به نام خدا

دانشکده‌ی علوم ریاضی

مقدمه‌ای بر رمزنگاری ۲۲ اسفند ۱۳۹۱

جلسه‌ی ۱۰: مولدهای رمز دنباله‌ای عملی

مدرس: دکتر شهرام خزائی *نگارنده: حسین اورعی*

این جلسه به معرفی مولدهای رمز دنباله‌ای که در عمل مورد استفاده قرار می‌گیرند اختصاص دارد. اما قبل از آن حمله جدیدی را به رمزهای دنباله‌ای تعریف می‌کنیم:

۱ حمله بده‌بستان زمان، حافظه و داده

فرض کنید، یک مولد رمز دنباله‌ای «نسبتاً خوب» در اختیار دارید که تعداد بیت‌های حالت آن n تاست. حالت اولیه‌ی چنین مولدی را به صورت «تقریباً یکتا» می‌توان از روی n بیت خروجی در زمان 2^n با استفاده از جستجوی کامل به دست آورد. برای توجیه این مسأله توجه کنید، که حدود نصف حالت‌ها، یعنی 2^{n-1} حالت، می‌توانند اولین بیت خروجی دنباله داده شده را تولید کنند؛ همچنین حدود یک‌چهارم حالت‌ها، یعنی 2^{n-2} حالت، می‌توانند دو بیت اول خروجی داده شده را تولید کنند؛ به همین ترتیب، به ازای هر $1 \leq t \leq n$ ، حدود $\frac{1}{2^t}$ حالت‌ها، یعنی 2^{n-t} حالت، می‌توانند t بیت اول دنباله خروجی داده شده را تولید کنند. به طور دقیق‌تر، به طور متوسط یک حالت اولیه وجود دارد که با n بیت داده شده از دنباله خروجی مطابقت دارد.
در حمله بده‌بستان زمان، حافظه و داده[1] مهاجم در دو مرحله عمل می‌کند. در مرحله پیش‌پردازش، در زمان $2^{n/2}$ اطلاعاتی را محاسبه و در یک حافظه (جدول) به اندازه $2^{n/2}$ ذخیره می‌نماید. سپس در مرحله دوم، با در دست داشتن حدود $2^{n/2}$ بیت خروجی از مولد، حالت اولیه مولد را در زمان سریع‌تر $2^{n/2}$ به دست می‌آورد. به طور دقیق‌تر تابع $f : \{0,1\}^n \to \{0,1\}^n$ را در نظر بگیرید که n بیت حالت اولیه را به n بیت اول از دنباله خروجی مولد می‌نگارد. فرض کنید مهاجم $D = 2^{n/2} + n - 1$ بیت از دنباله خروجی به صورت $z_1, \cdots, z_D$ مشاهده کرده است. معادلاً می‌توان تصور کرد که مهاجم $2^{n/2}$ نقطه خروجی از تابع f به صورت $\mathbf{z}_i = (z_i, z_{i+1}, \cdots, z_{i+n-1})$ برای $i = 1, 2, \cdots, 2^{n/2}$ در اختیار دارد. هدف مهاجم معکوس کردن تابع f در یکی از نقاط خروجی مشاهده شده است. به عبارت دیگر، اگر مهاجم بتواند یک حالت $\mathbf{s} = (s_1, s_2, \cdots, s_n)$ پیدا کند که $\mathbf{z}_i = f(\mathbf{s})$ موفق به شکستن رمز دنباله‌ای شده است. برای این منظور مهاجم در فاز پیش‌پردازش، به صورت زیر عمل می‌کند:

- به تعداد $M = 2^{n/2}$ حالت اولیه تصادفی $\mathbf{x}_1, \cdots, \mathbf{x}_M$ انتخاب کن.
- مقایر $\mathbf{y}_i = f(\mathbf{x}_i)$ را محاسبه کن (یعنی، حالت اولیه $\mathbf{x}_i$ را در مولد قرار بده و n بیت اول آن $\mathbf{y}_i$ را به‌دست آور.)
- زوج‌های $(\mathbf{y}_i, \mathbf{x}_i)$ را به صورت مرتب برحسب مؤلفه اول در یک جدول ذخیره کن.

برای اعمال حمله، در مرحله دوم مهاجم با مشاهده دنباله‌ی خروجی $z_1, \cdots, z_D$ از مولد به صورت زیر عمل می‌کند:

- نقاط خروجی $\mathbf{z}_i = (z_i, z_{i+1}, \cdots, z_{i+n-1})$ را تشکیل بده.
- به ازای $i = 1, 2, \cdots, 2^{n/2}$ اگر زوجی در جدول ذخیره شده با مؤلفه‌ی اول $\mathbf{z}_i$ وجود دارد، مؤلفه دوم آن را برگردان.

تناقض روز تولد بیان می‌کند اگر از مجموعه‌ای شامل N عضو، $\sqrt{N}$ عضو به تصادف و با جایگزینی انتخاب شوند، احتمال مشاهده عضو تکراری قابل توجه (حدود $1/2$) است. قضیه زیر، بیان دیگری از تناقض روز تولد است که موفقیت مهاجم را تضمین می‌کند.

