

رمزنگاری

و امنیت شبکه

جلد نخست

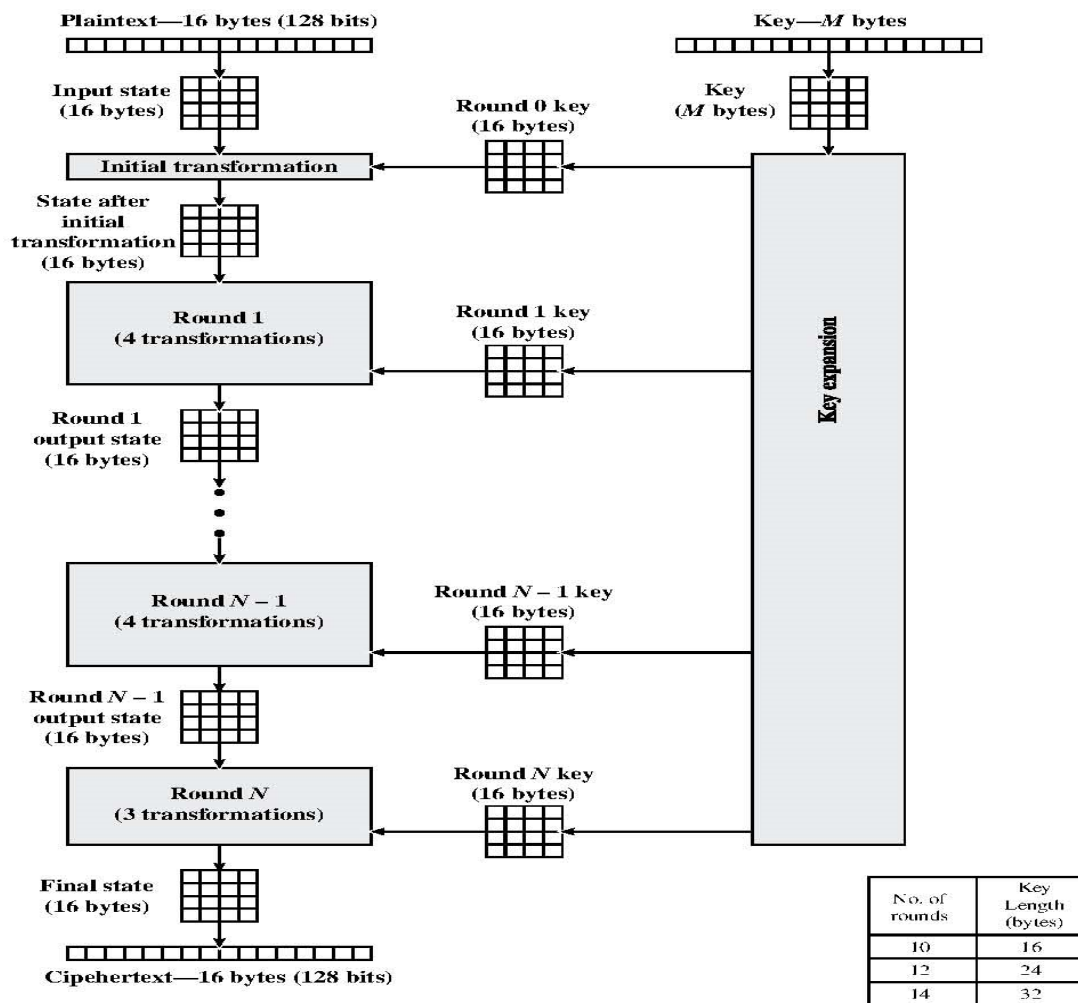


Figure 1. AES Encryption Process

برگردان: علیرضا اصغری

بهروز فروزان

رمزنگاری و امنیت شبکه

جلد نخست

بهروز فروزان

برگردان: علیرضا اصغری



سرشناسه	: فروزان، بهروز، ۱۳۲۳ -
	Forouzan, Behrouz A
عنوان و نام پدیدآور	: رمزنگاری و امنیت شبکه/ بهروز فروزان؛ برگردان علیرضا اصغری.
مشخصات نشر	: سنندج: علیرضا اصغری، ۱۳۹۵ -
مشخصات ظاهری	: ۲ ج.: مصور، جدول، نمودار.
شابک	: 978-600-04-5257-5
وضعیت فهرست نویسی	: فیپا
یادداشت	: عنوان اصلی: Introduction to cryptography and network security, 2008.
موضوع	: شبکه‌های کامپیوتری -- تدابیر ایمنی
موضوع	: Computer networks -- Security measures
موضوع	: رمزنگاری
موضوع	: Cryptography
شناسه افزوده	: اصغری، علیرضا، ۱۳۵۵ -، مترجم
رده بندی کنگره	: ۱۳۹۵ ر۸۴۹/ف۵۹/۵۱۰۵ TK
رده بندی دیویی	: ۰۰۵/۸
شماره کتابشناسی ملی	: ۴۲۷۹۲۴۱

رمزنگاری و امنیت شبکه / بهروز فروزان

برگردان : علیرضا اصغری

ویراستار: پرویز گل‌پسندی

ISBN: 978-600-04-5257-5

۱۱	دیباچه
----	-------	--------

۱۳	فصل ۱ پیشگفتار
----	-------	----------------

۱۵	چشم اندازهای امنیتی	۱-۱
۱۶	حملات	۲-۱
۲۰	سرویس ها و راهکارها	۳-۱
۲۵	راه کارها	۴-۱
۲۹	سایر قسمت های کتاب	۵-۱
۳۰	وب سایت پیشنهادی	۶-۱
۳۱	چکیده	۷-۱
۳۲	تمرین ها	۸-۱

۳۷	فصل ۲ ریاضیات رمزنگاری
----	-------	------------------------

۳۸	۱-۲ حساب عدد صحیح
۴۹	۲-۲ حساب پیمانه ای
۶۴	۳-۲ ماتریس ها
۶۹	۴-۲ تجانس خطی
۷۲	۵-۲ وب سایت پیشنهادی
۷۳	۶-۲ چکیده
۷۵	۷-۲ تمرین ها

۸۳	فصل ۳ رمزنگاری کلید متقارن سنتی
----	-------	---------------------------------

۸۴	۱-۳ مقدمه
۹۰	۲-۳ رمزهای جایگزین
۱۱۶	۳-۳ رمزهای مبتنی بر ترانهش
۱۲۵	۴-۳ رمزهای جریانی و بلوکی
۱۲۸	۵-۳ وب سایتهای پیشنهادی

۱۲۸	۳-۶ چکیده
۱۳۱	۳-۷ تمرین‌ها
۱۳۹	فصل ۴ ساختارهای جبری
۱۴۰	۴-۱ ساختارهای جبری
۱۵۴	۴-۲ میدان های $GF(2^n)$
۱۶۷	۴-۳ چکیده
۱۶۹	۴-۴ تمرین‌ها
۱۷۳	فصل ۵ مقدمه‌ای بر رمزهای کلید متقارن پیشرفته
۱۷۴	۵-۱ رمزهای بلوکی پیشرفته
۲۰۴	۵-۲ رمزهای جریانی پیشرفته
۲۱۲	۵-۳ وب سایت‌های پیشنهادی
۲۱۳	۵-۴ چکیده
۲۱۵	۵-۵ تمرین‌ها
۲۱۹	فصل ۶ استاندارد رمزگذاری داده‌ها (DES)
۲۱۹	۶-۱ مقدمه
۲۲۱	۶-۲ ساختار DES
۲۳۷	۶-۳ تجزیه و تحلیل DES
۲۴۶	۶-۴ DES چندگانه
۲۵۰	۶-۵ امنیت DES
۲۵۱	۶-۶ وب سایت‌های پیشنهادی
۲۵۳	۶-۷ چکیده
۲۵۴	۶-۸ تمرین

۲۵۷	فصل ۷ استاندارد رمزنگاری پیشرفته (AES)
۲۵۷	۱-۷ مقدمه
۲۶۴	۲-۷ تغییرات
۲۷۵	۳-۷ گسترش کلید
۲۸۱	۴-۷ رمز نویسی
۲۸۵	۵-۷ مثال ها
۲۸۷	۶-۷ تجزیه و تحلیل AES
۲۸۹	۹-۷ چکیده
۲۹۱	۱۰-۷ تمرین ها

۲۹۵	فصل ۸ رمز نویسی با استفاده از رمزهای کلید متقارن مدرن
۲۹۵	۱-۸ کاربرد رمزهای بلوکی پیشرفته
۳۱۱	۲-۸ کاربردهای رمزهای جریانی
۳۱۸	۳-۸ سایر چالش ها
۳۱۹	۴-۸ وب سایت های پیشنهادی
۳۲۰	۵-۸ چکیده
۳۲۲	۶-۸ تمرین ها

پیشکش به مادر و خواهران گرامیم، به پاس زحماتی که در این راه
کشیدند و مرایاری نمودند.

دیباچه

پیشینه نگرانی از امنیت، به سپیده دم تمدن بشری بازمی گردد. امنیت، آسایش، پاسداری، حفاظت و دیگر مترادف های لغوی این واژه، بیانگر معانی ای هستند که ابزارهای بهره گیری از آن در همه ی آفریده های ذی شعور به ودیعه نهاده شده است و پایداری نسبی آن موجب دستیابی به فراغت های ذهنی و نبود آن به همان اندازه موجب تشویش و تشتت خاطر در زندگی این موجودات خواهد بود. حال که به انسان و طیف گسترده ی مسائل امنیتی وی در اشکال گوناگونش می رسیم؛ با در نظر گرفتن تفاوت های وی با سایر موجودات زنده، درمی یابیم که انسان برای بقا به نیروهای برتر وجود خود مانند خلاقیت، ابتکار، آینده نگری و ... چنگ می اندازد تا با بهره گیری از این نهاده ی ذاتی، زندگی خود را از غارنشینی شروع و تا بدان جا رسانده است که امروزه دست در تسخیر کرات ناشناخته برده و در این سیر زمانی گنجینه ای مملو از ثروت دانش را برای آیندگان شکل داده است.

انسان در پرداختن به مقوله ی امنیت، شاخه های گوناگونی در دانش های مربوط به آن ایجاد کرده است. رمزنگاری یکی از زیرشاخه ها و مباحث شایان توجه در دانش های مختلف و مطرح امروزی ماست که در درون خود مباحث گسترده ای را دنبال می کند.

نمود عینی دانش رمزنگاری، امروزه آن چنان با علم کامپیوتر و فناوری های مرتبط با آن درهم تنیده شده است که گویی رمزنگاری نمود خود را فقط درون این دانش نو عینیت می بخشد. همچنین می توان ادعا کرد که این مقوله جایگاه ویژه ای در علوم انسانی نیز داشته است طوری که جای جای ادبیات منظوم ما با رمز و رمزنگاری مژدین شده است. اگر از تشبیهات و استعارات و کنایات، که هر کدام با صوری از رمز، بیانگر تصویری از هویت معشوق می باشند بگذریم؛ از معانی و واژه هایی می توان نام برد که در اشعار آمده و دریافت معنای اصلی آنها جز با کشف رمز امکان پذیر نیست و بیان مطلوب آن، به شیوه رمز و در قالب حروف ابجد است که با مراجعه به آثار شعرا می توان به وفور از این دست رمزنگاری ها یافت.

از نخستین شیوه‌های که بشر برای ارسال پیام و خواسته خود ابداع کرده است، مثل بهره‌گیری از ایجاد دود در معنا و کدهای خاص خود تا پیشرفته‌ترین روش‌های رمزنگاری در زندگی بشر امروزی مثل Dual ECC، همه و همه در نهایت، بیانگر غرض اصلی انسان با بهره‌گیری از هوش سرشار خود در رساندن اسرار در پرتو نهان سازی آن‌هاست که هیچ‌گاه حاضر نشده است اجازه دهد تا جبهه مقابل از کشف آن بهره جوید و هرگاه نیز احساس کرده باشد که جبهه یا فرد مقابل به رمزهای وی دست‌یافته و یا با این احساس که ایجاد رمز جدید ضرورت پیدا کرده است؛ همواره در مسیر تلاش برای بهینه‌سازی رمزها، به نوآوری‌های شگفت‌آوری دست یافته است و این دگرگونی و گوناگونی در ایجاد شیوه‌های رمزنگاری، دیواره‌ی سترگ و محکمی به وجود آورده است که امکان شکستن آن، یا محال است یا دست کم بسیار سخت و دشوار. جالب اینکه به موازات تولید دانش نو در رمزنگاری، شیوه‌های رمزشکنی و تحلیل رمز نیز در پیشرفت و حرکت شانه به شانه با آن بوده است.

بزرگواران محترم! کتابی که در پیش رو دارید برگردان بخش نخست کتاب رمزنگاری و امنیت شبکه آقای بهروز فروزان است که جلد دوم آن در آینده‌ی نزدیک به خدمت شما ارجمندان ارائه می‌گردد. چشم‌داشت حقیر از شما بزرگواران گرامی این است که بر کاستی‌های فراوان آن چشم فرو بسته و مزیت اندک آن را بزرگ پندارید. امید است که در سایه راهنمایی اندیشمندانه و به دلگرمی و پشتوانه دانش شما، جلد دوم این کتاب بیش از پیش پربار و درخور شما بزرگواران باشد.

علیرضا اصغری

مرداد ۹۴

Books2ara@yahoo.com

Books2ara@gmail.com

Books2ara@outlook.com

فصل اول

پیشگفتار

چشم‌انداز

فصل اول چشم‌اندازهای زیر را دنبال می‌کند:

- ✱ تعریف سه هدف امنیتی.
- ✱ تبیین حملات امنیتی، که چشم‌انداز امنیتی را تهدید می‌کند.
- ✱ معرفی سرویس‌های امنیتی و تشریح ارتباط این سرویس‌ها با چشم‌انداز امنیتی.
- ✱ تعیین سازوکارهای امنیتی جهت تأمین سرویس‌های امنیتی.
- ✱ معرفی دو کاربرد، رمزنگاری و استتار، جهت پیاده‌سازی سازوکارهای امنیتی.

مقدمه

ما در عصر اطلاعات زندگی می‌کنیم. اهمیت پاسداری از اطلاعات به عنوان ضرورتی گریزناپذیر در زندگی امروزی تا حدی مطرح است که می‌توان گفت که اطلاعات، دارایی باارزشی مانند سایر دارایی‌هاست و ما مکلف به حفظ آن در برابر حمله و دستبرد هستیم.

برای تأمین امنیت لازم است تا اطلاعات از دسترسی‌های غیرمجاز^۱ دور بماند (محرمانگی) و از اعمال تغییرات غیرمجاز در آن جلوگیری شود (یکپارچگی) و هنگام نیاز، فرد مجاز بتواند به اطلاعات دسترسی داشته باشد.

تا چند دهه قبل، اطلاعات جمع‌آوری شده توسط سازمان‌ها به صورت پرونده‌های مکتوب نگهداری می‌شد و طبقه‌بندی این پرونده‌ها به وسیله‌ی محدود کردن دسترسی به آن‌ها، تنها برای افراد انگشت شمار مورد اعتماد درون سازمان امکان‌پذیر بود. بدین ترتیب فقط تعداد اندکی از افراد، مجاز به دستیابی به محتوای این پرونده‌ها بودند. در دسترس بودن پرونده هم با گماردن دست کم یک نفر، که همیشه و در هر زمانی به این پرونده دسترسی داشت تأمین می‌گردید.

با پیدایش رایانه، ذخیره اطلاعات الکترونیکی شد و به جای ذخیره کردن اطلاعات در ابزارها و وسایل فیزیکی، در رایانه‌ها ذخیره شد. با این حال سه نیاز امنیتی تغییر نکرد، به این معنی که پرونده‌های ذخیره شده در رایانه هم نیازمند طبقه‌بندی، یکپارچگی و قابلیت دسترسی برای افراد مجاز هستند. اما تأمین نیازهای مذکور به شکلی متفاوت و چالش برانگیزتر متجلی گردید.

در دو دهه گذشته شبکه‌های رایانه‌ای، انقلابی در عرصه استفاده از اطلاعات ایجاد کردند. اکنون اطلاعات توزیع شده‌اند و افراد مجاز می‌توانند با استفاده از شبکه‌های رایانه‌ای، از راه دور اطلاعات را ارسال و بازیابی کنند. اگرچه هنوز هم سه نیاز امنیتی محرمانگی، یکپارچگی و در دسترس بودن تغییر نکرده‌اند، اما اکنون دارای ابعاد جدیدی شده‌اند که نه تنها اطلاعات در هنگام ذخیره شدن در رایانه می‌بایست محرمانه باقی بمانند، بلکه باید راهی برای حفظ محرمانگی آن‌ها در هنگام انتقال اطلاعات از رایانه‌ای به رایانه‌ی دیگر نیز وجود داشته باشد.

