SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

September 2024, Vol. 67, Iss. 9, 192101:1–192101:14
https://doi.org/10.1007/s11432-023-3901-0

LOL: a highly flexible framework for designing stream ciphers

Dengguo FENG[1], Lin JIAO[1*], Yonglin HAO[1*], Qunxiong ZHENG[2], Wenling WU[3], Wenfeng QI[2], Lei ZHANG[3], Liting ZHANG[4], Siwei SUN[5] & Tian TIAN[2]

[1]*State Key Laboratory of Cryptology, Beijing* 100878*, China;*
[2]*PLA Strategic Support Force Information Engineering University, Zhengzhou* 450001*, China;*
[3]*Institute of Software, Chinese Academy of Sciences, Beijing* 100190*, China;*
[4]*Westone Cryptologic Research Center, Beijing* 100006*, China;*
[5]*School of Cryptology, University of Chinese Academy of Sciences, Beijing* 100049*, China*

Received 28 August 2023/Revised 27 October 2023/Accepted 17 November 2023/Published online 13 August 2024

Abstract In this paper, we propose LOL, a general framework for designing blockwise stream ciphers. The proposed framework achieves ultrafast software implementations for ubiquitous virtual networks in 5G/6G environments and high-security levels for post-quantum cryptography. The LOL framework is structurally strong; furthermore, this framework and all its components enjoy high flexibility with various extensions. On the basis of the LOL framework, we propose new stream-cipher designs called LOL-MINI and LOL-DOUBLE with the support of the AES-NI and single instruction multiple data instructions. The former applies the basic LOL single mode, while the latter uses the extended parallel-dual mode. LOL-MINI and LOL-DOUBLE support 256-bit key length. Our thorough evaluations revealed that these cipher designs have 256-bit security margins against all existing cryptanalysis methods, including differential, linear, and integral. The software performances of LOL-MINI and LOL-DOUBLE can reach 89 and 135 Gbps. In addition to pure encryptions, the LOL-MINI and LOL-DOUBLE stream ciphers can be applied in a stream-cipher-then-MAC strategy to make AEAD schemes.

Keywords stream cipher, 5G/6G mobile system, fast software implementation

1 Introduction

Stream ciphers play crucial roles in various communication systems, including protecting integrity and confidentiality. For example, the A5/1 stream cipher in GSM [1], the SNOW3G [2] and ZUC-128 [3] in UMTS and LTE (usually known as 4G). Currently, the 5G mobile communication system has been extensively used globally. Compared with 4G, the 5G system is faster (upstream and downstream rates are $\geqslant$20 Gbps and $\geqslant$10 Gbps, respectively [4]), resulting in extensive use of software-defined virtual networks, where all services/applications are implemented on general CPUs rather than on dedicated hardware devices. This trend has led to high software speed requirements for stream ciphers and other cryptographic primitives. Additionally, the development of 5G has stimulated the study of the beyond-5G or 6G system. In 2019, the 6Genesisi project published the first white paper on 6G [5]; their findings raised several requirements for producing the 6G system, including data transmission speeds of over 100 Gbps to 1 Tbps. In addition to high software speeds, this system also requires high-security margins because the latest quantum-based cryptanalysis techniques have threatened several symmetric-key primitives (just name some as [6–17]).

Furthermore, development in software implementation technologies is ongoing. In the past few decades, CPU manufacturers have widely accepted the idea of single instruction multiple data (SIMD). Many modern CPUs are now equipped with extended accumulator instruction sets (e.g., SSE/SSE2/SSE4.2/AVX/AVX2/AVX512 for Intel and NEON for ARM CPUs, etc.) so that several

* Corresponding author (email: jiaolin_jl@126.com, haoyonglin@yeah.net)

repeated basic operations (e.g., modular add/minus, XOR, OR, etc.) can be implemented simultaneously using only one accumulator instruction. With the extensive utilization of advanced encryption standard (AES) block cipher, the AES round function, i.e., the sequential call of SubByte (SB), ShiftRow (SR), MixColumn (MC), and AddRoundKey (ARK), has also been adopted as a basic operation and can be realized using one AES-NI instruction supported by most modern CPUs. Some SoCs for mobile devices are equipped with an AES instruction set [18], and more SoCs will support the instruction by the time the 6G system is realized. The latest AVX512 instruction of Intel supports the running of four AES round function calls in parallel. Therefore, the AES round function has become a handy building block for designing new cryptographic primitives. A typical example of such AES-based designs is the authentication encryption design called AEGIS [19]. It applies the AES round function in parallel for high performances and is mounted to the final portfolio of the CAESAR competition [20].

Faced with the high efficiency and security challenges, the 3GPP standardization organization has proposed the requirement of 5G stream cipher standards, including a speed of over 20 Gbps and secure margins of 256 bits [21]. The 4G standards, SNOW3G and ZUC-128, do not satisfy the 256-bit secure margin requirement; hence, designers are urged to submit 256-bit key versions. In response, the ZUC design team proposed ZUC-256 [22] in 2018, which is hardware-oriented and has a similar structure to its predecessor, ZUC-128, for compatibility reasons. Subsequently, the SNOW design team has proposed the software-oriented stream cipher SNOW-V [23] along with its performance-improved variant SNOW-Vi [24], which are referred to as SNOW-V/Vi for short hereafter. In 2022, Sakamoto et al. [25] proposed another primitive, the authenticated-encryption scheme (AEAD) Rocca, supporting 256-bit keys for encryptions and 128-bit tags for authentications, which is the first symmetric cipher dedicated to 6G systems. Similar to AEGIS, Rocca and SNOW-V/Vi have adopted parallel AES round functions called the basic building block, enabling them to exhibit extremely high software performances. The SNOW-V encryption speed can reach 58 Gbps, while that of Rocca is higher than 100 Gbps, catering for 5G and 6G requirements [5].

Parallel AES round function calls provide high software performances on modern CPUs but do not guarantee the security of AES-based primitives. Some cryptanalysis results show that the above AEGIS, Rocca, and SNOW-V/Vi have structural and theoretical security issues. The simple output function of AEGIS results in linear biases in keystream bits, making AEGIS vulnerable to linear distinguishers [19,26]. It also results in weak key-recovery attacks mounting to half of the total initialization rounds [27]. For full SNOW-V/Vi, there are fast correlation attacks (FCAs), recovering the internal states within 2^{256} secure bounds [28, 29]. For Rocca, there is a full key-recovery attack that uses encryption–decryption oracles for finding nonce-repeated pairs with the differential properties of AES S-box and the meet-in-the-middle techniques against AES for low-complexity state recoveries in a single-key and nonce-respecting setting [30, 31][1)].

Motivations. Many lessons can be learned from the development of SNOW-V/Vi, Rocca, and AEGIS. For SNOW-V/Vi, the direct LFSR-FSM (linear feedback shift register-finite state machine) connection can easily yield high linear correlations, enabling easy conduction of FCAs. This phenomenon has already been detected and used in FCAs on SNOW 2.0 and SNOW 3G [32–34]. For Rocca and AEGIS, we observe that the differential properties and effective cryptanalysis techniques on AES can be more effective on designs with parallel AES round function calls. Therefore, special attention must be given to the secure design of the overall structure, especially for adding non-AES components properly, such as nonlinear feedback shift registers (NFSRs), to resist these attacks. Meanwhile, Rocca has a faster speed than SNOW-V/Vi, indicating that the non-AES components should be better designed to improve the overall software efficiency of stream ciphers. Additionally, Rocca tries to design an AEAD scheme directly, while SNOW-V/Vi is a simple conventional stream cipher design suitable for authentications in the existing GCM-like scheme [35]. The attack on Rocca in [30] is based on repeated nonce pairs acquired from querying decryption oracles, which cannot be applied to the well-developed GMAC-based AEAD mode of SNOW-V/Vi. Thus, although directly designing a secure AEAD may be faster, the process is more challenging than the relatively mature stream-cipher-then-GMAC strategy. To achieve such a dramatically fast encryption/decryption speed in a pure software environment for mobile systems beyond 5G or 6G, the support from the AES round function with SIMD instructions has become a must for high software performances on modern CPUs before more basic instructions appear.

Contribution. In this paper, we propose a general framework for designing blockwise stream ciphers

1) As a result, the Rocca designers provided a modified version [31] in which key feedforward is added in initialization, which turns the key-recovery attack [30] into a state recovery attack.

