مجله علمی پژوهشی «علوم و فناوری‌های پدافند نوین»
سال پنجم، شماره ۲، تابستان ۱۳۹۳؛ ص ۹۱-۸۱

طراحی و تحلیل یک الگوریتم جدید رمز جریانی امن

یوسف پورابراهیم*

مربی، گروه مهندسی برق و کامپیوتر، واحد مشکین شهر، دانشگاه آزاد اسلامی، مشکین شهر

(دریافت: ۹۲/۱۰/۱۱، پذیرش: ۹۳/۰۷/۱۶)

چکیده

در این مقاله اصول طراحی و تحلیل یک الگوریتم رمز جریانی جدید با طول کلید ۲۵۶ بیت ارائه گردیده است. این الگوریتم پیشنهادی مبتنی بر کلمه بوده و در برابر حملات معروف بسیار مقاوم می‌باشد. الگوریتم بر اساس ثبات‌های انتقال بهبود یافته کلمه‌ای در حالت کلاک کنترل با دوره تناوب بزرگ‌تر از $۲^{۱۲۱۴}$ طراحی شده است. با بهبود بخش غیرخطی هسته الگوریتم‌های خاص از قبیل SNOW2 و به‌کارگیری آن به عنوان بخش غیرخطی الگوریتم پیشنهادی، رمزنگار حاصل در برابر حملات تمایز و جبری بسیار مقاوم‌تر از نمونه‌های مشابه می‌باشد. همچنین طبق تحلیل‌های امنیتی صورت گرفته در خصوص پیچیدگی حملات، مرتبه حمله تمایز برابر $۲^{۹۰۳۳}$ و مرتبه حمله جبری برابر $۲^{۲۸۳۹}$ به دست آمده است.

کلید واژه‌ها: رمز جریانی، ثبات انتقال کلمه‌ای، حمله تمایز، حمله جبری.

Design and Analysis of a New Secure Stream Cipher Algorithm

Y. Pourebrahim*
Islamic Azad University of Meshkinshahr
(Received: 01/01/2014; Accepted: 08/10/2014)

Abstract

In this paper, a new word-oriented stream cipher algorithm with a key length of 256 bit is designed and analyzed. The proposed algorithm is highly resistant against the known attacks. The main principles of the algorithm are based on improved word-oriented shift registers in clock control mode with cyclic period greater than 2^{1214}. *With the improvement of certain nonlinear kernel of some algorithms, such as SNOW2 and using it as a nonlinear part of encryption, the proposed algorithm is much more resistant than other similar ciphers against distinguishing and algebraic attacks. According to the security analysis, the complexity of distinguishing attack is* $O(2^{9033})$ *and complexity of algebraic attacks is* $O(2^{2839})$.

Keywords: Stream Cipher, Word-Oriented Shift Register, Distinguishing Attack, Algebraic Attack.

* Corresponding Author E-mail: pourebrahim@meshkin-iau.ac.ir

۱. مقدمه

تولید اعداد تصادفی، در سامانه‌های رمزنگاری اهمیت ویژه‌ای دارد. دنباله اعداد تولید شده توسط یک مولد اعداد تصادفی، باید دارای توزیع آماری یکنواخت و همچنین دوره تناوب طولانی و غیرقابل پیش‌بینی باشد.

ثبات‌های انتقال با بازخورد خطی یکی از بهترین ابزار جهت تولید دنباله‌های تصادفی با توزیع آماری مناسب هستند. اما باید توجه داشت که دنباله‌های تولید شده توسط یک ثبات خطی، تصادفی نبوده و به دلیل ساختار ذاتی آنها، هر یک از سمبل های خروجی به راحتی با ترکیبی از سمبل‌های خروجی قبلی قابل تولید و در نتیجه قابل پیش‌بینی خواهند بود. با این وجود به دلیل خواص آماری خوب، ثبات‌های خطی در تولید اعداد تصادفی کاربرد زیادی دارند.

ثبات‌های انتقال با بازخورد خطی مبتنی بر میدان دو عضوی $F_2 = \{0,1\}$ در هر پالس ساعت بر اساس توابع بازخورد و انتقال، فقط یک بیت تولید می‌کنند. چنین ساختارهایی برای کاربردهای نرم‌افزاری بسیار کند می‌باشند. برای رفع این نقیصه، در مرجع [۱] نوعی طراحی ثبات‌های خطی ارائه شده که در آن استفاده از مکانیزم‌های موازی مبتنی بر کلمه برای پردازشگرهای جدید پیشنهاد شده است.

الگوریتم‌های SNOW1.0، SNOW2.0، SOSEMANUK و SNOW 3G اعضای خانواده SNOW هستند که دارای ساختار مشابهی هستند. این ساختار یک مولد از نوع فیلتر غیرخطی بوده که از یک ثبات خطی مبتنی بر کلمه و یک فیلتر غیرخطی تشکیل می‌شود. فیلتر غیرخطی در این الگوریتم‌ها از دو ثبات و عناصر غیرخطی‌ای مانند جمع پیمانه‌ای و جعبه جانشینی تشکیل شده است.

اولین الگوریتم خانواده SNOW تحت نام SNOW 1.0 در سال ۲۰۰۰ به عنوان یک الگوریتم رمزنگاری جریانی "مبتنی بر کلمه" معرفی شد [۲]. در همان سال این الگوریتم در پروژه NESSIE مطرح شده و در سال ۲۰۰۲ نسخه دوم آن تحت نام SNOW 2.0 ارائه گردید [۳] و به عنوان یکی از چهار الگوریتم رمزنگاری جریانی منتخب، در مرحله دوم پروژه NESSIE کاندید شد (هر چند در انتهای این پروژه به دلیل ضعف الگوریتم‌های رمزنگاری جریانی موجود، هیچ الگوریتمی به عنوان الگوریتم رمزنگاری جریانی امن معرفی نگردید [۴]). هدف نهایی از ارائه این الگوریتم، دستیابی به سرعتی بالاتر از AES با هزینه‌ای کمتر برای پیاده‌سازی، و امنیتی حداقل به اندازه امنیت AES بود.

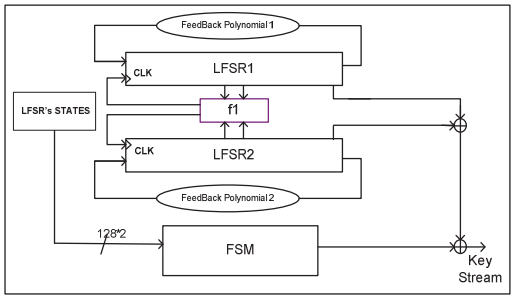
در این مقاله با توجه به ساختار مناسب الگوریتم SNOW2 و الگوریتم‌های مشابه آن یک الگوریتم رمز جریانی جدید امن ارائه شده است. در این الگوریتم تمامی ضعف‌های موجود در الگوریتم‌های خانواده SNOW2 برطرف گردیده است و کل مراحل طراحی بومی می‌باشد.

در طراحی الگوریتم پیشنهادی از دو ثبات انتقال خطی کلمه‌ای در یک قالب بدیع و کارا استفاده شده است. این ثبات‌ها حالت بهبود یافته ثبات‌های به‌کار برده شده در الگوریتم SNOW2 می‌باشند که در اصل بر پایه ثبات‌های خطی سیگما بوده که به نوعی در زمان طراحی بهبود داده شده‌اند. همچنین استفاده از حالت کلاک کنترلی برای ثبات‌ها، تابع جانشینی و تابع انتشار از نوع MDS با طراحی مناسب برای پیاده‌سازی کلمه به‌منظور افزایش سرعت رمزنگاری یکی از نقاط قوت الگوریتم می‌باشد. از نقاط قوت دیگر این الگوریتم استفاده از یک ثبات خطی است که طول ثبات بیتی آن عددی فرد و اول می‌باشد و به‌دلیل پیاده‌سازی کلمه‌ای سرعت رمزنگاری بالایی دارد.

ساختار ادامه مقاله به این شرح است که در بخش دوم ساختار الگوریتم جدید رمزنگاری تشریح گردیده است که شامل ثبات‌های انتقال به‌کار برده شده و ماشین حالت محدود و طرح تولید کلید می‌باشد. در بخش سوم اصول و معیارهای طراحی هر یک از اجزای الگوریتم بیان گردیده است و بخش چهارم تحلیل امنیتی الگوریتم ارائه شده است. در نهایت در بخش آخر نتیجه‌گیری آمده است.