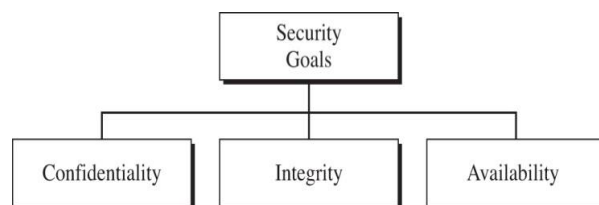
در این فصل، نخست به بحث در مورد چشم‌انداز اصلی امنیت اطلاعات می‌پردازیم، سپس به این مسئله که چگونه حملاتی می‌تواند این چشم‌انداز را تهدید کند نگاهی می‌اندازیم. در ادامه به بحث و بررسی سرویس‌های امنیتی مرتبط با چشم‌اندازهای امنیتی می‌پردازیم و سرانجام، سازوکارهایی را برای تأمین سرویس‌های امنیتی تعیین می‌کنیم و روش‌هایی را که می‌توانند در اجرای این سازوکارها به کاربرده شوند، معرفی می‌کنیم.

=====

¹ Unauthorized

۱-۱- چشم انداز امنیتی

اجازه دهید نخست به بحث در مورد سه هدف امنیتی بپردازیم؛ قابلیت اعتماد، یکپارچگی و در دسترس بودن (شکل ۱-۱).



شکل ۱-۱: رده بندی چشم اندازهای امنیتی

قابلیت اعتماد^۱

شاید رایج ترین جنبه امنیت اطلاعات، قابلیت اعتماد باشد، زیرا لازم است از اطلاعات سری حفاظت کنیم. سازمان ها می بایست خود را از فعالیت بد خواهان های که قابلیت اعتماد اطلاعات آن ها را به مخاطره می اندازد، پاسداری نمایند. پنهان کردن اطلاعات حساس وظیفه بنیادین در ارتش است؛ در صنعت، پنهان نمودن برخی از اطلاعات از رقبا برای فعالیت سازمان، حیاتی تلقی می گردد؛ در بانکداری، اطلاعات حساب مشتریان می بایست سری نگاه داشته شوند.

همان گونه که بعداً در این فصل خواهیم دید، قابلیت اعتماد نه تنها برای ذخیره اطلاعات، بلکه برای انتقال آن نیز به کار می رود. وقتی اطلاعاتی را که باید ذخیره شود به یک رایانه دور ارسال می کنیم یا هنگامی که اطلاعات را از یک رایانه دور دریافت می کنیم؛ در حین انتقال، باید اطلاعات را از دید دیگران پنهان نماییم.

یکپارچگی^۲

اطلاعات همواره و به طور مداوم تغییر می کنند. در بانک، وقتی مشتری ای پولی به حساب خود واریز یا از آن برداشت می کند، تراز حساب او باید تغییر کند. یکپارچگی یعنی انجام تغییرات به وسیله افراد مجاز و سازوکارهای مجاز. نقض یکپارچگی لزوماً نتیجه یک عمل خراب کارانه نیست. یک وقفه در کار سیستم مثلاً به علت افزایش ولتاژ برق، ممکن است باعث ایجاد تغییراتی ناخواسته در بخشی از اطلاعات شود.

=====

¹ Confidentiality

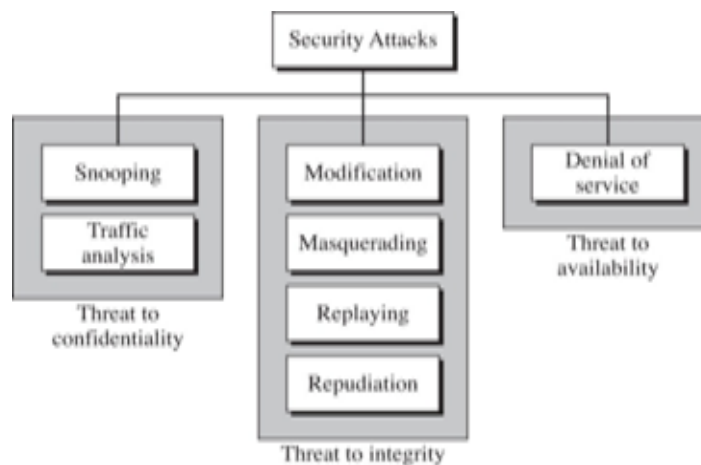
² Integrity

در دسترس بودن^۱

سومین رکن امنیت اطلاعات در دسترس بودن است. افراد مجاز باید بتوانند به اطلاعات جمع‌آوری و ذخیره شده به وسیله یک سازمان دسترسی داشته باشند. اگر اطلاعات در دسترس نباشد، بی‌فایده است. اطلاعات باید به طور مداوم تغییر کند، این بدین معناست که اطلاعات می‌بایست در دسترس افراد مجاز باشد. عدم دسترسی به اطلاعات می‌تواند به اندازه عدم وجود طبقه‌بندی یا یکپارچگی برای یک سازمان زیان‌بار باشد. تصور کنید، اگر مشتریان بانکی نتوانند به حساب‌های خود برای نقل و انتقال پول دسترسی داشته باشند چه سرنوشتی در انتظار بانک خواهد بود.

۲-۱- حملات

سه هدف امنیتی طبقه‌بندی، یکپارچگی و در دسترس بودن، ممکن است مورد تهدید حملات امنیتی قرار گیرند. اگرچه در ادبیات از شیوه‌های گوناگونی برای دسته‌بندی حملات استفاده می‌شود، اما نخست این تهدیدها را به گروه‌های مرتبط با چشم‌انداز امنیتی تقسیم‌بندی می‌کنیم؛ سپس بر مبنای تأثیراتشان بر سیستم، به دو مقوله گسترده‌تر تقسیم می‌کنیم. شکل ۲-۱ نخستین رده‌بندی را نشان می‌دهد.



شکل ۲-۱: رده‌بندی حملات مرتبط با چشم‌اندازهای امنیتی

=====

^۱ Availability

حملات تهدیدکننده طبقه‌بندی

به طور کلی دو نوع حمله، طبقه‌بندی اطلاعات را تهدید می‌کند: ۱- تجسس ۲- تجزیه و تحلیل نقل و انتقال اطلاعات.

تجسس^۱

تجسس عبارت است از دسترسی غیرمجاز به داده‌ها یا مختل کردن جریان داده‌ها. به عنوان مثال، فایلی که به وسیله اینترنت منتقل می‌شود ممکن است حاوی اطلاعات محرمانه باشد و این امکان وجود دارد که شخص غیرمجاز جریان ارسال اطلاعات را متوقف نموده و از محتوی آن به نفع خود استفاده کند. برای پیشگیری از تجسس، می‌توان با استفاده از روش‌های رمزنگاری که در این کتاب به آن خواهیم پرداخت، داده‌ها را برای جاسوس، غیرقابل فهم نمود.

تجزیه و تحلیل نقل و انتقال اطلاعات^۲

اگرچه رمزنویسی داده‌ها ممکن است باعث غیرقابل فهم شدن آن برای جاسوس شود، اما جاسوس می‌تواند با کنترل نقل و انتقالات آنلاین، اطلاعات دیگری به دست آورد. به عنوان مثال می‌تواند نشانی الکترونیکی (نظیر آدرس ایمیل) فرستنده و گیرنده را پیدا کند و یا می‌تواند با تولید زوج - درخواست و پاسخ - ماهیت فعالیت‌ها را حدس بزند.

حملات تهدیدکننده یکپارچگی

حملات گوناگونی، مانند: دستکاری، جعل هویت، بازپخش و انکار، می‌توانند یکپارچگی داده‌ها را تهدید کنند.

دستکاری^۳

پس از متوقف کردن ارسال جریان داده یا دسترسی به آن‌ها، مهاجم می‌تواند اطلاعات را به نفع خود تغییر دهد. به عنوان مثال، مشتری پیامی مبنی بر انجام عملیات بانکی به بانک ارسال می‌کند. مهاجم ارسال پیام را متوقف نموده و نوع عملیات بانکی را به سود خود تغییر می‌دهد. توجه داشته باشید که

=====

¹ Snooping

² Traffic Analysis

³ Modification

گاهی اوقات مهاجم برای آسیب رساندن به سیستم یا بهره‌برداری از پیام، تنها لازم است پیام را پاک کند یا دریافت آن را به تأخیر بیندازد.

جعل هویت^۱

جعل هویت زمانی اتفاق می‌افتد که مهاجم، خود را به جای شخص دیگری جا می‌زند. به عنوان مثال، ممکن است مهاجم کارت بانکی و رمز مشتری بانک را بدزد و طوری وانمود کند که او همان مشتری است. گاهی اوقات مهاجم خود را به جای مقصد جا می‌زند. به عنوان مثال، مشتری بانک سعی می‌کند با بانک ارتباط برقرار کند اما سایت دیگری در جای دیگر طوری وانمود می‌کند که انگار سایت بانک است و به این وسیله، اطلاعاتی را از مشتری به دست می‌آورد.

بازپخش^۲

بازپخش، نوع دیگری از حملات است. مهاجم کپی پیام‌های ارسالی کاربر را به دست آورده و می‌کوشد که بعداً آن را دوباره ارسال نماید. به عنوان مثال، شخصی به بانک پیامی ارسال می‌کند و از بانک درخواست پرداخت مبلغی را به مهاجم که برای او کاری انجام داده است می‌نماید. مهاجم کپی پیام را نگه می‌دارد و مجدداً برای دریافت دوباره همان مبلغ از بانک، آن را ارسال می‌کند.

انکار^۳

این نوع حمله، با انواع دیگر فرق دارد، زیرا یکی از دو طرف فرستنده یا گیرنده، این حمله را انجام می‌دهند و ممکن است فرستنده پیام، بعداً ارسال پیام را انکار کند یا ممکن است گیرنده پیام پس از دریافت پیام، دریافت آن را انکار کند.

به عنوان نمونه‌ای از انکار توسط گیرنده، می‌توان درخواست مشتری بانک برای ارسال پول به شخص ثالث را ذکر کرد. این فرد ممکن است بعداً چنین درخواستی را انکار کند. نمونه‌ای از انکار توسط فرستنده می‌تواند زمانی رخ دهد که خریدار محصولی را از کارخانه خریداری کرده و وجه آن را

=====

¹ Masquerading

² Replaying

³ Repudiation

به صورت الکترونیکی پرداخت کند. سپس کارخانه‌دار، دریافت پول را انکار می‌کند و خواستار پرداخت دوباره وجه محصول می‌شود.

حملات تهدیدکننده در دسترس بودن

در این بخش، تنها به یک نوع حمله‌ی تهدیدکننده در دسترس بودن اشاره می‌کنیم که عبارت است از: محرومیت از سرویس^۱.

محرومیت از سرویس، حمله بسیار رایجی است. این حمله ممکن است سرویس یک سیستم را کند یا به طور کامل قطع کند. مهاجم برای انجام این کار می‌تواند از راهکارهای گوناگونی استفاده کند. او می‌تواند درخواست‌های ساختگی زیادی برای سرور بفرستد، به طوری که سرور به علت حجم بالا از کار بیفتد و یا ممکن است که مهاجم، ارسال پاسخ سرور به یک کاربر را متوقف یا آن را پاک کند و کاربر تصور کند که سرور پاسخ نمی‌دهد. همچنین ممکن است مهاجم دریافت درخواست مشتریان را متوقف و باعث شود که مشتریان درخواست خود را چندین بار بفرستند و حجم بار سیستم بالا رود.

مهاجمان غیرفعال در برابر فعال

اجازه دهید مهاجمان را به دو گروه تقسیم کنیم: فعال^۲ و غیرفعال^۳. جدول ۱-۱ ارتباط بین این گروه‌بندی و دسته‌بندی پیشین را نشان می‌دهد.

جدول ۱-۱: دسته‌بندی حملات فعال و غیرفعال

<i>Attacks</i>	<i>Passive/Active</i>	<i>Threatening</i>
Snooping Traffic analysis	Passive	Confidentiality
Modification Masquerading Replaying Repudiation	Active	Integrity
Denial of service	Active	Availability

=====

¹ Denial of service

² Active

³ Passive

حملات غیرفعال

در حملات غیرفعال، هدف مهاجم تنها کسب اطلاعات است. این بدین معناست که حمله باعث تغییر داده‌ها یا آسیب رساندن به سیستم نمی‌شود و سیستم فعالیت عادی خود را ادامه می‌دهد. با این وجود، حمله ممکن است به فرستنده یا گیرنده پیام آسیب بزند. حملاتی که طبقه‌بندی را تهدید می‌کنند، جاسوسی و تجزیه و تحلیل نقل و انتقال است که حملات غیرفعال محسوب می‌شوند. لو رفتن اطلاعات می‌تواند به گیرنده یا فرستنده آسیب برساند، اما روی کار سیستم تأثیری ندارد؛ به همین دلیل، شناسایی این نوع حملات تا زمانی که فرستنده یا گیرنده به نشت اطلاعات پی نبرند بسیار مشکل است. با این وجود، می‌توان با استفاده از رمزنگاری داده‌ها، از بروز حملات غیرفعال جلوگیری کرد.

حملات فعال

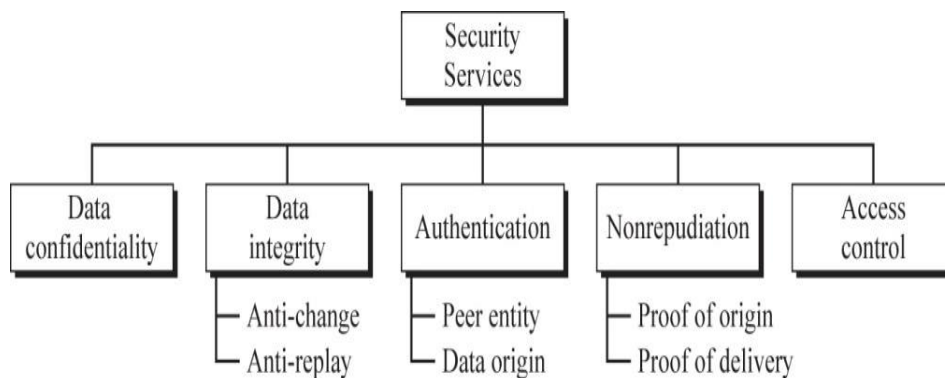
حمله فعال ممکن است داده‌ای را تغییر داده یا به سیستم آسیب برساند. حملاتی که یکپارچگی و در دسترس بودن داده‌ها را تهدید می‌کنند، حملات فعال محسوب می‌شوند. معمولاً شناسایی حملات فعال از پیشگیری از آن‌ها ساده‌تر است، زیرا مهاجم می‌تواند به شیوه‌های مختلف حمله را انجام دهد.

۳-۱- سرویس‌ها و راهکارها

اتحادیه مخابرات بین‌المللی، بخش استاندارد پست و مخابرات T-ITU، سرویس‌های امنیتی و راهکارهایی برای پیاده‌سازی این سرویس‌ها تهیه نموده است. راهکارها و سرویس‌های امنیتی ارتباط تنگاتنگی با هم دارند زیرا در ایجاد یک سرویس، یک راهکار یا ترکیبی از چند راهکار مورد استفاده قرار گرفته است. همچنین یک راهکار را می‌توان در یک یا چند سرویس به کار برد. در اینجا به منظور ارائه ایده‌ی کلی، به اختصار آن‌ها را توضیح می‌دهیم. در بخش‌های دیگر کتاب، به تفصیل در مورد راهکارها و سرویس‌ها صحبت شده است که در جای خود بررسی خواهیم کرد.

سرویس‌های امنیتی

ITUT-T (X.800)، پنج سرویس مرتبط با چشم انداز امنیتی و حملاتی را که در بخش‌های پیشین در مورد آن‌ها صحبت کردیم، مشخص کرده است. شکل ۳-۱ رده‌بندی این ۵ سرویس را نشان می‌دهد.



شکل ۱-۳: سرویس‌های امنیتی

مرتبط ساختن یک یا چند مورد از این سرویس‌ها با یک یا چند هدف امنیتی کار ساده‌ای است. همچنین تشخیص این مطلب که این سرویس‌ها به منظور پیشگیری از حملات امنیتی که پیشتر به آن‌ها اشاره شد، طراحی شده‌اند کار مشکلی نیست.

طبقه‌بندی داده‌ها

طبقه‌بندی داده‌ها برای محافظت از اطلاعات در برابر حملات افشاگرانه طراحی شده است. طبق تعریف X-800 این سرویس بسیار گسترده است و طبقه‌بندی تمام پیام یا قسمتی از پیام و همچنین محافظت در برابر تجزیه و تحلیل نقل و انتقال را دربر می‌گیرد؛ به این معنی که این سرویس برای پیشگیری از حملات تجسسی و تجزیه و تحلیل نقل و انتقال طراحی شده است.

یکپارچگی داده‌ها^۱

یکپارچگی داده‌ها برای جلوگیری از دستکاری، افزودن، حذف کردن و بازپخش توسط مهاجم طراحی شده است. این سرویس می‌تواند از کل پیام یا بخشی از پیام محافظت کند.

تصدیق^۲

این سرویس تأیید هویت طرف مقابل در آن طرف خط ارتباطی را ارائه می‌دهد. در ارتباط اتصال‌گرا^۱، هویت فرستنده یا گیرنده در زمان برقراری ارتباط مورد تأیید قرار می‌گیرد و در ارتباطات نااتصال‌گرا^۲، وظیفه تأیید هویت منبع فرستنده داده‌ها به وسیله این گونه سرویس‌ها به انجام می‌رسد.

