استفاده از ضریب پرش در توابع تی برای طراحی یک ساختار اولیه نوین در رمزهای جریانی

سید مهدی سجادیه[1]، علی هادی پور[2*] و راحله مرادعفیفی[3]

[1] استادیار، گروه آموزشی برق و کامپیوتر، دانشگاه آزاد واحد خوراسگان، خوراسگان، ایران
m.sajadieh@khuisf.ac.ir
[2] کارشناس ارشد ریاضی-رمز، گروه تحقیقات رمز خانه ریاضیات اصفهان، اصفهان، ایران
shakhesyar@chmail.ir
[3] مربی، گروه آموزشی فنی-مهندسی، موسسه غیرانتفاعی آل طه، تهران، ایران
rahele_4281@yahoo.com

چکیده

رمزهای جریانی دسته‌ای از الگوریتم‌های رمز متقارن هستند، که پیام محرمانه را به‌صورت دنباله‌ای از بیت‌ها دریافت کرده و عملیات رمز را با استفاده از تابعی پیچیده بر حسب کلید و IV و ترکیب XOR با دنباله بیت‌ها انجام می‌دهد. یکی از اهداف در طراحی رمزهای جریانی، به‌دست‌آوردن حداقل دوره تناوب بزرگ بوده که یکی از توابع اولیه استفاده از توابع تی است. از طرفی استفاده از ضریب پرش در طراحی LFSRها باعث پیچیده‌ترشدن تحلیل رمزهای جریانی مبتنی بر LFSRها شده است. در این مقاله سعی شده، با استفاده از مفاهیم توابع تی و به‌کارگیری ضرایب پرش، روش جدیدی برای طراحی توابع اولیه رمزهای جریانی با دوره تناوب بالا ارائه شود.

واژگان کلیدی: توابع تی، ضریب پرش، رمزهای جریانی، دوره تناوب بیشینه.

۱- مقدّمه

رمزهای جریانی کاربردهای فراوانی در بسترهای سخت‌افزاری و نرافزاری دارند و به‌طورکلی روش‌های بسیاری برای طراحی رمزکننده‌های جریانی وجود دارد. یک روش معمول، به‌کار بردن یک رمزکنندهٔ قالبی در حالت‌های بازگشتی مانند سبک عملکرد OFB است. بسیاری از رمزکننده‌های جریانی بر اساس ثبات‌های انتقال پیشنهاد شده‌اند که به دو صورت، با پس‌خورد غیرخطی (NLFSR) و ثبات انتقال با پسخورد خطی (LFSR) ساخته می‌شوند. این در حالی است که مولدهای مبتنی بر LFSR، دارای خواص آماری مناسب و پیاده‌سازی سخت‌افزاری آسانی هستند. رمزکننده‌های جریانی مبتنی بر LFSR دارای اصول طراحی متفاوتی هستند که می‌توان برای به‌دست‌آوردن رمزکننده‌های پیچیده‌تر و ایمن‌تر، ترکیبی از آن‌ها را به‌کار برد، به‌عنوان مثال با به‌کاربردن توابع غیرخطی بر روی ثبات‌های انتقال، پیچیدگی خطی بسیار بالاتر می‌رود. از طرفی با به‌کاربردن کلاک روی *LFSR*های یک رمز جریانی، می‌توان دنباله کلید را نیز غیرخطی کرده و پیچیدگی دنباله را به‌مراتب بیشتر کرد. مزیت به‌کاربردن کلاک نامنظم، علاوه‌بر این‌که باعث پیچیدگی خطی بالاتر می‌شوند، در مقابل حملات همبستگی مقاوم هستند؛ ولی در مقابل حملات کانال جانبی مقاوم نیستند؛ به‌همین دلیل از سال ۲۰۰۴ به بعد جانسن [1] سعی کرد راهی بیابد که علاوه‌بر داشتن مزیت کلاک نامنظم، مشکل آن را حل کند که در همین راستا مسئله پرش[1] را عنوان کرد و باعث شد رمزهای جریانی در مقابل حملات کانال جانبی نیز مقاوم باشند.

[1] Jumping

* نویسندهٔ عهده‌دار مکاتبات

[Downloaded from monadi.isc.org.ir on 2024-03-28]

خاصیت مهمی که در رمزکننده‌های جریانی وجود دارد، دوره تناوب بالای آن‌ها بوده که در بهترین حالت برای یک دنباله L بیتی دوره تناوب $2^L - 1$ به‌دست می‌آید. بنابراین برای تولید دنباله‌هایی با دوره تناوب‌های بیشتر به بررسی تابع تی[1] پرداخته که حداکثر دوره تناوب را دارد. از برتری‌های دیگر تابع تی آن است که عملیات معکوس آن بسیار سخت بوده و این برتری، مهم‌ترین برتری توابع تی نسبت به LFSRها است.

در این مقاله سعی شده با حفظ دوره تناوب بالا، به نحوی پلی بین پرش و تابع تی برقرار شود که دارای پیچیدگی مناسبی نیز باشد. از این‌رو بخش ۲، پس از بیان مباحث اولیه در رمزنگاری، به‌تفصیل به بررسی توابع تی و تعاریف آن پرداخته شده و در بخش ۳ مباحث مربوط به پرش، عنوان شده است. در بخش ۴ با ارائه نگاشتی از یک تابع تی، ارتباط بین آن و پرش را مطرح کرده و کران پایینی برای دوره تناوب ارایه می‌شود. در بخش ۵ نیز نتیجه‌گیری از مقاله ارایه شده است.

۲- توابع تی

هرم ابزارهای رمزنگاری را می‌توان به این صورت در نظرگرفت که پایین‌ترین سطح را اجزای سازندهٔ الگوریتم‌های رمزنگاری تشکیل می‌دهند، که از آن به‌عنوان Primitive یاد می‌شود. سطح بالاتر آن الگوریتم‌های رمزنگاری هستند، مانند الگوریتم رمز RSA و AES. بالاترین سطح از هرم را پروتکل‌های رمزنگاری ، مانند پروتکل ارسال یک پیام خصوصی از شخصی به شخص دیگر از طریق کانال ارتباطی مورد نظر تشکیل می‌دهند. به عبارت دیگر با ترکیبی از اجزای اولیه، الگوریتم‌های رمزنگاری طراحی شده و با استفاده از الگوریتم‌های رمزنگاری، پروتکل‌های رمزنگاری تولید می‌شوند. می‌توان گفت دو رویکرد اصلی برای طراحی الگوریتم‌های رمزنگاری وجود دارد؛ در رویکرد نخست سعی شده تنها اجزای اولیه‌های ساده (مانند LFSRها در رمزهای دنباله‌ای، و S-Boxها و P-Boxها در رمزهای قالبی و توابع درهم‌ساز) را با درک فهم خوبی به کار برده و همچنین قضایای ریاضی در رابطه با خواص رمزنگاری آن‌ها ثابت شود و در رویکرد دیگر نیز ترکیبی از عملیات مورد استفاده قرار داده شده است، به‌طوری‌که تنوعی از روش‌های غیرجبری و غیرخطی آمیخته می‌شوند؛ امید به این‌که نه طراح و نه حمله‌کننده قادر نباشند رفتار ریاضی طرح را تحلیل کنند. علاوه بر این‌که اثبات‌هایی نیز برای امنیت آن‌ها وجود دارد. لازم به ذکر است که طراحی الگوریتم‌های رمز با کلید مخفی اغلب از رویکرد دوم استفاده می‌شود. در این بخش دسته‌ای از توابع تی تعریف شده که شامل ترکیب‌های دلخواهی از عملیات جمع، تفریق، ضرب، or، and و جمع دودویی روی کلمات n بیتی هستند.