۲. ساختار الگوریتم جدید رمزنگاری

الگوریتم تشکیل شده از دو ثبات انتقال و یک ماشین حالت محدود با سه حافظه و یک تابع غیرخطی جهت کنترل کلاک‌ها می‌باشد. شکل (۱) ساختار کلی الگوریتم رمز را نشان می‌دهد که در آن از ثبات انتقال سیگمای بهبودیافته استفاده شده است.



شکل ۱. ساختار کلی الگوریتم رمز جریانی

۲-۱. ساختار ثبات انتقال سیگما

فرض کنید که m یک عدد صحیح مثبت و F_{2^m} میدان متناهی با 2^m عضو باشد. فرض کنید $M_m(F_2)$ یک حلقه ماتریسی روی F_2 بوده و $GL_m(F_2) \subseteq M_m(F_2)$ یک گروه خطی کلی باشد. نشان داده می‌شود که $F_{2^m} \subseteq M_m(F_2)$ می‌باشد [۵].

تعریف۱) فرض کنید که n یک عدد صحیح مثبت بوده و $C_0, C_1, \ldots, C_{n-1} \in M_m(F_2)$. اگر دنباله $s^\infty = s_0, s_1, \cdots$ روی F_{2^m} معادله (۱) را برآورده کند:

$$s_{i+n} = -(C_0 s_i + C_1 s_{i+1} + \cdots + C_{n-1} s_{i+n-1}) \quad for\ i = 0,1,\cdots \tag{۱}$$

آنگاه رابطه بالا مشخص‌کننده Sigma_LFSR از مرتبه n بوده و چند جمله‌ای معادله (۲) چندجمله‌ای ماتریسی آن می‌باشد:

$$f(x) = x^n + C_{n-1}x^{n-1} + \cdots + C_1 x + C_0 \in M_m(F_2)[x] \tag{۲}$$

قضیه (۱) دنباله تولید شده توسط Sigma_LFSR تناوبی است اگر و تنها اگر $C_0 \in M_m(F_2)$ معکوس‌پذیر باشد. به‌عبارتی $C_0 \in GL_m(F_2)$ است [۵].

تعریف۲) فرض کنید دنباله $s^{\infty} = s_0, s_1, \cdots$ توسط معادله (۱) تعریف شده باشد. اگر دوره تناوب آن برابر $2^{mn}-1$ باشد آنگاه دوره تناوب Sigma_LFSR بیشینه مقدار ممکن بوده و چندجمله‌ای آن اولیه خواهد بود.

قضیه (۲) فرض کنید دنباله $s^{\infty} = s_0, s_1, \cdots$ توسط Sigma_LFSR با چندجمله‌ای (۳) تولید شده باشد.

$$f(x) = x^n + C_{n-1}x^{n-1} + \cdots + C_1 x + C_0 \in M_m(F_2)[x] \tag{۳}$$

که در آن، $C_0 \in GL_m(F_2)$ بوده و $C_l = (c_l^{ij})_{m\times m}\ for\ i = 0,1,\cdots,n-1$ می‌باشد. در این صورت رابطه:

$$F(x) = (f^{ij}(x))_{m\times m} \in M_m(F_2)[x] \tag{۴}$$

ماتریس چندجمله‌ای معادل f(x) خواهد بود و در آن داریم:

$$f^{ij}(x) = \delta^{ij} x^n + \sum c_l^{ij} x^l \in F_2[x], \qquad \delta^{ij} = \begin{cases} 1, & i = j; \\ 0, & i \neq j. \end{cases} \tag{۵}$$

Sigma_LFSR رابطه (۱) اولیه خواهد بود اگر و تنها اگر دترمینان $|F(x)|$ چندجمله‌ای اولیه از درجه mn روی F_2 باشد [۵].

قضیه (۲) شرایط این که کدام سیگما ثبات انتقال اولیه است و کدام اولیه نیست را بیان می‌کند.

۲-۲. ثبات‌های انتقال مورد استفاده

در الگوریتم رمز از دو چندجمله‌ای اولیه به عنوان چندجمله‌ای بازخورد ثبات‌های انتقال مورد استفاده قرار گرفته است که در اصل ثبات‌های انتقالی از نوع سیگما می‌باشند.

با توجه به این اینکه در ثبات‌های انتقالی از نوع سیگما تنها از عملگرهای ساده انتقال به چپ یا راست و عملگر ضرب منطقی (AND) استفاده می‌شود، ممکن است در مقابل برخی حملات از خود ضعفی بروز دهند. بنابراین برای افزایش پیچیدگی حملات از عملگرهای مشابه آنچه که در SNOW2 و دیگر الگوریتم‌های مشابه آن به‌کار رفته، در ساختار ثبات‌های انتقال از نوع سیگما استفاده شده است. چندجمله‌ای‌های ثبات‌های انتقال بهبود یافته در معادلات (۶) و (۷) نشان داده شده‌اند. ویژگی این چندجمله‌ای‌ها قابلیت پیاده‌سازی آنها به صورت کلمه‌ای می‌باشد. معادل بیتی چندجمله‌ای‌های $f_1(x)$ و $f_2(x)$ به ترتیب از درجه ۶۰۸ و ۶۰۷ می‌باشد.

$f_1(x)$ به‌صورت یک ثبات انتقال با ۱۹ حالت بوده که هر حالت، خود ۳۲ بیتی می‌باشد. با توجه به چندجمله‌ای بازخورد محل تپ‌ها در حالت‌های ۰، ۲، ۶، ۱۱ و ۱۸ می‌باشد. چندجمله‌ای $f_2(s)$ دارای چهار تپ بازخورد حالت ۰، ۱، ۸ و ۱۱ می باشد.

$$f_1(X) = \alpha X^{19} + X^{17} + \&0x200000C5X^{13} + \alpha^{-1}X^8 + X + 1 \tag{۶}$$

$$f_2(S) = S^{19} + S^{11} + \alpha S^8 + (0x\,00000001 S^1 \,|\, 0xffffffffe\ S^0) \times [A]_{32\times 32} \tag{۷}$$

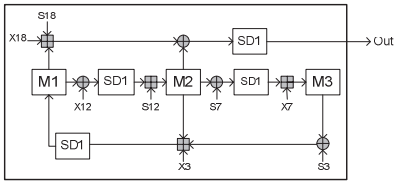
که در آن، & همان عمل AND بوده و | عملگر OR می‌باشد و α, α^{-1} به ترتیب انتقال به راست و انتقال به چپ شرطی به تعداد یک بیت می‌باشند؛ به این صورت که اگر بیت خروجی برابر ۱ باشد آنگاه حاصل انتقال به راست و انتقال به چپ به ترتیب با مقادیر 0xce3b1d99 و 0x9c763b33 جمع تحت میدان F_2 می‌شوند.

برای ثبات انتقال دوم پیاده‌سازی ۳۲ بیتی به این صورت است که از سلول صفر و یک تپ‌های بازخورد در هر کلاک ۳۲ بیت خارج و با انجام عمل & با مقادیر نشان داده شده ۳۱ بیت سلول صفرم به عنوان ۳۱ بیت با ارزش یک کلمه ۳۲ بیتی خواهند بود و کم ارزش‌ترین بیت این کلمه از مقدار سلول اول تأمین خواهد شد. این کلمه ۳۲ بیتی حاصل شده سپس از سمت راست در ماتریس A ضرب شده و حاصل بعد از xor شدن با مقادیر دیگر مقدار بازخورد را ایجاد خواهد کرد. شبه‌کد C پیاده‌سازی ثبات انتقال دوم در ادامه آورده شده است:

```
up = 0xffffffffe
lp = 0x00000001
x[19] = (x[0] & up) | (x[1] & lp)
x[19] = (x[19] >> 1) ⊕ x[11] ⊕ αx[8]
⊕ { 0          if LSB of x[19] = 0
  { 0x80000058 if LSB of x[19] = 1
```

۲-۳. ماشین حالت محدود

ماشین حالت محدود این رمزکننده دارای سه حافظه ۳۲ بیتی به نام‌های M1 و M2 و M3 می‌باشد. ورودی این ماشین حالت محدود، تعدادی از حالت‌های ثبات‌های انتقال است که در شکل (۲) مشخص می‌باشند.