^۱ Data Integrity

^۲ Authentication

عدم انکار^۳

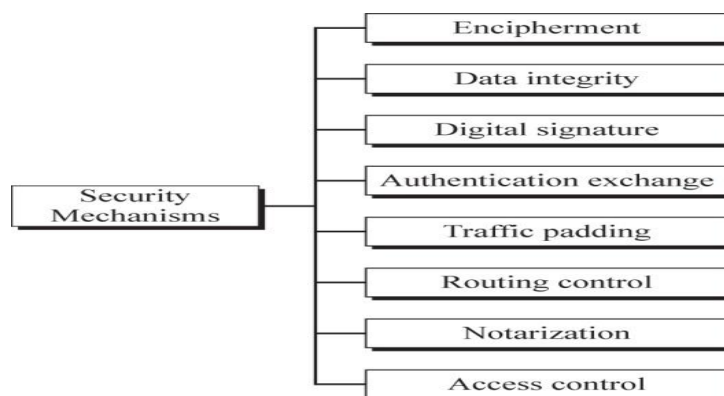
سرویس عدم انکار، مانع انکار ارسال یا دریافت اطلاعات توسط فرستنده یا گیرنده داده‌ها می‌شود. در سرویس عدم انکار مبتنی بر اصالت مبدأ، گیرنده داده‌ها می‌تواند در صورت انکار فرستنده هویت فرستنده را اثبات نماید. در سرویس عدم انکار مبتنی بر تحویل داده‌ها، فرستنده می‌تواند ثابت کند که داده‌ها به گیرنده‌ی مورد نظر تحویل شده است.

کنترل دسترسی^۴

کنترل دسترسی، داده‌ها را در برابر دسترسی غیرمجاز حفاظت می‌کند. واژه‌ی دسترسی در این تعریف بسیار گسترده است و می‌تواند شامل خواندن، نوشتن، ویرایش (تغییر)، اجرای برنامه و... شود.

سازوکارهای امنیتی

ITU (X88)، برخی راهکارهای امنیتی لازم برای ارائه‌ی سرویس‌های امنیتی مشروحه در بخش پیشین را پیشنهاد می‌کند. شکل ۴-۱ رده‌بندی این سازوکارها را نشان می‌دهد.



شکل ۴-۱: سازوکارهای امنیتی

¹ Connection-oriented

² Connectionless

³ Nonrepudiation

⁴ Access Control

رمزنویسی^۱

پنهان‌سازی یا پوشاندن داده‌ها می‌تواند اهداف محرمانگی را برآورده کند، همچنین از آن می‌توان برای تکمیل سایر سازوکارها به منظور ایجاد سرویس‌های دیگر استفاده کرد. امروزه برای رمزنویسی از دو شیوه رمزننگاری و استتار استفاده می‌شود که در بخش‌های آتی در مورد آن‌ها صحبت خواهیم کرد.

یکپارچگی داده‌ها

راهکار برآورد هدف یکپارچگی داده‌ها، اختصاص عدد ویژه‌ای به داده‌هاست. مقدار این عدد به وسیله فرآیند ویژه‌ای از خود داده‌ها تولید می‌شود. گیرنده، این عدد را دریافت می‌کند و از داده‌های دریافتی نیز عدد جدید ایجاد می‌کند، سپس عدد دریافتی را با عدد تولیدی جدید مقایسه می‌کند، اگر هر دو عدد یکسان باشند، معلوم می‌شود که یکپارچگی داده‌ها محفوظ مانده است.

امضاء دیجیتال^۲

امضاء دیجیتال ابزاری است که به وسیله آن فرستنده می‌تواند به صورت الکترونیک، داده‌ها را علامت‌گذاری کرده و گیرنده می‌تواند به صورت الکترونیک امضاء داده‌ها را بررسی کند. فرستنده از سازوکاری استفاده می‌کند که نشان می‌دهد فرستنده کلید اختصاصی مرتبط با کلید عمومی که به صورت همگانی اعلام کرده است را دارد. گیرنده از کلید عمومی فرستنده برای اثبات این موضوع که پیام را همان فرستنده‌ای که مدعی ارسال پیام است امضاء کرده، استفاده می‌کند.

تبادل هویت^۳

در تبادل هویت دو فرد، پیام‌هایی برای اثبات هویت خود رد و بدل می‌کنند. به عنوان مثال، یک طرف می‌تواند ثابت کند رازی را می‌داند که فقط او باید بداند.

افزودن لایه‌هایی به نقل و انتقال داده‌ها

افزودن لایه به نقل و انتقال، به معنای افزودن داده‌های ساختگی^۴ به داده‌های ارسالی است، تا با این وسیله بتوان تلاش دشمن برای استفاده از تجزیه و تحلیل ارسال و دریافت داده‌ها را خنثی نماید.

=====

¹ Encipherment

² Digital Signature

³ Authentication Exchange

⁴ Bogus Data

کنترل مسیریابی^۱

کنترل مسیریابی به معنی انتخاب و تغییر مداوم مسیرهای مختلف موجود بین فرستنده و گیرنده می‌باشد تا با این شیوه بتوان از استراق سمع شخص دیگری بر روی یک مسیر خاص پیشگیری نمود.

گواهی رسمی^۲

گواهی رسمی به معنی انتخاب شخص ثالث قابل اعتماد برای کنترل ارتباط بین دو فرد می‌باشد. به عنوان مثال، می‌توان این کار را به منظور پیشگیری از انکار انجام داد. گیرنده می‌تواند فرد قابل اعتمادی را مسئول ذخیره درخواست‌های فرستنده نماید، تا به این وسیله بتواند مانع از انکار فرستنده در خصوص ارسال چنین درخواستی شود.

کنترل دسترسی

در این روش برای اثبات این که کاربر دسترسی صحیحی به داده‌ها یا منابع تحت مالکیت یک سیستم داشته است، از روش‌های مختلفی استفاده می‌شود. گذر واژه و PIN، نمونه‌هایی از این اثبات‌ها هستند.

رابطه بین سرویس‌ها و سازوکارها

جدول ۱-۲ رابطه‌ی بین سرویس‌های امنیتی و راهکارهای امنیتی را نشان می‌دهد. این جدول نشان می‌دهد که می‌توان از سه سازوکار (رمزنویسی، امضای دیجیتالی، تبادل هویت) برای تأمین تصدیق استفاده کرد. همچنین این جدول نشان می‌دهد که سازوکار رمزنویسی را می‌توان در سه سرویس (طبقه‌بندی داده‌ها، یکپارچگی داده‌ها و تصدیق) به کار برد.

=====

¹ Routing Control

² Notarization

جدول ۱-۲: رابطه‌ی بین سرویس‌های امنیتی و سازوکارهای امنیتی

<i>Security Service</i>	<i>Security Mechanism</i>
Data confidentiality	Encipherment and routing control
Data integrity	Encipherment, digital signature, data integrity
Authentication	Encipherment, digital signature, authentication exchanges
Nonrepudiation	Digital signature, data integrity, and notarization
Access control	Access control mechanism

۱-۴- راهکارها

راهکارهای بیان شده در قسمت‌های پیشین، فقط دستورالعمل‌های تئوری در زمینه‌ی اجرای امنیت بودند. اجرای واقعی چشم‌اندازهای امنیتی مستلزم راهکارهایی است. دو راهکار پرکاربرد امروزی به صورت بسیار کلی (رمزنگاری) و به صورت خاص (استتار) است.

رمزنگاری^۱

برخی از راهکارهای بیان شده در بخش‌های پیشین را می‌توان با استفاده از رمزنگاری اجرا کرد. «cryptograpghy» واژه‌ای با ریشه‌ی یونانی و به معنی نوشتن رمزی است. هدف ما از بیان این واژه، اشاره به علم و هنر تبدیل پیام عادی به متن رمز برای محافظت و ایمن کردن آن‌ها در برابر حملات است. اگرچه در گذشته واژه‌ی رمزنگاری فقط برای بیان «رمزنویسی» و «رمزگشایی» پیام با استفاده از کلید رمز به کار می‌رفت، اما امروزه این واژه دارای سه سازوکار مجزا است: رمزنویسی کلید متقارن، رمزنویسی کلید نامتقارن و هش. در ادامه این سه سازوکار را به اختصار مورد بحث قرار می‌دهیم.

رمزنویسی کلید متقارن^۲

در رمزنویسی کلید متقارن (که گاهی به آن رمزنویسی کلید سری یا رمزنگاری کلید سری نیز می‌گویند)، شخصی مثل آلیس می‌تواند به شخص دیگری مثل باب با استفاده از کانال ناامن پیامی بفرستد. با فرض این که رقیب، مثلاً Eve نمی‌تواند محتوای پیام را به وسیله‌ی استراق سمع از کانال بفهمد. آلیس با

=====

^۱ Cryptography

^۲ Symmetric-Key Encipherment

استفاده از الگوریتم رمزنویسی، پیام را به شکل رمز درمی‌آورد، باب هم با استفاده از الگوریتم رمزگشایی، پیام را رمزگشایی می‌کند. رمزنویسی کلید متقارن، از کلید یکتا برای رمزنویسی و رمزگشایی استفاده می‌کند. رمزنویسی / رمزگشایی را می‌توان نوعی قفل الکترونیکی تلقی کرد. در رمزنویسی کلید متقارن، آلیس پیام را در جعبه‌ای قرار می‌دهد و با استفاده از کلید، رمز مشترک آن را قفل می‌کند. باب جعبه را با استفاده از همان کلید باز می‌کند و پیام آن را بیرون می‌آورد.

رمزنویسی کلید نامتقارن^۱

در رمزنویسی کلید نامتقارن (که گاهی به آن رمزنویسی کلید عمومی یا رمزنگاری کلید عمومی نیز می‌گویند) همان ساختار رمزنویسی کلید متقارن را داریم اما با چند استثنا. نخست، به جای یک کلید، دو کلید داریم؛ یک کلید عمومی و یک کلید اختصاصی برای ارسال پیام رمزی به باب. آلیس ابتدا با استفاده از کلید عمومی باب، پیام را رمزنویسی می‌کند. برای رمزگشایی پیام، باب از کلید اختصاصی خود استفاده می‌کند.

هش^۲ (درهم سازی)

هش خلاصه‌ای با طول ثابت را، از دل پیامی با طول دلخواه استخراج می‌نماید. همیشه خلاصه، کوتاه‌تر از اصل پیام است و برای این که مفید باشد، باید هم پیام و هم خلاصه‌ی آن برای گیرنده ارسال شود. برای تولید عدد ویژه‌ای که پیشتر در رابطه با برقراری یکپارچگی داده‌ها در مورد آن صحبت کردیم، از هش استفاده می‌شود.

اگرچه شالوده این کتاب، معرفی رمزنگاری به عنوان یک روش اجرای راهکارهای امنیتی است، اما روش دیگری که در ارتباطات سری در گذشته از آن استفاده می‌شد و تا به امروز نیز ادامه دارد، استتار است. واژه «Steganography» با ریشه یونانی به معنی نوشتن، در مقایسه با رمزنگاری که به معنی «نوشتن رمزی» است، به معنی «نوشتن در لفافه» می‌باشد. رمزنگاری یعنی پنهان کردن محتوای پیام به وسیله رمزنویسی. استتار به معنی پنهان کردن خود پیام با وسیله دیگری است.

=====

^۱ Asyrrnmetric-Key Encipherment

^۲ Hash

پیشینه تاریخی استتار

تاریخ سرشار از واقعیات و افسانه‌های استفاده از استتار است. در چین، پیام‌های جنگی روی تکه‌های ابریشم نوشته می‌شد، به شکل کروی در آمده سپس پیام‌رسان آن را قورت می‌داد و خود را به مقصد می‌رساند. در رم و یونان، پیام روی تکه چوبی حک می‌شد، سپس آن را آغشته به موم می‌کردند تا نوشته پنهان شود. از جوهرهای نامرئی (نظیر آب پیاز و نمک آمونیاک) هم برای نوشتن پیام بین خطوط متن دیگر یا بر پشت کاغذ استفاده می‌کردند. پیام پنهانی، با گرم کردن کاغذ یا مالیدن ماده دیگری روی کاغذ آشکار می‌شد.

اخیراً روش‌های دیگری نیز ارائه شده است. برخی حروف در یک پیام به ظاهر بی‌ضرر ممکن است حاوی متنی در اندازه نوک مداد باشد، که تنها زمانی قابل مشاهده است که در زاویه‌ی خاص در معرض نور قرار گیرد. برای پنهان کردن پیام رمزی در داخل پیام بی‌ضرر، از رمز تهی استفاده می‌شود. به عنوان مثال، حرف اول یا دوم هر واژه در پیام پوشش ممکن است پیام رمز را تشکیل دهد. از ریز خال‌ها هم برای این هدف استفاده می‌شود. از پیام‌های رمزی عکس می‌گرفتند و اندازه آن‌ها را به حد یک نقطه کاهش می‌دادند و آن را به شکل نقطه معمولی در پایان جملات متن پوششی قرار می‌دهند.

کاربرد مدرن استتار

امروزه هر شکلی از داده‌ها نظیر متن، عکس، فایل صوتی یا ویدئو را می‌توان دیجیتالی کرد و امکان درج اطلاعات دودویی رمزی در طول فرایند دیجیتالی کردن در آن‌ها وجود دارد. چنین اطلاعات پنهانی، الزاماً برای پنهان‌کاری مورد استفاده قرار نمی‌گیرد؛ بلکه می‌توان از آن برای حفاظت از مالکیت معنوی، جلوگیری از دستکاری یا افزودن اطلاعات دیگر به اصل داده‌ها استفاده کرد.

متن پوششی^۱

پوشش داده‌های رمزی می‌تواند در قالب متن عادی باشد. شیوه‌های مختلفی برای پوشش داده‌های رمزی متن وجود دارد. چنانچه شیوه‌های گوناگون برای درج داده‌های دودویی یک متن بی‌ضرر نیز وجود دارد. به عنوان مثال، می‌توانیم از یک فاصله بین کلمات برای نشان دادن ۰ و دو فاصله در بین کلمات برای

=====

^۱ Text Cover

نشان دادن ۱ استفاده کنیم. پیام کوتاه زیر نمایش دودویی ۸ بیتی از حرف A را در قالب کد ASCII نشان می‌دهد (۰۱۰۰۰۰۰۱).

This book is mostly about cryptography, not steganography.

□	□□□	□	□	□□□
0	1 0	0	0	0 1

در پیام بالا، دو فاصله بین «book» و «is» و بین «not» و «steganography» وجود دارد. البته نرم‌افزارهای خاصی می‌توانند فاصله‌هایی را در متن وارد کنند که تفاوت بسیار کمی با متن اصلی ایجاد می‌کند و با کمک آن‌ها می‌توان رمز را از کشف و شناسایی آنی رها کنید.

روش دیگر، استفاده از فرهنگ لغات تنظیم شده بر اساس کاربرد دستور زبان است. می‌توانیم فرهنگ لغتی داشته باشیم که دارای دو حرف تعریف، ۸ فعل، ۳۲ اسم و ۴ حرف اضافه است. سپس بر سر استفاده از متن پوششی‌ای توافق می‌کنیم که همیشه الگوی جملات آن «حرف تعریف - اسم - فعل - حرف تعریف - اسم» باشد. داده‌های دودویی رمزی را می‌توان به تکه‌های ۱۶ بیتی تقسیم کرد. یک حرف تعریف می‌تواند بیانگر اولین بیت داده‌های دودویی باشد (به عنوان مثال ۰ برای a و ۱ برای the). پنج بیت بعد را می‌توان با یک اسم (فاعل یک جمله) نشان داد، چهار بیت بعد را می‌توان به وسیله یک فعل نشان داد، بیت بعد با حرف تعریف دوم و آخرین پنج بیت را به وسیله یک اسم (مفعول) نشان داد. به عنوان مثال داده رمز «HI»، که ۰۱۰۰۱۰۰۰۰۱۰۰۱۰۰۱ در ASCII است، می‌تواند جمله‌ای نظیر جمله زیر باشد.

A friend called a doctor

01001 0 0001 10010 0

این یک مثال بسیار بدیهی و ساده است. در عمل، از طراحی پیچیده‌تر و انواع الگوها استفاده می‌شود.

عکس در قالب پوشش^۱

داده‌های رمز را می‌توان در داخل عکس رنگی پنهان کرد. عکس دیجیتالی متشکل از پیکسل (عناصر عکس) است؛ معمولاً هر پیکسل از ۲۴ بیت (۳ بایت) تشکیل می‌شود.

=====

^۱ Image Cover

هر بایت نشان دهنده‌ی یکی از رنگ‌های اصلی (قرمز، سبز یا آبی) است. به همین دلیل می‌توانیم طیفی از رنگ‌ها را تولید کنیم. در روش موسوم به **Least significant bit**، کم ارزش‌ترین بیت هر بایت مساوی صفر قرار داده می‌شود. این روش باعث می‌شود عکس در بعضی از نقاط روشن‌تر شود که معمولاً چندان مشهود نیست. اکنون می‌توانیم داده‌های دودویی را با تغییر کم ارزش‌ترین بیت، در عکس پنهان کنیم. اگر رقم دودویی ما ۰ باشد، بیت متناظر بدون تغییر باقی می‌ماند و اگر ۱ باشد، بیت متناظر را به ۱ تغییر می‌دهیم. بدین ترتیب می‌توانیم یک نویسه (۸ بیت اسکی) را در سه پیکسل پنهان کنیم. به عنوان مثال، سه پیکسل زیر می‌تواند حرف «M» را نشان دهد.