Table 1 Comparison of competitive ciphers on security and software performance

Cipher	Type	Security claim	Software implementation	Attack
SNOW-V/Vi ([23, 24])	Stream cipher	256-bit security against key-recovery ✗	58.25 Gbps [23] 92 Gbps [24]	FCAs [28, 29] (data/time/memory complexity: $2^{234.88}/2^{246.40}/2^{238.51}$)
Rocca ([25])	AEAD	† 256-bit security against key-recovery ✗ † 128-bit security against distinguishing † 128-bit security against forgery attacks in the nonce-respecting setting † No claim in the related-key and known-key settings	150.95 Gbps [25]	Key-recovery attack [30] (complexity: data 2^{128}, time 2^{128})
AEGIS-256 ([19])	AEAD	† 256-bit security for encryption ✗ † 128-bit security for authentication	56.17 Gbps [19]	Linear distinguisher [27] (complexity: 2^{162})
LOL-MINI (Section 3)	Stream cipher	256-bit security against key-recovery	89 Gbps Section 6	–
LOL-DOUBLE (Section 4)	Stream cipher	256-bit security against key-recovery	135 Gbps Section 6	–

and construction methods for its basic components suitable for SIMD implementations. We called our framework (LOL), which can be extended from a basic single mode with various extension methods for many flexible designs. The structure of the proposed LOL framework structure is a combination of a nonlinear driver on finite fields and an FSM with memories, avoiding the direct LFSR-FSM-connection weakness in SNOW-V/Vi. Moreover, all the underlying components of the LOL framework enjoy high flexibility and are equipped with specific extension methods for catering to different needs. Especially, the LFSR in LOL perfectly suits the SIMD instructions, enabling high efficiencies, resourceful parameter selections, and maximum periods. The updating functions of the nonlinear driver and FSM in the LOL framework are constructed on an SP network, which makes LOL stream ciphers directly inherit some security properties from structural conclusions and more effectively and favorably be evaluated with definitive security bounds using automatic analysis methods compared with previous stream ciphers.

In addition to the LOL framework, we propose two concrete stream cipher designs: LOL-MINI of the LOL single mode and LOL-DOUBLE of extended parallel-dual mode. The two designs support the 256-bit key length and have software encryption speeds well over 20 Gbps, which are 89 and 135 Gbps for LOL-MINI and LOL-DOUBLE, respectively. From the security aspect, LOL-MINI and LOL-DOUBLE take the first attempt to load the key and IV bits to FSM rather than NLFSRs, enabling us to give provable secure bounds against differential attacks. Our thorough evaluations further confirm that the LOL-MINI and LOL-DOUBLE stream ciphers can provide 256-bit secure margins against various existing cryptanalysis methods, including FCAs, integral, and guess-and-determine. Therefore, LOL-MINI and LOL-DOUBLE provide 256-bit secure margins and high software performances, satisfying the 3GPP 5G requirements, even targeting 6G systems. In addition to pure encryptions, LOL stream ciphers can be applied in AEAD schemes using the stream-cipher-then-MAC strategy, where MAC tags are generated with standard authentication modes, such as GCM [35] and NMH [36].

Although our current designs are software-oriented, we have also considered hardware compatibility from the perspective of circuit reuse in the architecture and component extensions. This is because the reusability of hardware components is important to reducing the area and cost of ASICs. For clarity, we present a comparison of the above competitive ciphers in Table 1.

Outline. The remainder of the paper is organized as follows. In Section 2, we describe the LOL framework. We present details of the LOL-MINI and LOL-DOUBLE stream cipher designs in Sections 3 and 4, respectively. The security of LOL-MINI and LOL-DOUBLE is evaluated against various cryptanalysis methods in Section 5. The software performance is evaluated in Section 6. Finally, Section 7 presents the conclusion.

2 The LOL framework

The LOL framework is a combination of a nonlinear driver on finite fields and an FSM with memories shown in Figure 1. The nonlinear driver is in the form of an LFSR in series with an NFSR shown in the dotted box of Figure 1.

The following notations are used hereafter.

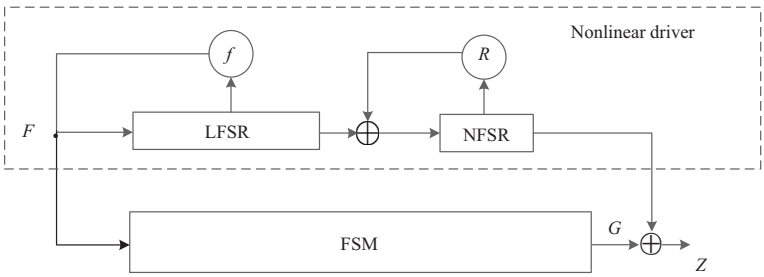

Figure 1 Form of cipher structure.

- Denote the state of LFSR by (H, L), the state of NFSR by N, the state of FSM by S, the keystream block by Z.
- The subscript corresponds to the state number and the superscript corresponds to the step.
- Denote the state updating function of LFSR and NFSR (FSM) by f and $\mathcal{R}$, respectively.
- Define two intermediate variables: the feedback of LFSR as F and the output of FSM as G.

2.1 Nonlinear driver

The LOL nonlinear driver takes the concatenated structure of an LFSR and an NFSR, which can benefit from both long periods and nonlinearities. We select LFSRs and NFSRs with the following features.

LFSR. The LOL LFSR is of Galois type defined over the finite field $\mathbb{F}_{2^m}$. It consists of an even number 2ℓ ($\ell > 0$) m-bit cells denoted as $a_{2\ell-1}, \ldots, a_0 \in \mathbb{F}_{2^m}$. At arbitrary time instance t, the 2ℓ cells are naturally stored in 2 registers according to their index parities as $L^t = \left(a^t_{2\ell-2}, \ldots, a^t_2, a^t_0\right)$ and $H^t = \left(a^t_{2\ell-1}, \ldots, a^t_3, a^t_1\right)$. Starting from (H^t, L^t), the LFSR states of the next time instance (H^{t+1}, L^{t+1}) is computed by taking the following 2 steps.

(1) Compute a feedback state F^t by calling the feedback function as follows:

$$F^t = f(H^t, L^t) = (C \times H^t) \oplus \sigma(L^t), \tag{1}$$

where C denotes a vector consisting of the roots of ℓ m-degree irreducible polynomials[2)] in $\mathbb{F}_2[x]$, $\times$ denotes a cell-to-cell finite field multiplication on $\mathbb{F}_{2^m}$, σ denotes a cell-wise permutation. Here, XOR integrates different expansion fields by the polynomial basis correspondence on $\mathbb{F}_{2^m}$.

(2) Then, L^t is output as the LFSR sequence and the internal state is updated as $(H^{t+1}, L^{t+1}) = (F^t, H^t)$ where F^t is computed as (1).

The whole LFSR state transformation from time instance t to $t+1$ can be represented as the Feistel-like structure in Figure 2. As can be seen, half of the state (H) is updated and another half (L) is output guaranteeing a high throughput and a specific adaptability for advanced parallel implementation (like SIMD). With proper selection of C and σ, the LFSR defined above can be an m-sequence with the maximum $2^{2\ell m} - 1$ period. The proof and generating method are shown in the full version[3)]. The advantages of such LOL LFSR can be summarized in two folds:

(1) `Software friendly`. The $C \times H^t$ can be implemented with parallel left-shift-then-XOR operations on modern CPUs supporting SIMD instructions[4)]. Moreover, $\times$ and σ operations can be executed synchronously.

(2) `Highly flexible`. Unlike traditional Fibonacci LFSRs where only 1 finite field is used, the Galois-style LOL LFSR allows ℓ finite fields defined by different irreducible polynomials, which has resulted in an enriched (C, σ) selections for maximizing the LFSR periods.

In summary, the LOL LFSR designing method balances the requirements of large-bit width, high parallelism, high speed, and high throughput for software implementation while taking into account the maximum period for security, and providing a resourceful solution.

NFSR. Concatenated with the L register of LFSR, the LOL NFSR consists of a register N sharing the same length with L. At time instance t, the whole register is updated from N^t to N^{t+1} as follows:

$$N^{t+1} = \mathcal{R}(N^t) \oplus L^t, \tag{2}$$

2) These irreducible polynomials can either be identical or different.

3) https://eprint.iacr.org/2023/1235.

4) This design strategy avoids the computations of both $a \times x$ and $a^{-1} \times x$ over the same field such as that in SNOW-V, which cannot be implemented together for different shift directions.