شکل ۲. ماشین حالت محدود

اصول کار این ماشین حالت به صورت زیر می‌باشد:

۱- حافظه‌های M1 و M2 و M3 به مقدار صفر ست می‌شوند.

۲- با استفاده از مقادیر سلول‌های X12 و S12 و تابع SD1 حافظه M2 از روی مقدار فعلی حافظه M1 به روز می‌شود. به عبارتی:

$$M2_{t+1} = SD1(M1_t \oplus X_{12}) + S_{12}$$

۳- با استفاده از مقادیر سلول‌های X7 و S7 و تابع SD1 حافظه M3 از روی مقدار فعلی حافظه M2 به روز می‌شود. به عبارتی:

$$M3_{t+1} = SD1(M1_t \oplus S_7) + X_7 \quad (۸)$$

۴- با استفاده از مقادیر سلول‌های X3 و S3 و تابع SD1 حافظه M1 از روی مقدار فعلی حافظه M3 و حافظه M2 به روز می‌شود. به عبارتی:

$$M1_{t+1} = SD1(M3_t \oplus S_3 + X_3 + M2_t) \quad (۹)$$

توجه شود که در مراحل بیان شده در بالا، اندیس t نشان دهنده مقدار فعلی بوده و اندیس t+1 مقدار بعدی (به روز شده) را نشان می‌دهد.

توابع SD1 به‌کار برده شده در FSM از چهار جعبه جانشانی Sbox و یک لایه انتشار از نوع MDS تشکیل شده است. شکل (۳) تابع SD1 را نشان می‌دهد. رابطه ریاضی بین ورودی و خروجی Sboxها تحت میدان 0x11B به صورت زیر است:

$$Sbox1(x) = [A \times ((x+128) \oplus 99)^{-1}] \oplus 59$$

$$Sbox2(x) = [B \times (((B \times x)+192) \oplus 74)^{-1}] \oplus 135$$

که تابع بولی A و B به صورت زیر می‌باشند:

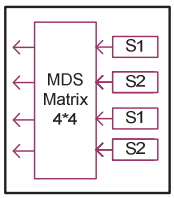
$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

لایه انتشار به‌کار رفته به صورت ماتریس (۱۰) است:

$$MDS = \begin{bmatrix} 196 & 130 & 227 & 164 \\ 130 & 196 & 164 & 227 \\ 227 & 164 & 196 & 130 \\ 164 & 227 & 130 & 196 \end{bmatrix} \quad (۱۰)$$

عناصر این ماتریس انتشار از میدان 0x163 می‌باشد.



شکل ۳. طرح تابع SD1

تابع f1 دو سویه بوده و خروجی آن (CLK) کلاک‌های ثبات‌ها را کنترل خواهد کرد و به صورت زیر تعریف شده است.

```
clk1 = LFSRi.V8[38]&0x01;
clk2 = (LFSRi.V8[42]&0x08)>>0x07;
CLK = 2*clk1+clk2+1;
```

در تابع فوق، متغیر LFSRi.V8[j] ، مربوط به zامین بایت از iامین ثبات انتقال است. به عبارتی ساده‌تر بیت نهم از سلول دهم و بیت شانزدهم از سلول یازدهم ورودی تابع f1 می‌باشند. خروجی این تابع یکی از مقادیر متعلق به مجموعه {۱،۲،۳،۴} می‌باشد که این خروجی به طور یکنواخت توزیع شده است.

۲-۴. طرح کلید

به منظور مقداردهی اولیه LFSRها و حافظه، فرض می‌شود ۵۱۲ بیت کلید اصلی اولیه در قالب یک ارائه ۶۴ بایتی به نام KeyInit[64] قرار می‌گیرید که اندیس کوچک‌تر مقدار کم ارزش کلید ورودی را در بر دارد. سپس طبق روال زیر دو LFSR مقداردهی می‌شوند:

```
for (i=0;i<16;++i)
{
LFSR1.V8[4*i+0] =KeyInit.V8[i*4+0];
LFSR1.V8[4*i+1] =KeyInit.V8[i*4+1]^0xff;
LFSR1.V8[4*i+2] =KeyInit.V8[i*4+2];
LFSR1.V8[4*i+3] =KeyInit.V8[i*4+3]^0xff;
LFSR2.V8[4*i+0] =KeyInit.V8[i*4+0]^0xff;
LFSR2.V8[4*i+1] =KeyInit.V8[i*4+1];
LFSR2.V8[4*i+2] =KeyInit.V8[i*4+2]^0xff;
LFSR2.V8[4*i+3] =KeyInit.V8[i*4+3];
}
LFSR1.V32[16] =KeyInit.V32[0];
LFSR1.V32[17] =KeyInit.V32[1];
LFSR1.V32[18] =KeyInit.V32[2];
LFSR2.V32[16] =KeyInit.V32[3];
LFSR2.V32[17] =KeyInit.V32[4];
LFSR2.V32[18] =KeyInit.V32[5];
```

متغیرهای LFSR.V8[i] آدرس بایت iام هریک از ثبات انتقال‌ها می‌باشد. بعد از آنکه مقادیر اولیه در ثبات انتقال‌ها قرار گرفتند، به تعداد ۱۹ کلاک عملیات زیر تکرار می‌شوند. توجه گردد مقادیر اولیه

حافظه‌ها برابر صفر قرار داده می‌شوند و مقدار اولیه متغیر Loop برابر صفر می‌باشد.

۱- مقدار Loop = Loop +1 قرار دهید.

۲- تابع FSM به روز شود.

۳- ثبات انتقال اول به طور منظم کلاک می‌خورد.

۴- ثبات انتقال دوم به طور منظم کلاک می‌خورد.

۵- خروجی تابع FSM با مقدار بازخورد ثبات انتقال اول (مقدار سلول نوزدهم) تحت میدان F_2 جمع شود.

۶- مقدار حافظه سوم با مقدار بازخورد ثبات انتقال دوم (مقدار سلول نوزدهم) تحت میدان F_2 جمع شود.

۷- اگر Loop==19 خروج و در غیر این صورت به گام اول برگردید.

بعد از اجرای مراحل فوق، تعداد یک گام دیگر مراحل فوق بدون اجرای مراحل پنجم و ششم اجرا می‌شود و سپس رمزکننده آماده تولید رشته کلید خواهد شد. این یک گام در واقع دور ریزی بعد از مقداردهی اولیه می‌باشد و با توجه به زمان مورد نیاز برای مقداردهی اولیه می‌تواند تعداد این دورریزی بیشتر شود.

۲-۵. تولید دنباله کلید

با اختصاص مقادیر اولیه به ثبات انتقال و حافظه‌های M1 و M2 و M3 در زمان $t = 0$ ترتیب عملیات جهت تولید رشته کلید به صورت ذیل می‌باشد:

۱- ماشین حالت محدود به روز می‌شود؛ به این صورت که مقادیر $M1_t$ و $M2_t$ و $M3_t$ با استفاده از مقادیر $M1_{t-1}$ و $M2_{t-1}$ و $M3_{t-1}$ و با اعمال مقادیر Stateهای بیان شده در بخش‌های قبلی محاسبه می‌گردد.

۲- با استفاده از دو بیت نهم و شانزدهم سلول‌های به ترتیب دهم و یازده ثبات انتقال دوم به عنوان ورودی تابع f1 تعداد کلاک‌های ثبات انتقال اول تعیین می‌شود.

۳- ثبات انتقال اول به تعداد مشخص شده در گام قبلی به روز می‌شود. در کلاک آخر و قبل از اعمال کلاک مقدار سلول اول از ثبات انتقال در یک متغیر نگه داشته می‌شود.

۴- با استفاده از دو بیت نهم و شانزدهم سلول‌های به ترتیب دهم و یازده ثبات انتقال اول به عنوان ورودی تابع f1 تعداد کلاک‌های ثبات انتقال دوم تعیین می‌شود.

۵- ثبات انتقال دوم به تعداد مشخص شده در گام قبلی به روز می‌شود. در کلاک آخر و قبل از اعمال کلاک مقدار سلول اول از ثبات انتقال در یک متغیر نگه داشته می‌شود.