0101001 1	1011110 0	0101010 1
0101111 0	1011110 0	0110010 1
0111111 0	0100101 0	0001010 1

البته، امروزه شیوه‌های پیچیده‌تری از استتار مورد استفاده قرار می‌گیرد.

سایر پوشش‌ها

پوشش‌های دیگری نیز مورد استفاده قرار می‌گیرند، مثال پیام رمز را می‌توان در پوشش فایل صوتی (صدا و موسیقی) و ویدئو پنهان کرد. امروزه هم فایل صوتی و هم ویدئو فشرده می‌شوند. پیام رمز را می‌توان در حین فشرده‌سازی یا پیش از آن در فایل حامل جاسازی نمود. بحث درباره این تکنیک‌ها را به کتاب-های تخصصی در زمینه استتار می‌سپاریم.

۱-۵- سایر قسمت‌های کتاب

کتاب به چهار قسمت تقسیم شده است.

قسمت اول: رمزنویسی کلید متقارن

این بخش در مورد رمزنویسی، هم کلاسیک و هم پیشرفته، با استفاده از رمزنگاری کلید متقارن به بحث می‌پردازد و نشان می‌دهد که چگونه با استفاده از این فن می‌توان نخستین هدف امنیتی را محقق کرد.

قسمت دوم: رمزنویسی کلید نامتقارن

این بخش، در مورد رمزنویسی با استفاده از رمزنگاری کلید نامتقارن بحث می‌کند و نشان می‌دهد که چگونه با استفاده از این روش می‌توان نخستین هدف امنیتی را محقق ساخت.

قسمت سوم: یکپارچگی، تصدیق و مدیریت کلید.

بخش سوم، سومین شیوه رمزنگاری یعنی «هش» را معرفی می‌کند و نشان می‌دهد که چگونه «هش» به همراه موارد بیان شده در بخش‌های اول و دوم، برای تحقق دومین هدف امنیتی به کار می‌رود.

قسمت چهارم: امنیت شبکه

این بخش نشان می‌دهد که چگونه می‌توان با استفاده از اینترنت، روش‌های آموخته شده در سه بخش اول کتاب را با هم ترکیب کرد و به این روش امنیت شبکه را برپا نمود.

۱-۶- وب سایت‌های پیشنهادی :

وب سایت‌های زیر، اطلاعات بیشتری در خصوص موضوعات این فصل را ارائه می‌دهند.

<http://www.faqs.org/rfcs/rfc2828.html>

fag.grm.hia.no/IKT7000/litteratur/paper/x800.pdf

۱-۷- چکیده

- * سه هدف کلی برای امنیت مشخص گردیده: محرمانگی، یکپارچگی و در دسترس بودن.
- * دو نوع حمله طبقه‌بندی اطلاعات را تهدید می‌کند: جاسوسی، تجزیه و تحلیل، نقل و انتقال اطلاعات. چهار نوع حمله می‌تواند یکپارچگی داده‌ها را به خطر بیندازد: دستکاری، جعل هویت، تکرار و انکار و حملات محرومیت از سرویس، دسترسی به اطلاعات را تهدید می‌کند.
- * برخی از سازمان‌های فعال در زمینه‌ی ارتباطات داده‌ای و شبکه‌ای نظیر ITU-T یا اینترنت، سرویس‌های امنیتی گوناگونی در راستای چشم‌انداز امنیتی و برای رویارویی با حملات امنیتی مشخص کرده‌اند. در این فصل به بحث در مورد پنج سرویس امنیتی رایج پرداخته شده است: محرمانگی داده‌ها، یکپارچگی داده‌ها، تصدیق، عدم انکار و کنترل دسترسی.
- * همچنین ITU-T برای تأمین امنیت، سازوکارهایی را توصیه می‌کند. هشت سازوکار از این نوع را مورد بحث قرار دادیم: رمزنویسی، یکپارچگی داده‌ها، امضای دیجیتالی، تبادل هویت، افزودن لایه به نقل و انتقال، کنترل مسیریابی، گواهی رسمی و کنترل دسترسی.
- * دو روش - رمزنگاری و استتار- وجود دارد که می‌توان از آن‌ها در برخی یا تمام سازوکارها استفاده کرد. رمزنگاری یا نوشتن رمزی، شامل به رمز درآوردن یک پیام یا ایجاد پیامی نامفهوم و به هم ریخته از پیام اصلی می‌شود. استتار یا نوشتن در لفافه به معنی پنهان کردن پیام به وسیله پوشاندن و نهان‌سازی آن در نوشته‌ی دیگری است.

۱-۸- مجموعه تمرین‌ها

پرسش‌های دوره‌ای

- (۱) سه هدف امنیتی را نام ببرید.
- (۲) تفاوت بین حمله فعال و غیرفعال را بیان کنید و چند حمله فعال و چند حمله غیرفعال را نام ببرید.
- (۳) پنج سرویس امنیتی را نام برده و تعریف کنید.
- (۴) هشت سازوکار امنیتی را نام ببرید.
- (۵) تفاوت بین رمزنگاری و استتار را بیان کنید.

تمرین‌ها

- (۶) از کدام سرویس / سرویس‌های امنیتی، هنگام ارسال بسته پستی در یک باجه پستی به روش‌های زیر استفاده شده است:
 - (a) پست عادی
 - (b) پست عادی همراه با تأیید تحویل
 - (c) پست عادی همراه با تحویل و امضای گیرنده
 - (d) پست سفارشی
 - (e) پست تضمینی
 - (f) پست ثبت شده
- (۷) نوع حمله‌ی امنیتی در هر یک از موارد زیر را بیان کنید.
 - (a) دانشجویی به قصد برداشتن کپی‌ای از امتحان روز بعد، وارد دفتر استاد شد.
 - (b) دانشجویی چکی به مبلغ ۱۰ دلار برای خرید یک کتاب دست دوم داد. بعد فهمید که چکش به مبلغ ۱۰۰ دلار نقد شده است.
 - (c) دانشجویی با استفاده از یک نشانی ایمیل ساختگی (جعلی)، هر روز صدها ایمیل برای دانشجویی دیگر می‌فرستاد.
- (۸) در هر یک از موارد زیر، چه نوع سازوکار امنیتی در نظر گرفته شده است؟

- (a) مدرسه‌ای از دانش‌آموزان خواست برای استفاده از سرور مدرسه، کارت شناسایی و یک گذر واژه ارائه دهند.
- (b) اگر دانش‌آموزی بیش از ۲ ساعت به سیستم وصل شود، سرور مدرسه ارتباط او را قطع می‌کند.
- (c) استادی از ارسال نمرات دانش‌آموزان به وسیله ایمیل خودداری می‌کرد، مگر این که دانش‌آموزان کارت شناسایی دانش‌آموزی را که پیش از آن توسط استاد طراحی شده باشد را تهیه نمایند.
- (d) مشتریان یک بانک می‌بایست برای برداشت وجه، امضاء ارائه دهند.
- (۹) در هر یک از موارد زیر، از کدام روش (رمزنگاری یا استتار) برای طبقه‌بندی استفاده شده است.
- (a) دانش‌آموزی، پاسخ پرسش را روی یک کاغذ کوچکی نوشت، آن را پیچیده و داخل خودکار گذاشت و خودکار را به دانش‌آموز دیگری داد.
- (b) برای ارسال یک پیام، جاسوس هر نشانه در متن را با نشانه‌ای که از پیش به عنوان جایگزین روی آن توافق شده بود عوض کرد.
- (c) شرکتی به منظور جلوگیری از جعل، از جوهر مخصوصی روی چک‌هایش استفاده می‌کرد.
- (d) دانشجوی کارشناسی ارشد، برای محافظت از پایان‌نامه‌ی خود که به وسیله وب پایگاهش ارسال کرده بود از واترمارک استفاده کرد.
- (۱۰) فردی فرم پر شده برای درخواست کارت اعتباری را امضاء می‌کند، چه سازوکار یا سازوکارهای امنیتی در نظر گرفته شده است؟

راهنمای فصل‌های کتاب

رمزنویسی کلید متقارن

فصل ۱ رمزنویسی کلید متقارن

در فصل پیش دیدیم که رمزنگاری به سه روش امکان‌پذیر است: رمزهای کلید متقارن، رمزهای کلید نامتقارن و هش. اول به رمزهای کلید متقارن اختصاص داده شده است. فصل‌های ۲ و ۴ مروری بر پیش-نیاز ریاضیاتی لازم برای درک سایر فصول این بخش دارد. فصل ۳ رمزهای سنتی به‌کاررفته در گذشته را مورد بررسی قرار می‌دهد. فصول ۵، ۶ و ۷ به توضیح رمزهای بلوکی که امروزه به کار می‌رود می‌پردازد. فصل ۸ نشان می‌دهد که چگونه می‌توان از رمزهای بلوکی و جریانی برای رمزگشایی پیام‌های طولانی استفاده کرد.

فصل ۲ ریاضیات رمزنگاری: قسمت ۱

فصل ۲ مروری بر برخی از مفاهیم ریاضی مورد نیاز برای درک چند فصل بعدی دارد. در این فصل عدد صحیح و حساب‌های پیمانه‌ای، ماتریس‌ها و روابط تجانس مورد بحث قرار می‌گیرد.

فصل ۳: رمزهای کلید متقارن سنتی

در فصل ۳ رمزهای کلید متقارن سنتی معرفی می‌شود. اگرچه این رمزها امروزه دیگر به کار نمی‌روند، اما پایه و اساس رمزهای کلید متقارن پیشرفته را تشکیل می‌دهند. این فصل، بر دو مقوله رمزهای سنتی تأکید دارد: رمزهای جایگزین و رمزهای جابجایی (انتقال). همچنین این فصل به معرفی مفاهیم رمزهای جریانی و رمزهای بلوکی می‌پردازد.

فصل ۴: ریاضیات رمزگذاری: قسمت ۲

فصل ۴ مرور دیگری بر ریاضیات مورد نیاز برای درک محتوای فصول بعدی است. در این فصل برخی از ساختارهای جبری مانند گروه‌ها، حلقه‌ها و میدان‌های متناهی که در رمزهای بلوکی پیشرفته مورد استفاده قرار می‌گیرد، بررسی می‌شود.

فصل ۵: معرفی رمزهای کلید متقارن پیشرفته

فصل ۵ رمزهای کلید متقارن را معرفی می‌نماید. درک عناصر خاص استفاده شده در رمزهای کلید متقارن، راه را برای درک و تجزیه و تحلیل بهتر رمزهای پیشرفته هموار می‌کند. این فصل مؤلفه‌های رمزهای بلوکی، نظیر جعبه‌های P و جعبه‌های S را معرفی می‌کند. همچنین بین دو دسته از رمزهای فرآورده تفاوت قائل می‌شود: رمزهای Feistel و Non-Feistel.

فصل ۶: استاندارد رمزنگاری داده‌ها

فصل ۶ عناصر تعریف شده در فصل ۵ را برای بحث و تجزیه و تحلیل یکی از رمزهای پرکاربرد مبتنی بر کلید، همچون استاندارد رمزگذاری داده‌ها (DES) که امروزه مورد استفاده قرار می‌گیرد به کار می‌برد. تأکید بر چگونگی استفاده‌ی رمزنگاری DES از ۱۶ گردش رمزهای Feistel است.

فصل ۷: استانداردهای رمزگذاری پیشرفته (AES)

فصل ۷ نشان می‌دهد که چگونه برخی از ساختارهای جبری بحث شده در فصل ۴ و برخی از عناصر بحث شده در فصل ۵ می‌توانند رمز بسیار قوی و استاندارد رمزگذاری پیشرفته را ایجاد می‌کنند. هدف این است که چگونه به وسیله‌ی ساختارهای جبری بحث شده در فصل ۴، به چشم‌انداز امنیتی AES می‌رسند.

فصل ۸: رمزگذاری با استفاده از رمزهای کلید متقارن پیشرفته

فصل ۸ نشان می‌دهد که چگونه می‌توان از رمزهای بلوکی و جریانی مدرن برای رمزنگاری پیام‌های طولانی استفاده کرد. این فصل، پنج راهکار طراحی شده جهت استفاده در رمزهای بلوکی پیشرفته را توضیح می‌دهد؛ همچنین دو رمز جریانی استفاده شده برای پردازش آنی داده‌ها را معرفی می‌کند.

ریاضیات رمزنگاری

قسمت ۱: حساب پیمانه‌ای، تجانس و ماتریس‌ها

چشم‌انداز

هدف این فصل، آماده‌سازی شما برای چند فصل بعدی در زمینه رمزنگاری است.

در این فصل چشم‌اندازهای زیر را دنبال می‌کنیم:

✿ مروری بر حساب اعداد صحیح با تمرکز بر بخش‌پذیری و یافتن بزرگ‌ترین مقسوم‌علیه مشترک با استفاده از الگوریتم اقلیدسی.

✿ درک این که چگونه الگوریتم اقلیدسی بسط‌یافته را می‌توان برای حل معادلات خطی سیال، حل معادلات خطی متجانس و یافتن وارون‌های ضربی به کار برد.

✿ نشان‌دادن اهمیت حساب پیمانه‌ای و عملگر پیمانه، به سبب کاربرد گسترده آن‌ها در رمزنگاری.

✿ بررسی ماتریس‌ها و عملیات روی ماتریس باقیمانده‌ها که در رمزنگاری کاربرد گسترده‌ای دارد.

✿ حل مجموعه‌ی معادلات متجانس با استفاده از ماتریس باقیمانده‌ها.

رمزنگاری مبتنی بر حوزه‌های خاصی در ریاضیات، شامل تئوری اعداد، جبرخطی و ساختارهای جبری می‌باشد. در این فصل تنها به بحث در مورد این حوزه‌ها، که برای درک محتوای فصول بعدی لازم است می‌پردازیم. کسانی که با این موضوعات آشنا هستند، می‌توانند از کل فصل یا قسمتی از این فصل چشم‌پوشی کنند. در صورت لزوم مطالب مشابهی در کل کتاب در نظر گرفته شده است. اثبات قضایا و الگوریتم‌ها را حذف و تنها به بیان کاربرد آن‌ها بسنده نموده‌ایم. علاقمندان می‌توانند اثبات این قضایا و الگوریتم‌ها را در پیوست آخر جلد دوم کتاب مشاهده نمایند.

۲-۱- حساب عدد صحیح

در حساب عدد صحیح، از مجموعه و تعدادی از عملگرها استفاده نموده‌ایم. اگرچه شما با این مجموعه و عملیات مرتبط با آن آشنا هستید، اما به منظور یادآوری و ایجاد پیش‌زمینه‌ی لازم برای معرفی حساب پیمانانه‌ای، آن‌ها را دوباره مرور می‌کنیم.

مجموعه اعداد صحیح

مجموعه اعداد صحیح که با Z نشان داده می‌شود، تمام اعداد صحیح (بدون قسمت اعشار)، از بینهایت منفی تا بینهایت مثبت را شامل می‌شود (شکل ۲-۱).

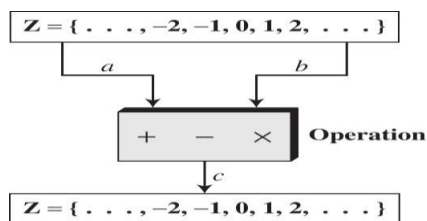
$$Z = \{ \dots, -2, -1, 0, 1, 2, \dots \}$$

شکل ۲-۱: مجموعه اعداد صحیح

عملیات دودویی

در رمزگذاری، به سه عملگر دودویی تعریف شده در مجموعه اعداد صحیح علاقمندیم. یک عمل دودویی، دو ورودی را دریافت و یک خروجی تولید می‌کند. سه عمل دودویی رایجی که برای اعداد صحیح تعریف شده‌اند، عبارت‌اند از: جمع، تفریق و ضرب. همان‌گونه که در شکل ۲-۲ نشان داده شده است، هر یک از این سه عمل، دو ورودی (a و b) را گرفته و یک خروجی (c) تولید می‌کنند. دو ورودی از مجموعه اعداد صحیح می‌باشد. خروجی هم به مجموعه اعداد صحیح تعلق دارد.

توجه داشته باشید که تقسیم، مناسب این مقوله نیست؛ زیرا همان‌گونه که خواهیم دید، در تقسیم به جای یک خروجی، دو خروجی تولید می‌شود.



