۱) الگوریتم های رمز متقارن

ساخت الگوریتم های رمز متقارن را می توان از دو جنبه مورد بحث و بررسی قرار داد:

- ساخت های عملی (Practical constructions): که با تکیه بر پیاده سازی عملیاتی (سرعت، سادگی، امن بودن) ارایه می شوند.
- ساخت های نظری (Theoritical constructions): که با تکیه بر اولیه های نظری (وجود توابع با خاصیت های مورد نیاز) و نه صرفاً عملیاتی ارایه می شوند.

در ادامه، بر روی ساخت های عملی متمرکز می شویم.

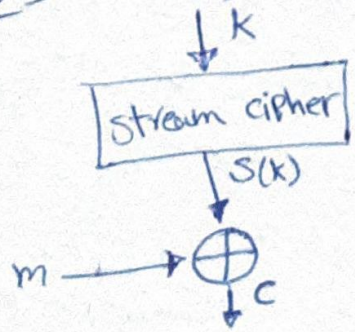
۲) رمزهای متقارن را می توان با دو روش عمده زیر تولید کرد:

۱. رمزهای قالبی (Block Cipher): پردازش پیام به صورت بلوک به بلوک.

۲. رمزهای جریانی (Stream cipher): پردازش پیام به صورت پیوسته.

۳) رمزهای جریانی

گونه ای از سیستم های رمز متقارن به حساب می آیند که در آنها برای هر واحد جریانی از پیام یک واحد جریانی کلید در نظر گرفته می شود. واحد جریانی می تواند بیت، کاراکتر، رقم و موارد مشابه باشد که به طور معمول در ادامه مباحث بیت در نظر گرفته می شود. شکل زیر، نمای کلی چنین سیستم های رمزی را به تصویر می کشد.

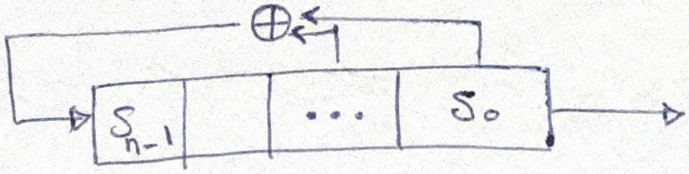


برای تولید واحد جریانی کلید ($S(k)$) به طور تاریخی در عمل از شیفت رجیستر فیدبک در خطی (Linear feedback shift Register) یا LFSR و گونه های مختلف آن استفاده می شود. در ادامه شرح مختصری از ایده LFSR ها برای تولید جریان کلید آورده شده است.

از ساختار و ایده های موجود در LFSR ها در واقع به دنبال آن هستیم که از یک کلید با طول مشخص یک کلید طولانی تر (با داشتن خاصیت شبه تصادفی یا همان کلید بودن) داشته باشیم.

LFSR (۴)

شکل زیر ساختار کلی یک LFSR نمونه را نشان می دهد.



همانطور که در شکل فوق قابل مشاهده است، هر LFSR حاوی یک آرایه n-تایی از رجیسترهاست که با S_0 و S_1 و ... و S_{n-1} نشان داده شده اند. هر رجیستر در این ساختار می تواند یک مقدار باینری را در خود جای دهد. علاوه بر این، هر LFSR یک مجموعه n-تایی از ضرایب بازخورد دارد که با c_0 و c_1 و ... و c_{n-1} نشان داده شده و می تواند یک مقدار باینری را به خود بگیرند. این ضرایب برای محاسبه تابع بازخورد و تولید بیت بازخورد مورد استفاده قرار می گیرد.