توابع تی نخستین بار توسط الکساندر کلیموف[2] تحت نظارت ادی شامیر[3] در سال ۲۰۰۲ با محوریت "طبقهٔ جدیدی از نگاشت‌های وارون‌پذیر" [6] معرفی شد. با توجه به تعریف این توابع، می‌توان آن‌ها را در طراحی اجزای اولیه رمزهای متقارن استفاده کرد. از جمله طرح‌هایی که مبتنی بر توابع تی بوده می‌توان به ABC (در سال ۲۰۰۵ توسط والدیمیر آناشین[4] و همکارانش) [7,12] و TSC (در سال ۲۰۰۵ توسط هانگ[5] و همکارانش) در نسخه‌های TSC-1، TSC-2 و TSC-3 [8] که خانواده‌ای از توابع تی مبتنی بر SBox است، اشاره کرد. Mir-1 (ساختاری مبتنی بر T-function و استفاده از SBox) [13] و Vest (ساختاری مبتنی بر NLFSR، SPN و T-function) [14] نیز از دیگر طرح‌هایی است که مبتنی بر توابع تی ارایه شد. به‌منظور آشنایی بیشتر، در بخش ۲-۱ خلاصه‌ای از الگوریتم‌های مذکور تشریح می‌شود.

فرض کنید $B^n = \{(x_{n-1},\dots,x_0) \mid x_i \in B\}$ یک مجموعه n تایی از عناصر $B = \{0,1\}$ باشند. در این صورت یک عنصر B یک بیت و یک عنصر B^n یک کلمه n بیتی نامیده می‌شود، به‌طوری‌که یک عنصر x از B^n به‌صورت $([x]_{n-1},[x]_{n-2},\dots,[x]_0)$ نمایش داده شده و هر بیت $[x]_{i-1}$، بیت شماره iام از کلمه x نامیده می‌شود. بنابراین $[x]_0$ را کم‌ارزش‌ترین بیت x و $[x]_{n-1}$ را پر ارزش‌ترین بیت x گویند. همان‌طور که بیان شد کلمه x برای نمایش بردار n بیتی $([x]_{n-1},\dots,[x]_0) \in B^n$ بیان می‌شود که می‌توان با استفاده از تابع تبدیل $x \leftrightarrow \sum_{i=0}^{n-1} 2^i [x]_i$ به پیمانه 2^n به این نمایش دست یافت.

[1] T-Function

[2] Alexander Klimov
[3] Adi Shamir
[4] Vladimir Anashin
[5] Hong

[Downloaded from monadi.isc.org.ir on 2024-03-28]

تعریف ۱. تابع f از $B^{m\times n}$ به $B^{l\times n}$ یک تابع تی نامیده می‌شود، اگر iامین ستون از خروجی $[f(x)]_{i-1}$ تنها به i ستون نخست از ورودی‌های $[x]_0,\ldots,[x]_{i-1}$ بستگی داشته باشد، به‌عبارت دیگر:

$$\begin{pmatrix}[x]_0\\[x]_1\\[x]_2\\M\\[x]_{n-1}\end{pmatrix}^T \rightarrow \begin{pmatrix}f_0([x]_0)\\f_1([x]_0,[x]_1)\\f_2([x]_0,[x]_1,[x]_2)\\M\\f_{n-1}([x]_0,\ldots,[x]_{n-2},[x]_{n-1})\end{pmatrix}^T$$

و به بیان دیگر هر بیت i از خروجی‌ها برای $0\le i<n$ می‌تواند تنها به بیت‌های $0,\ldots,i$ از ورودی‌ها وابسته باشد. درواقع نگاشت m کلمه به l کلمه است که هر کدام n بیتی‌اند و هر بیت خروجی به خودش و بیت‌های قبلی وابسته است[2]. در تعریف بالا، نگاشت یک کلمه n بیتی به یک کلمه n بیتی منظور شده است.

مثال ۱. $x+1$ یک تابع تی است؛ زیرا:

$$(x_2x_1\oplus x_0, x_1\oplus x_0, x_0\oplus 1)+(001)=(x_2x_1x_0).$$

همان طور که مشخص است، بیت نخست x_0+1 بوده که کم‌ارزش‌ترین بیت می‌باشد. بیت دوم به حاصل جمع بیت اول وابسته است؛ لذا x_1 فقط با x_0 که مجهول است، جمع می‌شود و همچنین برای بیت سوم، x_2 با ترکیب x_0 و x_1 جمع دودویی می‌شود. نمایش تعریف ۱ را برای این مثال می‌توان در زیر مشاهده کرد.

$$\begin{pmatrix}[x]_0\\[x]_1\\[x]_2\\M\end{pmatrix}^T \rightarrow \begin{pmatrix}[x_0]_0\oplus 1 & \oplus[x_1]_0\\ [x_0]_1\oplus[x_0]_0 & \oplus[x_1]_1\\ [x_0]_2\oplus[x_0]_0[x_0]_1 & \oplus[x_1]_2\\ & M\end{pmatrix}^T$$

مثال ۲. $x\oplus x^2$ یک تابع تی است. این مثال برای $x=(x_2x_1x_0)$ و $n=3$ بررسی می‌شود.

$$\begin{aligned}x\oplus x^2&=(x_2x_1x_0)\oplus(x_2x_1x_0)^2\\&=(x_2x_1x_0)\oplus(x_1\oplus x_1x_0\ \ 0\ \ x_0)\\&=(x_1\oplus x_1x_0\oplus x_2x_1\oplus x_0\ \ 0)\end{aligned}$$

به‌طوری‌که $(x_2x_1x_0)^2$ به‌صورت زیر محاسبه می‌شود و همان طور که مشخص است هر کدام از بیت‌های حاصل با احتساب بیت نقلی به بیت‌های قبلی بستگی دارد.

$$\begin{aligned}&(x_2x_1x_0)\\ \times\ &\frac{(x_2x_1x_0)\ mod\ 2^3}{x_0x_2x_0x_1x_0)}\\ \oplus\ &x_2x_1\quad x_1\quad x_1x_0\\ \oplus\ &\frac{(x_2\quad x_2x_1x_2x_0}{(x_1\oplus x_0x_1\quad 0\quad x_0)\ mod\ 2^3}\end{aligned}$$

مثال ۳. با توجه به دو مثال بالا، می‌توان از نگاشت‌های $x\rightarrow x\wedge x^2$ و $x\rightarrow\left(x\oplus x^2\right)+\left(x\wedge(3x\ \ 5)\vee(x-1)\right)$ به‌عنوان یک تابع تی نام برد. لازم به ذکر است، علامات $\wedge$، $\vee$ و (<<) به‌ترتیب نشان‌دهندهٔ عطف بیتی، یای منطقی و انتقال به چپ بیتی هستند. باید خاطر نشان کرد که انتقال بیتی به سمت راست یعنی $\supseteq$ (>>) خاصیت تابع تی ندارد.

تعریف ۲. یک نگاشت $\phi:B^k\rightarrow B^k$ معکوس‌پذیر است اگر داشته باشیم $\phi(x)=\phi(y)$ اگر و تنها اگر $x=y$. بنابراین باید بررسی شود که یک تابع تی داده شده، معکوس‌پذیر است یا نه. در ادامه به قضایایی در ارتباط با نگاشت‌های معکوس‌پذیر پرداخته می‌شود.