قضیه ۱ (روز تولد) *فرض کنید از بین گردایه‌ای N عضوی، دو لیست N_1 و N_2 عضوی به تصادف انتخاب می‌کنیم که لیست‌ها می‌توانند دارای عضو تکراری باشند. در این صورت اگر $N_1 \cdot N_2 = N$ آنگاه با احتمال حدود $1/2$ حداقل یک عضو مشترک در دو لیست وجود دارد. به طور خاص، اگر $N_1 = N_2 = \sqrt{N}$ این واقعه با احتمال قابل توجه حدود $1/2$ رخ می‌دهد.*

[1] Time-Memory-Data-Tradeoff

۲ مولدهای رمز دنباله‌ای معروف

در ادامه به معرفی رمزهای دنباله‌ای که در عمل مورد استفاده قرار می‌گیرند خواهیم پرداخت. این رمزهای دنباله‌ای به همراه برخی از مشخصات آن‌ها در جدول ۱ آمده است.

جدول ۱: جدول مقایسه رمزهای دنباله‌ای عملی

نام رمز دنباله‌ای	طول کلید	طول IV	تعداد بیت حالت	مورد استفاده
A5/1	۶۴ بیت	۲۲ بیت	۶۴	تأمین امنیت در مکالمات تلفن همراه
A5/2	۶۴ بیت	۲۲ بیت	۸۱	تأمین امنیت در مکالمات تلفن همراه
E۰	۱۲۸ بیت	–	۱۳۲	رمز کردن ارتباطات در بلوتوث
RC4	متغیر بین ۸ تا ۲۰۴۸ بیت	–	۲۰۶۴	پروتکل‌های امنیتی در اینترنت
Trivium	۸۰ بیت	۸۰ بیت	۲۸۸	سخت‌افزاری
Grain v1	۸۰ بیت	۶۴ بیت	۱۶۰	محیط‌های سخت‌افزاری دارای محدودیت

۳ A5/1

مولد رمز دنباله‌ای A5/1 برای رمز کردن ارتباطات در استاندارد GSM[۲] استفاده می‌شود. این مولد از سه ثبات خطی[۳] به نام‌های R_1، R_2 و R_3 به طول‌های ۱۹، ۲۲ و ۲۳ تشکیل شده است که چندجمله‌ای‌های مشخصه آن‌ها به ترتیب $1 + x + x^2 + x^5 + x^{19}$، $1 + x + x^{22}$ و $1 + x + x^2 + x^{15} + x^{23}$ می‌باشند. حالت اولیه این مولد به ۶۴ بیت کلید محرمانه[۴] و ۲۲ بیت شماره قاب[۵] که به عنوان مقدار اولیه[۶] (IV) استفاده می‌شود وابسته است.
در اینجا i-مین بیت کلید محرمانه را با K_i، i-مین بیت از سمت چپ از j-مین ثبات خطی را با $R_j[i-1]$ و i-مین بیت از مقدار اولیه را با IV_i نشان می‌دهیم.

۱.۳ تابع انتقال حالت

در رمز دنباله‌ای A5/1 ثبات‌ها نحوه انتقال (کلاک خوردن) خود را تعیین می‌کنند. بدین‌ترتیب انتقال ثبات‌ها به طور نامنظم انجام می‌شود. به طور دقیق‌تر، تابع انتقال حالت[۷]، ثبات‌ها را به صورت زیر به‌روز رسانی می‌کند:

- بیت‌های $R_1[8]$، $R_2[10]$ و $R_3[10]$ وارد تابع مد[۸] می‌شوند که مد[۹] (اکثریت) آن‌ها را می‌دهد.
- سپس، هر کدام از ثبات‌های خطی کلاک می‌خورند اگر و تنها اگر بیت ورودی‌شان به تابع مد، مساوی خروجی تابع مد باشد.

فرم نرمال جبری تابع مد به صورت زیر است:

$$\text{mahority}(x_1, x_2, x_3) = x_1x_2 + x_1x_3 + x_2x_3$$

۲.۳ الگوریتم بارگذاری کلید و مقدار اولیه

در این مرحله با توجه به الگوریتم بارگذاری کلید و مقدار اولیه[۱۰] (الگوریتم ۱) هر سه ثبات خطی مقداردهی اولیه می‌شوند. لازم به ذکر است که در ابتدا (قبل از شروع مرحله اول) هر سه ثبات خطی مقدار اولیه صفر دارند.

[۲] Global System for Mobile Communications
[۳] Linear Feedback Shift Register (LFSR)
[۴] Secret Key
[۵] Frame Number
[۶] Initial Value (IV)
[۷] update state function
[۸] majoriry
[۹] majoriry
[۱۰] Key-IV Setup