وضعیت LFSR در هر زمان به طور ساده توسط مجموع بیت های داخل رجیسترهای LFSR تعیین می شود. با هر کلاک نیز مقادیر جدید هر رجیستر از روی مقادیر فعلی، ضرایب و شیفت مقادیر به راست محاسبه می شود. به عنوان مثال، اگر در زمان t وضعیت رجیسترهای LFSR را به صورت $S_0^{(t)}$ و $S_1^{(t)}$ و ... و $S_{n-1}^{(t)}$ در نظر بگیریم، آنگاه بعد از اعمال یک کلاک وضعیت

رجیسترهای LFSR در زمان $t+1$ به صورت زیر خواهد بود:

$$s_i^{(t+1)} = s_{i+1}^{t} \quad \text{for } i=0,\dots,n-2$$

$$s_{n-1}^{(t+1)} = \bigoplus_{i=0}^{n-1} c_i s_i^{(t)}$$

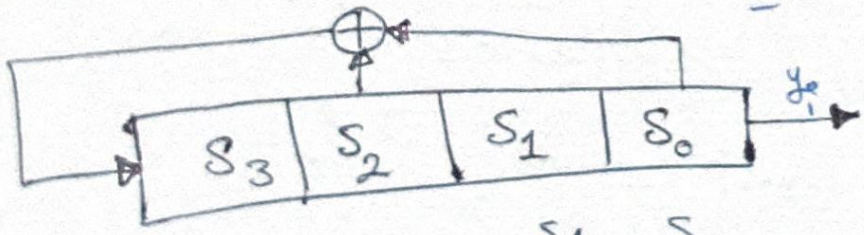
در هر کلاک، LFSR مقدار سمت راست ترین رجیستر (یعنی s_0) را تولید می کند. بنابراین اگر در زمان آغاز، رجیسترها وضعیت $s_0^{(0)}, \dots, s_{n-1}^{(0)}$ داشته باشند، n تا کلاک اول به ترتیب $s_0^{(0)}$ بعد $s_1^{(0)}$ تا $s_{n-1}^{(0)}$ را به خروجی ارسال می کند و بیت بعدی خروجی نیز $s_{n-1}^{(1)} = \bigoplus_{i=0}^{n-1} c_i s_i^{(0)}$ می باشد.

$$y_i = s_{i-1}^{(0)} \quad \text{for } i=1,\dots,n$$

$$y_i = \bigoplus_{j=0}^{n-1} c_j y_{i-n+j-1} \quad \text{for } i>n.$$

مثال برای LFSR

LFSR زیر را با وضعیت اولیه $(1,1,0,0)$ در نظر بگیرید.



$c_0 = c_2 = 1$,
$c_1 = c_3 = 0$

	s_3	s_2	s_1	s_0	
وضعیت اولیه:	0	0	1	1	
کلاک اول	1	0	0	1	$\longrightarrow$ خروجی $= 1 = y_1$
کلاک دوم	1	1	0	0	$\longrightarrow$ خروجی $= 1 = y_2$
کلاک سوم	1	1	1	0	$\longrightarrow$ خروجی $= 0 = y_3$
کلاک چهارم	1	1	1	1	$\longrightarrow$ خروجی $= 0 = y_4$

۶) نکته: یک LFSR می‌تواند حداکثر 2^n وضعیت داشته باشد. اگر از یک وضعیت ابتدایی یک n-رجیستری LFSR کار خود را شروع کند و بعد از چند کلاک، وضعیت تکراری برای آن پیش بیاید، پس واضح است که یک دنباله تکراری همواره از آن به بعد در خروجی دیده خواهد شد. بنابراین حداکثر طول دوره تکرار خروجی حاصل از یک LFSR 2^n-1 است (وضعیت صفر را از بین وضعیت‌هایی که LFSR می‌تواند بگیرد حذف کرده‌ایم)

۷) LFSR ها به نحویکه در بالا بیان گردید برای تولید دنباله شبه تصادفی (استفاده در رمزنگاری) مناسب نبوده و در واقع دنباله قابل پیش‌بینی است.

در واقع با روش‌های بالا، مهاجم به راحتی با گرفتن $2n$ بیت از خروجی می‌تواند مشخصات کامل یک LFSR درجه n (با n رجیستر) را بفهمد.