مثال ۴. نگاشت‌های یک متغیره $x\rightarrow x+2x^2$، $x\rightarrow x+(x^2\vee 1)$، $x\rightarrow x\oplus(x^2\vee 1)$ برای هر اندازه کلمه معکوس‌پذیرند، ولی نگاشت‌های $x\rightarrow x+x^2$، $x\rightarrow x+(x^2\wedge 1)$، $x\rightarrow x+(x^3\vee 1)$ معکوس‌ناپذیرند. برای نمونه معکوس‌پذیری نگاشت $x\rightarrow x+(x^2\vee 1)$ و معکوس‌ناپذیری نگاشت $x\rightarrow x+(x^2\wedge 1)$ بررسی می‌شود. معکوس‌پذیری و معکوس‌ناپذیری بقیه نگاشت‌ها به روش مشابه ثابت می‌شوند. در حالت یک بیتی برای نگاشت $x\rightarrow x+(x^2\vee 1)$ نتیجه می‌شود:

$$\forall x\in\{0,1\}: x^2\vee 1=1, \forall y\in\{0,1\}: y^2\vee 1=1\Rightarrow$$
$$x+\left(x^2\vee 1\right)=y+\left(y^2\vee 1\right)\Rightarrow x+1=y+1\Rightarrow x=y$$

با توجه به تعریف ۲ به روشنی نگاشت بالا معکوس‌پذیر است. برای نگاشت $x\rightarrow x+(x^2\wedge 1)$ در حالت یک بیتی نیز می‌توان بررسی کرد که:

[Downloaded from monadi.isc.org.ir on 2024-03-28]

$\forall x \in \{0,1\} : x = x^2 \wedge 1 \in \{0,1\}$,
$\forall y \in \{0,1\} : y = y^2 \wedge 1 \in \{0,1\}$:
$x + (x^2 \wedge 1) = y + (y^2 \wedge 1) \Rightarrow 2x = 2y$

حال چون $2 \bmod 2 = 0$ و $2 \times 1 = 2$ نیز $2 \bmod 2 = 0$ و $2 \times 0 = 0 \neq 1$. بنابراین با توجه به تعریف ۲ به روشنی نگاشت بالا معکوس‌ناپذیر است.

تعریف ۳. نگاشتی که گراف مرتبط به آن با یک تک‌دور[1] ایزومورف است، نگاشت تک‌دور نامیده می‌شود، به‌عبارتی نگاشت $x \to \Phi(x)$ یک نگاشت تک‌دور نامیده می‌شود [2] اگر دنباله به‌وسیله تکرار x_0، $x_1 = \Phi(x_0)$، $x_2 = \Phi(x_1) = \Phi(\Phi(x_0))$ و ... تولید شود که دوره تناوبی به اندازه 2^n دارد، که این مقدار، دوره تناوب ممکن بیشینه برای کلمات n بیتی است.

می‌توان نشان داد نگاشت $x \to x + (x^2 \vee 5) \bmod 2^n$ یک نگاشت تک‌دور برای هر اندازه کلمه n است؛ درحالی‌که نگاشت $x \to x + (x^2 \vee 1) \bmod 2^n$ این‌چنین نیست. همچنین می‌توان نگاشتی با m متغیر n بیتی ساخت که یک تک‌دور بیشینه‌شده از اندازه 2^{mn} دارد.

مثال ۵. نگاشت یک متغیره $x \to x + (x^2 \vee 5) \bmod 2^n$ یک نگاشت تک‌دور تعریف می‌شود. زیرا با فرض انتخاب $x=1$ و $n=3$ (به‌طور دلخواه) می‌توان دید بعد از هشت بار تکرار، نگاشت به $x=1$ برمی‌گردد. روند تغییرات x به‌صورت زیر است:

$x = 1 \to 6 \to 3 \to 0 \to 5 \to 2 \to 7 \to 4 \to 1$

همان‌طور که مشخص است دوره تناوب نگاشت بالا برابر 2^3 بوده که حداکثر دوره تناوب آن است.

مثال ۶. نگاشت سه‌متغیره زیر یک نگاشت تک‌دور است به طوری که داریم:

$s = ((x_0 \wedge x_1 \wedge x_2) - 1) \oplus (x_0 \wedge x_1 \wedge x_2)$، زیرا با انتخاب $\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$ و $n = 3$ (به‌طور دلخواه) پس از 2^9 بار تکرار، نگاشت به حات اولیه برمی‌گردد.

$$\begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \to \begin{pmatrix} x_0 & \oplus 2x_1 x_2 \\ x_1 \oplus (s \wedge x_0) & \oplus 2x_2 x_0 \\ x_2 \oplus (s \wedge x_0 \wedge x_1) & \oplus 2x_0 x_1 \end{pmatrix}$$

می‌توان گفت هر تابع تی معکوس‌پذیر، یک تک‌دور ندارد؛ یعنی می‌توان تابع تی معکوس‌پذیری یافت که یک تک‌دور تشکیل نمی‌دهد؛ مانند تابع تی $x + (x^2 \vee 1) \bmod 2^n$.

قضیه ۱. نگاشت $x \to x + C \pmod{2^n}$ یک نگاشت تک‌دور است اگر و تنها اگر C فرد باشد[3].

قضیه ۲. فرض کنید N_0 طوری باشد که نگاشت $x \to x \oplus r(x)$ یک تک‌دور به پیمانه 2^{N_0} تعریف شود. این نگاشت تک‌دوری به پیمانه 2^n برای همه nها تعریف می‌شود؛ اگر و تنها اگر برای همه $n \geq N_0$ تابع $r(x)$ یک پارامتر فرد باشد[3].

قضیه ۳. تابع تی $f(x) = x + (x^2 \vee C) \bmod 2^n$ مفروض است[4]. می‌توان برای این تابع ثابت کرد که اگر $[C]_0 = 1$ در این صورت f معکوس پذیر است. همچنین f یک نگاشت تک‌دور است؛ اگر و تنها اگر $[C]_0 = [C]_2 = 1$.

قضیه ۴. فرض کنید S یک Sbox تک دور و α یک پارامتر فرد باشند. اگر S^o توان فردی از S و S^e توان زوجی از S باشند، آنگاه نگاشت $T(X) = (\alpha(X).S^o(X)) \oplus ((\sim \alpha(X).S^e))$ یک تابع تی تک‌دور تعریف می‌شود.

برای فهم بهتر، نگاشت T را می‌توان به‌صورت زیر نشان داد.

$$[T(X)]_i = \begin{cases} S^e([X]_i), & [\alpha(X)]_i = 0 \\ S^o([X]_i), & [\alpha(X)]_i = 1 \end{cases}$$

لم ۱. فرض کنید S یک Sbox تک‌دور و α یک پارامتر فرد باشند. نگاشت $X \to X \oplus (\alpha(X).S(X))$ یک تابع تی تک‌دور تعریف می‌کند.

۲-۱- خلاصه‌ای از الگوریتم‌های مبتنی بر توابع تی

همان‌طورکه در بخش ۲ به آن اشاره شد، برخی از الگوریتم‌های رمز جریانی مبتنی بر توابع تی مطرح شد، که در زیر به برخی از آنها به‌اختصار اشاره می‌شود.