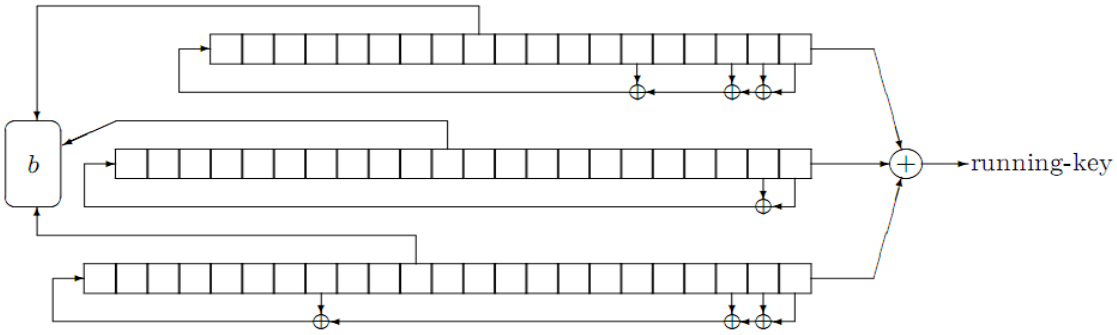
Algorithm 1 : The key-IV setup of A5/1

for $i = 1$ **to** 64 **do**
  $R_1[0] \leftarrow K_i \oplus R_1[0]$
  $R_2[0] \leftarrow K_i \oplus R_2[0]$
  $R_3[0] \leftarrow K_i \oplus R_3[0]$
  clock all 3 LFSRs
for $i = 65$ **to** 86 **do**
  $R_1[0] \leftarrow IV_{i-64} \oplus R_1[0]$
  $R_2[0] \leftarrow IV_{i-64} \oplus R_2[0]$
  $R_3[0] \leftarrow IV_{i-64} \oplus R_3[0]$
  clock all 3 LFSRs
for $i = 1$ **to** 99 **do**
  Apply update state function

۳.۳ الگوریتم تولید کلید اجرایی

مرحله دوم A5/1 که الگوریتم تولید کلید اجرایی[۱۱] نام دارد و در شکل ۱ نشان داده شده است، به صورت زیر اجرا می‌شود:

- حالت درونی ثبات‌ها با استفاده از تابع انتقال حالا به‌روز رسانی می‌شود.
- خروجی هر سه ثبات خطی با هم xor می‌شوند و یک بیت خروجی تولید می‌شود.



شکل ۱: مرحله دوم A5/1

معادلاً می‌توان گفت که پس از اینکه ۸۶ بیت کلید و مقدار اولیه به صورت خطی وارد ثبات‌ها می‌شوند (دو حلقه اول **for** الگوریتم ۱)، الگوریتم تولید کلید اجرایی ۳۲۸ بیت تولید می‌کند که ۱۰۰ بیت اول دور انداخته می‌شوند و ۲۲۸ بیت بعد، دنباله کلید اجرایی را تشکیل می‌دهند.

۴.۳ تحلیل A5/1

حمله جستجوی کامل علیه A5/1 با در دست داشتن ۶۴ بیت از دنباله خروجی، می‌تواند حالت اولیه مولد را در زمان 2^{64} به دست آورد. با استفاده از حمله حدس و تعیین[۱۲] می‌توان حالت درونی مولد را با حدس حدود ۴۱ بیت (هر بیت بیانگر یک رابطه خطی روی حالت

[۱۱] Running Key Generator Algorithm
[۱۲] guess-and-determine

درونی مولد است) به دست آورد. حمله بده‌بستان حافظه زمان، داده این کار را در زمان $۲^{۳۲}$ انجام می‌دهد اما به همان میزان حافظه و داده (بیت خروجی) نیاز دارد.

A5/2 ۴

در اینجا به معرفی مولد رمز دنباله‌ای A5/2 می‌پردازیم. این مولد، نسبت به مولد A5/1، یک ثبات خطی دیگر هم به نام $R_۴$ دارد ، که چندجمله‌ای مشخصه آن عبارت است از: $x^{۱۷} + x^{۵} + ۱$. در اینجا نیز مانند A5/1، مقدار دهی اولیه توسط **۶۴** بیت کلید محرمانه و همچنین **۲۲** بیت شماره قاب که به عنوان مقدار اولیه استفاده می‌شود انجام می‌گیرد.
مانند قسمت قبل i-مین بیت کلید محرمانه را با K_i، i-مین بیت از سمت چپ از j-مین ثبات خطی را با $R_j[i-۱]$ و i-مین بیت از کلید آشکار را با IV_i نشان می‌دهیم.

۱.۴ تابع انتقال حالت

در رمز دنباله‌ای A5/2 ثبات چهارم همواره به صورت منظم انتقال می‌یابد. اما نحوه انتقال سه ثبات دیگر توسط ثبات چهارم تعیین می‌شود. بدین‌ترتیب انتقال ثبات‌ها باز هم نامنظم است. به طور دقیق‌تر، تابع انتقال حالت، ثبات‌ها را به صورت زیر به‌روز رسانی می‌کند:

- بیت‌های $R_۴[۳]$، $R_۴[۷]$ و $R_۴[۱۰]$ وارد یک تابع مد می‌شوند و مد[۱۳] آن‌ها محاسبه می‌شود.
- ثبات $R_۱$ کلاک می خورد اگروتنهااگر $R_۴[۱۰]$ مساوی خروجی تابع مد باشد.
- ثبات $R_۲$ کلاک می خورد اگروتنهااگر $R_۴[۳]$ مساوی خروجی تابع مد باشد.
- ثبات $R_۳$ کلاک می خورد اگروتنهااگر $R_۴[۷]$ مساوی خروجی تابع مد باشد.
- سپس $R_۴$ کلاک می‌خورد.