در ابتدا وضعیت اولیه (n بیت) و ضرایب بازخورد (n بیت) برای مهاجم ناشناخته است. n خروجی اول LFSR دقیقاً وضعیت اولیه LFSR را فاش می‌کند و n خروجی بعدی باعث می‌شود که مهاجم قادر به تشکیل یک دستگاه معادله خطی n معادله n مجهولی به صورت زیر شود:

$$y_{n+1} = c_{n-1} y_n \oplus \cdots \oplus c_0 y_1$$

$$\vdots$$

$$y_{2n} = c_{n-1} y_{2n-1} \oplus \cdots \oplus c_0 y_n$$

جایی که c_0 تا c_{n-1} ضرایب بازخورد (مجهولات) و y_1 تا y_{2n} خروجی حاصل از LFSR برای $2n$ کلاک از وضعیت اولیه تعیین شده است.

بنابراین کل پیکربندی LFSR بدست آمده و می‌توان تمام بیت‌های خروجی LFSR را به راحتی تولید کرد.

نکته: می‌توان نشان داد که دستگاه معادلات خطی حاصل از جمله قبل، برای یک LFSR با ماکزیمم طول (پریود 2^n) مستقل خطی است و بنابراین ضرایب بازخورد (c_i ها) به صورت یکتا بدست می‌آیند.

Trivium و RC4 نمونه‌هایی از طراحی رمزهای جریانی با استفاده از تکنیک‌های تشریح شده در بالا هستند.

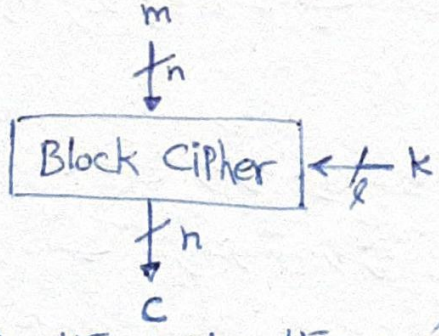
۸ رمزهای قالبی

گونه‌ای از سیستم‌های رمز متقارن به حساب می‌آیند که همانطور که از نام آنها پیداست بر روی قالب‌هایی از داده عمل می‌کنند. یک رمز قالبی در حالت کلی دارای دو مولفه اصلی است:

۱. اندازه قالب که آن را با n نشان می‌دهیم،

۲. و اندازه کلید یا طول کلید که آن را با $|k|$ نشان می‌دهیم.

شکل زیر نمای کلی چنین سیستم‌های رمزی را نشان می‌دهد:



برای یک کلید داده شده، یک رمز قالبی با اندازه قالب n، مجموعه فضای پیام $M=\{0,1\}^n$ (2^n ورودی) را به فضای متن رمزی $C=\{0,1\}^n$ (2^n خروجی) به نحوی تبدیل می‌کند که هر خروجی ممکن فقط و فقط یکبار تولید شود.

در واقع این تبدیل، یک جایگشت از مجموعه ورودی‌هاست و هنگامی که کلید تغییر کند، جایگشت‌های متفاوتی خواهیم داشت. بنابراین یک رمز قالبی، روشی برای تولید خانواده‌ای از جایگشت‌هاست که توسط کلید شاخص گذاری می‌شود.

۹) مفهوم دنباله‌زنی (Padding):

در رمزنگار قالبی، از آنجایی که ورودی (پیام) می‌بایست به بلوک‌هایی از اندازه n شکسته شوند، ممکن است بلوک آخر طولی کمتر از n داشته و می‌بایست این قطعه را به نحو مناسبی توسعه دهیم که طول آن n شود. به چنین عملی اصطلاحاً دنباله‌زنی یا پدینگ اطلاق می‌شود.

لازم به ذکر است که دنباله‌زنی می‌بایست به نحوی صورت گیرد که فاقد ابهام بوده و بتوانیم با اطمینان ۱۰۰ درصد به هنگام رمزگشایی پدینگ انجام شده را خنثی کنیم.