[1] Single Cycle

[Downloaded from monadi.isc.org.ir on 2024-03-28]

۲-۱-۱- الگوریتم رمز TSC

در این بخش الگوریتمی بر اساس قضیه ۴ در بخش قبل، توصیف می‌شود. خانوادهٔ جدیدی از الگوریتم‌های رمز جریانی مبتنی بر توابع تی در سال ۲۰۰۵ توسط هانگ و همکارانش ارائه شد.

الگوریتم TSC-1 از چهار کلمه ۳۲ بیتی x_0، x_1، x_2 و x_3 تشکیل شده است. در این الگوریتم، از پارامتر فردی به‌صورت $\alpha(X)=(p+C)\oplus p\oplus 23$ استفاده شده است، که $C=0x12488421$، $p=x_0\wedge x_1\wedge x_2\wedge x_3$ و $s=x_0+x_1+x_2+x_3$. باید توجه داشت که تمامی جمع‌ها به پیمانه 2^{32} انجام می‌شوند.

با تعریف یک Sbox، ۴×۴ تک‌دور می‌توان بررسی کرد که $S^o=S$ و $S^e=S^2$ برقرار است. این به صورت زیر تعریف شده است:

$$S_1[16]=\{3,5,9,13,1,6,11,15,4,0,8,14,10,7,2,12\}$$

بنابراین می‌توان یک تابع تی تک دور با استفاده از قضیه ۴ تعریف کرد. لازم به ذکر است فیلتر الگوریتم TSC-1 به‌صورت $f(x)=(x_{0<<<9}+x_1)_{<<<15}+(x_{2<<<7}+x_3)$ تعریف شده است، که در پایان ۳۲ بیت خروجی حاصل می‌شود.

الگوریتم TSC-2 به‌طورکامل مشابه الگوریتم TSC-1 بوده و از یک Sbox، ۴×۴ تک‌دور به‌صورت زیر استفاده کرده است.

$$S_2[16]=\{5,2,11,12,13,4,3,14,15,8,1,6,7,10,9,0\}$$

پارامتر فردی که در این الگوریتم به کار برده می‌شود، به‌صورت $\alpha_2(X)=(p+1)\oplus p\oplus 23$ و فیلتر تعریف‌شده به شکل زیر استفاده می‌شود.

$$f_2(x)=(x_{0<<<11}+x_1)_{<<<14}+(x_{0<<<13}+x_2)_{<<<22}+(x_{0<<<12}+x_3)$$

در سال ۲۰۰۵، هانگ و همکارانش در رقابت ECRYPT رمز جریانی TSC-3 را ارائه کردند[8]. این الگوریتم از چهار کلمه و هر کدام با اندازهٔ چهل بیت استفاده می‌کند. این طرح، معماری مبتنی بر الگوریتم‌های ۳۲ بیتی قبلی خود را تغییر داده و قاعدتاً به‌منظور پیاده‌سازی سخت‌افزاری طراحی شده است؛ به‌علاوه، اندازهٔ حالت این طرح به ۱۶۰ بیت افزایش یافته است. لذا سطح امنیتی 2^{80} برای آن جهت مقابله با حملاتی از جمله TMTO[1] مورد انتظار است. هر لایه به‌وسیله Sboxهایی به‌روز می‌شوند که پیچیدگی آن نسبت به دو الگوریتم قبلی خانواده خود بیشتر است. پارامتر از دو کلمه p_0 و p_1 ساخته می‌شود به‌طوری‌که برای لایه iام مقدار زیر محاسبه می‌شود.

$$tmp=2\times[p_1]_i+[p_0]_i\in\{0,3\}$$

بر اساس مقدار tmp، $[x]_i$ با استفاده از S، S^2، S^5 یا S^6به‌روز می‌شود، به‌طوری‌که S همان Sbox الگوریتم TSC-1 می‌باشد. تابع فیلتر TSC-3 بدین صورت به‌روز می‌شود که چهر متغیر ۳۲ بیتی y_i به‌نحوی برقرار می‌شوند، که هشت بیت کم ارزش از هر کدام از کلمات چهل بیتی x_i حذف می‌شوند؛ سپس y_iها بسته به مقدار کم‌ارزش‌ترین لایه حالت یعنی $[x]_0$، جایگشت داده می‌شوند. بنابراین شانزده جایگشت ممکن وجود خواهد داشت و تابع خروجی به‌صورت زیر حاصل می‌شود:

$$f(y)=(y_{0<<<9}+y_{1>>>2})_{<<<8}+(y_{2<<<7}+y_3)_{>>>9}$$

۲-۱-۲- الگوریتم رمز ABC

ABC یک رمز جریانی همزمان[2] بهینه شده برای کاربردهای نرم افزاری است. این الگوریتم با یک کلید ۱۲۸ بیتی و متغیرهای داخلی ۳۲ بیتی ارائه شده است. به همین دلیل از آن یک امنیت 2^{128} انتظار می‌رود. الگوریتم ABC از سه مولد اصلی A، B و C استفاده می‌کند. مولد A یک LFSR با طول ۱۲۸ که حالت‌های آن با z نشان داده شده‌اند. مولد B یک نگاشت تک‌دور مبتنی بر جمع حسابی در میدان $Z/2^{32}Z$ و جمع بیتی به پیمانه ۲ (xor) بوده و با متغیر x نشان داده می‌شود. مولد C نیز یک تابع فیلتر مبتنی بر جداول جستجو، جمع حسابی در میدان $Z/2^{32}Z$ و چرخش بیتی به سمت راست(>>>) است.

همان‌طورکه گفته شد، A یک LFSR با طول ۱۲۸ بوده و چندجمله‌ای مشخصه آن برابر $\phi(\theta)=\psi(\theta)\theta$ که $\psi(\theta)=\theta^{127}+\theta^{63}+1$ یک چندجمله‌ای اولیه است. ۳۲ بیت بعدی در یک مرتبه کلاک خوردن به‌صورت زیر به‌دست می‌آیند.

[1] Time Memory Data Trade Off

[2] Synchronous Stream Cipher

[Downloaded from monadi.isc.org.ir on 2024-03-28]

$\leftarrow \overline{z}_2 \oplus (\overline{z}_1 = 31) \oplus (\overline{z}_0 ? 1) \bmod 2^{32}$

$\overline{z}_0 \leftarrow \overline{z}_1$,

$\overline{z}_1 \leftarrow \overline{z}_2$,

$\overline{z}_2 \leftarrow \overline{z}_3$,

$\overline{z}_3 \leftarrow \overline{z}$

تابع تک‌دور B که در رمز ABC از آن استفاده شده است و در نوع خود یک تابع تی است، به‌صورت معادلهٔ زیر نشان داده می‌شود:

$$B(x) = ((x \oplus d_0) + d_1) \oplus d_2 \bmod 2^{32}$$

به‌طوری‌که $d_0, d_1, d_2 \in \ / 2^{32}$ و $d_0 \equiv 0 \pmod 4$ ، $d_1 \equiv 1 \pmod 4$ و $d_2 \equiv 0 \pmod 4$.