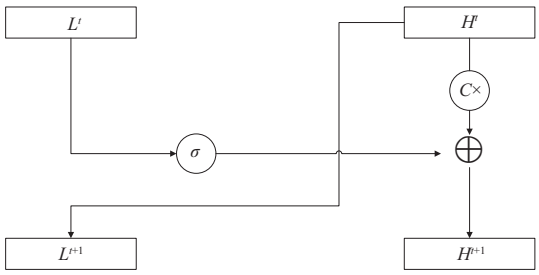

Figure 2 Feistel-like structure of LFSR updating function.

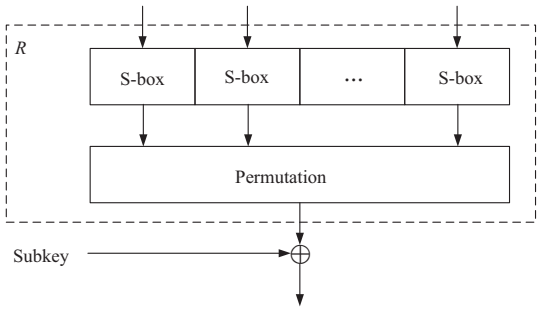

Figure 3 SP-network of $\mathcal{R}$ function.

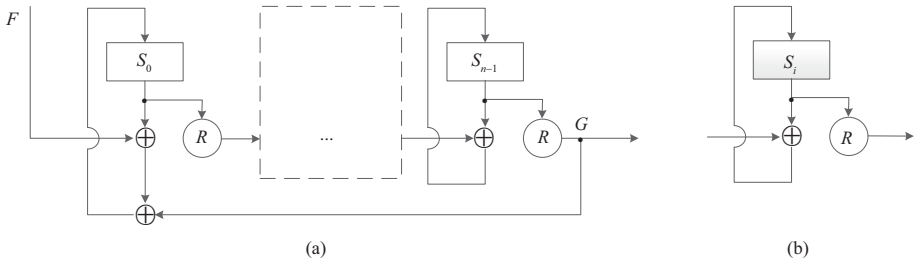

Figure 4 Connector and plugin of the FSM. (a) For the connector; (b) for the plugin.

where $\mathcal{R}$ is the SP-network function shown in Figure 3, and N^t is used as the output of NFSR. The flexibility of the LOL NFSR lies in the definition of $\mathcal{R}$ function and its length: both can be modified according to different efficiency/security requirements.

For the period of the whole nonlinear driver, the state equality $(H^t\|L^t, N^t) = (H^{t+T}\|L^{t+T}, N^{t+T})$ can only happen when $N^t = N^{t+T}$ with birthday collision as well as the LFSR reaches an integral period. Thus the average period of the concatenated LFSR and NFSR is about $2^{|N|/2} \times (2^{2\ell m} - 1)$, at least not less than the LFSR period. The large period of the NFSR and the randomness of the nonlinear round function work together to ensure the unpredictability of the keystream.

2.2 FSM

The LOL FSM uses parallel SP-network functions for fast diffusion and confusion inspired by [19]. A LOL FSM with n registers, denoted as $(S_0, S_1, \ldots, S_{n-1})$, is connected by two types of components: connectors and plugins. As can be seen in Figure 4, the connector is placed at both ends (S_0 and S_{n-1}) of the FSM so as to connect with the nonlinear driver or for further extensions; plugins are in the middle of FSM (i.e., the dotted box of Figure 4(a)) and can be added or removed according to the needs of security and efficiency. The $\mathcal{R}$ functions used in the FSM are also with SP-network shown in Figure 3 so the updating function of the FSM at time instance t can be described as follows:

$$S_i^{t+1} = \mathcal{R}(S_{i-1}^t) \oplus S_i^t, \tag{3}$$

where $1 \leqslant i \leqslant n-1$. Note that all $\mathcal{R}$ functions are not necessarily the same: different $\mathcal{R}$ definitions can be used simultaneously in one LOL framework depending on the application requirements equipping the designers with plenty of freedom.

The nonlinear driver and FSM are linked by the FSM connector as shown in Figure 1, where the feedback of LFSR is also the input of the connector, and the output of NFSR is combined with the output of the FSM to generate the keystream blocks. So, at time instance t, the updating function of S_0 and the keystream generation function can be represented as follows:

$$G^t = \mathcal{R}(S_{n-1}^t),\ S_0^{t+1} = S_0^t \oplus F^t \oplus G^t,\ Z^t = G^t \oplus N^t.$$

Such a combination of the nonlinear driver connected and the FSM makes up a single mode of the LOL framework.

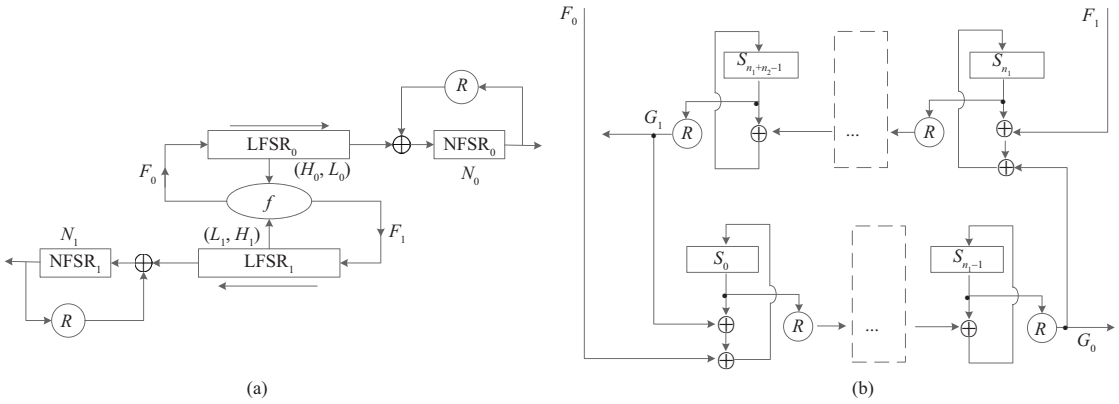

Figure 5 Form of extended (a) nonlinear driver and (b) extended FSM.

2.3 Expansion method

In Subsections 2.1 and 2.2, we have seen plenty of flexibilities in the underlying components of the LOL framework such as the LNFSR size, the $\mathcal{R}$ function definitions, the number of plugins in FSM, etc. Furthermore, the overall structure of the LOL framework is also expandable. For integer $\gamma \geqslant 2$, the LOL framework can expand from a single mode to a parallel-γ mode as follows:

(1) To construct an extended nonlinear driver, the LFSR can naturally expand by γ times while maintaining the Feistel-like overall structure with a maximum period through proper (C, σ) selection. The LFSR output sequence L is divided into γ parts and naturally fed to γ independent NFSRs.

(2) As to FSMs, we can link the connectors of γ single modes end to end in a circle, each feeding into the other. The feedback of the extended LFSR is also divided into γ parts and serves as the inputs of the connectors. The updates of the plugins in the FSM are unchanged. The output of each FSM in the single mode remains and is still XORed with the output of the corresponding NFSR, which are concatenated together to generate the keystream block.

Parallel-dual mode: expansion for $\gamma = 2$. We set $\gamma = 2$ and demonstrate the "parallel-dual" mode of the LOL framework. The expanded nonlinear driver is given in Figure 5(a), composed of 2 single mode drivers (H_0, L_0, N_0) and (H_1, L_1, N_1). Let the expanded LFSR be $L = (L_1, L_0),\ H = (H_1, H_0)$. The updating function of the LFSR remains the same form (1). The designers only need to properly select larger (C, σ)'s so as to maximize the LFSR period. The output of LFSR (H, L) is divided into 2 and fed to the NFSRs N_0, N_1 respectively as

$$N_0^{t+1} = \mathcal{R}(N_0^t) \oplus L_0^t,\ N_1^{t+1} = \mathcal{R}(N_1^t) \oplus L_1^t. \tag{4}$$

The expanded FSM for $\gamma = 2$ connects 2 single mode FSMs (consisting of n_1 and n_2 registers respectively) into a circle as in Figure 5(b). The connected updating function is then defined as

$$\begin{aligned}
&G_0 = \mathcal{R}(S_{n_1-1}^t),\ G_1 = \mathcal{R}(S_{n_1+n_2-1}^t),\\
&F^t = (F_1^t, F_0^t),\ S_0^{t+1} = F_0^t \oplus G_1^t \oplus S_0^t,\ S_{n_1}^{t+1} = F_1^t \oplus G_0^t \oplus S_{n_1}^t,\\
&S_i^{t+1} = \mathcal{R}(S_{i-1}^t) \oplus S_i^t,\ i = 1, \ldots, n_1 - 1, n_1 + 1, \ldots, n_1 + n_2 - 1,
\end{aligned}$$

where F^t is computed as (1). With both the nonlinear driver and FSM expanded by 2, the output keystream block Z^t is also expanded as

$$Z^t = (Z_0^t, Z_1^t) = (G_0^t \oplus N_0^t, G_1^t \oplus N_1^t).$$

Besides the parallel-dual mode above, there are also other diverse extended modes enriching the LOL framework catering to various efficiency and security demands.

