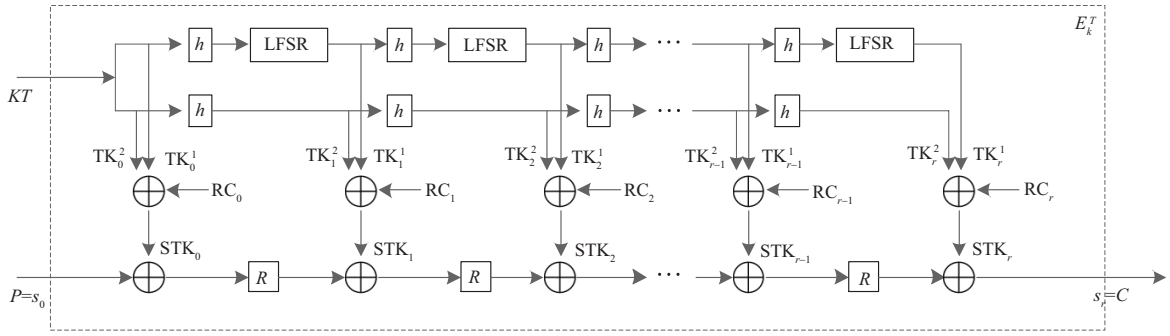

Figure 18 TWEAKEY framework for Deoxys-BC.

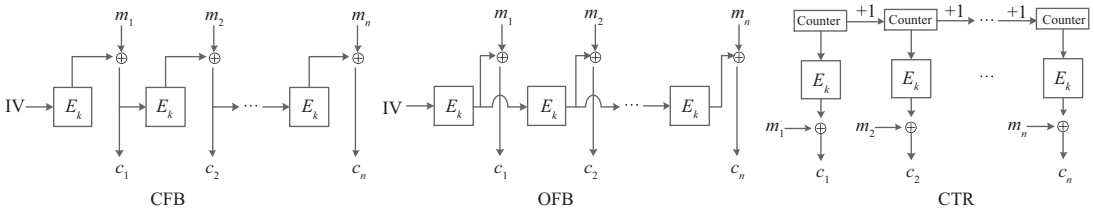

Figure 19 Block cipher work modes.

OFB. The output feedback (OFB) mode makes a block cipher into a synchronous stream cipher. It is quiet similar to the CFB mode, while it encrypts the previous keystream block rather than the previous ciphertext block as CFB mode. The concrete structure is shown in Figure 19.

CTR. The counter mode can turn a block cipher into a stream cipher in the way of generating the next keystream block by encrypting successive values of a counter with IV. The advantages of counter mode encryption (CTR) is safe, effective, parallelized, and suitable for any length of plaintext information. Moreover only encryption algorithm is used in the process of both encryption and decryption.

OCB. Offset codebook (OCB) [70] is one of the most classic authenticated encryption mode, which is adopted as ISO/IETF standards and in the finalists of CAESAR. It works as using maskings to disguise plaintext and ciphertext, where the last message block is a little different in cases, and generating the tag in the Checksum manner (in Figure 20). The processing of each message block just need some simple XOR operations beside invoking the encryption of AES. OCB does not need to pad the message, and is almost as fast as CTR mode. OCB has good parallel property, since most OCB calculations are independent from each other, which allow to accelerate the software and hardware implementation with more computing resources. OCB is online, which does not need to know the information of all the message or associated data before do the processing, and the processing can be done in any order. When multiple associated data in a series of encryption remain the same, it does not need to recalculate the associated data, which reduces an amount of calculation. Compared with Galois counter mode (GCM), the function and safety of OCB is similar, while its software achievement is more efficient. In addition, OCB tags can be truncated to shorter length, while GCM cannot. The security of OCB is fully demonstrated in 10 years of research and development. OCB requires non-reused nonce, and does not have birthday-beyond security. If the underlying block cipher is a strong pseudo-random permutation, OCB is a proven safe nonce-based authenticated encryption algorithm. Various key lengths of 128/192/256 bits and tag lengths of 64/96/128 with no more than 120-bit nonce are provided for different security levels in OCB.