به‌عبارت دیگر معادلات زیر در یک زمان باید برقرار باشند:

$d_{0,0} = d_{0,1} = 0$,

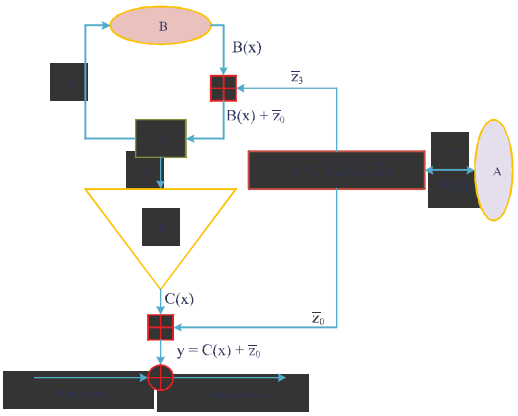
$d_{1,0} = 1$, $d_{1,1} = 0$,

$d_{2,0} = d_{2,1} = 0$.

برای نمایش فیلتر C رمز ABC، فرض کنید نگاشت $S : \ / 2^{32} \rightarrow \ / 2^{32}$ به وسیله $S(x) = e + \sum_{i=0}^{31} e_i \ _i(x) \bmod 2^{32}$ تعریف شده باشد به‌طوری که $e_{31} \equiv 2^{16} \pmod{2^{17}}$. تابع فیلتر C خروجی y را به‌صورت زیر تولید می‌کند.

$$y = >>>16 \ , \ S(x) =$$

مولد دنباله کلید الگوریتم ABC به‌صورت شکل زیر است.



(شکل-۱): نمایش الگوریتم رمز ABC

همان‌طورکه از شکل مشخص است، ورودی و خروجی الگوریتم به شرح زیر است:

Input : $z \in \ / 2^{128}$, $x \in \ / 2^{32}$

$z \leftarrow A(z)$

$x \leftarrow \overline{z}_3 + B(x) \bmod 2^{32}$

$y \leftarrow \overline{z}_0 + C(x) \bmod 2^{32}$

Output : $z \in \ / 2^{128}$, $x \in \ / 2^{32}$, $y \in \ / 2^{32}$

به‌منظور توصیف الگوریتم‌های معرفی شده دیگر به منابع [13] و [14] رجوع شود.

۳- پرش

همان طور که در بخش ۱ نیز اشاره شد، استفاده از روش کلاک نامنظم در رمزهای جریانی باعث حملات کانال جانبی خواهد شد. یکی از روش‌های نزدیک به روش کلاک نامنظم، استفاده از مفهوم پرش است که در این حالت وضعیت فعلی با وضعیت به‌روزشده جمع می‌شوند. برای توصیف این حالت فرض کنید A ماتریس انتقال یک ماشین حالت متناهی خطی[1] مستقل باشد و $f(x)$ چندجمله‌ای بازگشتی آن یعنی $f(x) = \det(xI + A)$ منظور گردد. در این صورت اگر توان J وجود داشته باشد به‌طوری‌که $A^J = A + I$، آن‌گاه می‌توان گفت با تغییر ماتریس انتقال LFSM، از A به $A + I$، به‌طور مؤثر J گام از فضای حالت LFSM اصلی، صرف نظر از حالت ابتدایی ساخته می‌شود.

نخستین‌بار جانسن[2] ایده پرش را مطرح کرد. او از این ایده به‌منظور مقاومت الگوریتم‌های رمز جریانی در مقابل حملات کانال جانبی استفاده کرد. از الگوریتم‌هایی که از این ایده استفاده شده است، می‌توان به Pomaranch [10] و Mickey [11] اشاره کرد.

بازگشتی خطی زیر حائز اهمیت است:

$$s_{j+n} = \sum_{i=1}^{n} c_i s_{j+n-i} \Leftrightarrow \sum_{i=0}^{n} c_i s_{j+n-i} = 0 : (c_0 = -1)$$

به‌طوری‌که ضرایب بازگشتی c_i به‌طورمعمول بیان می‌شوند و این ضرایب باعث می‌شوند چندجمله‌ای حاصل از آن، چندجمله‌ای بازگشتی نامیده شوند. چندجمله‌ای توصیف‌کننده $F(x)$ از درجه‌ای برابر با طول بازگشتی n

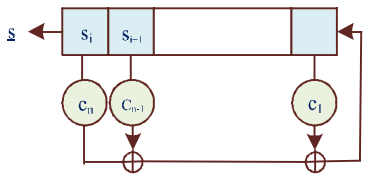
[1] Linear Finite State Machine (LFSM)
[2] JANSEN

[Downloaded from monadi.isc.org.ir on 2024-03-28]

است که در رابطه $F(x)=\sum_{i=0}^{n} c_i x^i$ مشهود است. مرتبه nام بازگشتی خطی به‌طورمعمول توسط چندجمله‌ای بازگشتی آن، $C(x)$، ارائه می‌شود که از درجه n با رابطه $C(x)=\sum_{i=0}^{n} c_i x^{n-i}$ بیان می‌شود.

LFSR را به‌عنوان یک ماشین حالت متناهی خطی می‌توان در نظر گرفت. در این حالت، حالتی از LFSM با یک بردار $\underline{t}=(t_{n-1}, t_{n-2}, \ldots, t_0)$ نمایش داده می‌شود؛ به‌طوری‌که t_i محتوای سلول حافظه M_i بعد از t انتقال را مشخص می‌کند. انتقال‌ها از یک حالت به حالت بعدی توسط یک ضرب بردار حالت با یک ماتریس انتقال T توصیف می‌شود؛ یعنی برای $t\geq 0$، رابطه $\underline{t+1}=\underline{t}\cdot T$ صادق است. ماتریس انتقال برای یک LFSR با نمایش زیر

به‌صورت $T=\begin{pmatrix} 0 & 0 & \ldots & 0 & c_n \\ 1 & 0 & \ldots & 0 & c_{n-1} \\ 0 & 1 & \ldots & 0 & c_{n-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & c_1 \end{pmatrix}$ مشخص شده است.



(شکل-۲): نمایش یک LFSR با n تپ

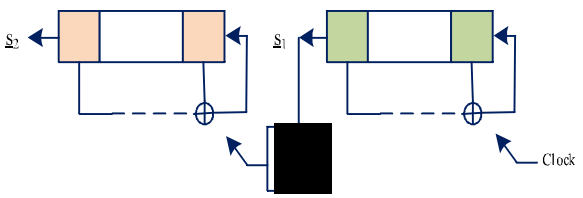
ماتریس بالا، ماتریس همراه چندجمله‌ای $x^n-c_1x^{n-1}-\ldots-c_{n-1}x-c_n=C(x)$ نامیده می‌شود. چندجمله‌ای بازگشتی T در مفهوم جبر خطی یعنی $C(x)=\det(xI-T)$، به‌طور دقیق برابر این چندجمله‌ای است و در نتیجه $C(T)=0$. بنابراین ماتریس همراه بالا نقش یک ریشه C را بازی می‌کند و به‌طور متوالی می‌تواند برای تشکیل حل معادلات بازگشتی به‌کار رود.

تعریف ۳. فرض کنید $f(x)$ یک چندجمله‌ای تحویل‌ناپذیر روی GF(۲) باشد. اگر $x^J\equiv x+1 \pmod{f(x)}$ برای J‌های صحیح برقرار باشد، آن‌گاه J، ضریب پرشی[1] f نامیده می‌شود[5].