۶- مقدار خروجی تابع FSM با مقادیر متغیرهای بیان شده در گام‌های سوم و پنجم xor شده و رشته کلید خروجی تعیین می‌شود.

مراحل بالا به تعداد لازم اجرا می‌گردند تا رشته کلیدهای مورد نیاز برای رمزنگاری به دست آیند. همچنین سرعت تولید رشته کلید خروجی با پیاده‌سازی غیربهینه به زبان C در پردازنده ۱.۸ گیگا هرتزی برابر 300 Mbit/sec به دست آمده است.

۳. معیارهای طراحی اجزای الگوریتم

در طراحی اجزا الگوریتم اصول و معیارهایی مد نظر قرار گرفته‌اند که از یک طرف ساختار الگوریتم را در برابر حملات مطرح مقاوم سازد و از طرف دیگر الگوریتم سرعت رمزنگاری مناسبی داشته باشد.

۳-۱. اصول انتخاب چندجمله‌ای بازخورد

چند جمله‌ای بازخورد مربوط به ثبات‌های انتقال کلمه‌ای باید تا حد ممکن تنک[1] باشد که به همین دلیل چندجمله‌ای اولیه به فرم زیر انتخاب شده است که در آن $0 < a < b < c < d < 19$ می‌باشند.

$$f(x) = X^{19} + w_a X^{n-a} + w_b X^{n-b} + w_c X^{n-c} + w_d X^{n-d} + 1 \quad (۱۱)$$

ضرایب $\{w_a, w_b, w_c, w_d\}$ از مجموعه عملگرهایی انتخاب شده‌اند که پیاده‌سازی نرم‌افزاری رمزکننده بهینه باشد. همچنین لازم است مقادیر a, b, c, d با طول ثبات انتقال نسبت به هم اولیه باشند. در غیر این صورت اگر $q = \gcd(a,n) > 1$ باشد رابطه بازگشتی (۱۲) چهار جمله از دنباله $(S_{qt+i})_{t>0}$ را به ازای بعضی از مقادیر i شامل خواهد شد که قابل تولید توسط یک LFSR از طول n/q می‌باشد که در واقع همان حمله ورچیدن[2] می‌باشد [۶ و ۷].

$$S_{t+n} = w_d S_{t+d} + w_c S_{t+c} + w_b S_{t+b} + w_a S_{t+a} + S_t \quad (۱۲)$$

قضیه ۳) اگر دوره تناوب دنباله خروجی دو LFSR متفاوت به ترتیب برابر $2^m - 1$ و $2^n - 1$ و نسبت به هم اول باشند آنگاه دوره تناوب دنباله حاصل از xor دو دنباله بیان شده برابر $(2^m - 1) \times (2^n - 1)$ بوده که در رابطه زیر صدق می‌کند:

$$2^{m+n-1} + 1 < (2^m - 1) \times (2^n - 1) < 2^{m+n} - 1 \quad (۱۳)$$

برای به دست آوردن دوره تناوب دو LFSR بیان شده در بالا از قضیه (۳) استفاده می‌کنیم. این دو LFSR دارای دوره تناوب به ترتیب $2^{۳۲\times۱۹-۱} - 1$ و $2^{۳۲\times۱۹} - 1$ می‌باشد. اگر خروجی این دو LFSR در هر بار کلاک‌زنی با هم جمع در مبنای ۲ شوند آنگاه دنباله حاصل دارای دوره تناوب بزرگ‌تر از $2^{۱۲۱۴}$ خواهد بود. برای اثبات این مسئله داریم:

[1] Sparse
[2] Decimation

$$LCM((2^{608}-1)\times(2^{607}-1)) = \frac{(2^{608}-1)\times(2^{607}-1)}{\gcd((2^{608}-1)\times(2^{607}-1))} = \frac{2^{608+607}-2^{608}-2^{607}+1}{1} > 2^{1215-1} \quad (۱۴)$$

بنابراین دوره تناوب بزرگ‌تر از 2^{1214} می‌باشد. البته توجه شود که این دوره تناوب به ازای کلاک زنی منظم دو ثبات انتقال به دست آمده است.

۳-۲. معیار طراحی ماشین حالت محدود

در طراحی این بخش سعی شده از عملگرها و توابعی استفاده شود که از نظر پیاده‌سازی نرم‌افزاری سرعت مناسبی داشته و همچنین متغیرهای لازم با پردازنده‌های متداول سازگاری داشته باشند.

تابع SD و Sbox: این تابع، ورودی ۳۲ بیتی را به ۳۲ بیت خروجی تبدیل می کند. در آن دو Sbox از اندازه ۸×۸ و یک لایه انتشار از نوع بایتی به‌کار برده شده است. با توجه به اینکه Sboxها به صورت جدول انتخاب مورد استفاده قرار می‌گیرند، بنابراین از لحاظ سرعت پیاده‌سازی مشکلی وجود نخواهد داشت. همچنین لایه انتشار نیز قابلیت پیاده‌سازی بایتی مناسبی را داراست. هدف از این تابع ایجاد اغتشاش روی ورودی تابع و انتشار آن روی بیت‌های خروجی می باشد. بیت‌های خروجی از این تابع درجه جبری حداقل ۷ را داشته و غیر خطینگی بیت‌های خروجی بسیار بالا است. برای مقدار غیر خطینگی یک تابع SD داریم [۸]:

$$NL_{SD} \geq 2^{km-1} - 2^{k-1}\prod_{i=1}^{k}(2^{m-1} - NL_{S_i}) \quad (۱۵)$$

که در آن k تعداد Sboxها و m تعداد بیت‌های ورودی به هر Sbox می‌باشند. بنابراین برای تابع SD که در آن D یک ماتریس بایتی از اندازه ۴×۴ است، مقدار غیر خطینگی برابر $2^{31} - 2^{21} > 2^{30.99}$ می‌باشد که این مقدار با فرض این که مقدار غیرخطی ۱۱۲ برای Sboxها محاسبه شده است.

تابع F1: این تابع به گونه‌ای انتخاب شده است که دو سویه باشد. ورودی تابع دو بیت از ثبات‌ها بوده و خروجی آن یکی از مقادیر متعلق به مجموعه $\{1,2,3,4\}$ می باشد. اگر خروجی را به صورت یک رشته از اعداد صحیح در نظر گرفته شود، به دلیل یک به یک بودن تابع F1 این رشته دارای دوره تناوب $2^{۶۰۷}-۱$ و یا $2^{۶۰۸}-۱$ وابسته به ثبات انتقال اول و یا ثبات انتقال دوم خواهد بود [۹].

معیار طراحی Sbox: در طراحی Sboxها نیاز می‌باشد که یک سری تمهیداتی مد نظر قرار گیرند تا الگوریتم رمزنگاری در برابر حملات رمزشکنی مقاوم باشد؛ زیرا که اصلی‌ترین عناصر غیرخطی الگوریتم Sboxها می‌باشند. ویژگی‌های Sboxهای به‌کار برده شده در ادامه آورده شده است. همچنین علت استفاده از چنین Sbox هایی سادگی پیاده‌سازی و نیاز به مقدار حافظه کمتر برای پیاده‌سازی در حالت جدول انتخاب می‌باشد.

الف- Sbox1 و Sbox2

۱- نداشتن نقطه ثابت و معکوس

۲- حداقل درجه غیرخطی برابر ۱۱۲

۴- حداکثر مقدار تفاضلی در ماتریس تقریب تفاضلی برابر چهار

۵- حداکثر بایاس در تقریب خطی برابر ۱۶

۶- داشتن بیشینه تعداد ضرائب نمایش جبری ۲۵۵

۷- داشتن بیشینه درجه جبری برابر ۷.

معیار طراحی لایه انتشار: لایه انتشار از نوع بایتی است. در طراحی این لایه ویژگی‌های زیر مد نظر قرار گرفته‌اند:

۱- پیاده‌سازی بایتی مناسبی داشته باشد.

۲- هر سطر و ستون شیفت چرخشی سطر و ستون قبلی باشد.

۳- با حداقل حافظه قابلیت پیاده‌سازی داشته باشد.

۴- حداکثر عدد انشعاب برابر ۵ را داشته باشد.