ارایه یک راهکار برای پدینگ: اضافه کردن یک بیت ۱ و به تعداد لازم بیت ۰ در انتهای پیام تا اینکه طول پیام مضربی از طول قالب گردد.

سؤال: آیا می‌توان راهکار بالا را با حذف بیت ۱ همچنان دنباله‌زنی معتبری دانست؟

حل: اگر آخرین قالب متن رمزگشایی شده تمام صفر باشد، ابهام پیش می‌آید که چند بیت از قالب آخر جزء متن رمزگشایی شده باید لحاظ شود و چند بیت به عنوان دنباله‌زنی است و باید حذف شود.

۱۰) رمزگذاری یک رمز قالبی

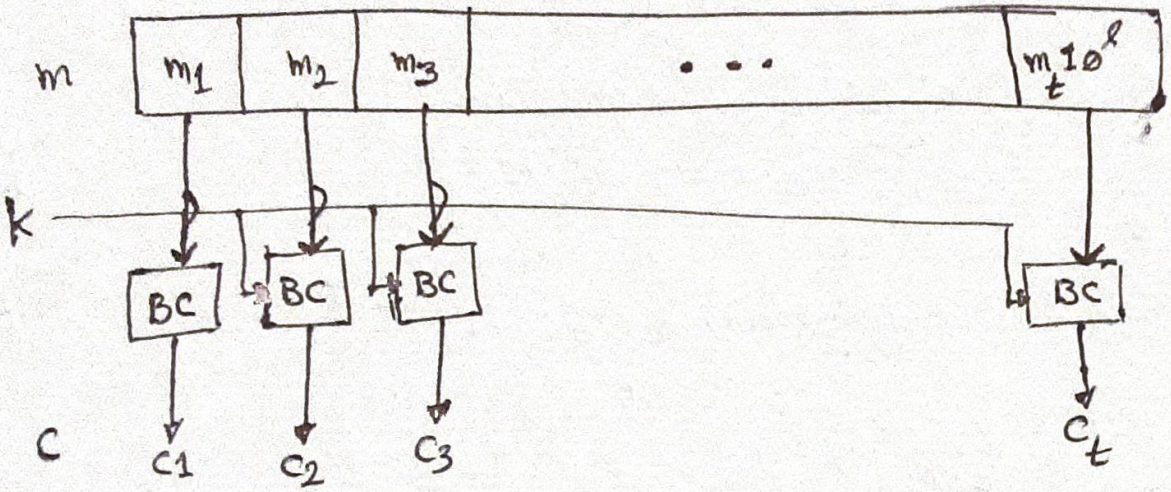
ساده‌ترین روش استفاده از یک رمز قالبی، اعمال آن به قالب‌های متن اصلی است که به نحو زیر انجام می‌شود:

۱. ابتدا با استفاده از دنباله‌زنی مناسب، طول پیام m را به مضربی از طول قالب می‌رسانیم. فرض کنید پیام دنباله‌زنی شده m' حاصل می‌شود.