پس با توجه به تعریف ۳ و مطالب ما قبل آن، اگر چنین توان Jای موجود باشد و بردار حالت $\underline{t}$ با A^J یا $A+I$ ضرب شود، به نتیجه یکسانی خواهد رسید. به‌علاوه تغییر A به $A+I$ به‌طور عمومی ساده‌تر از تبدیل A به A^J برای ماتریس انتقال دلخواه A است؛ لذا اگر $f(x)$ تحویل‌ناپذیر باشد، آنگاه A می‌تواند به‌عنوان ماتریس همراه $f(x)$ با به کاربردن ضرب ماتریسی مناسبی نوشته شود؛ به‌طوری‌که $A'=MAM^{-1}$ برقرار است. ماتریس‌های A و A' را ماتریس‌های مشابه نامند. توان‌های ماتریس همراه می‌تواند به‌عنوان نمایش همه عناصر میدان متناهی در نظر گرفته شوند.

لازم به ذکر است ضریب پرشی برای هر چندجمله‌ای تحویل‌ناپذیر، وجود ندارد؛ زیرا این مهم، بستگی به برقراری رابطه $x^J\equiv x+1 \pmod{f(x)}$ دارد، یا به‌طور معادل رابطه $f(x)\mid(x^J+x+1)$ برای برخی Jها برقرار باشد. به‌عبارت دیگر $J=+1$، که ریشه $f(x)$ است و بنابراین عنصری از میدان $GF(2^n)$ است. بنابراین معادل با عبارت‌های می‌توان گفت، مسئله، یافتن یک عنصر J در میدان متناهی $GF(2^n)$ است، به‌طوری‌که f تعریف چندجمله‌ای و $n=\deg(f)$ می‌باشند. عبارت $J=+1$ این مفهوم را می‌رساند که عنصری از $GF(2^n)$ بوده و با تغییر ماتریس انتقال LFSM از A به $A+I$، به‌طور مؤثر J گام از فضای حالت اصلی LFSM، صرف نظر از حالت ابتدایی ساخته می‌شود.

شکل زیر، نمایشی از یک مولد رمز جریانی با دو LFSR و به‌کار بردن کلاک نامنظم است.



(شکل-۳): مولد دنباله LFSR با کلاک نامنظم

[1] Jump Index

[Downloaded from monadi.isc.org.ir on 2024-03-28]

LFSR نخست یک دنباله دودویی $\underline{S_1}$ با دوره تناوب p_1 تولید می‌کند که با بعضی مقسوم علیه‌های از مرتبه $2^{L_1}-1$ حالتی از یک چندجمله‌ای بازگشتی تحویل‌ناپذیر را خواهد ساخت، به‌طوری‌که L_1 طول LFSR است. این دنباله N_0 تا صفر و N_1 تا یک دربر دارد که برای کلاک‌خوردن LFSR دوم به کار برده می‌شود، یعنی LFSR دوم از طریق فضای حالت آن گام بر می‌دارد که وابسته به بیت‌های نتیجه دنباله، با گام‌خوردن آن، c_0 بار یا c_1 بار خواهد بود، اگر به‌ترتیب بیت خروجی یک ۰ یا یک ۱ باشد. تعداد کل (N_s) گام‌ها توسط LFSR دوم با چندجمله‌ای در یک دوره تناوبی از LFSR نخست ساخته شده است که در رابطه $N_s = N_0 c_0 + N_1 c_1$ صدق می‌کند. فرض کنید LFSR دوم چندجمله‌ای بازگشتی تحویل‌ناپذیری دارد، از این جهت یک شرط ضروری در دنباله خروجی $\underline{S_2}$ از LFSR دوم برای داشتن دوره تناوب بیشینه‌شده $p_1 p_2$، باید $\gcd(N_s, p_2)=1$ باشد که یک شرط کافی نیز است.

LFSRهای به‌طول L، دوره تناوبی به‌اندازه 2^L-1 دارند. در این حالت تعداد صفرها و یک‌ها با $N_0 = \frac{p-1}{2}$ و $N_1 = \frac{p+1}{2}$ داده شده است. با توجه به اینکه حالت تماماً صفر اتفاق نمی‌افتد، دست‌کم مقدار اختلاف تعداد صفرها و یک‌ها اندازه یک واحد خواهیم داشت. بنابراین تعداد کل گام‌ها با رابطه $N_s = c_0 p + (c_1 - c_0)2^{L-1}$ بیان شده است. درنتیجه، اگر LFSR دوم با یک دوره تناوب p_2 برابر p، (یا یکی از مقسوم‌های آن) داشته باشد، آن‌گاه شرط ضروری برای دوره تناوب بیشینه $\underline{S_2}$ با رابطه $\gcd(N_s, p_2) = \gcd(c_1 - c_0, p) = 1$ به‌دست می‌آید. در حالت خاصی که LFSRهای پرشی به کار می‌روند، یک گام یا یک پرش معادل با J گام از فضای حالت ساخته می‌شود که می‌توان شرط قبلی را به‌صورت $\gcd(J-1, p)=1$ یا $\gcd(J^*, p)=1$ بیان کرد که در آن $J^* = 1-J \pmod{per(f)}$ ضریب پرشی چندجمله‌ای متقابل $f(x)$ یعنی $f^*(x)$ است. به‌عبارت دیگر ضریب پرشی چندجمله‌ای بازگشتی از LFSR پرشی بایستی نسبت به دوره تناوب آن، نخست باشد.

در بخش زیر الگوریتم Mickey تشریح‌شده که از ضریب پرش استفاده کرده است.

۳-۱-۱- الگوریتم رمز Mickey

الگوریتم Mickey به‌منظور کاربردهای سخت‌افزاری طراحی شده است و برای مولدهای دنباله کلید با کلاک نامنظم دو جانبه استفاده می‌شود.

Mickey دو پارامتر ورودی کلید و IV می‌گیرد. کلید استفاده شده در رمز Mickey برابر هشتاد بیت بوده که با k_0, K, k_{79} نشان داده می‌شود. مقدار IV نیز طولی بین صفر تا هشتاد بیت دارند که با $iv_0, K, iv_{IVLENGTH-1}$ نشان داده می‌شود. خروجی دنباله کلید با z_0, z_1, K نشان داده شده و درنهایت متن رمزشده با جمع دودیی بیت‌های دنباله کلید حاصل و متن اصلی به دست خواهد آمد؛ اما مولد دنباله کلید از دو ثبات R و S ساخته شده، به‌طوری‌که هر ثبات شامل هشتاد شبکه یک بیتی است. ثبات‌های R با r_0, K, r_{79} و ثبات‌های S با s_0, K, s_{79} برچسب گذاری می‌شوند. در اینجا ثبات R به‌عنوان ثبات خطی و ثبات S، ثبات غیرخطی فرض می‌شود.

به‌منظور کلاک ثبات R، مجموعه RTAPS به‌صورت زیر تعریف می‌شود:

$$\{0,2,4,6,7,8,9,13,14,16,17,20,22,24,26,27,28,34,35,37,39,41,43,49,51,52,54,56,62,67,69,71,73,76,78,79\}$$

حال یک عمل کلاک CLOCK_R(R, INPUT_BIT_R, CONTROL_BIT_R) به‌صورت زیر تعریف می‌شود:

با فرض اینکه r_0, K, r_{79} حالت ثبات R قبل از عمل کلاک و r'_0, K, r'_{79} بعد از عمل کلاک باشد، روابط زیر صادق هستند:

$$FEEDBACK_BIT = r_{79} \oplus INPUT_BIT_R$$

$$FOR\ 1 \le i \le 79,\ r'_i = r_{i-1};\ r'_0 = 0$$

$$FOR\ 0 \le i \le 79,\ \text{if } i \in RTAPS,\ r'_i = r'_i \oplus FEEDBACK_BIT$$

$$\text{if } CONTROL_BIT_R = 1:$$

$$FOR\ 0 \le i \le 79,\ r'_i = r'_i \oplus r_i$$

همچنین به‌منظور کلاک ثبات S، چهار دنباله COP_0, K, COP_{78}، COP_1, K, COP_{78} و $FB0_0, K, FB0_{79}$ و $FB1_0, K, FB1_{79}$ به صورت جدولی در [11] تعریف می‌شوند.