معیار طراحی مولدها: مولدها به گونه‌ای طراحی شده‌اند که دنباله کلید خروجی حداکثر دوره تناوب را برآورده سازند. به همین دلیل مولد دوم دارای دوره تناوب عدد اول است که با این انتخاب دوره تناوب دو مولد نسبت به هم اول خواهند بود. همچنین سعی شده وزن همینگ چند جمله‌ای معادل بیتی آنها نزدیک به نصف طول مولد معادل بیتی باشد تا عمل اغتشاش به خوبی انجام گیرد.

از موارد دیگر که در طراحی مد نظر قرار داشته مقاومت در برابر حمله خطی بیان شده برای SNOW2 بوده و در عین حال پیاده‌سازی نرم‌افزاری آن ساده باشد.

برای مقابله با حمله حدس و تعیین طول ثبات‌های انتقال بزرگ انتخاب شده و محل تپ‌ها به گونه‌ای برگزیده شده که FPD باشند.

با توجه به نوع طراحی و انتخاب اجزای سرعت تولید رشته کلید خروجی با پیاده‌سازی به زبان C در پردازنده ۱/۸ GHz و با رم ۲Gig برابر ۳۰۰Mbit/sec به‌دست آمده است

۴. تحلیل امنیتی الگوریتم

به منظور برآورد امنیتی الگوریتم آن دسته از حملاتی که در مورد الگوریتم‌های خانواده SNOW2 کارآمدی مناسبی داشته‌اند، برای الگوریتم ارائه شده در این مقاله نیز مورد تحلیل قرار گرفته‌اند. در بررسی بعضی از جملات به دلیل ساختار خاص الگوریتم و همچنین محدودیت‌های حافظه مجازی از قبیل RAM شکل الگوریتم و یا ساختار آن کوچک‌سازی شده است. با بررسی این ساختار کوچک شده نتیجه برای الگوریتم اصلی تعمیم داده شده است. البته در صورتی که ساختار کوچک‌سازی شده در برابر حملات امن باشد در آن صورت الگوریتم اصلی نیز امن خواهد بود.

۴-۱. حمله تمایز

حمله تمایز یک روش عمومی می‌باشد که جهت تمیز دادن خروجی یک مولد رشته کلید از دنباله کاملاً تصادفی با طول یکسان مورد کاربرد است. یکی از روش‌های اعمال این حمله روش تشکیل ماسک خطی که برای اولین بار در مرجع [۱۰] به‌کار رفته می‌باشد و در واقع ماسک خطی تمایز دهنده است. در روش ماسک خطی تقریب خطی بیت‌های خروجی مورد توجه بوده تا از این طریق رشته خروجی مولد از دنباله کاملاً تصادفی تمیز داده شود. این حمله در صورتی که حالت داخلی مولد رشته کلید به دو قسمت خطی و غیرخطی قابل تفکیک باشد، اعمال می‌شود. با این روش موفق ترین حمله صورت گرفته روی الگوریتم SNOW2 حمله تمایز با مرتبه 2^{179} بوده است [۱۱]. این حمله روی الگوریتم SNOW3G دیگر قابل اعمال نمی‌باشد [۱۲]. روش دیگری نیز برای اعمال حمله تمایز در [۱۳] بیان شده است. این روش سامانه‌ای بوده و بر مبنای آن مرتبه حمله برابر 2^{202} به دست آمده است.

کاربرد نسخه ساده شده الگوریتم: برای اعمال حمله، ابتدا ساده‌سازی‌های زیر به‌کار برده شده است:

۱- تغییر سامانه از ۳۲ بیت به ۱۶ بیت

۲- استفاده از چندجمله‌ای‌های زیر:

$$\begin{aligned} p(x) &= \alpha x^{19} + x^{17} + x^{13} + \alpha^{-1}x^8 + x + 1 \\ q(x) &= x^{19} + x^{18} + \alpha x^{11} + x^8 + 1 \end{aligned} \quad (۱۶)$$

که در آن، α ریشه چندجمله‌ای $r(x) = x^{16} + x^5 + x^3 + x^2 + 1$ می‌باشد

۳- حذف عملگرهای & و ضرب ماتریسی از توابع بازخورد LFSRها

۴- استفاده از توابع زیر به جای تابع SD

$$F_1(w) = F_2(w) = \begin{pmatrix} x & 1 \\ 1 & x \end{pmatrix} \begin{pmatrix} S_R(w_0) \\ S_R(w_1) \end{pmatrix} \quad (۱۷)$$

جایی که w_1, w_0 به ترتیب هشت بیت کم ارزش و هشت بیت پر ارزش w می‌باشند و S_R مربوط به Sbox استفاده شده در الگوریتم AES می‌باشد.

با توجه به چند جمله‌ای بازخورد معادله تشکیل شده برای ثبات‌های انتقال به صورت زیر خواهد بود:

$$\begin{aligned} s_{t+19} + s_{t+18} + \alpha^{-1}s_{t+11} + s_{t+6} + s_{t+2} + \alpha s_t &= 0 \\ x_{t+19} + x_{t+11} + \alpha x_{t+8} + x_{t+1} + x_t &= 0 \end{aligned} \quad (۱۸)$$

نحوه به روزرسانی حافظه‌ها و تقریب‌های به‌کار برده شده به صورت زیر می‌باشد:

$$\begin{aligned} M_1^{t+1} &= F_1(x_{t+3} + M_2^t + (s_{t+3} \oplus M_3^t)) \\ &= x_{t+3} \oplus M_2^t \oplus s_{t+3} \oplus M_3^t \oplus N_{F_1} \oplus N_{c3} \\ M_2^{t+1} &= s_{t+12} + F_1(x_{t+12} \oplus M_1^t) \\ &= x_{t+12} \oplus s_{t+12} \oplus M_1^t \oplus N_{F_1} \oplus N_{c2} \\ M_3^{t+1} &= x_{t+7} + F_2(s_{t+7} \oplus M_2^t) \\ &= x_{t+7} \oplus s_{t+7} \oplus M_2^t \oplus N_{F_2} \oplus N_{c2} \end{aligned} \quad (۱۹)$$

در اینجا $N_{F_1}(x)$ نویز حاصل از تقریب $F_1(x)$ با x می باشد. با استفاده از خروجی مرحله‌های t، t+1 و t+3 معادلات زیر را خواهیم داشت:

$$\begin{aligned} z_t &= x_t \oplus s_t \oplus F_2(M_2^t \oplus (M_1 + x_{t+18} + s_{t+18})) \\ &= x_t \oplus x_{t+18} \oplus s_t \oplus s_{t+18} \oplus N_{F_2} \oplus N_{c3} \oplus M_1 \oplus M_2 \\ z_{t+1} &= \alpha x_t \oplus \alpha s_{t+8} \oplus \alpha^{-1}x_{t+11} \oplus x_{t+1} \oplus x_{t+2} \oplus x_{t+3} \\ &\quad \oplus x_{t+6} \oplus x_{t+12} \oplus x_{t+18} \oplus s_t \oplus s_{t+3} \oplus s_{t+11} \oplus s_{t+12} \\ &\quad \oplus M_1 \oplus M_2 \oplus M_3 \oplus 2N_{F_1} \oplus N_{F_2} \oplus N_{c2} \oplus 2N_{c3} \\ z_{t+3} &= \alpha x_t \oplus \alpha x_{t+1} \oplus \alpha x_{t+2} \oplus \alpha s_{t+10} \oplus \alpha^{-1}x_{t+11} \\ &\quad \oplus \alpha^{-1}x_{t+12} \oplus \alpha^{-1}x_{t+13} \oplus x_{t+2} \oplus x_{t+3} \oplus x_{t+5} \\ &\quad \oplus x_{t+6} \oplus x_{t+13} \oplus x_{t+14} \oplus x_{t+18} \oplus s_{t+2} \oplus s_{t+3} \\ &\quad \oplus s_{t+4} \oplus s_{t+5} \oplus s_{t+7} \oplus s_{t+8} \oplus s_{t+14} \\ &\quad \oplus M_3 \oplus 7N_{F_1} \oplus 3N_{F_2} \oplus 6N_{c2} \oplus 4N_{c3} \end{aligned}$$

منظور از $7N_{F_1}$، جمع پیمانه‌ای هفت تابع N_{F_1} با آرگومان‌های مختلف است که برای رعایت اختصار این‌گونه نوشته شده است. این کار برای توابع نویز دیگر نیز انجام شده است.