شکل ۲-۲: سه عملیات دودویی برای مجموعه اعداد صحیح

مثال ۱-۲

نمونه‌های زیر نتایج سه عمل دودویی را بر روی دو عدد صحیح نشان می‌دهد. از آنجا که هر ورودی می‌تواند مثبت یا منفی باشد، چهار حالت برای هر عمل داریم:

جمع:	$5 + 9 = 14$	$(-5) + 9 = 4$	$5 + (-9) = -4$	$(-5) + (-9) = -14$
تفریق:	$5 - 9 = -4$	$(-5) - 9 = -14$	$5 - (-9) = 14$	$(-5) - (-9) = +4$
ضرب:	$5 \times 9 = 45$	$(-5) \times 9 = -45$	$5 \times (-9) = -45$	$(-5) \times (-9) = 45$

تقسیم اعداد صحیح

در حساب اعداد صحیح اگر a را به n تقسیم کنیم، q و r حاصل می‌شود. رابطه فی‌مابین این چهار عدد صحیح را می‌توان به شکل زیر نشان داد:

$$a = q \times n + r$$

در این رابطه a مقسوم، q خارج‌قسمت، n مقسوم‌علیه و r باقیمانده است. توجه داشته باشید که تقسیم یک عملگر دودویی نیست، زیرا نتیجه تقسیم a بر n دو عدد صحیح q و r است. می‌توانیم آن را رابطه بخش‌پذیری بنامیم.

مثال ۲-۲

فرض کنید $a = 255$ و $n = 11$ است. همان‌گونه که در شکل ۲-۳ نشان داده شده است، می‌توانیم با استفاده از الگوریتم تقسیم که در حساب یاد گرفته‌ایم، $q = 23$ و $r = 2$ را به دست آوریم.

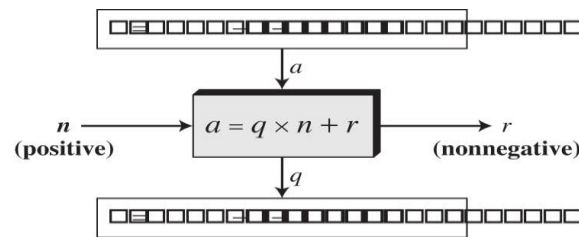
$$\begin{array}{r}
 23 \longleftarrow q \\
 n \longrightarrow 11 \quad \overline{) 255} \\
 \underline{22} \\
 35 \\
 \underline{33} \\
 2 \longleftarrow r
 \end{array}$$

شکل ۲-۳: یافتن خارج‌قسمت و باقیمانده

بیشتر زبان‌های برنامه‌نویسی با استفاده از دستورات خود می‌توانند خارج قسمت و باقیمانده را به دست آورند. به عنوان مثال، در زبان C عملگر / خارج قسمت و عملگر % باقیمانده را به دست می‌آورد.

محدودیت‌ها

وقتی از معادله‌ی تقسیم در رمزنگاری استفاده کنیم، با دو محدودیت روبرو می‌شویم. نخست باید مقسوم‌علیه یک عدد صحیح مثبت باشد ($n > 0$)، دوم باقیمانده باید یک عدد غیر منفی باشد ($r > 0$). شکل ۲-۴ این رابطه را با دو محدودیت فوق‌الذکر نشان می‌دهد.



شکل ۲-۴: الگوریتم تقسیم برای اعداد صحیح.

مثال ۲-۳

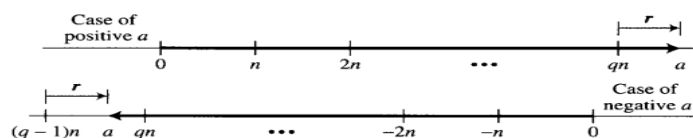
وقتی که از رایانه یا ماشین حساب استفاده می‌کنیم، اگر a منفی باشد r و q هم منفی هستند. چگونه محدودیتی که r باید مثبت باشد را رعایت کنیم؟ راه‌حل ساده است. مقدار q را به یک واحد کاهش می‌دهیم و برای مثبت کردن آن، یک واحد به r اضافه می‌کنیم.

$$-255 = (-23 \times 11) + (-2) \leftrightarrow -255 = (-24 \times 11) + 9$$

۲۳- را به ۲۴- کاهش دادیم و ۱۱ را به ۲- اضافه کردیم و ۹ را به دست آوردیم. رابطه فوق همچنان معتبر است.

نمودار روابط

همان‌گونه که در شکل ۲-۵ نشان داده شده است، می‌توانیم با استفاده از دو نمودار، رابطه بالا را با دو محدودیت بر روی n و r نشان دهیم.



شکل ۲-۵: بردار الگوریتم تقسیم

با شروع از نقطه صفر، شکل نشان می‌دهد که چگونه می‌توانیم به نقطه‌ای که عدد صحیح a را روی خط نشان داده‌ایم برسیم. اگر a مثبت باشد، باید $q \times n$ واحد به راست حرکت کنیم و سپس r واحد اضافه را نیز در همان مسیر طی کنیم. اگر a منفی باشد، باید $(q-1) \times n$ واحد به سمت چپ حرکت کنیم (q در این حالت منفی است) و سپس r واحد به سمت مخالف برویم.

بخش‌پذیری

اجازه دهید به اختصار در مورد بخش‌پذیری، موضوعی که اغلب اوقات در رمزگذاری با آن روبرو هستیم صحبت کنیم. اگر در رابطه تقسیم، a صفر نباشد و $r=0$ قرار دهیم، آنگاه: $a = q \times n$ سپس می‌گوییم که n عدد a را عاد می‌کند (n مقسوم‌علیه a است). همچنین می‌توانیم بگوییم که a بر n بخش‌پذیر است. اگر مقدار q برای ما مهم نباشد، می‌توانیم رابطه فوق را به این شکل بنویسیم $a | n$. اگر باقیمانده صفر نباشد، n بر a بخش‌پذیر نیست و می‌توانیم رابطه را به این شکل بنویسیم: $a \nmid n$.

مثال ۲-۴

(a) عدد صحیح ۳۲ بر ۴ بخش‌پذیر است زیرا $8 \times 4 = 32$ این رابطه را به این شکل $4 | 32$ نشان می‌دهیم.

(b) عدد ۴۲ بر ۸ قابل تقسیم نیست زیرا $5 \times 8 + 2 = 42$. در این رابطه باقیمانده، یعنی عدد ۲ در معادله وجود دارد. آن را به شکل $8 \nmid 42$ نشان می‌دهیم.

مثال ۲-۵

(a) داریم $13 | 78$ ، $7 | 98$ ، $6 | 24$ ، $4 | 44$ و $11 | (-33)$

(b) داریم $13 \nmid 27$ ، $7 \nmid 50$ ، $6 \nmid 23$ ، $4 \nmid 41$ و $11 \nmid (-32)$

ویژگی‌ها:

موارد زیر ویژگی‌های گوناگونی از بخش‌پذیری را نشان می‌دهد. علاقه‌مندان می‌توانند برای مشاهده اثبات آن‌ها به پیوست آخر جلد دوم کتاب مراجعه نمایند:

ویژگی ۱: اگر $a | 1$ پس $a = \pm 1$

ویژگی ۲: اگر $a | b$ و $b | a$ پس $a = \pm b$

ویژگی ۳: اگر $a | b$ و $b | c$ پس $a | c$

ویژگی ۴: اگر $a \mid b$ و $a \mid c$ ، پس $a \mid (m \times b + n \times c)$ ، طوری که m و n اعداد صحیح دلخواه

هستند.

مثال ۶-۲

- (۱) از آنجا که $3 \mid 15$ و $15 \mid 45$ ، بر اساس ویژگی سوم داریم $3 \mid 45$.
- (۲) از آنجا که $3 \mid 15$ و $3 \mid 9$ بر اساس ویژگی چهارم $3 \mid (15 \times 2 + 9 \times 4)$ ، و به معنی $3 \mid 66$ می-باشد.

مجموعه‌ی مقسوم‌علیه‌ها

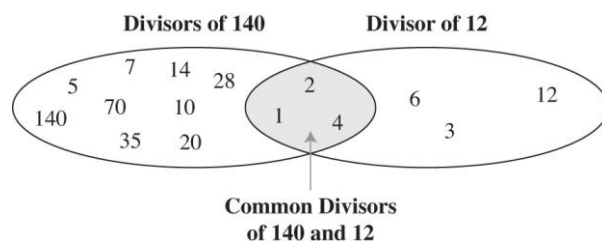
یک عدد صحیح مثبت می‌تواند بیش از یک مقسوم‌علیه داشته باشد. به عنوان مثال، عدد صحیح ۳۲ شش مقسوم‌علیه ۱، ۲، ۴، ۸، ۱۶ و ۳۲ دارد. در این قسمت می‌توانیم به دو واقعیت جالب در مورد مقسوم-علیه‌های اعداد صحیح مثبت اشاره کنیم:

واقعیت ۱: عدد صحیح ۱ تنها یک مقسوم‌علیه دارد و مقسوم‌علیه، خود عدد ۱ است.

واقعیت ۲: هر عدد صحیح مثبت حداقل ۲ مقسوم‌علیه دارد، ۱ و خودش (اما می‌تواند بیشتر هم داشته باشد).

بزرگ‌ترین مقسوم‌علیه مشترک^۱

عدد صحیحی که اغلب در رمزگذاری لازم است، بزرگ‌ترین مقسوم‌علیه مشترک دو عدد صحیح می-باشد. دو عدد صحیح مثبت می‌توانند چندین مقسوم‌علیه مشترک داشته باشند، اما فقط یکی از آن‌ها بزرگ‌ترین مقسوم‌علیه مشترک است؛ مثلاً مقسوم‌علیه‌های مشترک ۱۲ و ۱۴۰ عبارت‌اند از ۱، ۲ و ۴، پس بزرگ‌ترین مقسوم‌علیه مشترک ۴ است. به شکل ۶-۲ توجه کنید.



شکل ۶-۲: مقسوم‌علیه‌های مشترک دو عدد صحیح ۱۲ و ۱۴۰

^۱ Greatest Common Divisor

بزرگ‌ترین مقسوم‌علیه مشترک دو عدد صحیح مثبت عبارت است از بزرگ‌ترین عدد صحیحی که هر دو عدد به آن بخش‌پذیر باشند.

الگوریتم اقلیدسی^۱

وقتی دو عدد صحیح بزرگ باشند، یافتن بزرگ‌ترین مقسوم‌علیه مشترک آن‌ها، به وسیله لیست کردن تمام مقسوم‌علیه‌های آن‌ها عملی نیست. خوشبختانه ۲۰۰۰ سال پیش ریاضیدانی به نام اقلیدس الگوریتمی ارائه نمود، که می‌تواند بزرگ‌ترین مقسوم‌علیه مشترک دو عدد صحیح مثبت را پیدا کند. الگوریتم اقلیدسی مبتنی بر دو واقعیت زیر می‌باشد:

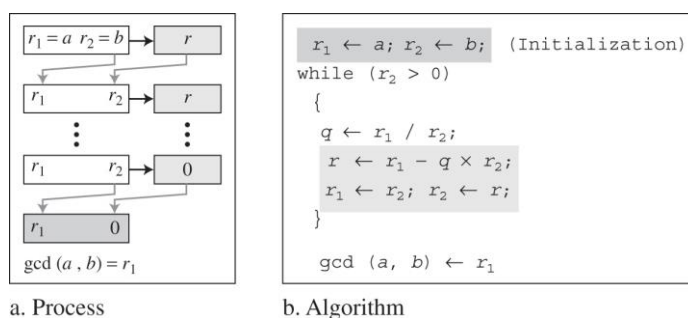
قانون ۱: $\gcd(a, 0) = a$.

قانون ۲: $\gcd(a, b) = \gcd(b, r)$ به طوری که r باقیمانده تقسیم a بر b است.

قانون اول به ما می‌گوید که اگر عدد صحیح دوم صفر باشد، بزرگ‌ترین مقسوم‌علیه مشترک، اولی است. قانون دوم به ما این امکان را می‌دهد که مقدار b و a را آن قدر تغییر دهیم تا b صفر شود. به عنوان مثال، برای محاسبه $\gcd(10, 36)$ همان‌گونه که در زیر نشان داده شده است، می‌توانیم قانون دوم را چندین بار و قانون اول را فقط یک بار به کار ببریم.

$$\gcd(10, 36) = \gcd(6, 10) = \gcd(4, 6) = \gcd(2, 4) = \gcd(0, 2) = 2$$

به عبارت دیگر $\gcd(10, 36) = 2$ ، $\gcd(6, 10) = 2$ والی آخر. این بدین معناست که به جای محاسبه $\gcd(36$ و $10)$ می‌توانیم $\gcd(0, 2)$ را پیدا کنیم. شکل ۷-۲ نشان می‌دهد که چگونه از دو قانون بالا برای محاسبه $\gcd(a, b)$ استفاده می‌کنیم.



a. Process

b. Algorithm

شکل ۷-۲: الگوریتم اقلیدسی

¹ Euclidean Algorithm

از دو متغیر r_1 و r_2 برای نگه داشتن مقادیر میانی، در طول فرایند کاهش استفاده می کنیم. مقدار اولیه a و b تعیین شده است. در هر مرحله، باقیمانده تقسیم r_1 بر r_2 را محاسبه می کنیم و نتیجه را در متغیر r ذخیره می کنیم، سپس r_1 را جایگزین r_2 و r_2 را جایگزین r می کنیم و این مراحل را آن قدر ادامه می دهیم تا زمانی که r_2 صفر شود. در پایان بزرگ ترین مقسوم علیه مشترک (a, b) ، r_1 حاصل می شود.

وقتی که $\gcd(a, b) = 1$ باشد، می گوئیم a و b نسبت به هم اول هستند.

مثال ۲-۷

بزرگ ترین مقسوم علیه مشترک ۲۷۴۰ و ۱۷۶۰ را پیدا کنید.

حل:

مراحل بالا را با استفاده از یک جدول اعمال می کنیم. مقدار اولیه r_1 را ۲۷۴۰ و r_2 را ۱۷۶۰ در نظر می گیریم. مقدار q را در هر مرحله نشان داده ایم، در پایان $\gcd(2740, 1760) = 20$ حاصل می شود.

q	r_1	r_2	r
1	2740	1760	980
1	1760	980	780
1	980	780	200
3	780	200	180
1	200	180	20
9	180	20	0
	20	0	

مثال ۲-۸

بزرگ ترین مقسوم علیه ۲۵ و ۶۰ را پیدا کنید.

حل:

این مثال را انتخاب کردیم تا نشان دهیم که اگر عدد اول کوچک تر از عدد دوم باشد اشکالی پیش نمی آید. بلافاصله ترتیب درست در مرحله ۲ شکل گرفته و $\gcd(25, 60) = 5$ به دست می آید.

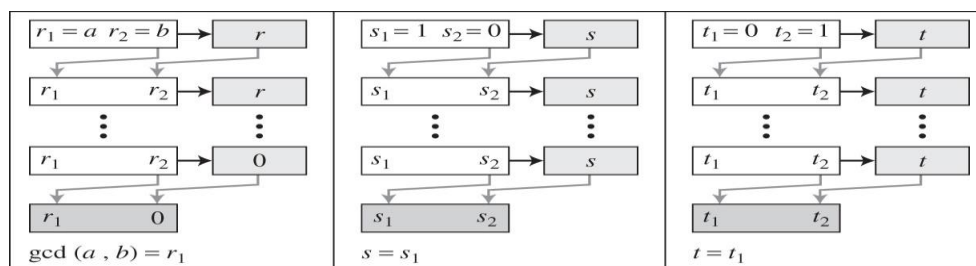
q	r_1	r_2	r
0	25	60	25
2	60	25	10
2	25	10	5
2	10	5	0
	5	0	

الگوریتم اقلیدسی بسط یافته^۱

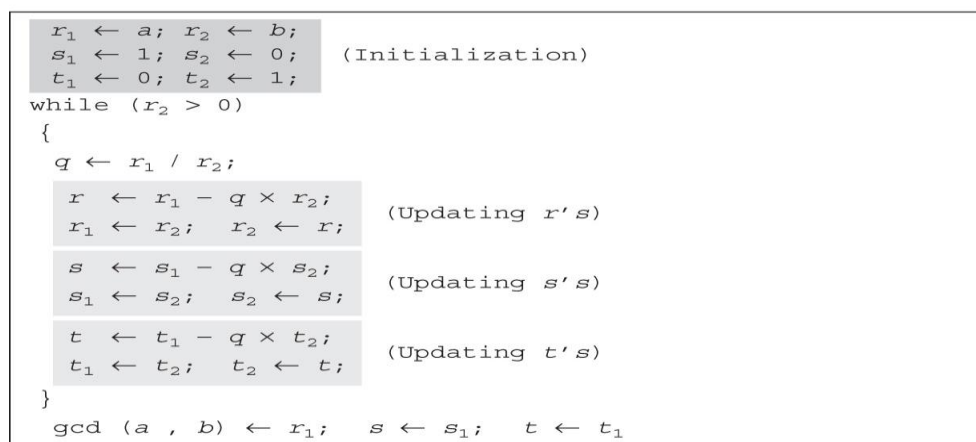
بیشتر وقت‌ها با در اختیار داشتن دو عدد صحیح a و b ، می‌بایست دو عدد صحیح دیگر t و s را پیدا کنیم، به گونه‌ای که:

$$s \times a + t \times b = \gcd(a, b)$$

الگوریتم اقلیدسی بسط یافته می‌تواند $\gcd(a, b)$ را پیدا کند و در عین حال، مقادیر s و t را هم محاسبه می‌نماید. با این حال، در هر مرحله به جای یک مجموعه، از سه مجموعه محاسباتی و تبدیلی استفاده می‌کنیم. این الگوریتم از سه مجموعه متغیر r, s, t استفاده می‌کند.



a. Process



b. Algorithm

شکل ۲-۸: الگوریتم اقلیدسی بسط یافته

در هر مرحله متغیرهای r_1, r_2 و r دارای مقدار یکسانی با الگوریتم اقلیدسی هستند. مقدار اولیه‌ی متغیرهای r_1 و r_2 ، به ترتیب a و b در نظر گرفته می‌شود. مقدار اولیه‌ی s_1 و s_2 به ترتیب ۱ و ۰ می‌باشد. مقدار اولیه‌ی متغیرهای t_1 و t_2 هم به ترتیب ۰ و ۱ در نظر گرفته می‌شود. محاسبه r, s, t

^۱ The Extended Euclidean Algorithm

مشابه‌اند؛ اما با یک تفاوت و آن این است که اگرچه r باقیمانده تقسیم r_1 بر r_2 است، اما چنین رابطه‌ای بین دو مجموعه‌ی دیگر وجود ندارد بلکه تنها یک خارج قسمت q وجود دارد که آن هم به صورت r_1/r_2 محاسبه شده و برای دو محاسبه دیگر به کار می‌رود.