۲.۴ الگوریتم بارگذاری کلید و مقدار اولیه

ابتدا هر چهار ثبات خطی مقدار اولیه صفر دارند. در مرحله اول طبق الگوریتم بارگذاری کلید و مقدار اولیه (الگوریتم ۲) هر چهار ثبات خطی مقداردهی اولیه می‌شوند.

Algorithm 2 : The key-IV setup of A5/2

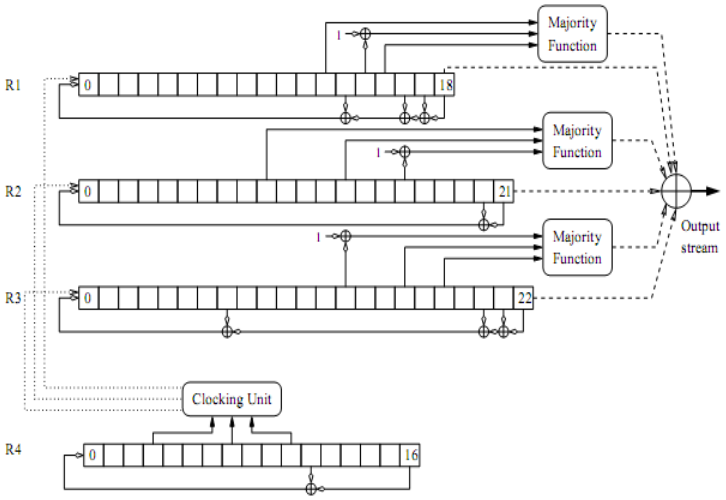
for $i = 0$ **to** 63 **do**
  $R_1[0] \leftarrow K_i \oplus R_1[0]$
  $R_2[0] \leftarrow K_i \oplus R_2[0]$
  $R_3[0] \leftarrow K_i \oplus R_3[0]$
  $R_4[0] \leftarrow K_i \oplus R_4[0]$
  clock all 4 LFSRs
for $i = 0$ **to** 21 **do**
  $R_1[0] \leftarrow IV_i \oplus R_1[0]$
  $R_2[0] \leftarrow IV_i \oplus R_2[0]$
  $R_3[0] \leftarrow IV_i \oplus R_3[0]$
  $R_4[0] \leftarrow IV_i \oplus R_4[0]$
  clock all 4 LFSRs
Set $R_1[18], R_2[16], R_3[15], R_4[10]$ to 1
for $i = 1$ **to** 99 **do**
  Apply update state function

نحوه بارگذاری مشابه A5/1 است. تفاوت قابل توجه این است که قبل از انتقال نامنظم ثبات‌ها یک بیت از هر ثبات برابر **۱** قرار داده می‌شود که به وضوح باعث کاهش امنیت می‌شود.

[۱۳] majoriry

۳.۴ الگوریتم تولید کلید اجرایی

مشابه A5/1 برای تولید یک بیت از دنباله کلید اجرایی، ابتدا ثبات‌ها به‌روزرسانی می‌شوند سپس با اعمال یک تابع فیلتر، یک بیت خروجی تولید می‌شود. برعکس A5/1، تابع فیلتر دیگر خطی نیست؛ بلکه یک تابع درجه ۲ است که در شکل ۲ قابل مشاهده است.



شکل ۲: الگوریتم تولید کلید اجرایی A5/2

۴.۴ تحلیل A5/2

با توجه به اینکه تعداد بیت‌های مفید نامعلوم حالت A5/2 برابر با ۷۸ بیت است، حمله جستجوی کامل با در دست داشتن ۷۸ بیت از دنباله خروجی، می‌تواند حالت اولیه مولد را در زمان $۲^{۷۸}$ به دست آورد. با استفاده از حمله تقسیم و حل[۱۴] می‌توان حمله بهتری با جستجوی کامل روی ۱۶ بیت مفید مجهول ثبات چهارم انجام داد. برای این کار نیاز است که به‌ازای هر حالت اولیه یک دستگاه معادلات درجه ۲ را حل نمود. این دستگاه غیرخطی را می‌توان با استفاده از تکنیک خطی‌سازی به یک دستگاه معادلات خطی شامل

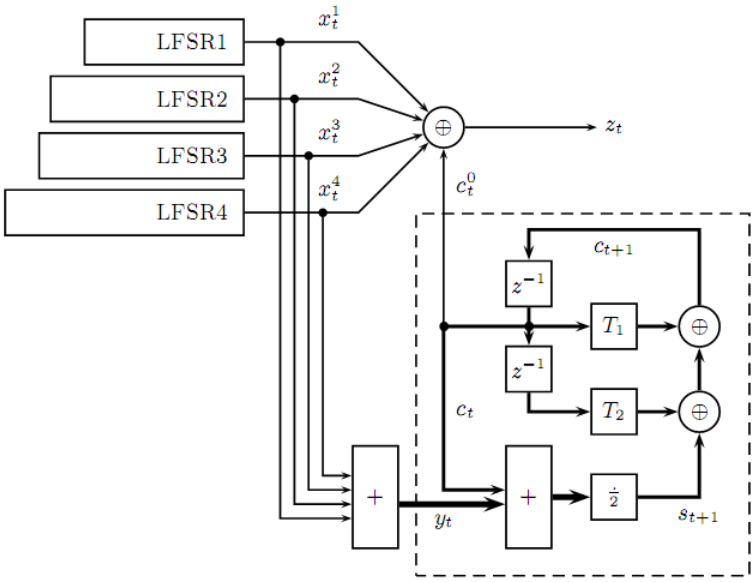
$$۱۸ + \binom{۱۸}{۲} + ۱۸ + \binom{۲۲}{۲} + ۱۸ + \binom{۲۱}{۲} = ۶۵۵$$