حال یک عمل کلاک CLOCK_S(S, INPUT_BIT_S, CONTROL_BIT_S) به‌صورت زیر تعریف می‌شود:

[Downloaded from monadi.isc.org.ir on 2024-03-28]

با فرض اینکه $s_0,\ldots,s_{79}$ حالت ثبات S قبل از عمل کلاک و $s'_0,\ldots,s'_{79}$ بعد از عمل کلاک باشد، و نیز از $\hat{s}_0,\ldots,\hat{s}_{79}$ به‌عنوان متغیرهای میانی برای سادگی توصیف استفاده شود، روابط زیر صادق هستند:

$$FEEDBACK_BIT = s_{79} \oplus INPUT_BIT_S$$
$$FOR\ 1 \le i \le 78,\ \hat{s}_i = s_{i-1} \oplus$$
$$((s_i \oplus COMP0_i).(s_{i+1} \oplus COMP1_i));\ \hat{s}_0 = 0;\ \hat{s}_{79} = s_{78}$$
$$if\ CONTROL_BIT_S = 0:$$
$$FOR\ 0 \le i \le 79,\ s'_i = \hat{s}_i \oplus (FB0_i.FEEDBACK_BIT)$$
$$if\ CONTROL_BIT_S = 1:$$
$$FOR\ 0 \le i \le 79,\ s'_i = \hat{s}_i \oplus (FB1_i.FEEDBACK_BIT)$$

پس از بیان توابع ثبات‌ها، یک کلاک مولد به نام $CLOCK_KG(R,S,MIXING,INPUT_BIT)$ به‌صورت زیر تعریف می‌شود:

$$CONTROL_BIT_R = s_{55} \oplus r_{53}$$
$$CONTROL_BIT_S = s_{53} \oplus r_{79}$$
$$if\ MIXING = TRUE: INPUT_BIT_R = INPUT_BIT \oplus s_{40}$$
$$if\ MIXING = FALSE: INPUT_BIT_R = INPUT_BIT$$
$$CLOCK_R(R, INPUT_BIT_R, CONTROL_BIT_R)$$
$$CLOCK_S(S, INPUT_BIT_S, CONTROL_BIT_S)$$

برای توصیف بیشتر الگوریتم به [11] رجوع شود. همچنین از الگوریتم Pomaranch به‌عنوان الگوریتم‌های مبتنی بر توابع تی یاد شد که می‌توان به [10] رجوع کرد.

۴- ارتباط بین توابع تی و پرش

قضیه ۳. فرض کنید تابع $f: Z_{2^n} \to Z_{2^n}$ یک تابع تی دوسویه[1] باشد. در این صورت تابع $f^i: Z_{2^n} \to Z_{2^n}$ با ضابطه $f^i(x) = \begin{cases} x & if\ \ i = 0 \\ f(f^{i-1}(x)) & if\ \ i > 1 \end{cases}$ برای هر i یک تابع تی دوسویه خواهد بود[4].

هدف اصلی این مقاله با ذکر این قضیه بیان خواهد شد که پرش در توابع تی است. از توابع یک‌متغیره‌ای که دارای دوره تناوب بیشینه است، می‌توان تابع $x \to x + (x^2 \vee 5) \mod 2^n$ را نام برد.

یکی از ویژگی‌های جالب در توابع $x \to x + (x^2 \vee 5) \mod 2^n$، آن است که دو تابع $x \to x + (x^2 \vee 5) \mod 2^n$ و $x + (x^2 \vee (5 + 2^{n-1})) \mod 2^n$ دارای اختلاف 2^{n-1} است.

اثبات این مشاهده بسیار راحت است؛ زیرا اگر بیت پرارزش x^2 برابر صفر باشد، اثبات مطلب بالا مشخص است؛ ولی اگر بیت پرارزش x^2 برابر ۱ باشد، حاصل جمع به بیت nام رفته و با توجه به پیمانه 2^n، این بیت حذف شده و اختلاف همان مقدار 2^{n-1} خواهد بود.

نکته مهم دیگری که برای همه توابع تی با دوره تناوب 2^n و در پیمانه 2^n برقرار است و ازجمله در تابع $f(x) = x + (x^2 \vee (5 + 2^{n-1})) \mod 2^n$ می‌توان بیان کرد، آن است که تساوی رابطه $f^{2^{n-1}}(x) = (x + 2^{n-1}) \mod 2^n$ برقرار می‌باشد. دلیل این تساوی آن است که می‌دانیم $f^{2^n}(x) = x$. حال اگر فرض کنیم $f^{2^{n-1}}(x) = (x + 2^{n-1}) \mod 2^n$ در این صورت رابطه

$$x = f^{2^{n-1}}(f^{2^{n-1}}(x)) = (f^{2^{n-1}}(x) + 2^{n-1}) \mod 2^n$$
$$= (x + 2^{n-1} + 2^{n-1}) \mod 2^n = x$$

برقرار است. البته با توجه به استقرا باید برای یک نقطه اولیه تأییدکننده باشد و درنهایت، برای هر تابع f باید این امر تحقیق شود. بر اساس دو نکته بالا لم زیر به‌راحتی اثبات می‌شود.

لم ۱. اگر $f(x) = x + (x^2 \vee 5) \mod 2^n$ در این صورت $f^{2^{n-1}+1}(x) = x + (x^2 \vee (5 + 2^{n-1})) \mod 2^n$.

قضیه ۴. فرض کنید نگاشت به‌صورت: $x \to x + (x^2 \vee (5 + a)) \mod 2^n$ تعریف شود، به‌طوری که یک بیت تصادفی تعیین کند که a برابر صفر و یا 2^{n-1} باشد؛ در این صورت دوره تناوب تابع فوق حداقل برابر 2^{n-1} است.

اثبات:

با توجه به اینکه اگر تابع $x + (x^2 \vee 5) \mod 2^n$ انتخاب شود، میزان پرش به اندازه ۱ و اگر تابع $x + (x^2 \vee (5 + 2^{n-1})) \mod 2^n$ انتخاب شود میزان پرش برابر $c = 2^{n-1} + 1$ نسبت به نگاشت $x \to x + (x^2 \vee 5) \mod 2^n$ خواهد بود. دلیل برقراری مقدار c طبق لم ۱ مشخص است.

در این صورت برای پیداکردن دورهٔ تناوب این تابع فرض کنیم که t بار پرش به میزان یک و s بار پرش به اندازه

[1] Bijective

[Downloaded from monadi.isc.org.ir on 2024-03-28]

$۲^{n-۱}+۱$ باشد. در این صورت باید کوچکترین مقدار s+t را به طوری پیدا کرد که

$$t + s\,(۲^{n-۱}+۱) = k\,.۲^{n}$$

همچنین مقادیر s، t و s+t باید مثبت باشند. در این صورت رابطه زیر به دست خواهد آمد:

$$t + s = k\,.۲^{n} - s\,.۲^{n-۱} = k'(۲^{n-۱})$$

به این ترتیب می‌توان نشان داد که s+t حداقل مقدار مثبتی که می‌تواند داشته باشد برابر $۲^{n-۱}$ است.