COLM. COLM [71] is selected in the finalists of CAESAR, which combines PHASH and EME (encrypt-mix-encrypt) modes. It first processes the associated data and nonce with parallelizable message authentication code (PMAC) to achieve the intermediate variable IV, then executes the first layer of electronic codebook (ECB) after masking the plaintext, next linearly mixes the IV and the successive accumulation output from the first layer of ECB as the inputs of the second layer of ECB, finally masks the output of the second layer of ECB to generate the ciphertext (in Figure 21). COLM is designed with the goal to achieve online misuse resistance, to be fully parallel and secure against blockwise adaptive

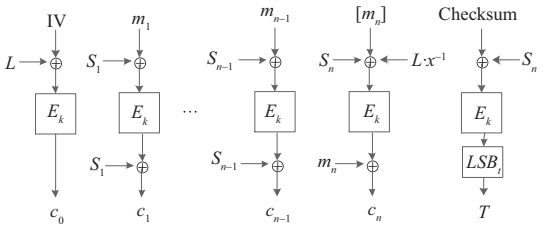

Figure 20 Schematic of OCB.

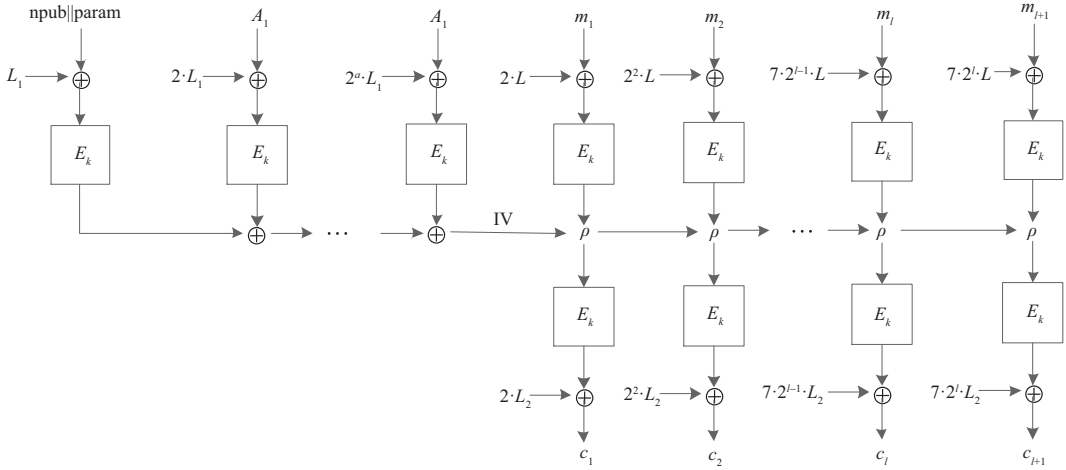

Figure 21 Schematic of COLM.

adversaries. COLM uses AES-128 with a key and state of 128 bits, takes a nonce of 64 bits and has two parameters for intermediate tag interval and length to set the param of 64 bits, aiming to achieve 64-bit security, which is supported by security proofs corresponding to the birthday bound security of the block size of AES.

JAMBU. JAMBU [72] is a lightweight authenticated encryption mode submitted to CAESAR, motivated by developing secure structure that can convert lightweight block ciphers to lightweight authenticated encryption ciphers. It uses SIMON/AES-128 to construct the specific ciphers. JAMBU contains four parts of initialization, associated data processing, plaintext processing and tag generation. The initialization processes the nonce with an encryption, and then does a simple state expansion. The associated data is padded and divided into data blocks, which are embedded to update the state sequentially, while the plaintext processing is similar and the ciphertext blocks are output simultaneously. For the last block of message, its ciphertext block should be truncated to ensure the same length with the plaintext. The tag is generated by once state update without data embedding, one more encryption on the state, and an XOR of the words of state. JAMBU accepts the underlying block ciphers with $2n$-bit block size, and a key size same with the block cipher, a tag length of n bits, and the message length is limited with 2^n bits under a single key. For SIMON-JAMBU, the designer recommends SIMON64/96, SIMON96/96, SIMON128/128 with parameters of 96/96/128-bit key, 32/48/64-bit nonce, 96/144/192-bit state and 32/48/64-bit tag. For AES-JAMBU, AES-128 is used with parameters of 128-bit key, 64-bit nonce, 192-bit state and 64-bit tag. The padding mode of JAMBU does not need to store the data length. It uses different constants to distinguish the four stages to resist sliding attacks, etc. For lightweight purposes, the internal state of JAMBU has not extended to twice the length of key, so it is necessary to explore its security in the tweakable key mode.