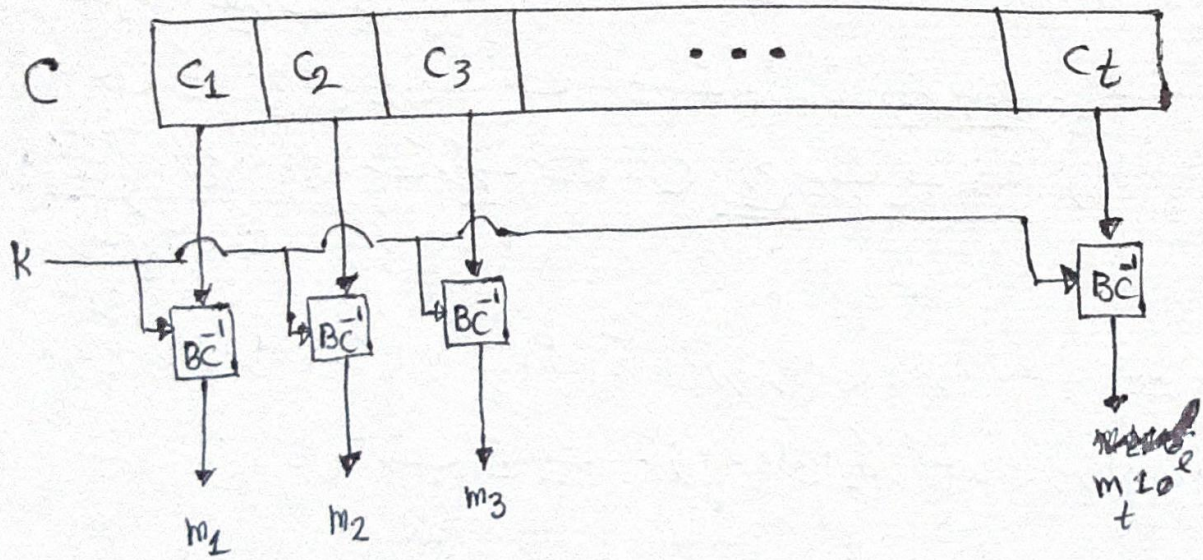
۲. در گام بعدی، پیام m' را به قالب‌های n-بیتی m_i تفکیک می‌کنیم، جایی که $1 \le i \le \frac{|m'|}{n}$ می‌باشد.

۳. با اعمال رمز قالبی روی هر قالب m_i، متن رمزی معادل آن c_i را بدست می‌آوریم.

اشکال زیر به ترتیب نمای کلی الگوریتم رمزگذاری و الگوریتم رمزگشایی رمز قالبی مشروح شده در بالا را نشان می‌دهند.



(نمای کلی الگوریتم رمزگذاری)



(نمای کلی الگوریتم رمزگشایی)

(۱۱) روش های طراحی رمزهای قالبی

دو روش کلی برای طراحی رمزهای قالبی عبارتند از:

۱) شبکه‌های جانشینی - جایگشتی یا SPN:
(Substitution-Permutation Network)

ایده این روش، ساخت رمزهای قالبی با طول قالب بزرگ از تعداد زیادی رمز قالبی با طول قالب کوچکتر است.

۲) شبکه‌های فایستلی (Feistel Network): ایده این روش، ساخت رمزهای قالبی با استفاده از توابع شبه تصادفی (به جای جایگشت شبه تصادفی که اساس روش‌های جانشینی - جایگشتی است) می‌باشد.

در ادامه خلاصه‌ای از هر دو روش فوق آورده شده است.

(۱۲) شبکه جانشینی - جایگشتی

شانون یک طرح کلی برای ساخت جایگشت شبه تصادفی با طول قالب بزرگ از جایگشت شبه تصادفی با طول قالب کوچک تحت عنوان طرح پرشانی - پخش (Confusion-Diffusion) را مطرح کرد.

فرض کنید می‌خواهیم یک جایگشت شبه تصادفی (رمز قالبی) با طول قالب ۱۲۸ از ۱۶ جایگشت شبه تصادفی با طول قالب ۸ بیت بسازیم. برای اینکار گام‌های زیر را دنبال می‌کنیم:

۱) ابتدا ورودی $x \in \{0,1\}^{128}$ را به ۱۶ قسمت ۸ بیتی متوالی $x_1, x_2, \dots, x_{16}$ تقسیم می‌کنیم.