با xor نمودن سه خروجی معادله بالا داریم:

$$\begin{aligned} Z(t) &= z_t \oplus z_{t+1} \oplus z_{t+1} \\ &= \alpha x_{t+1} \oplus \alpha x_{t+2} \oplus \alpha x_{t+8} \oplus \alpha x_{t+10} \\ &\quad \oplus \alpha^{-1}x_{t+12} \oplus \alpha^{-1}x_{t+13} \oplus x_t \oplus x_{t+1} \\ &\quad \oplus x_{t+5} \oplus x_{t+12} \oplus x_{t+13} \oplus x_{t+14} \\ &\quad \oplus x_{t+18} \oplus s_{t+2} \oplus s_{t+4} \oplus s_{t+5} \oplus s_{t+7} \\ &\quad \oplus s_{t+8} \oplus s_{t+11} \oplus s_{t+12} \oplus s_{t+14} \oplus s_{t+18} \\ &\quad \oplus 9N_{F_1} \oplus 5N_{F_2} \oplus 7N_{c2} \oplus N_{c3} \end{aligned} \quad (۲۰)$$

$$N_1 = 9N_{F_1} \oplus 5N_{F_2} \oplus 7N_{c2} \oplus 7N_{c3} \quad (۲۱)$$

$$\begin{aligned} Z'(t) &= \alpha Z(t) \oplus Z(t+2) \oplus Z(t+6) \\ &\quad \oplus \alpha^{-1}Z(t+11) \oplus Z(t+18) \oplus Z(t+19) \\ &= \left(\bigoplus_{i \in I_1} c_i x_{t+i} \right) \oplus \alpha N_1 \oplus \alpha^{-1}N_1 \oplus 4N_1 \end{aligned} \quad (۲۲)$$

$$N_2 = \alpha N_1 \oplus \alpha^{-1}N_1 \oplus 4N_1 \quad (۲۳)$$

$$\begin{aligned} Z''(t) &= Z'(t) \oplus Z'(t+1) \oplus \alpha Z'(t+8) \oplus Z'(t+19) \\ &= \alpha N_2 \oplus 4N_2 \end{aligned} \quad (۲۴)$$

$$N_3 = \alpha N_2 \oplus 4N_2 \quad (۲۵)$$

برای محاسبه توزیع توابع نویز از روابط (۲۱)، (۲۳) و (۲۵) استفاده می‌شود. البته توجه شود که فرض یکنواخت بودن توزیع آرگومان‌ها و مستقل بودنشان از همدیگر در این محاسبات در نظر گرفته شده است.

برای تابع نویز N_2 مقدار اریبی برابر $\varepsilon \approx 0.2171832704 \times 10^{-1359}$ می‌باشد که نشان می‌دهد برای حمله موفق تقریباً به $1/\varepsilon^2 \approx 2^{9033}$ بیت از دنباله خروجی نیاز می‌باشد. این مقدار در مقایسه با حمله

مشابه انجام شده روی SNOW2 بسیار بزرگ‌تر است. بنابراین الگوریتم ارائه شده در این مستند بسیار مقاوم‌تر از الگوریتم SNOW2 در برابر حمله تمایز است.

۴-۲. حمله TMTO و TMDTO

بیرکف و شامیر [۱۴] نشان داده‌اند که اگر D بیت از رشته کلید تولید شده توسط مولد که حالت داخلی آن توسط یک کلید خاص مقداردهی اولیه شده و در طول تولید رشته کلید تغییر نکرده است، می‌توان در زمان T و با اختیار داشتن مقدار حافظه M کلید را بازیابی نمود. مقدار T و M و D در رابطه زیر صدق می‌کند که در آن N همان تعداد بیت‌های حالت داخلی مولد رشته کلید است:

$$TM^2D^2 = N^2 \quad , \; D^2 \le T \le N \qquad (۲۶)$$

همچنین این حمله نیاز به فضای پیش پردازشی با پیچیدگی $P = N/D$ دارد. بنابراین با توجه به طرح مقدار N برابر طول رجیسترها و فضای حافظه‌ها در FSM می‌باشد. از طرفی با فرض اینکه در هر بار تعویض کلید مقدار $D = 2^{100}$ بیت رشته کلید تولید گردد خواهیم داشت:

$$N = 2^{607+608+32+32+32} = 2^{1311}$$
$$D = 2^{100}$$
$$\Rightarrow T = M = 2^{807}$$

مشخص است که این حمله با طول رشته کلید حداکثر برابر $D = 2^{100}$ به هیچ عنوان قابل اعمال نخواهد بود. برای اینکه این حمله دارای پیچیدگی برابر طول بیت کلید گردد لازم است که حمله کننده حداقل به اندازه $D = 2^{511}$ بیت رشته کلید تولید شده به ازای یک کلید خاص در اختیار داشته باشد.

۴-۳. حمله جبری

در حالت کلی برای به دست آوردن حالت داخلی یک رمزکننده جریانی که فضای حالت آن N بیتی می‌باشد لازم است جستجوی کامل تمامی 2^N مقدار ممکن انجام گیرد. حملات جبری این پیچیدگی را با استفاده از یک سری روابط جبری و استفاده از تکنیک‌های خطی‌سازی کاهش می‌دهند. در سال ۲۰۰۳ یک روش جهت حمله به رمزکننده‌های جریانی بر مبنای روش‌های جبری ارائه شد [۱۵ و ۱۶]. طبق آن اگر یک رمزکننده جریانی با n بیت ثبات و یک بخش غیرخطی شامل L بیت حافظه، k بیت ورودی و m بیت خروجی وجود داشته باشد آنگاه باند بالایی پیچیدگی حملات جبری به صورت زیر به دست می‌آید:

۱- M و d را با استفاده از روابط زیر می‌توان محاسبه کرد:

$$M = \left\lceil \frac{L+1}{m} \right\rceil \qquad (۲۷)$$

$$d = \left\lceil \frac{Mk}{2} \right\rceil \qquad (۲۸)$$

۲- کوچک‌ترین مقدار ممکن را برای d با استفاده از رابطه زیر به دست آورید و آن را $d_{\min}$ بنامید:

$$2^{Mm} \sum_{i=0}^{d} (C_i^{Mk}) > 2^{Mk+L} \qquad (۲۹)$$

۳- باند بالایی برای پیچیدگی زمانی (T) و پیچیدگی دیتا (D) حمله جبری به صورت زیر به دست می‌آید:

$$T = 2^{\omega(Mk+L)} + (C_{d_{\min}}^{n})^{\omega} \qquad (۳۰)$$

$$D = (C_{d_{\min}}^{n})^{\omega} \qquad (۳۱)$$

در روابط بالا ، C_d^n ترکیب d از n می‌باشد و ω به مقدار کمتر از ۲/۳۷۶ تنظیم می‌شود. مطابق روابط بالا پیچیدگی زمانی و دیتای حملات جبری روی طرح ارائه شده به همراه طرح SNOW2 جهت مقایسه در جدول (۱) آورده شده‌اند.

روش بیان شده در فوق یک برآورد از باند بالایی حمله جبری است که این احتمال وجود دارد که یک حمله سریع‌تر با استفاده از روش‌های ابتکاری قابل اجرا باشد. به عنوان مثال در مرجع [۱۷] یک حمله جبری روی الگوریتم ساده شده SNOW2 بیان شده است که دارای پیچیدگی $2^{۵۰}$ می باشد و در آنجا تأکید شده است که هیچ روش مؤثری برای حل دستگاه معادلات جبری به دست آمده برای الگوریتم اصلی، با پیچیدگی کمتر از جستجوی کامل فضای کلید وجود ندارد. بنابراین با توجه به افزایش تعداد حافظه‌ها و تعداد جعبه‌های جانشینی در تابع غیر خطی، اجرای حمله جبری روی الگوریتم این مقاله بسیار مشکل‌تر از اجرای حمله جبری روی الگوریتم SNOW2 می‌باشد که جدول (۱) نیز بیان کننده این موضوع می‌باشد.