12 Stream ciphers for fully homomorphic encryption systems

Homomorphic encryption (HE) technology can directly operate the ciphertexts without decryption, and the result is the same as directly operate the corresponding plaintext, which makes it of great importance to areas such as cloud computing, ciphertext search, electronic voting, and multi-party computing. However its actual application is still faced with two problems of realization consumption and homomorphism capacity limitation. In order to solve the problems, some scholars proposed to apply symmetrical ciphers as fully homomorphic encryption, in order to minimize the current inherent cost of the homomorphism implementation, such as time and storage, meanwhile improve the homomorphism ability, i.e., limit the noise level.

For example, in cloud applications, it is often assumed that some data is sent encrypted under an HE scheme to the cloud to be processed in a way or another. It is thus typical to consider in the first step that a user (Alice) encrypts some data m under some other user's public key pk (Bob) and sends some homomorphic ciphertext $c = \mathrm{HE}_{\mathrm{pk}}(m)$ to a third-party evaluator in the Cloud (Charlie). However, all HE schemes proposed so far suffer from a very large ciphertext expansion; the transmission of c between Alice and Charlie is therefore a very significant bottleneck in practice. The problem of reducing the size of c as efficiently as possible is to encrypt m with a symmetric encryption scheme E under some key k randomly chosen by Alice, who then sends a much smaller ciphertext $c' = (\mathrm{HE}_{\mathrm{pk}}(k), E_k(m))$ to Charlie. Given c', Charlie then exploits the homomorphic property of HE and recovers the original $c = \mathrm{HE}_{\mathrm{pk}}(m) = \mathcal{C}_{E^{-1}}(\mathrm{HE}_{\mathrm{pk}}(k), E_k(m))$ by homomorphically evaluating the decryption circuit. This can be assimilated to a compression method for homomorphic ciphertexts, c' being the result of applying a compressed encryption scheme to the plaintext m and c being recovered from c' using a ciphertext decompression procedure. In that approach obviously, the new encryption rate $|c'|/|m|$ becomes asymptotically close to 1 for long messages, which leaves no significant margin for improvement. However, the paradigm of ciphertext compression leaves totally open the question of how to choose E in a way that minimizes the decompression overhead, while preserving the same security level as originally intended. Typical algorithms used in fully homomorphic encryption are represented by the block cipher LowMC [73], the aforementioned stream cipher Kreyvium and the following new structure stream cipher FLIP, which is designed according to a natural idea that possesses both the constant noise properties of block cipher sequence and the low noise properties of stream cipher sequence. A generic construction based on an HE with stream ciphers is shown in Figure 22.

FLIP. FLIP [39], a new stream cipher structure named filter permutator is proposed recently. The structure consists of a constant key register, a permutation generator, a pseudo-random generator and a nonlinear filter function (in Figure 23). In each clock cycle, the permutation generator first creates a bit-permutation according to the output of the pseudo-random generator, then uses the permutation to transform once the key register, finally uses the nonlinear filter function on the key register to produce one keystream-bit . The key length of FLIP is much more than its security level, such as the latest version uses 530-bit key length to achieved 80-bit security. From the perspective of the current analysis results, the filter function used in this kind of structure is of strictly high requirement, otherwise the cipher is easy to be broken for revealing a lot of key information. The main advantage of FLIP is that the filter function always acts directly on the key bit, which makes the output noise always a constant, and with a great homomorphism ability.

13 Provable secure stream cipher

The current status of stream cipher design is characterized by a considerable discrepancy between theory and practice. On the theoretical side, seminal work by Shamir [74], Blum et al. [75], Yao [76], and Goldreich et al. [77] in the early 1980s produced the well founded theory of pseudo-random generators, which represents one of the major achievements in the area of provable security. A pseudo-random number generator (PRNG) can be viewed as an IV-less stream cipher. It expands a short seed, e.g., a

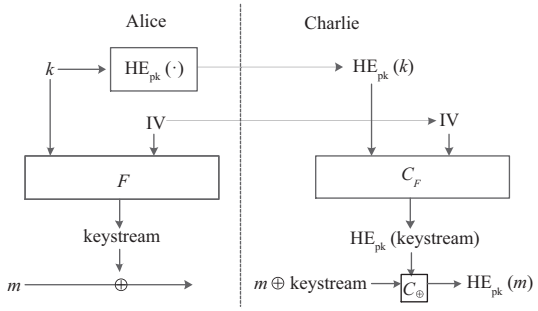

Figure 22 HE with stream ciphers.