۲) سپس با استفاده از ۱۶ جایگشت شبه تصادفی با طول قالب ۸ به صورت زیر جایگشت شبه تصادفی را محاسبه می‌کنیم که به این عمل پرشانی (یا Confusion) گفته می‌شود.

$$E_K(x) = E_K^1(x_1) \| E_K^2(x_2) \| \dots \| E_K^{16}(x_{16})$$

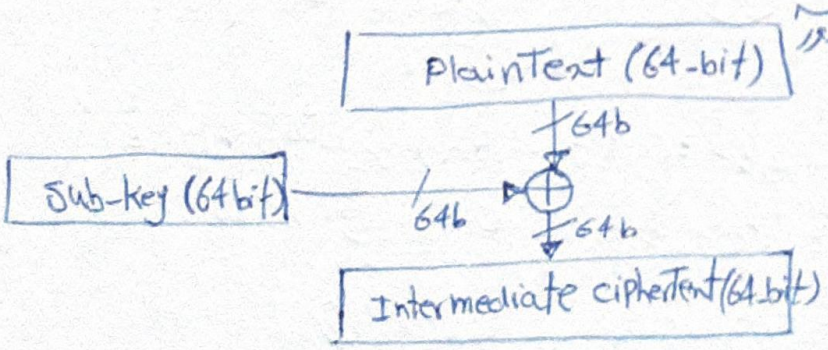
۳ جایگشت بدست آمده از مرحله قبل، شبه تصادفی نیست. زیرا اگر ورودی x و x' تنها در یک بیت اختلاف داشته باشند، $E_k(x)$ و $E_k(x')$ تنها در یک بایت ~~مرحله~~ اختلاف خواهند داشت. بنابراین در این گام می‌بایست بیت‌های خروجی E_k^i ها را باهم قاطی کرده و یک جایگشت از آنها در نظر بگیریم. به این عمل اصطلاحاً پخش (یا Difusion) گفته می‌شود.

۴ مرحله ۲ و ۳ (پریشانی و پخش) که باهم دور (Round) نامیده می‌شوند را چندین بار تکرار کنیم. در واقع این گام تضمین می‌کند که اعمال تغییرات کوچک در ورودی باعث اعمال تغییرات اساسی در خروجی می‌شود.