جدول ۱. مقایسه الگوریتم طراحی شده با SNOW2 از لحاظ حمله جبری

	پارامترها				مرتبه حمله	
	N	L	K	M	T	D
SNOW2	۵۱۲	۶۴	۹۶	۳۲	۲۸۳۶	۲۳۴۳
الگوریتم این مقاله	۱۲۱۵	۹۶	۳۲۰	۳۲	۲۳۲۶۹	۲۲۸۳۹

۴-۴. حمله همبستگی

حمله همبستگی از مهم‌ترین حملات عمومی روی رمزهای جریانی است. برای اینکه یک حمله همبستگی قابل اعمال باشد باید رشته کلید z1, z2, ... با دنباله خروجی s1, s2, ... یک LFSR همبسته باشد. دو دنباله همبسته هستند اگر احتمال $P(z_i = s_i) \neq 1/2$ باشد. در این صورت ممکن است حالت اولیه ثبات انتقال قابل بازیابی

باشد. یکی از حملات همبستگی که بسیار خوب عمل کرده، روش ارائه شده در مرجع [۱۸] می‌باشد. به منظور اعمال این حمله همبستگی لازم است ابتدا احتمال همبستگی بین حالت‌های ثبات انتقال مولد و رشته کلید خروجی محاسبه شود. این احتمال برای اعمال حمله همبستگی سریع‌تر از جستجوی جامع فضای کلید مهم می‌باشد. در این مرجع دو روش برای اعمال حمله بیان شده که از نظر نویسنده روش بیان شده بسیار ساده و در عین حال مؤثر و کارآمد می‌باشد.

طبق روش مرجع [۱۸] با فرض L به عنوان طول ثبات انتقال مولد، k و t به عنوان پارامترهای الگوریتم، اگر احتمال همبستگی برابر $1/2+\varepsilon$ باشد آنگاه طول رشته کلید خروجی مورد نیاز برای موفقیت حمله به صورت زیر به دست می‌آید:

$$N=\frac{1}{4}\cdot(2kt!\ln 2)^{1/t}\cdot\varepsilon^{-2}\cdot 2^{\frac{L-k}{t}} \qquad (۳۲)$$

حمله‌کننده یک تعداد معادلات بررسی توازن در مرحله پیش پردازش جمع‌آوری نماید. پیچیدگی این مرحله تقریباً برابر $N^{\lceil (t-1)/2\rceil}$ بوده و نیازمند $N^{\lceil (t-1)/2\rceil}$ حافظه می‌باشد. مقدار حافظه مورد نیاز با افزایش پیچیدگی محاسباتی می‌تواند کاهش داده شود. همچنین تعداد معادلات بررسی توازن که در مرحله کدگشایی لازم است ذخیره گردد، برابر $(N^t/t!)\cdot 2^{-(L-k)}$ می‌باشد. پیچیدگی مرحله کدگشایی نیز 2^k برابر تعداد معادلات بررسی توازن خواهد بود.

برای الگوریتم مورد نظر این مستند با فرض کلاک منظم و حذف ثبات دوم و با فرض مقدار حداکثر $N=2^{50}$ جدول (۲) همبستگی مورد نیاز برای اعمال حمله سریع‌تر از جستجوی جامع فضای کلید را نشان می‌دهد.

برای محاسبه پیچیدگی حمله لازم است که حداقل پیچیدگی کلید در نظر گرفته شود. برای مثال اگر فرض کنیم کلید برابر ۲۵۶ بیت باشد حداکثر پیچیدگی حمله از مرتبه ۲۵۵ گردد. با این مقدار و با استفاده از اینکه حداکثر تعداد رشته بیت موجود در دسترس چقدر می‌باشد (معمولاً 2^{50} بیت) با استفاده از رابطه زیر مقدار K محاسبه می‌شود و در نتیجه پیچیدگی حمله به دست می‌آید.

$$\begin{aligned}&2^k\times(N^t/t!)\times 2^{-(L-k)}=2^{KEY-1}\\&2^{2k-L+(50\times t)}=t!2^{KEY-1}\end{aligned} \qquad (۳۳)$$

اگر تعداد بیت‌های !t برابر T باشد آنگاه داریم:

$$\begin{aligned}&2^{2k-L+(50\times t)}=2^{KEY+T-1}\\&2k-L+(50t)=KEY+T-1\\&k=\frac{L+KEY+T-50t-1}{2},\ T=\log 2(t!)\end{aligned} \qquad (۳۴)$$

همان‌طوری که از جدول مشخص است برای رسیدن به یک اربیی قابل قبول برای اجرای حمله همبستگی، پیچیدگی داده برابر 2^{450} بیت و زمان مورد نیاز برای اجرای حمله برابر 2^{500} است که دلیل امن‌بودن الگوریتم در برابر این‌گونه حملات می‌باشد.

جدول ۲. پیچیدگی حمله همبستگی

همبستگی لازم	تعداد پریتی چک‌ها	پیچیدگی داده	پیچیدگی زمان	مقدار t
–	۰	۰	۵۰	۲
–	۰	۵۰	۵۰	۳
⋮	⋮	⋮	⋮	⋮
(k=۱۹۳) ε=۰/۰۷۳۹۵	۲۶۴	۲۰۰	۲۵۰	۱۰
⋮	⋮	⋮	⋮	⋮
(k=۸) ε=۰/۰۰۴۵۴۹	۲۲۴۸	۴۵۰	۵۰۰	۱۸

۴-۵. حمله حدس و تعیین

در این حمله هدف حدس تعداد کمی از متغیرهای مجهول در رمز است که از روی این مقادیر حدس زده‌شده، مقادیر متغیرهای مجهول را تعیین می‌کنند. این حمله اغلب سریع‌تر از جستجوی کامل کلید به دلیل ویژگی غیرخطی در رمز، نمی‌باشد. به‌خاطر این مساله برای انجام این حمله فرض می‌شود که رمز خطی است. اگر این فرض با احتمال p درست باشد آنگاه تعداد حدس و تلاش‌های لازم برای موفقیت حمله برابر 1/p می‌باشد. بعد از تعیین مقادیر مجهول، فرآیند رمزنگاری تکرار و خروجی الگوریتم با خروجی واقعی مقایسه می‌شود. این حمله دارای سه مرحله به‌صورت زیر است:

۱- بعضی از بخش‌های کلید یا حالت اولیه حدس زده می‌شود.

۲- دیگر بخش‌های کلید یا حالت اولیه تحت بعضی از فرض‌ها به دست می‌آید. فرض بر این است که زوج (کلید/IV) بعضی از زیر مجموعه‌ها از کل مجموعه‌ای که رمز را تشکیل می‌دهند ضعیف می‌باشد.

۳- با محاسبه رشته کلید از مقادیر تعیین شده و مقایسه با کلید مجهول می‌توان چک کرد که حدس درست بوده یا نه و فرض برقرار است یا نه. این حمله موفق خواهد بود اگر $2^g\cdot(1/p)\cdot w<2^k$ باشد که در آن g تعداد بیت‌های حدس زده شده است. w نیز کار لازم جهت تعیین مجهولات باقی مانده است اگر حدس درست بوده و فرض برقرار باشد. k نیز طول کلید اصلی بوده که برابر ۲۵۶ بیت است. برای طرح مذکور با فرض حذف ثبات انتقال دوم و فرض کلاک منظم، حمله‌کننده قبل از هر عملی لازم است ورودی تابع FSM و مقدار حافظه‌ها را حدس بزند تا بتواند اولین معادله را به ازای اولین خروجی داشته باشد. این حدس برابر تعداد ۲۲۴ = (۳×۳۲) + ۱۲۸ بیت می‌باشد (سه حافظه ۳۲ بیتی به همراه چهار مقدار ۳۲ بیتی ورودی از ثبات انتقال). به خوبی مشخص می‌باشد برای تکمیل این معادله لازم است مقادیر سلول اول ثبات انتقال خطی اول حدس زده شود. بنابراین حمله کننده در همین مرحله ناتوان از حمله با

پیچیدگی کمتر از طول کلید (۲۵۶ بیت) می‌باشد. دلیل عدم موفقیت حمله حدس و تعیین می‌توان از یک طرف به انتخاب محل بازخوردها در محاسبه حالت بعدی ثبات‌های انتقال خطی استناد نمود که بر مبنای تفاضل مثبت کامل (FPD) است. (مرجع [۱۹] با بیان علت ضعف SNOW در برابر حمله حدس و تعیین به این نتیجه رسیده است) و از طرفی دیگر استفاده از تابع FSM با سه حافظه و چهار ورودی سی و دو بیتی باعث شده بدون حدس ورودی‌های آن نتوان معادله‌ای بین سلول‌ها و خروجی رشته کلید بیان نمود. همچنین این حمله روی SNOW2 دیگر موفق نبوده است [۲۰].