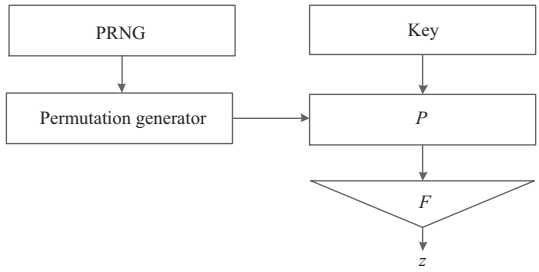

Figure 23 Schematic of FLIP.

key, into a larger bit string in such a way that if the input seed is secret and randomly drawn, then the resulting output is computationally indistinguishable from a perfect random sequence. The research effort on security proofs for PRNGs has not only led to remarkable generic results, it has also led to "provably secure" PRNG constructions exploiting the conjectured one-wayness of some specific permutation or function f, which generally rely on the iteration of f and the extraction of a few bits at each iteration. However, current provably secure PRNGs are still generally regarded as too complex and inefficient to provide really practical stream ciphers. The lack, for these various algorithms, of an extra IV parameter represents an additional drawback.

On the practical side, extremely efficient stream ciphers have been proposed, which allow much faster software encryption than existing block ciphers such as AES and/or require much lower computing resources for hardware implementations. However, the design of secure stream ciphers is not currently as well understood as the design of secure block ciphers. The state of the art of the cryptanalysis of stream ciphers has evolved significantly over the last ten years with the development of attack techniques such as algebraic attacks, fast correlation and linear masking attacks, resynchronization attacks against IV dependent stream ciphers, and it turns out that many recent proposals still suffer from security weaknesses. The main design objective of provable secure stream ciphers is to contribute to reducing the discrepancy between practical stream ciphers and provably secure PRNG constructions depicted above by specifying a practical stream cipher with provable security arguments.

QUAD. QUAD [78] is a stream cipher designed with provable security arguments. QUAD relies on the iteration of a randomly chosen multivariate quadratic system $S = (Q_1, \ldots, Q_{kn})$ of kn equations in n unknowns over a finite field $\mathrm{GF}(q)$. The keystream generation consists of iterating the steps that computes the kn-tuple of $\mathrm{GF}(q)$ values $S(x) = (Q_1(x), \ldots, Q_{kn}(x))$ where x is the current internal state, outputs the sequence $(Q_{n+1}(x), \ldots, Q_{kn}(x))$ as $(k-1)n$ keystream blocks on $\mathrm{GF}(q)$, updates the internal state x with the sequence of $(Q_1(x), \ldots, Q_n(x))$. Before generating any keystream, the internal state x is initialized with the key and IV, using two additional multivariate quadratic functions described by two systems of n randomly chosen polynomials in n variables over $\mathrm{GF}(q)$. The designer recommends to use a version of QUAD with an 80-bit key, an 80-bit IV and an internal state of $n = 160$ bits. It outputs 160 keystream bits ($k = 2$) at each iteration until 2^{40} bits of keystream have been produced in order to provide 2^{80} security. The security of QUAD is provably reducible to the conjectured intractability of the MQ problem (solving a multivariate system of quadratic equations).

14 Stream ciphers cryptanalysis

Cryptanalysis plays an essential role in the design of ciphers. A good cipher should be designed by taking into account all the known cryptanalysis techniques. Thus we briefly describe some well-known attacks on stream ciphers here due to the space limitation. The general attacks include exhaustive search, time-memory trade-off, algebraic, correlation and linear, guess-and-determine, resynchronization (differential, slide), cube and fault attacks. Exhaustive search is also called brute force that searches through all