متغیر تبدیل کرد. تعداد بیت‌های مورد نیاز برای اعمال حمله نیز به تعداد متغیرهای خطی‌سازی شده است. با استفاده از روش گاوس به زمان $۲^{۴۴} \approx ۲^{۱۶۶۵۵^{۳}}$ نیاز است. در عمل زمان مورد نیاز را می‌توان با پیش‌پردازش از این هم کمتر کرد. حمله بده‌بستان حافظه زمان، داده علیه این مولد در زمان $۲^{۴۰/۵}$ قابل اعمال می‌باشد اما به همان میزان حافظه و داده (بیت خروجی) نیاز دارد.

E۰ ۵

مولد رمز دنباله‌ای E۰ برای رمزنگاری در ارتباطات به وسیله بلوتوث مورد استفاده قرار می‌گیرد. در اینجا صرفاً به توصیف الگوریتم تولید کلید اجرایی بسنده می‌کنیم. شکل ۳ طرز کار مولد E۰ را نشان می‌دهد.
همان‌طور که در شکل دیده می‌شود، در این مولد چهار ثبات خطی به طول‌های ۲۵، ۳۱، ۳۳ و ۳۹ وجود دارد که چند جمله‌ای‌های فیدبک آن‌ها به ترتیب به صورت زیر می‌باشد:

[۱۴] divide-and-conquer



شکل ۳: طرز کار مولد E۰

$$
\begin{aligned}
&۱ + x^{۸} + x^{۱۲} + x^{۲۰} + x^{۲۵},\\
&۱ + x^{۱۲} + x^{۱۶} + x^{۲۴} + x^{۳۱},\\
&۱ + x^{۴} + x^{۲۴} + x^{۲۸} + x^{۳۳},\\
&۱ + x^{۴} + x^{۲۸} + x^{۳۶} + x^{۳۹}.
\end{aligned}
$$

خط‌های پررنگ حامل دوبیت و خط‌های کمرنگ حامل یک بیت هستند. نماد $\oplus$ و $+$ به‌ترتیب بیانگر جمع مبنای ۲ و جمع معمولی است. همچنین $z^{-۱}$ نشان دهنده یک واحد تأخیر زمانی است؛ یعنی، دو بیت خروجی آن در لحظه t همان مقادیر ورودی آن در لحظه $t-۱$ است. بنابراین، علاوه بر ۱۲۸ بیت حالت ثبات‌ها، ۴ بیت دیگر هم استفاده می‌شود.

در هر زمان t خروجی ثبات‌ها را به ترتیب با $x_t^۱$، $x_t^۲$، $x_t^۳$ و $x_t^۴$ نشان می‌دهیم. چهار بیت حالت را به صورت

$$(\mathbf{c}_{t-۱}, \mathbf{c}_t) = (c_{t-۱}^۱, c_{t-۱}^۰, c_t^۱, c_t^۰)$$

در نظر می‌گیریم که تشکیل یک ماشین حالت متناهی می‌دهد. این ماشین حالت متناهی با دریافت دو بیت $\mathbf{y}_t$ به عنوان ورودی، به صورتی که در زیر توضیح داده می‌شود بروز می‌شود. بیت‌های خروجی ثبات‌ها با بیت $c_t^۰$ از ماشین حالت متناهی XOR می‌شود تا یک بیت از دنباله کلید اجرایی تولید شود:

$$z_t = x_t^۱ \oplus x_t^۲ \oplus x_t^۳ \oplus x_t^۴ \oplus c_t^۰ .$$

برای به‌روز رسانی ماشین حالت متناهی، همان‌طور که در شکل دیده می‌شود، ابتدا مقدار $\mathbf{c}_{t+۱} = (c_{t+۱}^۱, c_{t+۱}^۰)$ به صورت زیر محاسبه می‌شود:

$$\mathbf{c}_{t+۱} = (c_{t+۱}^۱, c_{t+۱}^۰) = \mathbf{s}_{t+۱} \oplus T_۱(\mathbf{c}_t) \oplus T_۲(\mathbf{c}_{t-۱}),$$

که در آن:

$$
\begin{aligned}
\mathbf{y}_t &= x_t^۱ + x_t^۲ + x_t^۳ + x_t^۴\\
\mathbf{s}_{t+۱} &= \lfloor (\mathbf{y}_t + \mathbf{c}_t)/۲ \rfloor
\end{aligned}
$$