2.4 Design rationale

Notably, the LOL framework adapts from the classical source-filter and generator-with-memory stream cipher models. It integrates the underlying nonlinear function with the classical SP network of block

ciphers and connects the blockwise components with XOR operations. When the S-boxes are the only nonlinear operations in the SP-network functions of Figure 3, the security evaluation of LOL stream ciphers becomes quite easy because off-the-shelf proofs and automatic cryptanalytic tools can be used easily. Such an easy-to-analyze characteristic makes the selection of LOL stream cipher parameters (e.g., initialization round number, key/IV placement, etc.) better grounded than that of SNOW (S-boxes and modular are added) and Grain (bit-oriented) parameters.

As to the component selection, the LOL framework utilizes the advantages of existing designs, such as SNOW, Grain, and AEGIS, while avoiding their disadvantages: the nonlinear driver avoiding the highly linear correlation of SNOW, large-memory FSM avoiding the dedicated FCA on Grain, and LNFSR-FSM combinations avoiding the linear distinguishing attacks on AEGIS. The blockwise structure and non-AES component design strategy also make LOL ciphers highly flexible and parallelable, resulting in software efficiency on modern CPUs.

Therefore, the LOL framework provides highly flexible components, software-efficient structures, and various expansions. It also enjoys hospitality to cryptanalysis for drawing secure bounds. This indicates that the LOL framework is friendly to ordinary users and stream cipher designers.

2.5 Instantiations

In the remainder of this paper, we present 2 stream cipher instances following the LOL framework. The 2 stream ciphers are named LOL-MINI and LOL-DOUBLE respectively: the former takes the LOL single mode while the latter follows the parallel-dual mode. We first present some universal preliminaries here.

- For the suitability of SIMD implementation, the ciphers use 128-bit registers as storage units in the description.
- According to our experiments, 16-bit cells usually make LFSRs with higher efficiencies in comparison with the 32/64-bit counterparts. So we select $m = 16$.
- State correspondence: a 128-bit register state S can be divided into eight 16-bit words $w_7, \ldots, w_0$ where w_0 is the least significant word, or into sixteen 8-bit bytes $b_{15}, \ldots, b_0$ where b_0 is the least significant byte, i.e., $S = (w_7, \ldots, w_0) = (b_{15}, \ldots, b_0)$.
- Specifically for extremely high software speeds, the SP-network can be set as a standard AES round function that can be accomplished with a single SIMD instruction on many modern CPUs like Intel and AMD (AES-NI), and so far as to some SoCs for mobile devices. Thus $\mathcal{R}$ functions are all specified as a compound of AES SB, SR, and MC operations (equivalent to the AES round function without AddRoundKey (ARK)), i.e., $\mathcal{R}(S) = \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{SB}(S)$.
- The mapping between a 128-bit register state $S = (w_{15}, \ldots, w_0)$ and the 4×4-byte state array of the AES round function is as follows:

$$S = \begin{pmatrix} w_0 & w_4 & w_8 & w_{12} \\ w_1 & w_5 & w_9 & w_{13} \\ w_2 & w_6 & w_{10} & w_{14} \\ w_3 & w_7 & w_{11} & w_{15} \end{pmatrix}. \tag{5}$$

3 LOL-MINI cipher

LOL-MINI cipher adopts the single mode of the LOL framework, which supports a 256-bit key and a 128-bit initialization vector (IV). We denote the 256-bit master key as $K = (K_h, K_\ell)$ and the 128-bit IV as IV. The internal state of LOL-MINI cipher consists of six 128-bit registers: 2 for the LFSR denoted as (H, L), 1 for the NFSR N and 3 for the FSM (S_0, S_1, S_2) (corresponding a connector with 1 plugin or equivalently $n = 3$ for Figure 4). LOL-MINI cipher outputs a 128-bit keystream block at each step.

3.1 Nonlinear driver in LOL-MINI

The LFSR of LOL-MINI cipher consists of 16 16-bit cells denoted as $(a_{15}, \ldots, a_1, a_0)$. The 16 cells are stored in the two 128-bit registers (H, L) according to the parities of their subscripts as follows:

$$H = (a_{15}, \ldots, a_3, a_1),\ L = (a_{14}, \ldots, a_2, a_0).$$

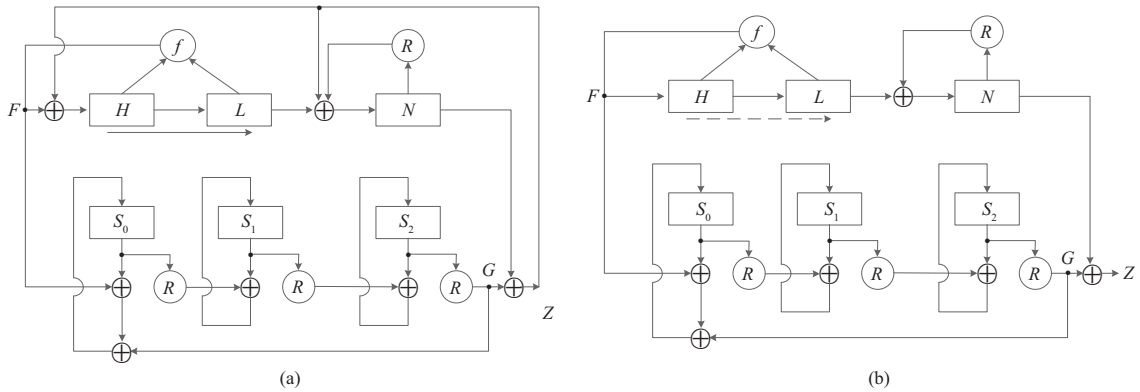

Figure 6 Schematic of LOL-MINI cipher in (a) initialization phase and (b) keystream generation phase.

Let $\alpha_0, \alpha_1, \ldots, \alpha_7$ be the roots of the irreducible polynomials $g_0(y), g_1(y), \ldots, g_7(y) \in \mathbb{F}_2[y]$ with algebraic degree 16, respectively,

$$\begin{aligned}
g_0(y) &= y^{16} + y^{13} + y^{12} + y^{10} + y^8 + y^7 + y^6 + y^3 + 1,\\
g_1(y) &= y^{16} + y^{15} + y^{12} + y^{10} + y^8 + y^5 + y^3 + y + 1,\\
g_2(y) &= y^{16} + y^{15} + y^{14} + y^{12} + y^{10} + y^7 + y^5 + y^4 + 1,\\
g_3(y) &= y^{16} + y^{14} + y^{11} + y^9 + y^7 + y^5 + y^4 + y^2 + 1,\\
g_4(y) &= y^{16} + y^{15} + y^{13} + y^9 + y^7 + y^4 + 1,\\
g_5(y) &= y^{16} + y^{14} + y^{13} + y^{12} + y^{11} + y^{10} + y^9 + y^7 + y^6 + y^5 + y^3 + y^2 + 1,\\
g_6(y) &= y^{16} + y^{15} + y^{13} + y^9 + y^8 + y^4 + y^3 + y + 1,\\
g_7(y) &= y^{16} + y^{14} + y^{13} + y^{12} + y^{11} + y^{10} + y^7 + y^5 + 1.
\end{aligned} \tag{6}$$

Let $C = (\alpha_7, \ldots, \alpha_1, \alpha_0)$ and define the operation $C\times : \mathbb{F}_{2^{16}}^8 \to \mathbb{F}_{2^{16}}^8$ as

$$(x_7, \ldots, x_1, x_0) \xrightarrow{C\times} (\alpha_7 \cdot x_7, \ldots, \alpha_1 \cdot x_1, \alpha_0 \cdot x_0), \tag{7}$$

where $\alpha_i \cdot x_i$ denotes the multiplication over the finite field $\mathbb{F}_{2^{16}} = \mathbb{F}_2[y]/g_i(y)$ defined by the polynomial $g_i(y)$ in (6) $(0 \leqslant i \leqslant 7)$. Let the permutation $\sigma : \mathbb{F}_{2^{16}}^8 \to \mathbb{F}_{2^{16}}^8$ be

$$(x_7, \ldots, x_1, x_0) \xrightarrow{\sigma} (x_5, x_0, x_3, x_6, x_4, x_7, x_2, x_1)$$

with (C, σ) defined above, the feedback function f can be defined as (1), which is primitive, making the structure of LOL-MINI LFSR updating function identical to Figure 2. The NFSR updating function is identical to (2).