possible states/keys and checks for a match with keystream, which is always a baseline. Time-memory tradeoff attack can be seen as a tradeoff exhaustive search that has two phases: in the preprocessing phase, one explores the general structure of the cryptosystem, and summarize the findings in large tables not tied to particular keys; in the realtime phase, given actual data one tries to find the key using the precomputed tables. To resist such attack, it requires that the state size is at least twice of the key size [79]. Algebraic attack is used in LFSR-based stream cipher since any stream cipher can be expressed as a system of multivariate algebraic equations depending on the secret key and known keystream, and the system can be solved to recover the secret key, where the algebraic immunity was proposed for resistance of the crucial degree reducing [80, 81]. Correlation attack exploits statistical dependence between the keystream and the output of LFSR and fast correlation attack develops with parity check equations created from feedback polynomial of LFSR. Linear attack tries to take advantage of high probability occurrences of linear relations involving keystream bits and initial state bits. To resist such attacks, a Boolean functions used should be correlation immune, have large distance to affine functions and algebraic degree to counter Berlekamp-Massey synthesis [82, 83]. Differential cryptanalysis is a general method to analyze the initialization phase of stream ciphers, and the keystream allows computation of the output difference. Guess-and-determine attack is defined as guessing a set of state elements of the cryptosystem and determining the remaining state elements by the keystream, which is efficient for block-wise stream ciphers. Resynchronization attack targets the key/IV setup of stream ciphers. The reuse of key with many different IVs allows the key/IV setup being attacked similar to attacks on block ciphers as chosen-IV attacks. Differential cryptanalysis and linear cryptanalysis are two powerful attacks where differential cryptanalysis exploits the biased distribution of the difference of keystream pairs corresponding to a particular IV difference pattern; and linear cryptanalysis exploits the biased distribution of the parity of some IV and keystream bits. To resist the resynchronization attack, there should be sufficient confusion and diffusion in the key/IV setup [84]. Slide attack works as sliding one copy of the encryption process against another valid setup to find relations between these two keystreams to recover some information of the key, and adding the round constant is the most effective way for resistance [85]. Cube attack exploits the existence of low degree polynomial representation of a single output bit as a function of the key and IV bits and sums this bit over all possible values of a subset of the IV bits to derive linear equations and recover the secret key [86]. Fault attack is a powerful cryptanalytic tool that can apply some bit flipping faults to either the RAM or the internal register of the cryptographic device but have only a partial control over their number, location and timing [87]. In recent years, the analysis technology of stream ciphers has not achieved breakthrough development. The research of security analysis mainly focuses on two aspects: firstly, the improvement of traditional analysis methods by using new technologies or tools; secondly, the analysis of new design methods or new algorithms.

15 Conclusion and outlook on the future trend

In conclusion, it introduced many famous design methods of stream ciphers up to now, and presented some typical instances of these classic design methods in this paper. According to the different design methods, we analyzed their advantages and shortages in order to inspire the further research in stream cipher designs. For now though, both the stream cipher design and the analysis strategy are gradually diversified. In the hardware aspect, stream ciphers purely based on LFSR design have been out of historical stage over time, but it still can be used as a cipher component, and the stream ciphers based on NFSRs have become the mainstream design. In the software aspect, the groups of stream cipher designs based on the operations (ARX), components (S-box, diffusion layer), structures (Feistel, SPN), round functions (especially the AES round function), entire encryptions and work mode of the block ciphers have been rapidly grown, which call for correspondingly appropriate security analysis methods. Besides, stream cipher designs based on the fixed permutation of hash functions are also a hot research direction. On the whole, the stream cipher designs mainly focus on the cipher structure and the basic operations employed, which must assure the security that resists to all the existing attack category, and keep easy to

be evaluated further cryptographic properties. Moreover, it is in urgent need of lightweight designs, which puts emphasis on implement considering state storage, component consumption. In addition, leakage-resilient techniques also should be taken into account in the process of stream cipher design. Meanwhile, the design and analysis of authenticated encryption algorithms based on stream ciphers are still a hot spot. In the view of stream ciphers workflow, it can be concluded that there always take two stages in modern stream ciphers, which contains an initialization of key and IV loading, and a keystream generation that limits the keystream length with a pair of key and IV. For the initialization, it should consider the loading mode, number of blank rounds, and the key length which should be quantum-resistant after split-half. Compared with block ciphers, we believe that although the implementation efficiency of block ciphers is considerable in terms of hardware and software speed, the inherent advantages of stream ciphers will not be diminished. In contrast, stream ciphers will show a greater speed advantage under new programming instructions. New design ideas and new evaluated methods are needed by deepening the understanding of stream ciphers, such as new cipher structure and key/IV diffusion mode.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 61902030).