که مقادیر دوبیتی به عنوان یک عدد صحیح در مجموعه $\{۰, ۱, ۲, ۳, ۴\}$ تفسیر می‌شود. همچنین $T_۱, T_۲ : \mathbb{Z}_۴^۲ \to \mathbb{Z}_۴^۲$ دوسویی‌هایی خطی هستند که به صورت زیر تعریف می‌شوند:

$$
\begin{aligned}
T_۱ &: (x_۱, x_۰) \mapsto (x_۰, x_۱),\\
T_۲ &: (x_۱, x_۰) \mapsto (x_۰, x_۱ \oplus x_۰).
\end{aligned}
$$

RC4 ۶

الگوریتم RC4 یکی از پرکاربردترین الگوریتم‌ها در زمینه‌ی رمزهای دنباله‌ای می‌باشد که از آن در پروتکل‌هایی مانند پروتکل SSL در اینترنت استفاده می‌شود. RC4 نیز از دو بخش اصلی تشکیل شده است: الگوریتم بارگذاری کلید و الگوریتم تولید کلید اجرایی. حالت داخلی این مولد شامل دو اشاره‌گر بایتی i و j و آرایه‌ی $S = (S[۰], S[۱], \cdots, S[۲۵۵])$ از بایت‌ها است که در هر لحظه تشکیل یک جایگشت از بردار $(۰, ۱, \cdots, ۲۵۵)$ می‌دهند.

۱.۶ الگوریتم بارگذاری کلید

طول کلید RC4 می‌تواند بین ۱ تا ۲۵۶ بایت باشد که با $(K[۰], \cdots, K[l-۱])$ نشان داده می‌شود. این الگوریتم مقدار اولیه ندارد و در بعضی کاربردها بعضی از بایت‌های کلید به عنوان مقدار اولیه در نظر گرفته می‌شود. در الگوریتم بارگذاری کلید، ابتدا S توسط جایگشت همانی پر می‌شود ($S[i] = i$) و سپس اعضای S مطابق الگوریتم ۳ جایگشت داده می‌شوند.

Algorithm 3 : The key setup of RC4

Input: $(K[0], \cdots, K[l-1])$ where l is key length in bytes
  for $i = 0$ **to** 255 **do**
    $S[i] \leftarrow i$
  $j \leftarrow 0$
  for $i = 0$ **to** 255 **do**
    $j \leftarrow (j + S[i] + K[i \mod l]) \mod 256$
    Swap $S[i] \leftrightarrow S[j]$

۲.۶ الگوریتم مولد دنباله کلید اجرایی

الگوریتم ۴ مولد دنباله کلید اجرایی را نشان می‌دهد.. در این الگوریتم اشاره‌گرهای i و j به دو خانه از این آرایه اشاره می‌کنند که مقدار ابتدایی آن‌ها صفر است. ابتدا مقادیر i و j به‌روز شده و سپس مقادیر $S[i]$ و $S[j]$ جابه جا می‌شوند. در نهایت الگوریتم مقدار $S[(S[i] + S[j]) \mod ۲۵۶]$ به عنوان یک بایت از دنباله کلید اجرایی معرفی می‌شود.

Algorithm 4 : The Running Key Generator of RC4

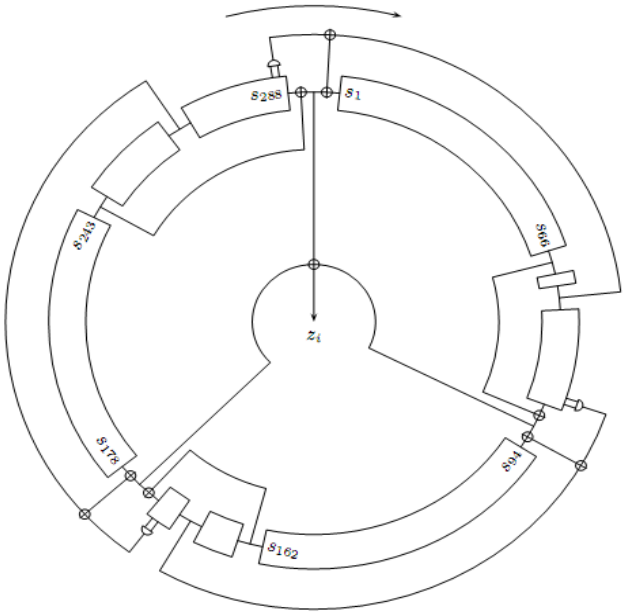
  $i \leftarrow 0$
  $j \leftarrow 0$
  for $i = 0$ **to** N **do**
    $i \leftarrow (i + 1) \mod 256.$
    $j \leftarrow (j + S[i]) \mod 256$
    Swap $S[i] \leftrightarrow S[j]$
    $Z_i \leftarrow S[S[i] + S[j] \mod 256]$
    Output Z_i

حمله جستجوی کامل به این مولد به زمان $۲۵۶! \cdot ۲۵۶^۲ \simeq ۲^{۱۷۰۰}$ نیاز دارد. بهترین حمله موجود، حالت اولیه را در زمان $۲^{۲۴۱}$ پیدا می‌کند. البته حمله‌های تمایز عملی زیادی علیه این مولد وجود دارد که دنباله خروجی آن را به راحتی از یک دنباله کاملا تصادفی تشخیص می‌دهد.