هرچند اثبات بالا برای توابع تی یک‌متغیره ارائه شد، می‌توان همین بحث را برای توابع تی n متغیره اثبات کرد.

فرض کنیم رابطه $(x_1,x_2,\ldots,x_n)=f^{۲^n}(x_1,x_2,\ldots,x_n)$ صادق باشد، زیرا بعد از ۲^n بار تکرار $f(x_1,x_2,\ldots,x_n)$، به n‌تایی $(x_1,x_2,\ldots,x_n)$ خواهیم رسید و لذا روابط زیر به دست خواهند آمد:

$$(x_1,x_2,\ldots,x_n)=f^{۲^n}(x_1,x_2,\ldots,x_n)$$

$$=f^{۲^{n-1}}\left(f^{۲^{n-1}}(x_1,x_2,\ldots,x_n)\right)=f^{۲^{n-1}}(x_1,x_2,\ldots,x_n)+۲^{n-1}$$

$$=(x_1,x_2,\ldots,x_n)+۲^{n-1}+۲^{n-1}\ \ (mod\ ۲^n)=(x_1,x_2,\ldots,x_n)\ .$$

$$\Rightarrow f^{۲^{n-1}}(x_1,x_2,\ldots,x_n)=(x_1,x_2,\ldots,x_n)$$

$$+۲^n \Rightarrow (x_1,x_2,\ldots,x_n)+\begin{pmatrix}(x_1,x_2,\ldots,x_n)^{۲}\vee\\((a_1,a_2,\ldots,a_n)+۲^{n-1})\end{pmatrix}\ \ (mod\ ۲^n)$$

۵- نتیجه

در این مقاله پس از بیان کوتاهی از توابع تی، تشریح چند الگوریتم رمز مبتنی بر توابع تی، مفاهیم پرش و تشریح یک الگوریتم مبتنی بر پرش، یک ساختار پرش با استفاده از توابع تی ارایه شد. این ساختار دارای حد پایینی برای حداقل دوره تناوب است و همچنین از لحاظ پیاده‌سازی، عملیات اضافه‌ای ندارد و می‌تواند به‌عنوان یک ساختار اولیه مناسب در طراحی رمزهای جریانی آینده به کار رود. این ساختار به‌منظور طراحی یک مولد تصادفی بسیار مناسب است. داشتن بیشترین دوره تناوب بیشینه در یک دنباله L بیتی و مقاومت علیه حملات کانال جانبی به دلیل استفاده از پرش، علت مناسب‌تربودن این ساختار نسبت به ساختارهای معمول است.

۶-منابع

[1] Jansen, C.J.A : Streamcipher design: Make your LFSRs jump! In The State of the Art of Stream Ciphers, Workshop Record, ECRYPT Network of Excellence in Cryptology (2004), pp. 94–108 http://www.ecrypt.eu.org/stvl/sasc/sasc-record.zip.

[2] A. Klimov and A. Shamir : Applications of T-functions in Cryptography ,PHD Thesis, Weizmann Institute of Science, 2005.

[3] A. Klimov and A. Shamir, New Cryptographic Primitives Based on Multiword T-functions, FSE (Fast Software Encryption) 2004, LNCS 3017, Springer-Verlag, pp. 1–15.

[4] Min Surp Rhee : on a Characterization of T-Functions with one Cycle Property, JOURNAL OF THE CHUNGCHEONG MATHEMATICAL SOCIETY Volume 21, No. 2, June 2008, pp. 259-268.

[5] Cees J.A. Jansen : Stream Cipher Design based on Jumping Finite State Machines, http://eprint.iacr.org/2005/267.

[6] A Kilmov and A. Shamir, A New Class of Invertible Mappings, CHES 2002, LNCS 2523, 470-483, 2003.

[7] Vladimir Anashin, Andrey Bogdanov, Ilya Kizhvatov, and Sandeep Kumar, ABC: A new fast flexible stream cipher. eSTREAM, ECRYPT Stream Cipher Project, Report 2005/001,2005.http://www.ecrypt.eu.org/strea.

[8] J. Hong, D. Lee, Y. Yeom, D. Han, and S. Chee. T-function Based Stream Cipher TSC-3. ECRYPT Stream Cipher Project Report 2005/031,2005,http://www.ecrypt.en.org/strea.

[9] J. Hong, D. Lee, Y. Yeom, and D. Han (2005). A New Class of Single Cycle T-functions. Fast Software Encryption, FSE 2005, LNCS 3557. Springer-Verlag. pp. 68–82.

[10] C.J.A. Jansen, T. Helleseth, and A. Kholosha, Cascade Jump Controlled Sequence Generator and Pomaranch Stream Cipher, LNCS 4986, pp. 224–243, 2008.

[11] S. Babbage, M. Dodd, The MICKEY Stream Ciphers, LNCS 4986, pp. 191–209, 2008.

[12] Vladimir Anashin, Andrey Bogdanov, Ilya Kizhvatov, and Sandeep Kumar. ABC: A new fast flexible stream cipher. Version 2, 2005. http://crypto.rsuh.ru/papers/abc-spec-v2.pdf.

[13] Maximov, A.: A New Stream Cipher “Mir-1”. eSTREAM submission, 2005.

[14] S ONeil, B. Gittins and H. Landman. VEST Hardware-Dedicated Stream Ciphers. Available at www.ecrypt.eu.org/stream/ciphers/vest/vest.pdf.

[Downloaded from monadi.isc.org.ir on 2024-03-28]



سید مهدی سجادیه مدارک کارشناسی، کارشناسی ارشد و دکترای خود را از دانشگاه صنعتی اصفهان در سال‌های ۸۲، ۸۴ و ۹۱ در رشته مهندسی برق مخابرات کسب کرده است. وی از سال ۱۳۹۰ تا کنون عضو هیات علمی دانشگاه ازاد اسلامی واحد اصفهان (خوراسگان) است. زمینه پژوهش‌های وی الگوریتم ها و پروتکل‌های رمزنگاری بویژه بررسی رمزهای قالبی و دنباله‌ای است.



علی هادی‌پور مدرک کارشناسی را در سال ۸۷ در رشته ریاضی محض از دانشگاه پیام نور مرکز نطنز و کارشناسی ارشد خود را در سال ۹۰ در رشته ریاضی رمز از دانشگاه صنعتی مالک اشتر کسب کرده است. وی از سال ۱۳۹۱ تا کنون در یک مرکز تحقیقاتی مشغول به فعالیت بوده است. علایق پژوهشی ایشان، پژوهش در زمینه طراحی الگوریتم‌های رمزنگاری متقارن و نیز طراحی الگوریتم‌های پنهان‌نگاری است.



راحله مرادعفیفی مدرک کارشناسی را در سال ۸۵ در رشته ریاضی کاربردی از دانشگاه صنعتی امیرکبیر و کارشناسی ارشد خود را در سال ۹۰ در رشته ریاضی رمز از دانشگاه صنعتی مالک اشتر کسب کرده است. وی از سال ۱۳۹۳ تا کنون در دانشگاه غیرانتفاعی آل طه مشغول به فعالیت بوده است. علایق پژوهشی ایشان، پژوهش بر روی طراحی الگوریتم‌های رمزنگاری متقارن و نیز تحقیقات بر روی کدینگ کانال می‌باشد.

[Downloaded from monadi.isc.org.ir on 2024-03-28]

[Downloaded from monadi.isc.org.ir on 2024-03-28]