3.2 Working process of LOL-MINI

The whole LOL-MINI can be described as Algorithm 1 and summarized as follows.

Firstly, the secret key and public IV are loaded to the FSM registers while all nonlinear driver bits are set to 0. Then, LOL-MINI runs a 12-round initialization phase during which the 128-bit keystream block Z^t's $(t = -12, \ldots, -1)$ are not output but fed back to the nonlinear driver for thorough diffusion. It is noticeable that LOL-MINI adopts the FP-(1) mode [37] by XORing the master key to the internal state at $t = -1$ so as to make the whole initialization irreversible. Finally, LOL-MINI generates keystream blocks $Z^0, \ldots, Z^{N-1} \in \mathbb{F}_2^{128}$. Same with the existing stream cipher designs, LOL-MINI also limits the keystream length to $N \leqslant 2^{64}$ for a single key-IV pair, and each key may be used with a maximum of 2^{64} different IVs, although breaking such limitations does not cause any security issues according to our thorough cryptanalysis.

The initialization and keystream generation phases of LOL-MINI are shown as Figures 6(a) and (b), respectively.

Algorithm 1 Working process of LOL-MINI

Input: A key-IV pair $(K, \mathrm{IV}) \in \mathbb{F}_2^{256} \times \mathbb{F}_2^{128}$.
Output: 128-bit keystream blocks $Z^0, \ldots, Z^{N-1}$.

1: Set $t = -12$;
2: Divide $K = (K_h, K_\ell) \in \mathbb{F}_2^{128} \times \mathbb{F}_2^{128}$ and initialize the LOL-MINI registers as $S_0^t = \mathrm{IV}$, $S_1^t = K_h$, $S_2^t = K_\ell$, $H^t = L^t = N^t = 0$;
3: **for** $t = -12, \ldots, -1, 0, 1, \ldots, N-1$ **do**
4: Compute the 128-bit intermediate variables $G^t = \mathcal{R}(S_2^t)$, $F^t = f(H^t, L^t)$;
5: Compute the 128-bit keystream block $Z^t = G^t \oplus N^t$;
6: Update the nonlinear driver:

$$L^{t+1} = H^t,\ H^{t+1} = \begin{cases} F^t \oplus Z^t, & -12 \leqslant t < -1, \\ F^t \oplus Z^t \oplus K_h, & t = -1, \\ F^t, & t \geqslant 0, \end{cases}$$
$$N^{t+1} = \begin{cases} \mathcal{R}(N^t) \oplus L^t \oplus Z^t, & -12 \leqslant t \leqslant -1, \\ \mathcal{R}(N^t) \oplus L^t, & t \geqslant 0; \end{cases}$$

7: Update the FSM:

$$S_1^{t+1} = \mathcal{R}(S_0^t) \oplus S_1^t,\ S_2^{t+1} = \mathcal{R}(S_1^t) \oplus S_2^t,$$
$$S_0^{t+1} = \begin{cases} F^t \oplus G^t \oplus S_0^t \oplus K_\ell, & t = -1, \\ F^t \oplus G^t \oplus S_0^t, & \text{otherwise}; \end{cases}$$

8: **end for**
9: **return** $Z^0, \ldots, \ldots, Z^{N-1}$.

4 LOL-DOUBLE cipher

LOL-DOUBLE cipher supports a 256-bit key and a 256-bit IV, denoted as $K = (K_h, K_\ell)$ and $\mathrm{IV} = (\mathrm{IV}_h, \mathrm{IV}_\ell)$, respectively. The LFSR 512-bit (H, L) is composed of 4 128-bit registers, denoted as $H = (H_1, H_0)$, $L = (L_1, L_0)$. There are two 128-bit NFSR registers of N_0 and N_1. The two FSMs, each having two 128-bit registers of (S_0, S_1) or (S_2, S_3), form the circular in Figure 5(b) (equivalent to the $n_1 = n_2 = 2$ setting with no plugins). As can be seen, LOL-DOUBLE follows exactly the parallel-dual mode of the LOL framework: the combinations (H_0, L_0, N_0, S_0, S_1), and (H_1, L_1, N_1, S_2, S_3) form 2 standard LOL single modes accordingly; the 256-bit output keystream block of LOL-DOUBLE is the concatenation of the two 128-bit output keystream blocks from the 2 LOL single modes as $Z = (Z_0, Z_1)$.

4.1 Nonlinear driver in LOL-DOUBLE

The LFSR of LOL-DOUBLE cipher consists of 32 16-bit cells denoted as $(a_{31}, \ldots, a_0)$ and are stored in H, L as follows:

$$\begin{aligned} H &= (H_1, H_0) = ((a_{31}, \ldots, a_{17}), (a_{15}, \ldots, a_3, a_1)), \\ L &= (L_1, L_0) = ((a_{30}, \ldots, a_{16}), (a_{14}, \ldots, a_2, a_0)). \end{aligned} \tag{8}$$

Let $\alpha_0, \alpha_1, \ldots, \alpha_{15}$ be the roots of the 16-degree irreducible polynomials $g_0(y), g_1(y), \ldots, g_{15}(y) \in \mathbb{F}_2[x]$, where $g_0(y), g_1(y), \ldots, g_7(y)$ are the same with those in (6) and $g_8(y), g_9(y), \ldots, g_{15}(y)$ are defined as (9).

$$\begin{aligned} g_8(y) &= y^{16} + y^{15} + y^{14} + y^{10} + y^8 + y^6 + y^4 + y + 1, \\ g_9(y) &= y^{16} + y^{14} + y^{13} + y^{12} + y^{11} + y^{10} + y^8 + y^7 + y^6 + y^2 + 1, \\ g_{10}(y) &= y^{16} + y^{11} + y^{10} + y^8 + y^7 + y + 1, \\ g_{11}(y) &= y^{16} + y^{15} + y^{13} + y^{12} + y^9 + y^7 + y^6 + y^5 + y^3 + y + 1, \\ g_{12}(y) &= y^{16} + y^{15} + y^{14} + y^{12} + y^{10} + y^8 + y^5 + y^3 + y^2 + y + 1, \\ g_{13}(y) &= y^{16} + y^{15} + y^{12} + y^{11} + y^{10} + y^9 + y^8 + y^7 + y^5 + y^4 + y^2 + y + 1, \\ g_{14}(y) &= y^{16} + y^{14} + y^{10} + y^7 + y^6 + y^5 + 1, \\ g_{15}(y) &= y^{16} + y^{15} + y^{14} + y^{13} + y^{12} + y^6 + y^5 + y^3 + 1. \end{aligned} \tag{9}$$

In this way, we can define $C = (\alpha_{15}, \ldots, \alpha_1, \alpha_0)$ and the corresponding $C\times : \mathbb{F}_{2^{16}}^{16} \to \mathbb{F}_{2^{16}}^{16}$ operation as

$$(x_{15}, \ldots, x_1, x_0) \xrightarrow{C\times} (\alpha_{15} \cdot x_{15}, \ldots, \alpha_1 \cdot x_1, \alpha_0 \cdot x_0), \tag{10}$$