Trivium ۷

رمز دنباله‌ای Trivium برای استفاده در کاربردهای سخت‌افزاری طراحی شده است و یکی از هفت الگوریتم نهایی مسابقه eStream است. در این مولد ۸۰ بیت کلید محرمانه که آن‌ها را با $(K_۱, \cdots K_{۸۰})$ نشان می‌دهیم و همچنین ۸۰ بیت مقدار اولیه که آن‌ها را با

$(IV_1, \cdots, IV_{80})$ نشان می‌دهیم، وجود دارد که دنباله کلید اجرایی با استفاده از آن‌ها تعیین می‌شود. شکل ۴ ساختار Trivium را که شامل ۲۸۸ بیت حالت $(s_1, \cdots, s_{288})$ می‌باشد، نشان می‌دهد.



شکل ۴: طرز کار مولد Trivium

Algorithm 5 : The key-IV setup of Trivium

$(s_1, s_2, \cdots, s_{93}) \leftarrow (K_1, \cdots, K_{80}, 0, \cdots, 0)$
$(s_{94}, s_{95}, \cdots, s_{177}) \leftarrow (IV_1, \cdots, IV_{80}, 0, \cdots, 0)$
$(s_{178}, s_{179}, \cdots, s_{288}) \leftarrow (0, \cdots, 0, 1, 1, 1)$
for $i = 1$ **to** 4×288 **do**
  $t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$
  $t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$
  $t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$
  $(s_1, s_2, \cdots, s_{93}) \leftarrow (t_3, s_1, \cdots, s_{92})$
  $(s_{94}, s_{95}, \cdots, s_{177}) \leftarrow (t_1, s_{94}, \cdots, s_{176})$
  $(s_{178}, s_{179}, \cdots, s_{288}) \leftarrow (t_2, s_{178}, \cdots, s_{287})$

مولد Trivium دارای یک تابع انتقال حالت غیرخطی و یک تابع فیلتر خطی برای تولید خروجی است. بیت‌های حالت ابتدا با استفاده از بیت‌های کلید، مقدار اولیه و تعدادی بیت ثابت پر می‌شوند. سپس بیت‌های حالت به تعداد ۴×۲۸۸ بدون تولید خروجی به‌روز رسانی می‌شود. الگوریتم ۵ نحوه مقداردهی اولیه و را الگوریتم ۶ نحوه تولید دنباله کلید اجرایی را نشان می‌دهد که مولد مجاز به تولید حداکثر $N \leq ۲^{۶۴}$ است که با $z_1, \cdots, z_N$ نشان داده می‌شوند. همانطور که مشاهده می‌شود، تابع فیلتر یک تابع خطی است که ۶ بیت حالت داخلی را برای تولید یک بیت از دنباله کلید اجرایی با xor می‌کند.

Algorithm 6 : The Running Key Generator of Trivium

for $i = 1$ **to** N **do**

$t_1 \leftarrow s_{66} + s_{93}$
$t_2 \leftarrow s_{162} + s_{177}$
$t_3 \leftarrow s_{243} + s_{288}$
$z_i \leftarrow t_1 + t_2 + t_3$
$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$
$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$
$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$
$(s_1, s_2, \cdots, s_{93}) \leftarrow (t_3, s_1, \cdots, s_{92})$
$(s_{94}, s_{95}, \cdots, s_{177}) \leftarrow (t_1, s_{94}, \cdots, s_{176})$
$(s_{178}, s_{179}, \cdots, s_{288}) \leftarrow (t_2, s_{178}, \cdots, s_{287})$

۸ Grain

رمز دنباله‌ای Grain یکی دیگر از هفت الگوریتم نهایی مسابقه eStream است که برای محیط‌های سخت‌افزاری با منابع محدود طراحی شده است. در این مولد ۸۰ بیت کلید که آن‌ها را با $(K_0, \cdots, K_{79})$ نشان می‌دهیم و همچنین ۶۴ بیت مقدار اولیه (IV) که آن‌ها را با $(IV_0, \cdots, IV_{79})$ نمایش می‌دهیم، وجود دارد. این مولد از یک ثبات خطی، یک ثبات غیرخطی[15] (NFSR) و یک تابع فیلتر[16] تشکیل شده است.

۱.۸ مولد کلید اجرایی

شکل ۵ شمای الگوریتم تولید کلید اجرایی Grain را نشان می‌دهد. محتویات ثبات خطی با $(s_i, s_{i+1}, \cdots, s_{i+79})$ و محتویات ثبات غیرخطی با $(b_i, b_{i+1}, \cdots, b_{i+79})$ نشان داده می‌شوند.
چندجمله‌ای اولیه ثبات خطی از درجه ۸۰ است که به صورت زیر تعریف می‌شود:

$$f(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}.$$

که رابطه بازگشتی زیر را برای به‌روز رسانی ثبات خطی معین می‌کند:

$$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i.$$

چندجمله‌ای فیدبک ثبات غیرخطی به صورت زیر می‌باشد:

$$\begin{aligned} g(x) \quad = \quad & 1 + x^{17} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} \\ & + x^{17}x^{20} + x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} \\ & + x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} \\ & + x^{28}x^{35}x^{43}x^{47}x^{52}x^{59}\,, \end{aligned}$$