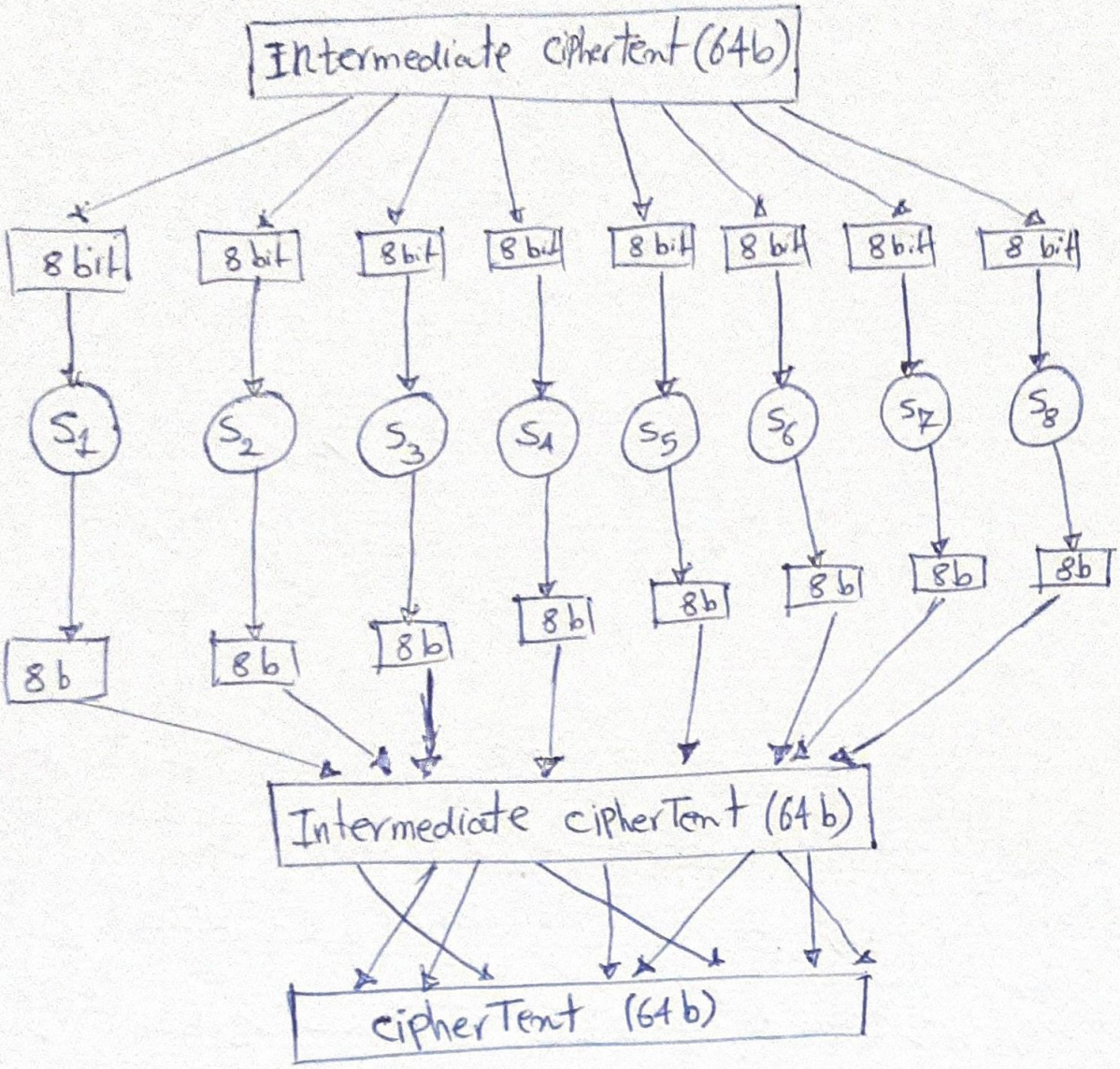
شبکه‌های جانشینی-جایگشتی از اجرای مستقیم طرح پریشانی-پخش حاصل می‌شود. تفاوت اصلی این طرح‌ها با طرح پریشانی-پخش آن است که در اینجا از جایگشت‌های ثابت (شبه تصادفی) استفاده می‌شود که اصطلاحاً S-box (subsitution-Box) نام دارند.

در نظر گرفتن S-box ها نه تنها کار پیاده‌سازی را راحت‌تر و اجرای آن را سریعتر می‌کند بلکه باعث می‌شود با انتخاب مناسب اجزای رمز قالبی، بتوان آنها را امن‌تر نیز طراحی کرد.

طریقه عملکرد شبکه‌های جانشینی-جایگشتی (SPN): یک SPN با طول بلاک ۶۴ بیت که بر پایه مجموعه S-boxهای ۸ بیتی S_1 و S_2 و ... و S_8 ساخته شده را در نظر بگیرید. شکل زیر چنین ساختی را (برای یک دور) نشان می‌دهد.



ادامه شکل صفحه بعد



(یک دور از شبکه SPN)

همانطور که در شکل بالا مشخص است، سه عمل اصلی در هر دور به قرار زیر است:

۱. عمل ترکیب کلید (key mixing): در این عمل پیام اصلی (x) ۶۴ بیتی با زیرکلید (sub-key) مربوط به دور جاری ترکیب می‌شود (یعنی $x = x \oplus k$) (k ۶۴ بیتی)

۲. عمل جانشینی (subsitution): در این عمل x مرحله قبل را دریافت و با استفاده از s-boxهای موجود عمل جانشینی را به صورت زیر انجام می‌دهد:

$$x = S_1(x_1) \| S_2(x_2) \| \dots \| S_8(x_8)$$

۳. عمل جایگشت (Permutation): در این عمل، بیت‌های x بدست آمده از مرحله قبل را جایگشت داده و به عنوان خروجی دور در نظر می‌گیریم.