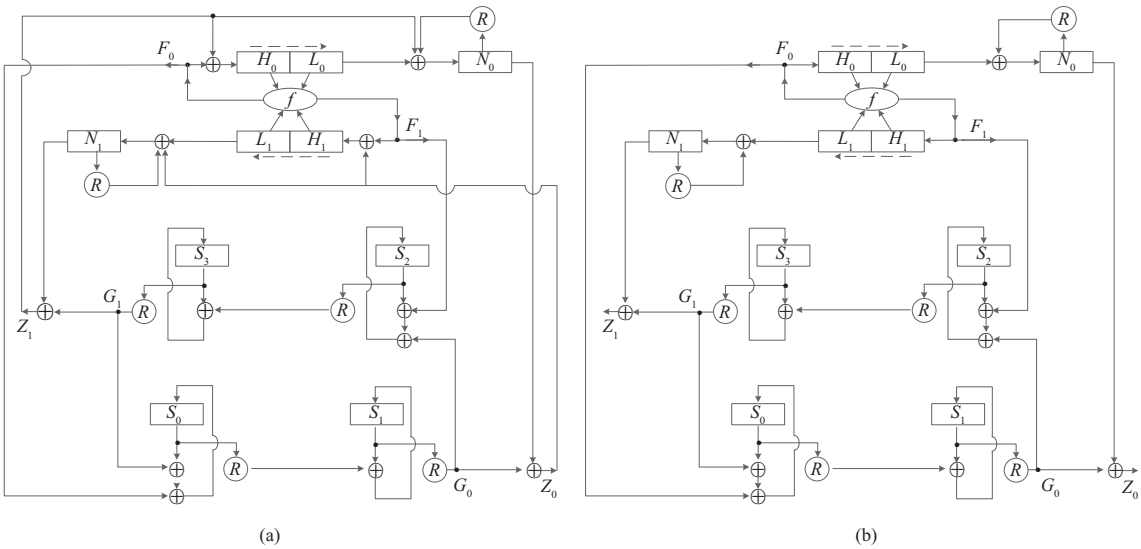

Figure 7 Schematic of LOL-DOUBLE cipher in (a) initialization phase and (b) keystream generation phase.

where $\alpha_i \cdot x_i$ denotes the multiplication over the finite field $\mathbb{F}_{2^{16}} = \mathbb{F}_2[y]/g_i(y)$ $(0 \leqslant i \leqslant 15)$. The shared definition of $\alpha_0, \ldots, \alpha_7$ enables the circuit reuse in the hardware implementations of LOL-MINI and LOL-DOUBLE in some highly compatible application scenarios. The permutation $\sigma : \mathbb{F}_{2^{16}}^{16} \to \mathbb{F}_{2^{16}}^{16}$ of LOL-DOUBLE is defined as

$$(x_{15}, \ldots, x_0) \xrightarrow{\sigma} (x_{11}, x_6, x_{15}, x_{14}, x_2, x_8, x_0, x_9, x_4, x_7, x_{10}, x_{13}, x_1, x_5, x_{12}, x_3).$$

With (C, σ) above, the $f : \mathbb{F}_2^{256} \times \mathbb{F}_2^{256} \to \mathbb{F}_2^{256}$ is of the same form as (1). Such an LFSR has a period of $2^{512} - 1$. The updating function of the NFSRs is the same with (4).

4.2 Working process of LOL-DOUBLE

The whole LOL-DOUBLE can be described as Algorithm 2. As can be seen, LOL-DOUBLE shares many similarities with LOL-MINI such as loading key-IV to FSM, 12 initialization rounds, and FP-(1) mode. The output keystream blocks are also $Z^0, \ldots Z^{N-1}$ but doubling the size to 256 bits. It is also noticeable that the 256-bit keystream block Z is the concatenations of 2 independently computed 128-bit blocks Z_0 and Z_1 where $Z_0 = \mathcal{R}(S_1) \oplus N_0$ is placed at the higher half of Z and $Z_1 = \mathcal{R}(S_3^t) \oplus N_1$ at the lower half: such a rearrangement, according to our experiments, provides a faster diffusion. As a tradition, LOL-DOUBLE limits $N \leqslant 2^{64}$ for a single key-IV pair and each key can only be used with no more than 2^{64} different IVs, although breaking such limitations does not cause any security issues according to our thorough cryptanalysis. The schematic of LOL-DOUBLE in the initialization phase is shown in Figure 7(a) and that of the keystream generation phase is in Figure 7(b).

5 Security analysis

Security has always been the main concern of newly designed stream ciphers. We evaluate the resistance of LOL-MINI and LOL-DOUBLE against various cryptanalysis methods to prove their 256-bit security margins. In this section, we briefly describe the cryptanalysis results and present details in the full version[3)].

Maximum period proof of the LFSR. We first deduce the LFSR generation matrix G and its characteristic polynomial $\eta(x) = \det(xI - G)$. Then, proving the maximum period is equivalent to proving that $\eta(x)$ is a primitive. The characteristic polynomials of LOL-MINI and LOL-DOUBLE are primitives of degree 128 and 256, respectively; therefore, the corresponding LFSRs have the maximum periods of $2^{256} - 1$ and $2^{512} - 1$, respectively.

Full diffusion round analysis. For LOL-MINI, five initialization rounds are necessary to make every input-output bit correlated; therefore, the full diffusion round required is five. For LOL-DOUBLE, the required full diffusion round is six.

Algorithm 2 Working process of LOL-DOUBLE

Input: A key-IV pair $(K, \mathrm{IV}) \in \mathbb{F}_2^{256} \times \mathbb{F}_2^{256}$.
Output: 256-bit keystream blocks $Z^0, \ldots, Z^{N-1}$.

1: Set $t = -12$;
2: Divide $K = (K_h, K_\ell), \mathrm{IV} = (\mathrm{IV}_h, \mathrm{IV}_\ell) \in \mathbb{F}_2^{128} \times \mathbb{F}_2^{128}$ and initialize the LOL-DOUBLE registers as $S_3^t = K_h$, $S_2^t = K_\ell$, $S_1^t = \mathrm{IV}_h$, $S_0^t = \mathrm{IV}_\ell$, $H_0^t = L_0^t = H_1^t = L_1^t = N_0^t = N_1^t = 0$;
3: **for** $t = -12, \ldots, -1, 0, 1, \ldots, N-1$ **do**
4: Compute the 128-bit intermediate variables $G_0^t = \mathcal{R}(S_1^t)$, $G_1^t = \mathcal{R}(S_3^t)$;
5: Compute 256-bit value $F^t = f(H^t, L^t) = f((H_1^t, H_0^t), (L_1^t, L_0^t))$ and divide it as $F^t = (F_1^t, F_0^t) \in \mathbb{F}_2^{128} \times \mathbb{F}_2^{128}$;
6: Compute the 256-bit keystream block $Z^t = (Z_0^t, Z_1^t)$ where $Z_0^t = G_0^t \oplus N_0^t$, $Z_1^t = G_1^t \oplus N_1^t$;
7: Update the nonlinear driver:

$$N_j^{t+1} = \begin{cases} \mathcal{R}(N_j^t) \oplus L_j^t \oplus Z_{j-1}^t, & -12 \leqslant t \leqslant -1, \\ \mathcal{R}(N_j^t) \oplus L_j^t, & t \geqslant 0, \end{cases}$$

$$L^{t+1} = H^t,$$

$$H^{t+1} = \begin{cases} F^t \oplus Z^t, & -12 \leqslant t < -1, \\ F^t \oplus Z^t \oplus K, & t = -1, \\ F^t, & t \geqslant 0; \end{cases}$$

8: Update the FSM:

$$S_0^{t+1} = F_0^t \oplus G_1^t \oplus S_0^t,\ S_1^{t+1} = \mathcal{R}(S_0^t) \oplus S_1^t,$$
$$S_2^{t+1} = F_1^t \oplus G_0^t \oplus S_2^t,\ S_3^{t+1} = \mathcal{R}(S_2^t) \oplus S_3^t;$$

9: **end for**
10: **return** $Z^0, \ldots, \ldots, Z^{N-1}$.

Differential attack on initialization. We consider differential attacks under the related-key chosen-IV model, where differences can be injected in key and IV bits. The resistance to differential attacks can be bounded with the minimum number of active S-boxes in the SP-network functions. With truncated differential and MILP-based automatic search techniques, we can derive the minimum number of active S-boxes for r-round LOL-MINI/LOL-DOUBLE, where r is an arbitrary positive integer. We observe that LOL-MINI has no differential characteristic with probability larger than 2^{-256} for $r \geqslant 6$ so $r = 6$ is the secure bound for LOL-MINI against differential attacks. The secure bound for LOL-DOUBLE is $r = 4$.

Integral attack. LOL-MINI and LOL-DOUBLE use the AES round function as the basic SP-network component $\mathcal{R}$. We that, for LOL-MINI, IV needs to go through 6 AES rounds for Z^{-7}. Similarly, for LOL-DOUBLE, IV_h and IV_ℓ need to go through at least 5 AES rounds for the keystream block $Z^{-7} = (Z_0^{-7}, Z_1^{-7})$. According to [38], there is no integral distinguisher on 5 or more AES rounds [38]. Our evaluations demonstrate that it is impossible to prevent IV from going through $\leqslant 5$ AES rounds for Z^{-7}, making it impossible to conduct integral attacks. Therefore, 12 initialization rounds are sufficient for LOL-MINI and LOL-DOUBLE to resist integral attacks.