References

1 Rivest R. The RC4 encryption algorithm. Rsa Data Secur Inc Doc No, 1992, 20: 86–96
2 Anderson D P, Herrtwich R G. Internet communication with end-to-end performance guarantees. In: Telekommunikation und multimediale Anwendungen der Informatik. Berlin: Springer, 1991
3 ETSI/SAGE. Specification of the 3GPP confidentiality and integrity algorithms UEA2&UIA2. Document 2: SNOW 3G Specification, Version 1.1, 2006. http://www.gsmworld.com/using/algorithms/docs/etsi_sage_06_09_06.pdf
4 Feng X T. ZUC algorithm: 3GPP LTE international encryption standard. China Inform Secur, 2011, 19: 45–46
5 Bluetooth. Specification of the Bluetooth system. 2005. https://www.bluetooth.com/specifications/adopted-specifications
6 Ekdahl P, Johansson T. A new version of the stream cipher SNOW. In: Proceedings of International Workshop on Selected Areas in Cryptography, 2002. 47–61
7 Ekdahl P, Johansson T. SNOW-a new stream cipher. 2007. https://pdfs.semanticscholar.org/900e/081fa7ba0d0b45e36185e327e1081bf55d28.pdf
8 European Commission. First open NESSIE workshop. 2000. https://www.cosic.esat.kuleuven.be/nessie/workshop/
9 Hawkes P, Rose G G. Guess-and-determine attacks on SNOW. In: Proceedings of International Workshop on Selected Areas in Cryptography, 2002. 37–46
10 Markku-Juhani O S. A time-memory tradeoff attack against LILI-128. In: Proceedings of International Workshop on Fast Software Encryption, 2002
11 Tsunoo Y, Saito T, Shigeri M, et al. Shorter bit sequence is enough to break stream cipher LILI-128. IEEE Trans Inform Theory, 2005, 51: 4312–4319
12 Imai H, Yamagishi A. CRYPTREC project — cryptographic evaluation project for the japanese electronic government. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology, 2000. 399–400
13 Watanabe D, Furuya S, Yoshida H, et al. A new keystream generator MUGI. In: Proceedings of International Workshop on Fast Software Encryption, 2002. 179–194
14 van Tilborg H C A, Jajodia S. ECRYPT Stream Cipher Project. Berlin: Springer. 2011
15 Robshaw M. The eSTREAM project. In: New Stream Cipher Designs. Berlin: Springer, 2008
16 Hell M, Johansson T, Meier W. Grain: a stream cipher for constrained environments. Int J Wirel Mobile Comput, 2007, 2: 86–93
17 Canniere C D, Preneel B. TRIVIUM specifications. eSTREAM, ECRYPT Stream Cipher Project, 2006. http://www.ecrypt.eu.org/stream/
18 Babbage S, Dodd M. The stream cipher MICKEY 2.0. eSTREAM, ECRYPT Stream Cipher Project, 2006. http://www.ecrypt.eu.org/stream/
19 Robshaw M, Billet O. New Stream Cipher Designs. Berlin: Springer, 2008
20 Berbain C, Billet O, Canteaut A, et al. SOSEMANUK, a fast software-oriented stream cipher. In: New Stream Cipher Designs. Berlin: Springer, 2008. 98–118
21 Boesgaard M, Vesterager M, Pedersen T, et al. Rabbit: a new high-performance stream cipher. In: Proceedings of International Workshop on Fast Software Encryption, 2003. 307–329
22 Wu H J. The stream cipher HC-128. In: New Stream Cipher Designs. Berlin: Springer, 2008. 39–47
23 Cruz J R C. Keccak: the new SHA-3 encryption standard. 2014
24 Chakraborti A, Chattopadhyay A, Hassan M, et al. TriviA: a fast and secure authenticated encryption scheme. In: Proceedings of International Workshop on Cryptographic Hardware and Embedded Systems, 2015. 330–353
25 Wu H J. ACORN: a lightweight authenticated cipher (v3). CAESAR Submission, 2016. http://competitions.cr.yp.to/caesar-submissions.html