مثال ۲-۹

فرض کنید $a=161$ و $b=28$ است، $\gcd(a,b)$ و همچنین مقادیر s و t را به دست آورید.

حل:

$$r = r_1 - q \times r_2 \quad s = s_1 - q \times s_2 \quad t = t_1 - q \times t_2$$

برای پیگیری مراحل الگوریتم، از جدولی به شکل زیر استفاده می‌کنیم.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
5	161	28	21	1	0	1	0	1	-5
1	28	21	7	0	1	-1	1	-5	6
3	21	7	0	1	-1	4	-5	6	-23
	7	0		-1	4		6	-23	

$\gcd(161, 28) = 7$, $s = -1$ و $t = 6$ را به دست آوردیم. پاسخ‌ها را برای بررسی درستی آن‌ها می-

توان آزمایش کرد؛ در نتیجه: $7 = 161 \times (-1) + 28 \times 6$

مثال ۲-۱۰

فرض کنید $a=17$ و $b=0$ باشد، $\gcd(a,b)$ و مقادیر s, t را پیدا کنید.

حل:

برای پیگیری مراحل الگوریتم از جدولی به شکل زیر استفاده می‌کنیم.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
	17	0		1	0		0	1	

توجه داشته باشید که نیازی به محاسبه‌ی s, r, q نداریم. اولین مقدار r_2 ، منطبق با معیار پایانی

الگوریتم است. $\gcd(17, 0) = 17$, $s = 1$ و $t = 0$ حاصل می‌شود که حاصل نشان می‌دهد که چرا باید مقدار

اولیه‌ی s_1 را ۱ و t_1 را ۰ قرار دهیم. پاسخ را می‌توانیم به شکل زیر آزمایش کنیم.

$$(1 \times 17) + (0 \times 0) = 17$$

مثال ۲-۱۱

فرض کنید $a=0$ و $b=45$ باشد. $\gcd(a,b)$ و مقادیر s, t را پیدا کنید.

حل:

برای پیگیری الگوریتم از جدول زیر استفاده می‌کنیم.

q	r_1	r_2	r	s_1	s_2	s	t_1	t_2	t
0	0	45	0	1	0	1	0	1	0
	45	0		0	1		1	0	

$\gcd(45, 0) = 45$, $s=0$ و $t=1$ حاصل می‌شود. حاصل نشان می‌دهد که چرا باید مقدار اولیه‌ی s_2

را ۰ و t_2 را ۱ قرار دهیم. پاسخ را می‌توانیم به شکل زیر آزمایش کنیم:

$$(0 \times 0) + (1 \times 45) = 45$$

معادلات خطی سیال^۱

اگرچه کاربرد بسیار مهم الگوریتم اقلیدسی بسط‌یافته را در بخش بعد خواهیم دید اما یک کاربرد فوری آن یافتن راه‌حل‌های معادلات خطی سیال دو متغیری می‌باشد؛ معادله‌ای نظیر $ax+by=c$. ما باید اعداد صحیحی برای y, x پیدا کنیم که در معادله صدق کند. این نوع معادلات یا راه‌حلی ندارند یا بی‌نهایت راه‌حل دارند. فرض کنید $d = \gcd(a, b)$. اگر $d \nmid c$ در این صورت معادله پاسخی ندارد. اگر $d \mid c$ در آن صورت بی‌نهایت راه‌حل داریم. یکی از آن‌ها، پاسخ ویژه و بقیه پاسخ کلی نامیده می‌شود.

معادله‌ی خطی سیال دو متغیره، به صورت $ax+by=c$ است.

پاسخ ویژه^۲

اگر $d \mid c$ باشد، می‌توان با استفاده از مراحل زیر پاسخی ویژه را برای معادله بالا یافت:

(۱) با تقسیم هر دو طرف معادله بر d ، معادله‌ی اولیه را به معادله $a_1x+b_1y=c_1$ کاهش دهید.

این تقسیم امکان‌پذیر است، زیرا می‌دانیم که c, b, a همگی بر ۱ بخش‌پذیرند.

(۲) با استفاده از الگوریتم اقلیدسی بسط‌یافته، رابطه‌ی $a_1s+b_1t=1$ را حل و مقادیر s, t پیدا کنید.

(۳) اکنون پاسخ ویژه را می‌توان به دست آورد:

$$x_0 = (c/d) s \text{ و } y_0 = (c/d) t$$

¹ Linear Diophantine Equations

² Particular Solution

پاسخ‌های کلی^۱

پس از یافتن پاسخ ویژه، پاسخ‌های کلی را می‌توان به شکل زیر به دست آورد:

K عدد صحیح دلخواه است $x = x_0 + k(b/d)$ and $y = y_0 - k(a/d)$: پاسخ‌های کلی

مثال ۲-۱۲

پاسخ ویژه و کلی معادله $21x + 14y = 35$ را پیدا کنید.

حل:

$d = \gcd(21, 14) = 7$ از آنجا که $7 | 35$ است، پس معادله دارای بینهایت راه حل می‌باشد. برای یافتن

معادله $3x + 2y = 5$ می‌توانیم هر دو طرف را به ۷ تقسیم کنیم. با استفاده از الگوریتم اقلیدسی بسط‌یافته،

s, t را به این صورت به دست آوریم: $3s + 2t = 1$.

$s = 1$ و $t = -1$ را داریم، پس راه‌حل‌ها عبارت‌اند از:

پاسخ ویژه $x_0 = 5 \times 1 = 5$ and $y_0 = 5 \times (-1) = -5$ since $35 | 7 = 5$

K عدد صحیح دلخواه است $x = 5 + k \times 2$ and $y = -5 - k \times 3$: پاسخ کلی

بنابراین، راه‌حل‌ها، $(5, -5)$ ، $(7, -8)$ ، $(9, -11)$ و می‌باشد. به سادگی می‌توانیم آزمایش کنیم که

آیا هر یک از این راه‌حل‌ها در معادله صدق می‌کند یا نه؟

مثال ۲-۱۳

یک کاربرد بسیار جالب در زندگی روزمره، زمانی است که بخواهیم ترکیبات مختلف اشیاء دارای مقادیر

مختلف را پیدا کنیم. مثلاً فرض کنید می‌خواهید یک اسکناس ۱۰۰ دلاری را خرد کنید و چند اسکناس

۲۰ دلاری و ۵ دلاری بگیرید. انتخاب‌های گوناگونی داریم که می‌توانیم با حل معادله‌ی سیال به

شکل $20x + 5y = 100$ آن‌ها را پیدا کنیم. از آنجا که $d = \gcd(5, 20) = 5$ می‌باشد، معادله دارای بینهایت

راه‌حل است، اما در این مورد تنها تعداد کمی از آن‌ها (تنها پاسخ‌هایی که در آن‌ها هم x و هم y ، اعداد

صحیح غیر منفی باشند) پذیرفتنی‌اند. با تقسیم هر دو طرف بر ۵، معادله‌ی $4x + y = 20$ را به دست می-

آوریم؛ سپس معادله $4s + t = 1$ را حل می‌کنیم. با استفاده از الگوریتم اقلیدسی بسط‌یافته می‌توانیم $s = 0$

=====

¹ General Solutions

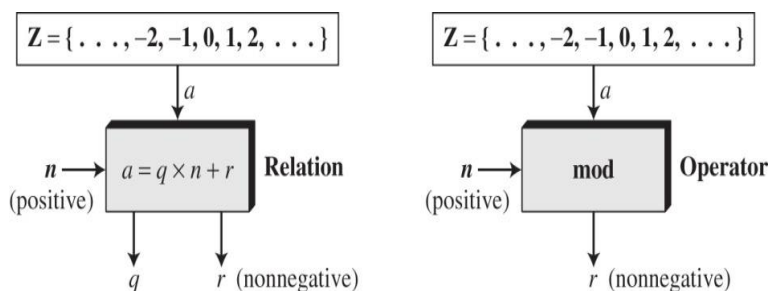
و $t=1$ را به دست آوریم. راه‌حل‌های کلی با x و y غیر منفی عبارت‌اند از $(۱۲,۲)$ ، $(۱۶,۱)$ ، $(۲۰,۰)$ ، $(۸,۳)$ ، $(۴,۴)$ و $(۰,۵)$. سایر راه‌حل‌ها قابل قبول نیستند، زیرا y منفی می‌شود. مشتری در بانک باید بگوید که کدام یک از ترکیبات فوق را می‌خواهد. اولین ترکیب فوق، اسکناس ۲۰ دلاری ندارد، آخرین ترکیب هم اسکناس ۵ دلاری ندارد.

۲-۲- حساب پیمانه‌ای^۱

معادله تقسیم $a = q \times n + r$ که در بخش پیش در مورد آن بحث شد، دارای دو ورودی (n, a) و دو خروجی (q, r) است. در حساب پیمانه‌ای، تنها یک خروجی برای ما اهمیت دارد و آن هم باقیمانده r است. خارج قسمت q برای ما اهمیتی ندارد؛ به عبارت دیگر، می‌خواهیم بدانیم که وقتی a را بر n تقسیم می‌کنیم مقدار r چند است. پس امکان تبدیل معادله تقسیم با دو ورودی را به یک عملگر دودویی با دو ورودی n, a و یک خروجی r را داریم.

عملگر پیمانه‌ای^۲

عملگر دودویی بالا، عملگر پیمانه‌ای نامیده می‌شود و آن را به صورت mod نشان می‌دهند. ورودی دوم (n) را پیمانه می‌نامند. خروجی r مانده نامیده می‌شود. شکل ۲-۹ معادله تقسیم را در مقایسه با عملگر پیمانه‌ای نشان می‌دهد.



شکل ۲-۹: رابطه‌ی تقسیم و عملگر مد

همان‌گونه که شکل ۲-۹ نشان می‌دهد، عملگر mod ، یک عدد صحیح a از مجموعه Z و یک

پیمانه مثبت n را گرفته و مانده‌ی غیر منفی r را محاسبه می‌نماید پس می‌توانیم بگوییم:

$$a \bmod n = r$$

¹ Modular Arithmetic

² Modulo Operator

مثال ۲- ۱۴

نتیجه اعمال زیر را به دست آورید:

- a) $27 \bmod 5$
- b) $36 \bmod 12$
- c) $-18 \bmod 14$
- d) $-7 \bmod 10$

حل:

ما به دنبال باقیمانده‌ی «r» هستیم. می‌توانیم a را بر n تقسیم کنیم و q و r را به دست آوریم و سپس q را نادیده گرفته و r را نگه می‌داریم.

(a) با تقسیم ۲۷ بر ۵، $r=2$ می‌شود. این بدین معنی است که $27 \bmod 5 = 2$.

(b) با تقسیم ۳۶ بر ۱۲، $r=0$ می‌شود. این بدین معنی است که $36 \bmod 12 = 0$.

(c) با تقسیم -18 بر ۱۴، $r=-4$ می‌شود. با این وجود می‌بایست پیمانه ۱۴ را به آن اضافه کنیم تا حاصل را غیر منفی نماییم؛ در نتیجه $r = -4 + 14 = 10$ حاصل می‌شود. این بدین معنی است که $-18 \bmod 14 = 10$.

(d) با تقسیم -7 بر ۱۰، $r=-7$ می‌شود. پس از افزودن مقدار پیمانه به -7 ، $r=3$ حاصل می‌شود. این بدین معنی است که $-7 \bmod 10 = 3$.

مجموعه‌ی باقیمانده‌ها^۱

نتیجه عملگر پیمانه‌ای به پیمانه‌ی n، همیشه یک عدد صحیح بین ۰ و $n-1$ است؛ به عبارت دیگر، نتیجه‌ی $a \bmod n$ همیشه یک عدد صحیح بین ۰ و $n-1$ است؛ به بیان دیگر، نتیجه $a \bmod n$ همیشه یک عدد صحیح غیر منفی کمتر از n است. پس عملگر پیمانه‌ای مجموعه‌ای ایجاد می‌کند که در حساب پیمانه‌ای به آن، مجموعه حداقل باقیمانده‌ها به پیمانه n^2 یا Z_n گفته می‌شود. ولی باید به خاطر

¹ Set Of Residues

² Set Of Least Residues modulo n

داشته باشیم که اگرچه فقط یک مجموعه اعداد صحیح Z داریم، اما برای هر مقدار n بی نهایت نمونه از مجموعه‌ی باقیمانده‌ها (Z_n) داریم. شکل ۱۰-۲ مجموعه Z_n و سه نمونه Z_2, Z_6, Z_{11} را نشان می‌دهد.

$$Z_n = \{ 0, 1, 2, 3, \dots, (n-1) \}$$

$$Z_2 = \{ 0, 1 \} \quad Z_6 = \{ 0, 1, 2, 3, 4, 5 \} \quad Z_{11} = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 \}$$

شکل ۱۰-۲: برخی از مجموعه‌های Z_n

تجانس^۱

در رمزنگاری، بیشتر از مفهوم تجانس به جای تساوی استفاده می‌کنیم. نگاشت از Z به Z_n یک به یک نیست بلکه بی نهایت عضو از Z می‌توانند متناسب با یکی از اعضای Z_n باشد. به عنوان مثال نتیجه‌ی $10 \bmod 2 = 0$ ، $12 \bmod 2 = 0$ ، $22 \bmod 2 = 0$ است و الی آخر.

در حساب پیمانه‌ای به اعداد صحیحی مانند ۲، ۱۲ و ۲۲ متجانس به پیمانه ۱۰ می‌گویند. برای نشان دادن این که دو عدد صحیح متجانس هستند، از عملگر تجانس^۲ استفاده می‌شود. برای تعیین مقدار پیمانه‌ای که رابطه را معتبر می‌کند، عبارت $(\bmod n)$ را به سمت راست معادله تجانس اضافه می‌کنیم، مثلاً می‌نویسیم:

$$\begin{array}{llll} 2 \equiv 12 \pmod{10} & 13 \equiv 23 \pmod{10} & 34 \equiv 24 \pmod{10} & -8 \equiv 12 \pmod{10} \\ 3 \equiv 8 \pmod{5} & 8 \equiv 13 \pmod{5} & 23 \equiv 33 \pmod{5} & -8 \equiv 2 \pmod{5} \end{array}$$

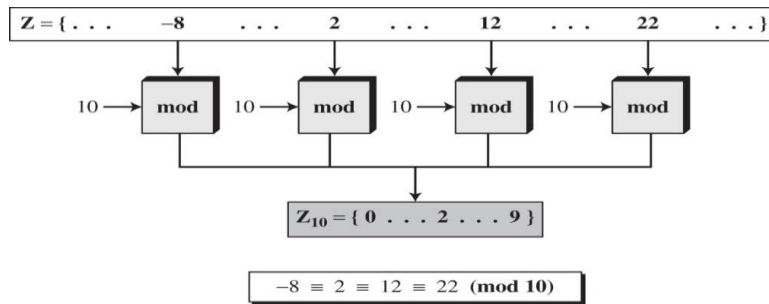
شکل ۱۱-۲ رابطه تجانس را نشان می‌دهد. لازم است چند نکته را توضیح دهیم.

(a) عملگر تجانس شبیه عملگر تساوی است، اما با تفاوت‌هایی. نخست: عملگر تساوی یک عضو از مجموعه Z را به خودش نگاشت می‌کند؛ اما عملگر تجانس یک عضو از Z را به عضوی از Z_n نگاشت می‌کند. دوم: عملگر مساوی یک به یک است اما عملگر تجانس، چند به یک^۳ است.