Slide attack. The idea of slide attacks is to form the same state at a shift of steps for different key/IV pairs, which can be detected using similar keystreams for recovering some key information. LOL ciphers can be secure against slide attacks by adding a feeding-key-forward operation in initialization, or using different update functions for initialization and key generation. For LOL-MINI and LOL-DOUBLE, such sliding properties would be difficult to find due to the update of large blocks at each step and the two special arrangements in the initialization: the feedback of the keystream blocks and the FP(1)-mode.

Correlation attack. FCA utilizes high linear correlations between keystream and LFSR bits for internal state recoveries. Similar to differential attacks, the resistance against correlation attacks can also be bounded with the minimum number of active S-boxes. We describe the truncated linear propagation rules of the AES round function with MILP models and lower bound the number of active S-boxes. Our analysis reveals that 4 (3) LOL-MINI (LOL-DOUBLE) keystream blocks are necessary for a correlation attack, and the corresponding linear masks contain at least 80 (94) active S-boxes. Considering the linear properties of AES S-boxes, no linear distinguisher or FCA can be conducted within a complexity of 2^{256}. Therefore, no effective linear distinguishers or correlation attacks are applicable to LOL-MINI and LOL-DOUBLE.

Guess-and-determine attack. Guess-and-determine attacks are common tools for achieving state recovery in the keystream generation phase. We first conduct the guess-and-determine attack following the blockwise strategy. At each step, a keystream block of LOL-MINI (LOL-DOUBLE) involves 2 (4) state units. Since the internal state consists of 6 (10) units, the adversary needs to consider at least 3 (3) consecutive steps to involve the information about the entire internal state. For LOL-MINI, one

should guess N^t, N^{t+1}, N^{t+2} according to Z^t, Z^{t+1}, Z^{t+2} with a time complexity of 2^{384}, and another three keystream blocks are needed for filtering. Furthermore, we consider the bytewise strategy only to find that for LOL-MINI and LOL-DOUBLE, two consecutive keystream blocks are involved in at least 2 AES round functions, and each requires guessing a 128-bit block. Therefore, the guess-and-determine attacks on LOL-MINI and LOL-DOUBLE are well above the 2^{256} bound.

Time/memory/data tradeoff attacks. The time memory data tradeoff attacks have two phases: preprocessing and real-time phases. In the preprocessing phase, the mapping table from different secret keys or internal states to keystreams is computed and stored with time complexity P and memory M. In the real-time phase, attackers have intercepted D keystreams and searched them in the table with time complexity T, expecting to get some matches and further recover the corresponding input. To balance the parameters, the most popular tradeoffs are Babbage-Golic [39,40] with $TM = N, P = M, T \leqslant D$ and Biryukov-Shamir [41] with $MT^2D^2 = N^2, P = N/D, T \leqslant D^2$, where N is the input space. To reconstruct the internal state, LOL-MINI and LOL-DOUBLE are naturally immune because their internal state sizes are well over twice the secret key size. Additionally, we use FP(1)-mode in the initialization phase. The attacks cannot benefit from reconstructing the internal state to recover the key. To recover the secret key directly, there is a trivial (but unrealistic way of achieving it in practice) attack on LOL-MINI since the IV size of 128 bits is smaller than the key size of 256 bits. In contrast, it is resisted by LOL-DOUBLE since the IV size of 256 bits is equal to the key size of 256 bits.

Algebraic attacks. In an algebraic attack, the adversary derives several nonlinear equations in either unknown key or unknown state bits and solves the system of equations. It is well-known that each AES S-box can be defined by a system of 23 linearly independent bi-affine quadratic equations between the input and output bits in 80 terms, which contains 64 quadratic and 16 linear terms, and there are no more such equations [42]. Based on such statistics, for LOL-MINI and LOL-DOUBLE, we construct a system of equations and compute the total number of terms. We find that the number of terms and equations expands significantly with rounds. Moreover, whether to recover internal states or secret keys, the complexity of the equation-solving process is higher than 2^{256}. Therefore, LOL-MINI and LOL-DOUBLE are immune to algebraic attacks.

6 Software implementation and performance

In this section, we evaluate the software performance of LOL-MINI and LOL-DOUBLE implemented in C++ (Visual Studio 2022) utilizing AVX2/AVX512/AES-NI/PCLMULQDQ intrinsic on a laptop with Intel i7-11800H CPU 2.30 GHz with Turbo Boost up to 4.6 GHz. All experiments are conducted in a single process/thread manner. The software performance is measured in Gbps as stream ciphers. Test vectors and reference implementations of LOL-MINI/LOL-DOUBLE are given in the full version[3)]. This section is written with Intel intrinsic notation, but similar implementations can be made on other CPUs, such as AMD and ARM.

Encryption/decryption speeds. Unlike block ciphers, stream ciphers are more sensitive to data lengths: the longer the data is processed, the higher will be the speed acquired. Therefore, we evaluate the software performances using message lengths ranging from 32 to 16384 bytes. Although encountered with security issues, it is still true that the SNOW-V stream cipher and pure encryption modes of AEGIS and Rocca remain the fastest stream ciphers. Hence, we conduct the same experiments on SNOW-V, AEGIS, and Rocca on the same platform using exactly the C++ source codes from the origins [19,23,25] for fair comparisons. The data are shown in Table 2.

The encryption/decryption speeds of LOL-MINI and LOL-DOUBLE can reach 89 and 99 Gbps using the AVX2 instructions, respectively. When using AVX512 instructions, the speed of LOL-DOUBLE can reach 135 Gbps, catering to 5G and 6G requirements. The LOL-MINI and AEGIS exhibit similar efficiency (LOL-MINI is only slightly faster than AEGIS on short messages); however, they are slower than LOL-DOUBLE but much faster than SNOW-V. Rocca still demonstrates the highest speed but with different security targets, such as 128-bit security against distinguishing attacks. The high speeds of LOL-MINI and LOL-DOUBLE indicate that the LNFSR+AES structure can be competitive even when pure AES-based primitives are used. Considering the security issues listed in the introduction section and the diversity of cipher designs (not relying only on the AES encryption round function), LOL-MINI and LOL-DOUBLE are more competitive. Notably, the AEGIS performance on our platform is much better than the original 56 Gbps in [19], implying that the latest CPUs have made progress for supporting

Table 2 Software performance (Gbps) of AEGIS, SNOW-V, Rocca, LOL-MINI and LOL-DOUBLE in the pure encryption mode[a)]

Bytes	AEGIS256	SNOW-V	Rocca	LOL-MINI	LOL-DOUBLE	LOL-DOUBLE†
32	12.58	5.35	12.47	11.22	6.77	9.36
64	19.94	9.61	**23.20**	19.58	13.04	17.37
96	**24.70**	13.31	31.25	**24.34**	18.16	**23.52**
128	29.30	16.14	40.26	30.88	**22.55**	29.47
160	34.17	19.14	47.25	36.53	26.85	34.88
192	38.34	**21.54**	54.78	40.55	31.27	38.86
224	41.90	23.62	60.84	44.56	34.57	43.41
256	45.75	25.47	66.82	45.20	37.08	47.93
1024	70.77	40.73	109.51	71.22	68.30	92.40
2048	79.54	45.44	129.83	79.62	82.39	110.54
4096	84.96	49.52	141.78	85.01	91.43	122.42
8192	87.83	51.39	147.85	87.84	96.71	129.52
16384	89.38	52.37	152.04	89.42	99.59	135.00

a) Note that the LOL-DOUBLE is implemented in both AVX2 and AVX512 manners while others only use the AVX2 instructions. The shortest message length reaching 20 Gbps is represented in a bold format. †: Implemented with the AVX512 instructions.

AES-NI instructions, and such improvements can be fully used in the frequent utilization of the parallel AES operations. By contrast, SNOW-V shares the same speed as the original 58 Gbps, reflecting the necessity of optimal non-AES component designs.

AEAD speeds. Moreover, combined with standard GCM and NMH modes, LOL-MINI and LOL-DOUBLE stream ciphers can also be applied to AEAD schemes generating 128-bit tags at speeds over 40 Gbps, which is much better than that of the similar AEAD application of SNOW-V [23], please refer to the full version[3)] for more information.