۵. ارزیابی آماری الگوریتم

اگر یک الگوریتم رمز جریانی حداقل یکی از آزمون‌های آماری مطرح را با موفقیت نگذراند به این معنی است که ضعف ساختاری دارد و خروجی‌های آن رفتار آماری مطلوب مطابق با توزیع یکنواخت ندارد. چنانچه یک الگوریتم رمز جریانی تمام آزمون‌های آماری موردنظر را با موفقیت بگذراند در حقیقت طراحی آن دارای استحکام ساختاری است و خروجی‌های آن مشخصه‌های آماری مطلوب را دارد. جهت اطمینان از امن بودن و استحکام ساختاری الگوریتم رمز از نظر تصادفی بودن خروجی، لازم است پس از طراحی، آن را مورد ارزیابی آماری قرار داد. در این ارزیابی هدف بررسی بی‌قاعدگی رشته کلیدهای تولیدی از نظر رفتار آماری است که این امر با انجام پانزده آزمون مختلف استاندارد NIST [۲۱] بر روی ۴ نوع داده مختلف بررسی می‌شود و نتایج آن به صورت قابل قبول و یا غیرقابل قبول ارائه می‌گردد. این آزمون‌ها به ازای چهار نوع داده ۱- بهمنی کلید، ۲- کلید کم چگال، ۳- کلید پر چگال، ۴- کلید تصادفی اجرا می‌شوند. در جدول (۳) نتایج حاصل از ارزیابی آماری به ازای چهار نوع داده ذکر شده طبق آزمون‌های استاندارد NIST با سطح معنی‌داری یک درصد آمده است.

جدول ۳. نتایج آزمون‌های آماری *NIST* برای الگوریتم جدید

Test Name	p-value	Result
Frequency	0.623	Pass
Frequency within a Block	0.701	Pass
Runs	0.665	Pass
Longest Run	0.812	Pass
FFT	0.496	Pass
Binary Matrix Rank	0.699	Pass
Non Overlapping Template Matching	0.958	Pass
OverlappingTemplate Matching	0.042	Pass
Universal	0.817	Pass
Linear Complexity	0.993	Pass
Serial	0.762	Pass
Approximate Entropy	0.741	Pass
Cumulative Sums Forward	0.663	Pass
Cumulative Sums Backward	0.673	Pass
Random Excursions	0.698	Pass
Random Excursions Variant	0.945	Pass

۶. نتیجه‌گیری

در این مقاله یک الگوریتم جریانی جدید بومی طراحی و تحلیل نمودیم که اصول کلی آن مبتنی بر ثبات‌های کلمه‌ای جدید استوار بوده و در حالت کلاک کنترلی مورد استفاده قرار گرفته‌اند. تحلیل‌های امنیتی نشان دادند که این الگوریتم در برابر حملات شناخته شده معروف، نسبت به الگوریتم‌های مشابه بسیار مقاوم‌تر می‌باشد. این سطح از امنیت به‌دلیل استفاده از ثبات‌های مبتنی بر کلمه بهبودیافته در حالت کلاک کنترلی به همراه استفاده از صافی غیرخطی شبیه الگوریتم‌های قالبی حاصل گردیده است. همچنین با توجه به انجام تحلیل محاسباتی در خصوص مرتبه پیچیدگی الگوریتم؛ مرتبه حمله تمایز برابر $2^{۹۰۳۳}$ و مرتبه حمله جبری برابر $2^{۲۸۳۹}$ به دست آمده است. مرتبه پیچیدگی حمله جبری الگوریتم این مقاله در مقایسه با الگوریتم SNOW2 بسیار بزرگ بوده و از آنجا که تا به حال هیچ حمله جبری موفق روی SNOW2 اجرا نشده، بنابراین این حمله روی الگوریتم جدید پیشنهادی در این مقاله هم قابل اجرا نیست.

۷. تشکر و قدردانی

نویسندگان از دانشگاه آزاد اسلامی واحد مشگین شهر برای حمایت از این اثر، کمال تشکر و قدردانی را دارند.

۸. مراجع

[1] Preneel, B. “Introduction to the Proceeding of the Fast Software Encryption”; 1994 Workshop, LNCS 1008, 1995.

[2] Ekdahl, P.; Johansson, T. “SNOW - A New Stream Cipher”; Proceedings of First Open NESSIE Workshop, 2000.

[3] Ekdahl, P.; Johansson, T. “A New Version of the Stream Cipher SNOW”; In Selected Areas in Cryptography, SAC 2002, LNCS 1233, Springer-Verlag, Pages 37–46, 2002.

[4] Final Report European Project Number IST-1999-12324, “NESSIE: New European Schemes for Signatures Integrity and Encryption”; Available at: https://www.cosic.esat.kuleuven.be./nessie/bookv015.pdf.

[5] Filiol, E. “Decimation Attack of Stream Ciphers”; Rapport de recherche n° 3990, Aout 2000.

[6] Guang, Z.; KaiCheng, H.; WenBao, H. “A Trinomial Type of σ - LFSR Oriented Toward Software Implementation”; Science in China Series F: Information Sciences, 2007.

[7] Robshaw, M.; Billet, O. “New Stream Cipher Designs. The E Stream Finalists”; Springer-Verlag Berlin Heidelberg 2008, pp. 98-118.

[8] Dawson, E. “The LILI-128 Key Stream Generator”; Information Security Research Centre, Queensland University of Technology, 2002.

[9] Kontak, M. “Nonlinearity of Round Function”; Institute of Mathematics and Cryptology, Poland, 2006.

[10] Coppersmith, D. “Cryptanalysis of Stream Ciphers with Linear Masking"; IBM T. J. Watson Research Center, NY, USA, 2002.

[11] Nyberg, K.; Wall, J. “Improved Linear Distinguishers for SNOW 2.0”; Fast Software Encryption, Lecture Notes in Computer Science 2006, 4047, 144-162

[12] ETSI/SAGE "Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2. Document 5: Design and Evaluation Report"; Version: 1.0 http://www.3gpp.org/ftp/tsg_sa/WG3_Security/TSGS3_42_Bangalore/Docs/S3-06018%0.zip.

[13] Maximov, A.; Johansson, T. "Fast Computation of Large Distributions and Its Cryptographic Applications"; In Asiacrypt 2005, LNCS 3788, Springer-Verlag, 2005, pp. 313–332,.

[14] Biryukov, A.; Shamir, A. "Cryptanalytic Time/Memory/Data Tradeoffs for Stream Ciphers"; Asiacrypt 2000, LNCS Vol. 1976, pp. 1-13.

[15] Courtois, N. T. "Algebraic Attacks on Stream Ciphers with Memory and Several Outputs"; Eurocrypt 2003, LNCS Vol. 2656, pp. 345-359.

[16] Courtois, N. T. "Algebraic Attacks on Combiners with Memory and Several Outputs"; Extended Version of the Paper Published in ICISC 2004.

[17] Billet, O.; Gilbert, H. "Resistance of SNOW 2.0 Against Algebraic Attacks"; CT-RSA 2005, 2004, pp. 19–28.

[18] Chepyzhov, V.; Johansson, T.; Smeets, B. "A Simple Algorithm for Fast Correlation Attacks on Stream Ciphers"; Fast Software Encryption, FSE'2000, to Appear in Lecture Notes in Computer Science, Springer-Verlag, 2000.

[19] Hawkes, P.; Rose, G. "Guess-and-Determine Attacks on SNOW"; In: Nyberg, K., Heys, H. (eds.) SAC 2002, LNCS Vol. 2595, pp. 37–46.

[20] Ahmadi, H.; Eghlidos, T. "Advanced Guess and Determine Attacks on Stream Ciphers"; IST 2005, pp. 87-91.

[21] Rukhin, A.; Soto, J.; Nechvatal, J.; . Smid, M.; . Barker, E.; Leigh, S.; Leven-son, M.; Vangel, M.; Banks, D.; Heckert, A.; Dray, J.; Voo, S. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications"; NIST Special Publication 800-22, Revision 1a, 2010.