دقت کنید که چندجمله‌ای فوق صرفاً یک نمایش نمادین برای تعیین رابطه بازگشتی ثبات غیرخطی است. با توجه به اینکه بیت خروجی ثبات خطی با بیت ورودی ثبات غیرخطی xor می‌شود، چندجمله‌ای فوق بیانگر رابطه بازگشتی زیر برای به‌روز زسانی حالت ثبات غیرخطی است:

$$\begin{aligned} b_{i+80} \quad = \quad & s_i + b_{i+63} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} \\ & + b_{i+15} + b_{i+9} + b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + b_{i+15}b_{i+9} \\ & + b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} + b_{i+63}b_{i+45}b_{i+28}b_{i+9} \\ & + b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} \\ & + b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} + b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} \\ & + b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}\,. \end{aligned}$$

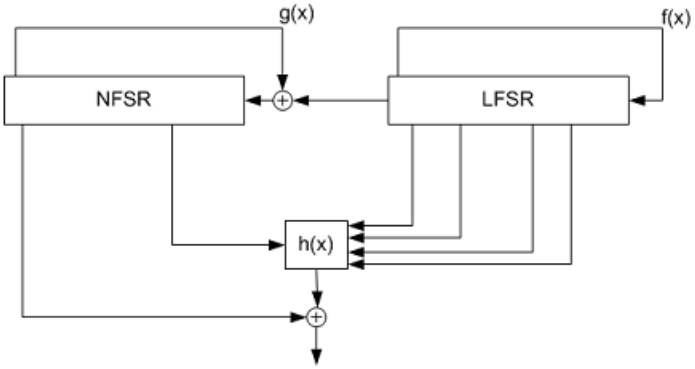
[15] Nonlinear Feedback Shift Register (NFSR)
[16] Filter Function

خروجی با اعمال یک تابع فیلتر بر روی ۱۲ بیت از بیت‌های حالت حاصل می‌شود. تابع فیلتر با استفاده از تابع بولی ۵-متغیره زیر تعریف می‌شود:

$$h(x) = x_۱ + x_۴ + x_۰x_۳ + x_۲x_۳ + x_۳x_۴ + x_۰x_۱x_۲ + x_۰x_۲x_۳ + x_۰x_۲x_۴ + x_۱x_۲x_۴ + x_۲x_۳x_۴$$

که در آن $x_۰$، $x_۱$، $x_۲$، $x_۳$ و $x_۴$ به ترتیب متناظر با بیت‌های حالت $s_{i+۳}$، $s_{i+۲۵}$، $s_{i+۴۶}$، $s_{i+۶۴}$ و $b_{i+۶۳}$ می‌باشد. هفت بیت دیگر حالت $\{b_{i+j}\}_{j\in A}$ هستند که $A = \{۱, ۲, ۴, ۱۰, ۳۱, ۴۳, ۵۶\}$ که با خروجی تابع بولی xor می‌شوند. بنابراین بیت iام دنباله کلید اجرایی به صورت زیر تولید می‌شود:

$$z_i = \sum_{j\in A} b_{i+j} + h(s_{i+۳}, s_{i+۲۵}, s_{i+۴۶}, s_{i+۶۴}, b_{i+۶۳})$$



شکل ۵: مرحله تولید کلید اجرایی Grain

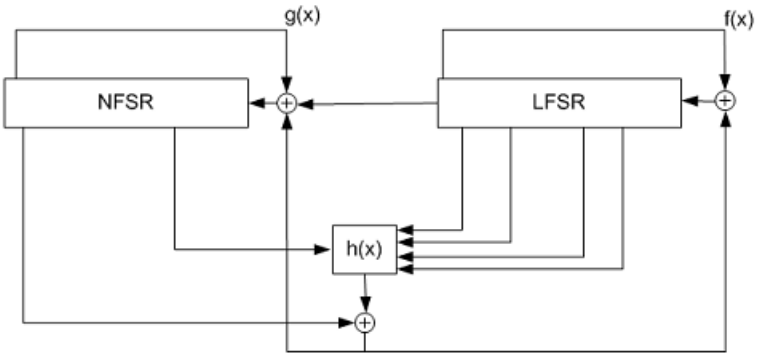
۲.۸ مرحله بارگذاری کلید و مقدار اولیه

قبل از اینکه کلید اجرایی تولید شود باید کلید و همچنین مقدار اولیه بارگذاری شوند. برای این منظور ابتدا ثبات‌های خطی و غیر خطی به صورت زیر مقداردهی می‌شوند:

$$(b_۰, b_۱, \cdots, b_{۷۹}) \leftarrow (K_۰, K_۱, \cdots, K_{۷۹}) ,$$

$$(s_۰, s_۱, \cdots, s_{۷۹}) \leftarrow (IV_۰, IV_۱, \cdots, IV_{۶۳}, ۱, \cdots, ۱) .$$

سپس مانند شکل ۶، مولد ۱۶۰ بار کلاک می‌خورد، بدون اینکه خروجی تولید شود. در واقع به جای تولید کلید اجرایی، در این مرحله خروجی تابع فیلتر خروجی با ورودی ثبات‌ها xor می‌شود.



شکل ۶: مقداردهی اولیه Grain