26 Shannon C E. A mathematical theory of communication. Bell Syst Technical J, 1948, 27: 379–423
27 Fontaine C. Synchronous stream cipher. In: Encyclopedia of Cryptography and Security. Beilin: Springer, 2005. 1274–1275
28 Millan W, Dawson E. On the security of self-synchronous ciphers. In: Proceedings of Australasian Conference on Information Security and Privacy, 1997. 159–170
29 Massey J. Shift-register synthesis and BCH decoding. IEEE Trans Inform Theory, 1969, 15: 122–127
30 Ere C D, Johansson T, Preneel B. Cryptanalysis of the Bluetooth Stream Cipher. Cosic Internal Report, 2001
31 Lu Y, Vaudenay S. Cryptanalysis of an E0-like combiner with memory. J Cryptol, 2008, 21: 430–457
32 Armknecht F, Mikhalev V. On lightweight stream ciphers with shorter internal states. In: Proceedings of International Workshop on Fast Software Encryption, 2015. 451–470
33 Ghafari V A, Hu H G, Chen Y. Fruit-v2: ultra-lightweight stream cipher with shorter internal state. 2016. https://eprint.iacr.org/2016/355
34 Ghafari V A, Hu H G. Fruit-80: a secure ultra-lightweight stream cipher for constrained environments. Entropy, 2018, 20: 180
35 Mikhalev V, Armknecht F, Müller C. On ciphers that continuously access the non-volatile key. IACR Trans Symmetric Cryptol, 2016, 2016: 52–79
36 Zhang B, Gong X X. Another tradeoff attack on Sprout-like stream ciphers. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, 2014. 561–585
37 Lallemand V, Naya-Plasencia M. Cryptanalysis of full Sprout. In: Proceedings of Annual Cryptology Conference, 2015. 663–682
38 Esgin M F, Kara O. Practical cryptanalysis of full Sprout with TMD tradeoff attacks. In: Proceedings of International Conference on Selected Areas in Cryptography, 2015. 67–85
39 Méaux P, Journault A, Standaert F X, et al. Towards stream ciphers for efficient FHE with low-noise ciphertexts. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2016. 311–343
40 Duval S, Lallemand V, Rotella Y. Cryptanalysis of the FLIP family of stream ciphers. In: Proceedings of Annual International Cryptology Conference, 2016. 457–475
41 Yu Y, Pereira O, Yung M. Practical leakage-resilient pseudorandom generators. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, 2010. 141–151
42 Faust S, Pietrzak K, Schipper J. Practical leakage-resilient symmetric cryptography. In: Proceedings of International Conference on Cryptographic Hardware and Embedded Systems, 2012. 213–232
43 Yu Y, Standaert F X. Practical leakage-resilient pseudorandom objects with minimum public randomness. In: Proceedings of Cryptographers' Track at the RSA Conference, 2013
44 Qu L J, Feng K Q, Liu F, et al. Constructing symmetric boolean functions with maximum algebraic immunity. IEEE Trans Inform Theory, 2009, 55: 2406–2412
45 Peng J, Wu Q S, Kan H B. On symmetric Boolean functions with high algebraic immunity on even number of variables. IEEE Trans Inform Theory, 2011, 57: 7205–7220
46 Wang H, Peng J, Li Y, et al. On $2k$-variable symmetric Boolean functions with maximum algebraic immunity k. IEEE Trans Inform Theory, 2012, 58: 5612–5624
47 Li N, Qi W F. Symmetric Boolean function with maximum algebraic immunity on odd number of variables. 2005. arXiv:cs/0511099
48 Rueppel R A. Analysis and Design of Stream Ciphers. Berlin: Springer, 1986
49 Simpson L R, Dawson E, Golic J D, et al. LILI keystream generator. In: Proceedings of International Workshop on Selected Areas in Cryptography, 2000. 248–261
50 Ekdahl P, Johansson T, Maximov A, et al. A new SNOW stream cipher called SNOW-V. 2018. https://eprint.iacr.org/2018/1143.pdf
51 Hell M, Johansson T, Maximov A, et al. A stream cipher proposal: Grain-128. In: Proceedings of IEEE International Symposium on Information Theory, 2006. 1614–1618
52 Ågren M, Hell M, Johansson T, et al. Grain-128a: a new version of Grain-128 with optional authentication. Int J Wirel Mobile Comput, 2011, 5: 48–59
53 Hamann M, Krause M, Meier W. LIZARD — a lightweight stream cipher for power-constrained devices. IACR Trans Symmetric Cryptol, 2017, 2017: 45–79
54 Hamann M, Krause M. On stream ciphers with provable beyond-the-birthday-bound security against time-memory-data tradeoff attacks. Cryptogr Commun, 2018, 10: 959–1012
55 Canteaut A, Carpov S, Fontaine C, et al. Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression. In: Proceedings of International Conference on Fast Software Encryption, 2016
56 Arnault F, Berger T P. F-FCSR: design of a new class of stream ciphers. In: Proceedings of International Workshop on Fast Software Encryption, 2005. 83–97
57 Hell M, Johansson T. Breaking the F-FCSR-H stream cipher in real time. In: Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, 2008. 557–569
58 Daemen J, Clapp C. Fast hashing and stream encryption with PANAMA. In: Proceedings of the 5th International Workshop on Fast Software Encryption, Paris, 1998. 60–74
59 Rivest R L, Schuldt J C N. Spritz — a spongy RC4-like stream cipher and hash function. 2016. https://eprint.iacr.org/2016/856