¹ Congruence

² Congruence Operator

³ Many-To-One



شکل ۲-۱۱: مفهوم تجانس

(b) عبارت $(\text{mod } n)$ که به سمت راست عملگر تجانس اضافه کرده‌ایم، تنها نشانه‌ای از مجموعه‌ی مقصد (Z_n) می‌باشد. پس برای نشان دادن این که چه پیمانه‌ای در نگاشت به کار رفته است، این عبارت را اضافه نمودیم. علامت mod که در اینجا به کار رفته است، معنای یکسانی با عملگر دودویی ندارد، به عبارت دیگر، علامت mod در $10 \text{ mod } 12$ یک عملگر است، در حالی که عبارت $(\text{mod } 10)$ در $12 \equiv 2 \pmod{10}$ به این معنی است که مجموعه مقصد، Z_{10} است.

کلاس‌های باقیمانده^۱

کلاس‌های باقیمانده $[a]$ یا $[a]_n$ مجموعه اعداد صحیح متجانس به پیمانه n است، به عبارت دیگر، مجموعه‌ی تمام اعداد صحیح نظیر $x = a \pmod{n}$ می‌باشد. به عنوان مثال اگر $n=5$ باشد، همان‌گونه که مشاهده می‌شود پنج مجموعه $[0], [1], [2], [3], [4]$ به شکل زیر داریم:

$$[0] = \{ \dots, -15, -10, -5, 0, 5, 10, 15, \dots \}$$

$$[1] = \{ \dots, -14, -9, -4, 1, 6, 11, 16, \dots \}$$

$$[2] = \{ \dots, -13, -8, -3, 2, 7, 12, 17, \dots \}$$

$$[3] = \{ \dots, -12, -7, -2, 3, 8, 13, 18, \dots \}$$

$$[4] = \{ \dots, -11, -6, -1, 4, 9, 14, 19, \dots \}$$

وقتی که عملگر باقیمانده به پیمانه ۵ را بر روی مجموعه‌ی نخست اعمال می‌کنیم، کل مجموعه به صفر نگاشت می‌شود. نتیجه‌ی نگاشت به مجموعه‌ی دوم یک است و الی آخر. در هر مجموعه، عضوی وجود دارد که به آن کوچک‌ترین باقیمانده^۲ (غیر منفی) می‌گویند. در مجموعه اول، این عضو ۰

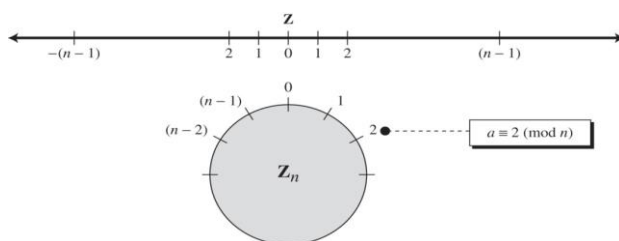
^۱ Residue Classes

^۲ least residue

است و در مجموعه ۱، این عضو ۱ است و الی آخر. مجموعه تمام کوچک‌ترین باقیمانده‌ها را به شکل $Z_5 = \{4, 3, 2, 1, 0\}$ نشان می‌دهیم؛ به عبارت دیگر مجموعه Z_n ، مجموعه تمام کوچک‌ترین باقیمانده به پیمانه n است.

نمایش مفهوم تجانس به شکل دوار^۱

می‌توان با استفاده از دایره، مفهوم تجانس را بهتر درک کرد. همان‌گونه که از بردار، برای نشان دادن توزیع اعداد صحیح در Z استفاده می‌کنیم؛ می‌توانیم از دایره برای نشان دادن توزیع اعداد صحیح در Z_n استفاده کنیم. شکل ۱۲-۲ مقایسه‌ی بین این دو را نشان می‌دهد.



شکل ۱۲-۲: مقایسه Z و Z_n به شکل نمودار

اعداد صحیح ۰ تا $n-1$ به طور منظم در اطراف یک دایره قرار گرفته‌اند. تمام اعداد صحیح به پیمانه n ، همان نقاط روی بردار را بر روی دایره هم اشغال کرده‌اند. اعداد صحیح مثبت و منفی از مجموعه Z به گونه‌ای با دایره تطابق داده شده‌اند که تقارن بین آن‌ها وجود داشته باشد.

مثال ۱۵-۲

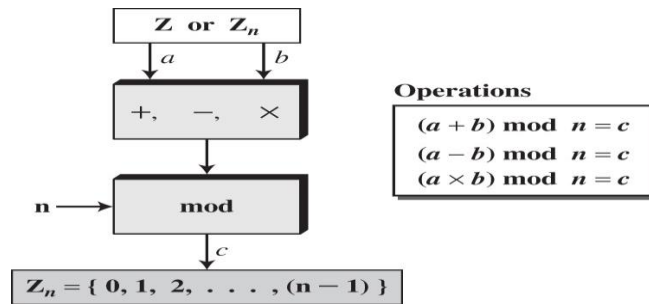
در زندگی روزمره از حساب پیمانه‌ای استفاده می‌کنیم. به عنوان مثال، از ساعت برای اندازه‌گیری زمان استفاده می‌کنیم. سیستم ساعتی ما از حساب پیمانه‌ای به پیمانه‌ی ۱۲ استفاده می‌کند ولی به جای ۰ از عدد ۱۲ استفاده می‌کنیم؛ بنابراین سیستم ساعتی ما از ۰ یا ۱۲ شروع می‌شود و تا ۱۱ ادامه می‌یابد. از آنجا که روز ۲۴ ساعت است، دو بار این دایره را دور می‌زنیم؛ دور نخست را با A.M و دور دوم را با P.M نشان می‌دهیم.

=====

¹ Circular Notation

عملگرها در Z_n

سه عملگر دودویی (جمع، تفریق و ضرب) را که برای مجموعه Z در مورد آن‌ها بحث کردیم، می‌توانیم برای مجموعه Z_n نیز تعریف کنیم. همان‌گونه که در شکل ۲-۱۳ نشان داده شده است، شاید برخی اوقات لازم باشد با استفاده از عملگر پیمانه، نتیجه را با Z_n نگاشت کنیم.



شکل ۲-۱۳: اعمال دودویی در Z_n

در واقع، دو مجموعه از اعمال در اینجا مورد استفاده قرار می‌گیرد. در مجموعه اول، یکی از عملگرهای $+$ ، $-$ ، \times و در دومین مجموعه عملگر mod استفاده می‌شود. برای نشان دادن ترتیب عملگرها باید از پرانتز استفاده کنیم. همان‌گونه که شکل ۲-۱۳ نشان می‌دهد، ورودی‌های عملگر (b, a) می‌توانند عضو Z یا Z_n باشند.

مثال ۲-۱۶

اعمال زیر را انجام دهید (ورودی‌ها از Z_n در نظر بگیرید).

(a) ۷ را در Z_{15} به ۱۴ اضافه کنید.

(b) ۱۱ را از ۷ در Z_{13} کم کنید.

(c) ۱۱ را در Z_{20} در ۷ ضرب کنید.

حل:

راه‌حل زیر نشان می‌دهد که پیدا کردن پاسخ در هر مورد شامل دو مرحله است:

$$(14 + 7) \bmod 15 \rightarrow (21) \bmod 15 = 6$$

$$(7 - 11) \bmod 13 \rightarrow (-4) \bmod 13 = 9$$

$$(7 \times 11) \bmod 20 \rightarrow (77) \bmod 20 = 17$$

مثال ۲-۱۷

اعمال زیر را انجام دهید (ورودی‌ها را از Z یا Z_n در نظر بگیرید).

(a) ۱۷ را به ۲۷ در Z_{14} اضافه کنید.

(b) ۳۴ را از ۱۲ در Z_{13} کم کنید.

(c) ۱۲۳ را در ۱۰- در Z_{19} ضرب کنید.

حل:

راه‌حل زیر نشان می‌دهد که هر مورد شامل دو مرحله است :

$$(17+27) \bmod 14 \rightarrow (44) \bmod 14 = 2$$

$$(12-43) \bmod 13 \rightarrow (-31) \bmod 13 = 8$$

$$(123 \times (-10)) \bmod 19 \rightarrow (-1230) \bmod 19 = 5$$

ویژگی‌ها

پیشتر اشاره کردیم که دو ورودی به سه عمل دودویی در حساب پیمانه‌ای، می‌توانند هم از Z و هم از Z_n باشند. ویژگی‌های زیر به ما این امکان را می‌دهد که پیش از انجام اعمال $+$, $-$, \times ، اگر ورودی عضوی از Z باشد، نخست آن‌ها را به Z_n نگاشت کنیم سپس عملیات را انجام دهیم. علاقمندان می‌توانند برای مشاهده اثبات این ویژگی‌ها به پیوست آخر جلد دوم کتاب مراجعه کنند.

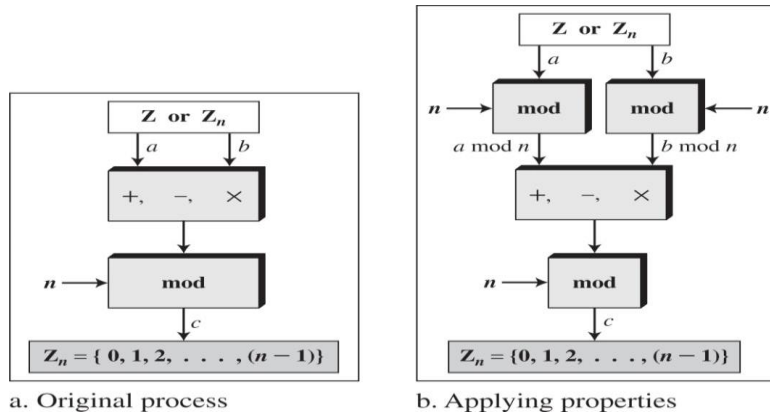
ویژگی نخست: $(a + b) \bmod n = [(a \bmod n) + (b \bmod n)] \bmod n$

ویژگی دوم: $(a - b) \bmod n = [(a \bmod n) - (b \bmod n)] \bmod n$

ویژگی سوم: $(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

شکل ۲-۱۴ مراحل پیش و پس از اعمال ویژگی‌های بالا را نشان می‌دهد. اگر چه شکل نشان می‌دهد که در صورت اعمال ویژگی‌های بالا مراحل طولانی‌تر می‌شود، اما باید به خاطر داشته باشیم که در رمزگذاری، با اعداد صحیح بسیار بزرگ روبرو هستیم. مثلاً اگر یک عدد صحیح بسیار بزرگی را در عدد صحیح بسیار بزرگ دیگری ضرب کنیم، ممکن است حاصل عدد صحیحی باشد که برای ذخیره شدن در رایانه، بیش از اندازه بزرگ باشد. به کار بردن ویژگی‌های بالا، نخست اعداد صحیح بزرگ را به اعداد صحیح کوچک‌تر تبدیل کرده و پس از آن دو عدد کوچک را در هم ضرب می‌کند؛ به عبارت دیگر

استفاده از ویژگی‌های بالا باعث می‌شود که با اعداد کوچک‌تر کار کنیم. این مزیت، در بحث‌های آتی در خصوص عملگرهایی که بعداً در این فصل بررسی می‌کنیم خود را نمایان‌تر بروز می‌دهد.



a. Original process

b. Applying properties

شکل ۲-۱۴: ویژگی‌های عملگر mod

با اعمال ویژگی‌های بالا، اندازه‌ی دو عملوند، پیش از انجام ضرب کوچک‌تر می‌شود؛ به عبارت دیگر این ویژگی‌ها به ما امکان کار کردن با اعداد کوچک‌تر را می‌دهند.

مثال ۲-۱۸

موارد زیر کاربرد ویژگی‌های بالا را نشان می‌دهند:

$$(1'723'345 + 2'124'945) \bmod 11 = (8+9) \bmod 11 = 6$$

$$(1'723'345 - 2'124'945) \bmod 16 = (8-9) \bmod 11 = 10$$

$$(1'723'345 \times 2'124'945) \bmod 16 = (8 \times 9) \bmod 11 = 6$$

مثال ۲-۱۹

در حساب، اغلب باید باقیمانده عدد صحیح را وقتی که بر توان‌هایی از ۱۰ تقسیم می‌شود، پیدا کنیم. به عنوان مثال، باید $10^3 \bmod 3$ ، $10^2 \bmod 3$ ، $10 \bmod 3$ و الی آخر را پیدا کنیم. همچنین باید $10^3 \bmod 7$ ، $10^2 \bmod 7$ ، $10 \bmod 7$ و الی آخر را پیدا کنیم. سومین ویژگی عملگر پیمانه، که در بالا به آن اشاره شد، کار را بسیار ساده‌تر می‌کند.

$$10^n \bmod x = (10 \bmod x)^n$$

در اینجا سومین ویژگی را n بار به کار می‌بریم؛ بنابراین داریم:

$$10 \bmod 3 = 1 \rightarrow 10^3 \bmod 3 = (10 \bmod 3)^n = 1$$

$$10 \bmod 9 = 1 \rightarrow 10^3 \bmod 9 = (10 \bmod 9)^n = 1$$

$$10 \bmod 7 = 1 \rightarrow 10^3 \bmod 7 = (10 \bmod 7)^3 = 1^3 \bmod 7 = 1$$

مثال ۲-۲۰

در حساب به ما گفته شده است که باقیمانده‌ی تقسیم یک عدد صحیح بر ۳، با باقیمانده‌ی تقسیم مجموع ارقام آن بر ۳ یکی است؛ به عبارت دیگر باقیمانده‌ی تقسیم ۶۳۷۱ بر سه، با باقیمانده‌ی تقسیم ۱۷ بر ۳ یکی است، زیرا $1+7+3+6=17$ می‌باشد. با استفاده از ویژگی‌های عملگر \bmod می‌توانیم این ادعا را ثابت کنیم. یک عدد صحیح را به شکل مجموع ارقام آن ضرب در توان ۱۰ می‌نویسیم:

$$a = a_n \times 10^n + \dots + a_1 \times 10^1 + a_0 \times 10^0$$

$$\text{مثلاً } 6371 = 6 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 1 \times 10^0$$

اکنون می‌توانیم عملگر پیمانه را برای هر دو طرف معادله اعمال کنیم و از مثال قبلی، که $10 \bmod 3 = 1$ برابر است با ۱ استفاده کنیم.

$$a \bmod 3 = (a_n \times 10^n + \dots + a_1 \times 10^1 + a_0 \times 10^0) \bmod 3$$

$$= (a_n \times 10^n) \bmod 3 + \dots + (a_1 \times 10^1) \bmod 3 + (a_0 \times 10^0) \bmod 3$$

$$= (a_n \bmod 3) \times (10^n \bmod 3) + \dots + (a_1 \bmod 3) \times (10^1 \bmod 3) + (a_0 \bmod 3) \times (10^0 \bmod 3)$$

$$= a_n \bmod 3 + \dots + a_1 \bmod 3 + a_0 \bmod 3$$

$$= (a_n + \dots + a_1 + a_0) \bmod 3$$

وارون‌ها

وقتی که با حساب پیمانه‌ای سروکار داریم، غالباً باید وارون یک عدد را نسبت به یک عمل پیدا کنیم. معمولاً به دنبال وارون جمع^۱ یا وارون ضرب^۲ هستیم.

وارون نسبت به عمل جمع

در مجموعه Z_n ، دو عدد a, b وارون جمعی یکدیگر هستند اگر $a+b \equiv 0 \pmod{n}$.

در مجموعه Z_n وارون نسبت به جمع عدد a را می‌توان به شکل $b = n-a$ محاسبه کرد. مثلاً

وارون نسبت به جمع عدد ۴ در مجموعه Z_{10} ، برابر $6 = 10-4$ است.

=====

^۱ وارون نسبت به عمل جمع [م]

^۲ وارون نسبت به عمل ضرب [م]

در حساب پیمانه‌ای، هر عدد صحیح یک وارون در عمل جمع دارد.

مجموع عدد صحیح و وارون آن متجانس صفر است به پیمانه n

توجه داشته باشید که در حساب پیمانه‌ای، هر عدد یک وارون جمع دارد و این وارون، منحصر به فرد است. هر عدد یک و تنها یک وارون نسبت به جمع دارد؛ اگرچه ممکن است وارون یک عدد، خود آن عدد باشد.

مثال ۲-۲۱

تمام زوج‌های وارون یکدیگر نسبت به جمع را در مجموعه Z_{10} پیدا کنید.

حل:

شش زوج از وارون‌ها نسبت به جمع عبارت‌اند از: $(0,0)$ ، $(1,9)$ ، $(2,8)$ ، $(3,7)$ ، $(4,6)$ ، $(5,5)$. در این لیست ۰ وارون خودش نسبت به عمل جمع است؛ ۵ نیز همین وضعیت را دارد. توجه داشته باشید که وارون‌ها نسبت به جمع، دو طرفه هستند. اگر ۴ وارون نسبت به جمع ۶ است، ۶ هم وارون نسبت به جمع ۴ است.

وارون نسبت به ضرب

در مجموعه Z_n ، دو عدد a, b وارون ضربی همدیگر هستند، اگر $a \times b \equiv 1 \pmod{n}$.

مثلاً اگر پیمانه عدد ۱۰ باشد، در این صورت وارون ضربی ۳، برابر ۷ است. زیرا

$$(3 \times 7) \bmod 10 = 1$$

در حساب پیمانه‌ای، ممکن است یک عدد صحیح، وارون نسبت به ضرب داشته باشد یا نداشته

باشد. وقتی که عدد، وارون ضربی داشته باشد، حاصل ضرب عدد صحیح و وارون ضربی‌اش متجانس

است با یک به پیمانه n .