7 Conclusion

In this paper, we propose the LOL framework for designing extendable, software-efficient stream ciphers with high parallelism and flexibility. The proposed framework is friendly to automatic cryptanalysis tools for conducting security evaluations. Specifically, we provide two concrete stream cipher designs, LOL-MINI and LOL-DOUBLE, which support 256-bit keys and exhibit software performances of 89 and 135 Gbps, respectively. They also satisfy the speed and security requirements of mobile systems beyond the 5G/6G. Combined with the standard GCM and NMH modes, LOL-MINI and LOL-DOUBLE stream ciphers can also be applied to AEAD schemes that generate 128-bit tags at speeds greater than 40 Gbps, which is much faster than the 28-Gbps SNOW-V-GCM counterpart. We expect that more designs will adopt the LOL framework in the future.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 62002024, 62072445, 62372464, 12371526).

References

1 Martin S. Global system for mobile communications (GSM). In: From GSM to LTE-Advanced: An Introduction to Mobile Networks and Mobile Broadband. Hoboken: Wiley, 2014
2 ETSI/SAGE Specification. Specification of the 3GPP confidentiality and integrity algorithms UEA2 & UIA2. Document 1: UEA2 and UIA2 specification. 2009. https://www.gsma.com/about-us/wp-content/uploads/2014/12/uea2uia2d1v21.pdf
3 ETSI/SAGE Specification. Specification of the 3GPP confidentiality and integrity algorithms 128-EEA3 & 128-EIA3. Document 3: implementor's test data. 2011. https://www.gsma.com/about-us/wp-content/uploads/2014/12/eea3eia3testdatav11.pdf
4 ITU. Minimum Requirements Related to Technical Performance for IMT-2020 Radio Interface(s). Mobile, Radiodetermination, Amateur and Related Satellite Services, Report ITU-R M.2410-0(11/2017), 2017
5 Aazhang B, Ahokangas P, Alves H, et al. Key Drivers and Research Challenges for 6G Ubiquitous Wireless Intelligence (White Paper). Oulu: 6G Flagship, 2019
6 Targhi E E, Tabia G N, Unruh D. Quantum collision-resistance of non-uniformly distributed functions. In: Proceedings of the 7th International Workshop, 2016. 79–85
7 Kaplan M, Leurent G, Leverrier A, et al. Breaking symmetric cryptosystems using quantum period finding. In: Proceedings of Annual International Cryptology Conference, 2016. 207–237
8 Chailloux A, Naya-Plasencia M, Schrottenloher A. An efficient quantum collision search algorithm and implications on symmetric cryptography. In: Proceedings of Annual International Cryptology Conference, 2017. 211–240
9 Dong X, Dong B, Wang X. Quantum attacks on some Feistel block ciphers. Des Codes Cryptogr, 2020, 88: 1179–1203

10 Balogh M, Eaton E, Song F. Quantum collision-finding in non-uniform random functions. In: Proceedings of the 9th International Conference on Post-Quantum Cryptography, 2018. 467–486
11 Ni B, Ito G, Dong X, et al. Quantum attacks against type-1 generalized Feistel ciphers and applications to CAST-256. In: Proceedings of on Cryptology in India, 2019. 433–455
12 Liu Q, Zhandry M. On finding quantum multi-collisions. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2019. 189–218
13 Dong X, Sun S, Shi D, et al. Quantum collision attacks on AES-like hashing with low quantum random access memories. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2020. 727–757
14 Hosoyamada A, Sasaki Y. Quantum collision attacks on reduced SHA-256 and SHA-512. In: Proceedings of Annual International Cryptology Conference, 2021. 616–646
15 Hirose S, Kuwakado H. A note on quantum collision resistance of double-block-length compression functions. In: Proceedings of the 18th IMA International Conference on Cryptography and Coding, 2021. 161–175
16 Bonnetain X, Schrottenloher A, Sibleyras F. Beyond quadratic speedups in quantum attacks on symmetric schemes. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2022. 315–344
17 Schrottenloher A, Stevens M. Simplified MITM modeling for permutations: new (quantum) attacks. In: Proceedings of Annual International Cryptology Conference, 2022. 717–747
18 ARM. ARM architecture reference manual for ARMv8-A (64-bit). 2013. https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/arm-architecture-reference-manual-for-armv8-a-64-bit-publicly-released
19 Wu H, Preneel B. AEGIS: a fast authenticated encryption algorithm. In: Proceedings of International Conference on Selected Areas in Cryptography, 2014. 185–201
20 CAESAR. CAESAR: competition for authenticated encryption: security, applicability, and robustness. 2014. https://competitions.cr.yp.to/caesar.html
21 Feng C. Support of 256-bit algorithm in 5G mobile communication system. J Inform Secur Res, 2020, 6: 716–721
22 Design team T Z. The zuc-256 stream cipher. 2018. http://www.is.cas.cn/ztzl2016/zouchongzhi/201801/W020180416526664982687.pdf
23 Ekdahl P, Johansson T, Maximov A, et al. A new SNOW stream cipher called SNOW-V. IACR Trans Symm Cryptol, 2019, 2019: 1–42
24 Ekdahl P, Maximov A, Johansson T, et al. SNOW-Vi: an extreme performance variant of SNOW-V for lower grade CPUs. In: Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2021. 261–272
25 Sakamoto K, Liu F, Nakano Y, et al. Rocca: an efficient AES-based encryption scheme for beyond 5G. IACR Trans Symm Cryptol, 2021, 2021: 1–30
26 Eichlseder M, Nageler M, Primas R. Analyzing the linear keystream biases in AEGIS. IACR Trans Symm Cryptol, 2019, 2019: 348–368
27 Liu F, Isobe T, Meier W, et al. Weak keys in reduced AEGIS and Tiaoxin. IACR Trans Symm Cryptol, 2021, 2021: 104–139
28 Shi Z, Jin C, Zhang J, et al. A correlation attack on full SNOW-V and SNOW-vi. In: Proceedings of Annual International Conference on the Theory and Applications of Cryptographic Techniques, 2022. 34–56
29 Zhou Z, Feng D, Zhang B. Efficient and extensive search for precise linear approximations with high correlations of full SNOW-V. Des Codes Cryptogr, 2022, 90: 2449–2479
30 Hosoyamada A, Inoue A, Ito R, et al. Cryptanalysis of Rocca and feasibility of its security claim. IACR Trans Symm Cryptol, 2022, 2022: 123–151
31 Sakamoto K, Liu F, Nakano Y, et al. Rocca: an efficient aes-based encryption scheme for beyond 5G (full version). IACR Cryptol, 2022, 2022: 116
32 Nyberg K, Wallén J. Improved linear distinguishers for SNOW 2.0. In: Proceedings of International Workshop on Fast Software Encryption, 2006. 144–162
33 Zhang B, Xu C, Meier W. Fast correlation attacks over extension fields, large-unit linear approximation and cryptanalysis of SNOW 2.0. In: Proceedings of Annual Cryptology Conference, 2015. 643–662
34 Gong X, Zhang B. Comparing large-unit and bitwise linear approximations of SNOW 2.0 and SNOW 3G and related attacks. IACR Trans Symm Cryptol, 2021, 2021: 71–103
35 Dworkin M. Recommendation for Block Cipher Modes of Operation: Galois/counter Mode (GCM) and GMAC. NIST SP 800-38D, 2007
36 Halevi S, Krawczyk H. MMH: software message authentication in the Gbit/second rates. In: Proceedings of International Workshop on Fast Software Encryption, 1997. 172–189
37 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
38 Sun B, Liu Z, Rijmen V, et al. Links among impossible differential, integral and zero correlation linear cryptanalysis. In: Proceedings of Annual Cryptology Conference, 2015. 95–115
39 Babbage S. Improved "exhaustive search" attacks on stream ciphers. In: Proceedings of European Convention on Security and Detection, 1995
40 Golic J D. Cryptanalysis of alleged A5 stream cipher. In: Proceedings of International Conference on the Theory and Applications of Cryptographic Techniques, 1997
41 Biryukov A, Shamir A. Cryptanalytic time/memory/data tradeoffs for stream ciphers. In: Proceedings of International Conference on the Theory and Application of Cryptology and Information Security, 2000
42 Courtois N T, Pieprzyk J. Cryptanalysis of block ciphers with overdefined systems of equations. In: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, 2002. 267–287