60 Banik S, Isobe T, Morii M. Analysis and improvements of the full spritz stream cipher. IEICE Trans Fund, 2017, 100: 1296–1305
61 Wu H J. A new stream cipher HC-256. In: Proceedigns of International Workshop on Fast Software Encryption, 2004. 226–244
62 Bernstein D J. ChaCha, a variant of Salsa20. 2009. http://cr.yp.to/chacha/chacha-20080120.pdf
63 Mileva A, Dimitrova V, Velichkov V. Analysis of the authenticated cipher MORUS (v1). In: Proceedings of International Conference on Cryptography and Information Security in the Balkans, 2016. 45–59
64 Dobraunig C, Eichlseder M, Mendel F, et al. Ascon — submission to the CAESAR competition. 2016
65 Wu H J, Preneel B. AEGIS: a fast authenticated encryption algorithm. In: Proceedings of International Conference on Selected Areas in Cryptography, 2013. 185–201
66 Ivica N. Tiaoxin-346, version 2.1. CAESAR Submission, 2016
67 Biryukov A. A New 128-bit Key Stream Cipher LEX. Estream Ecrypt Stream Cipher Project Report, 2008
68 Halevi S, Coppersmith D, Jutla C S. Scream: a software-efficient stream cipher. In: Proceedings of International Workshop on Fast Software Encryption, 2002. 195–209
69 Jean J, Nikolić I, Peyrin T, et al. Deoxys v1.41. 2016. http://competitions.cr.yp.to/round3/deoxysv141.pdf
70 Krovetz T, Rogaway P. OCB (v1.1). 2016. https://competitions.cr.yp.to/round3/ocbv11.pdf
71 Andreea E, Bogdanov A, Datta N, et al. COLM v1. 2016. http://competitions.cr.yp.to/caesar-submissions.html
72 Wu H J, Huang T. The JAMBU lightweight authentication encryption mode (v2.1). 2016. http://competitions.cr.yp.to/caesar-submissions.html
73 Albrecht M R, Rechberger C, Schneider T, et al. Ciphers for MPC and FHE. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2015
74 Shamir A. The generation of cryptographically strong pseudo-random sequences. In: Proceedings of IEEE Workshop on Communications Security, Santa Barbara, 1981
75 Blum M, Micali S. How to generate cryptographically strong sequences of pseudo-random bits. SIAM J Comput, 1984, 13: 850–864
76 Yao A C. Theory and applications of trapdoor functions (extended abstract). In: Proceedings of the 23rd Annual Symposium on Foundations of Computer Science, Chicago, 1982. 80–91
77 Goldreich O, Levin L A. A hard-core predicate for all one-way functions. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, Seattle, 1989. 25–32
78 Berbain C, Gilbert H, Patarin J. QUAD: a multivariate stream cipher with provable security. J Symb Comput, 2009, 44: 1703–1723
79 Biryukov A, Shamir A. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Proceedigns of International Conference on the Theory and Application of Cryptology and Information Security, 2000
80 Courtois N T, Meier W. Algebraic attacks on stream ciphers with linear feedback. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, 2003. 345–359
81 Courtois N T. Fast algebraic attacks on stream ciphers with linear feedback. In: Proceedings of Annual International Cryptology Conference, 2003. 176–194
82 Meier W, Staffelbach O. Fast correlation attacks on certain stream ciphers. J Cryptology, 1989, 1: 159–176
83 Berbain C, Gilbert H, Maximov A. Cryptanalysis of grain. In: Proceedings of International Workshop on Fast Software Encryption, 2006
84 Biham E, Dunkelman O. Differential Cryptanalysis in Stream Ciphers. Technical Report CS-2007-10, 2007
85 Biryukov A, Wagner D. Slide attacks. In: Proceedings of International Workshop on Fast Software Encryption, 1999. 245–259
86 Dinur I, Shamir A. Cube attacks on tweakable black box polynomials. In: Proceedigns of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2009. 278–299
87 Barenghi A, Breveglieri L, Koren I, et al. Fault injection attacks on cryptographic devices: theory, practice, and countermeasures. Proc IEEE, 2012, 100: 3056–3076