می‌توان ثابت کرد که a وارون ضربی در مجموعه Z_n دارد، اگر و تنها اگر $\gcd(n, a) = 1$ باشد. در

این حالت a, n نسبت به هم اول هستند.

مثال ۲-۲۲

وارون ضربی عدد ۸ در مجموعه Z_{10} را پیدا کنید.

حل:

در اینجا هیچ وارون ضربی وجود ندارد، زیرا $\gcd(10, 8) = 2 \neq 1$ می باشد؛ به عبارت دیگر نمی توانیم عددی بین ۰ و ۹ پیدا کنیم که وقتی در ۸ ضرب شود، نتیجه آن متجانس با یک باشد.

مثال ۲-۲۳

تمام وارون های ضربی در مجموعه Z_{10} را پیدا کنید.

حل:

در اینجا تنها سه زوج وارون ضربی وجود دارد: $(1, 1)$ ، $(3, 7)$ ، $(9, 9)$. اعداد ۰، ۲، ۴، ۵، ۶، ۸ وارون ضربی ندارند. می توانیم مشاهده کنیم که رابطه ای $1 = 10 \bmod (1 \times 1)$ برقرار است.

مثال ۲-۲۴

تمام وارون های ضربی در مجموعه Z_{11} را پیدا کنید.

حل:

هفت زوج وارون ضربی دارند: $(1, 1)$ ، $(2, 6)$ ، $(3, 4)$ ، $(5, 9)$ ، $(7, 8)$ ، $(9, 5)$ ، $(10, 10)$. در مقایسه Z_{10} با Z_{11} ، تعداد \gcd ها دو برابر شد. علت این است که در Z_{11} ، $\gcd(11, a)$ برای تمام مقادیر a به غیر از ۰ وارون ضربی وجود دارند.

عدد صحیح a در Z_n وارون ضربی دارد، اگر و تنها اگر $\gcd(n, a) = 1 \bmod n$ باشد.

الگوریتم اقلیدسی بسط یافته، که پیشتر در این فصل در مورد آن صحبت کردیم، می تواند وارون ضربی b در Z_n را پیدا کند، به شرطی که b, n معلوم باشد و وارون وجود داشته باشد. برای نشان دادن این مطلب اجازه دهید n (پیمانه) را جایگزین اولین عدد صحیح یعنی a کنیم. می توانیم بگوییم که این الگوریتم می تواند t, s را به این صورت به دست آورد: $s \times n + b \times t = \gcd(n, b)$ ؛ با این حال، اگر وارون ضربی b وجود داشته باشد، $\gcd(n, b)$ باید ۱ باشد؛ بنابراین رابطه به شکل زیر است:

$$(s \times n) + (b \times t) = 1$$

اکنون عملگر پیمانه ای را بر هر دو طرف اعمال می کنیم؛ به عبارت دیگر هر طرف را به Z_n نگاشت می کنیم.

$$(n \times s + t \times b) \bmod n = 1 \bmod n$$

$$[(s \times n) \bmod n] + [(b \times t) \bmod n] = 1 \bmod n$$

$$+ [(b \times t) \bmod n] = 1$$

$(b \times t) \bmod n = 1$ ← به این معنی است که t ، وارون ضربی عدد b در مجموعه Z_n است.

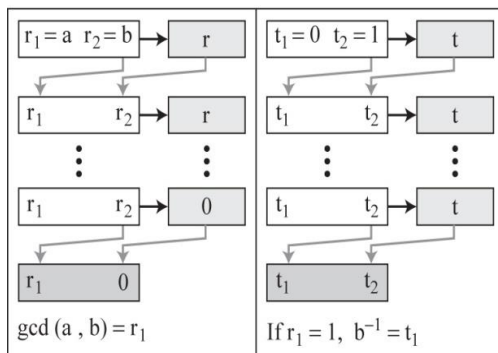
توجه داشته باشید که حاصل $[(s \times n) \bmod n]$ در خط سوم ۰ است، زیرا اگر $(s \times n)$ را بر n تقسیم کنیم خارج قسمت برابر S ، اما باقیمانده ۰ است.

الگوریتم اقلیدسی بسط یافته می تواند وارون ضربی b در Z_n را پیدا کنید، به شرطی که b, n مشخص شده باشد و $\gcd(n, b) = 1$ باشد.

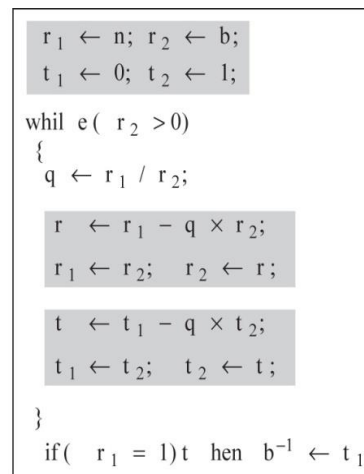
وارون ضربی b پس از نگاشت به Z_n برابر با مقدار t است.

شکل ۲-۱۵ نشان می دهد که چگونه می توان با استفاده از الگوریتم اقلیدسی بسط یافته، وارون

ضربی یک عدد را پیدا کرد.



a. Process



b. Algorithm

شکل ۲-۱۵: استفاده از الگوریتم اقلیدسی بسط یافته برای یافتن وارون ضربی

مثال ۲-۲۵

وارون ضربی عدد ۱۱ در مجموعه Z_{26} را پیدا کنید.

حل:

از جدولی شبیه به جدولی که پیشتر برای $r_1 = 26$ و $r_2 = 11$ به کار بردیم، استفاده می کنیم. تنها مقدار t برای ما اهمیت دارد.

q	r_1	r_2	r	t_1	t_2	t
2	26	11	4	0	1	-2
2	11	4	3	1	-2	5
1	4	3	1	-2	5	-7
3	3	1	0	5	-7	26
	1	0		-7	26	

$\gcd(26, 11)$ برابر ۱ است و این بدان معنی است که وارون ضربی ۱۱ وجود دارد. در الگوریتم

اقلیدسی بسط یافته $t_1 = -7$ به دست می آید. وارون ضربی $19 = 26 \bmod (-7)$ می باشد؛ به عبارت دیگر

۱۱ و ۱۹ وارون ضربی در Z_{26} هستند. چون $1 = 26 \bmod 26 = 209 \bmod 26 = (11 \times 19) \bmod 26$ است.

مثال ۲-۲۶

وارون ضربی عدد ۲۳ در مجموعه Z_{100} را پیدا کنید.

حل:

از جدولی شبیه به جدولی که پیشتر برای $r_1 = 100$ و $r_2 = 23$ به کار بردیم، استفاده می کنیم. فقط مقدار t

برایمان اهمیت دارد.

q	r_1	r_2	r	t_1	t_2	t
4	100	23	8	0	1	-4
2	23	8	7	1	-4	19
1	8	7	1	-4	9	-13
7	7	1	0	9	-13	100
	1	0		-13	100	

$\gcd(100, 23)$ برابر ۱ است و این بدان معنی است که وارون ۲۳ وجود دارد. الگوریتم اقلیدسی

بسط یافته $t_1 = -13$ را نتیجه می دهد. وارون ضربی $87 = 100 \bmod (-13)$ می باشد؛ به عبارت دیگر ۱۳ و

۸۷ وارون ضربی هم در Z_{100} هستند. پس داریم:

$$(23 \times 87) \bmod 100 = 201 \bmod 100 = 1$$

مثال ۲-۲۷

وارون ضربی عدد ۱۲ در مجموعه Z_{26} را پیدا کنید.

حل:

از جدولی با $r_1 = 26$ و $r_2 = 12$ که شبیه به جدولی است که پیشتر به کار بردیم استفاده می کنیم.

q	r_1	r_2	r	t_1	t_2	t
2	26	12	2	0	1	-2
6	12	2	0	1	-2	13
	2	0		-2	13	

$\gcd(12, 26) = 2 \neq 1$ می باشد و به این معنی است که وارون ضربی برای عدد ۱۲ در مجموعه Z_{26}

وجود ندارد.

جدول های وارون جمع و ضرب

شکل ۲-۱۶ دو جدول جمع و ضرب را در Z_{10} نشان می دهد. در جدول مربوط به جمع، هر عدد صحیح یک وارون نسبت به جمع دارد. زوج های وارون را زمانی می توان یافت که حاصل جمعشان صفر باشد؛ در نتیجه زوج های $(0,0)$ ، $(9,1)$ ، $(8,2)$ ، $(7,3)$ ، $(6,4)$ ، $(5,5)$ حاصل می شود. در جدول ضرب، فقط سه زوج $(1,1)$ ، $(3,7)$ ، $(9,9)$ را داریم. زوج هایی مورد نظر است که حاصل ضرب آنها ۱ باشد. هر دو جدول نسبت به قطر عناصر، که از بالای سمت چپ به قسمت پایین سمت راست کشیده شده است متقارن هستند و این نشان دهنده ویژگی جابجایی جمع و ضرب است ($a \times b = b \times a$, $a + b = b + a$). همچنین جدول جمع نشان می دهد که هر سطر و ستون، جایگشت سطر و ستون دیگری است. این نکته در مورد جدول ضرب درست نیست.

	0	1	2	3	4	5	6	7	8	9
0	0	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9	0
2	2	3	4	5	6	7	8	9	0	1
3	3	4	5	6	7	8	9	0	1	2
4	4	5	6	7	8	9	0	1	2	3
5	5	6	7	8	9	0	1	2	3	4
6	6	7	8	9	0	1	2	3	4	5
7	7	8	9	0	1	2	3	4	5	6
8	8	9	0	1	2	3	4	5	6	7
9	9	0	1	2	3	4	5	6	7	8

Addition Table in Z_{10}

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7	8	9
2	0	2	4	6	8	0	2	4	6	8
3	0	3	6	9	2	5	8	1	4	7
4	0	4	8	2	6	0	4	8	2	6
5	0	5	0	5	0	5	0	5	0	5
6	0	6	2	8	4	0	6	2	8	4
7	0	7	4	1	8	0	2	9	6	3
8	0	8	6	4	2	0	8	6	4	2
9	0	9	8	7	6	5	4	3	2	1

Multiplication Table in Z_{10}

شکل ۲-۱۶: جدول های جمع و ضرب برای Z_{10}

مجموعه های متفاوت برای جمع و ضرب

در رمزنگاری، بیشتر با وارون ها سروکار داریم. اگر عملگر مورد استفاده در الگوریتم رمزنویسی / رمزگشایی جمع باشد، فرستنده از یک عدد صحیح (به عنوان کلید رمزنویسی) استفاده می کند و گیرنده

از وارون آن عدد (به عنوان کلید رمزگشایی) استفاده می‌کند. Z_n می‌تواند مجموعه‌ی کلیدهای ممکن باشد، زیرا هر عدد صحیح در این مجموعه دارای وارون نسبت به جمع است؛ از طرف دیگر، اگر عملگر مورد استفاده در الگوریتم رمزنویسی / رمزگشایی ضرب باشد، Z_n دیگر نمی‌تواند مجموعه‌ای از کلیدهای ممکن باشد زیرا فقط برخی از اجزای این مجموعه دارای وارون نسبت به ضرب هستند، پس به مجموعه‌ی دیگری نیاز داریم. این مجموعه‌ی جدید زیرمجموعه Z_n بوده و تنها شامل اعداد صحیحی در Z_n می‌باشد که یک وارون ضربی منحصر به فرد داشته باشد. این مجموعه جدید را Z_n^* می‌نامند. شکل ۱۷-۲ چند نمونه از این دو مجموعه را نشان می‌دهد. توجه داشته باشید که Z_n^* را می‌توان از جدول ضرب نظیر آنچه که در شکل ۱۶-۲ نشان داده شده است به دست آورد.

هر عضو از مجموعه Z_n ، یک وارون نسبت به جمع دارد، اما تنها تعدادی از اعضا دارای وارون نسبت به ضرب هستند. هر عضو از مجموعه Z_n^* ، یک وارون نسبت به ضرب دارد، اما فقط برخی از آن‌ها دارای وارون نسبت به جمع می‌باشند.

پس زمانی که وارون نسبت به جمع نیاز داریم، باید از مجموعه Z_n استفاده کنیم و وقتی وارون نسبت به ضرب نیاز است باید از مجموعه Z_n^* استفاده کنیم.

$Z_6 = \{0, 1, 2, 3, 4, 5\}$	$Z_6^* = \{1, 5\}$
$Z_7 = \{0, 1, 2, 3, 4, 5, 6\}$	$Z_7^* = \{1, 2, 3, 4, 5, 6\}$
$Z_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$	$Z_{10}^* = \{1, 3, 7, 9\}$

شکل ۱۷-۲: برخی از مجموعه‌های Z_n و Z_n^*

مجموعه‌های دیگر

غالباً در رمزنگاری از دو مجموعه‌ی دیگر نیز استفاده می‌کند: Z_p ، Z_p^* . پیمانه‌ی این دو مجموعه، یک عدد اول است. در فصول بعدی در مورد اعداد اول بحث خواهیم کرد، در اینجا همین قدر کافی است که بدانیم عدد اول فقط دو مقسوم‌علیه دارد؛ عدد صحیح یک و خود عدد.

مجموعه Z_p مانند Z_n است فقط n عدد اول است. Z_p تمام اعداد صحیح از ۰ تا $p-1$ را شامل می‌شود. هر عضو Z_p یک وارون نسبت به جمع دارد و هر عضو، به غیر از صفر، یک وارون نسبت به ضرب دارد.

مجموعه Z_p^* همان مجموعه Z_n^* است، فقط n عدد اول است. Z_p^* تمام اعداد صحیح از 1 تا $p-1$ را شامل می‌شود. هر عضو از Z_p^* یک وارون نسبت به جمع و یک وارون نسبت به ضرب دارد. پس وقتی مجموعه‌ای نیاز داشته باشیم که هم وارون نسبت به جمع و هم وارون نسبت به ضرب را پشتیبانی کند، Z_p^* گزینه بسیار خوبی است.

اعضای این دو مجموعه وقتی که $p=13$ باشد به شکل زیر است.

$$Z_{13} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

$$Z_{13}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

۲-۳ ماتریس‌ها

در رمزنگاری، باید از ماتریس‌ها نیز پشتیبانی کنیم. اگرچه ماتریس‌ها به شاخه خاصی از جبر که به آن جبر خطی می‌گویند تعلق دارد؛ اما در اینجا خلاصه‌ای از مبحث ماتریس‌ها که جهت آمادگی شما برای مطالعه رمزنگاری لازم است را بررسی می‌کنیم. کسانی که با این مبحث آشنا هستند می‌توانند از قسمتی یا کل این بخش چشم‌پوشی کنند. این بخش، با تعاریف آغاز می‌شود و سپس نشان می‌دهیم که چگونه از ماتریس‌ها در حساب پیمانه‌ای استفاده می‌شود.

تعاریف

ماتریس آرایه‌ای مستطیل شکل از عناصر در اندازه‌ی $L \times m$ است، که L تعداد سطرها و m تعداد ستون‌ها را نشان می‌دهد. معمولاً ماتریس را با حروف بزرگ و توپر، مانند **A** نشان می‌دهند. عنصر a_{ij} در i امین سطر و j امین ستون ماتریس قرار دارد. اگر چه این عناصر می‌توانند مجموعه‌ای از اعداد باشند، اما فقط ماتریس‌هایی که عناصرشان در Z است را بررسی می‌کنیم. شکل ۲-۱۸ این ماتریس را نشان می‌دهد.

$$\text{Matrix A: } \begin{matrix} & \text{m columns} \\ \begin{matrix} \text{rows} \\ \vdots \\ \vdots \end{matrix} & \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \vdots & & \vdots \\ a_{l1} & a_{l2} & \dots & a_{lm} \end{bmatrix} \end{matrix}$$

شکل ۲-۱۸: ماتریسی به اندازه $l \times m$

اگر ماتریسی فقط یک سطر داشته باشد ($l=1$) به آن ماتریس سطری می‌گویند و اگر فقط یک ستون داشته باشد ($m=1$) به آن ماتریس ستونی می‌گویند. در ماتریس مربع، تعداد سطرها و ستونها برابرند ($l=m$) و عناصر $a_{11}, a_{22}, \dots, a_{mm}$ قطر اصلی را تشکیل می‌دهند. یک ماتریس صفر که آن را با 0 نشان می‌دهند، ماتریسی است که عناصر تمام سطرها و ستون‌های آن صفر است. ماتریس یک که با I نشان داده می‌شود، ماتریس مربعی است که عناصر قطر اصلیش یک و بقیه عناصرش صفر هستند. شکل ۱۹-۲ ماتریس‌هایی را با عناصری از Z نشان می‌دهد.

$$\begin{array}{ccccc} \begin{bmatrix} 2 & 1 & 5 & 11 \end{bmatrix} & \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} & \begin{bmatrix} 23 & 14 & 56 \\ 12 & 21 & 18 \\ 10 & 8 & 31 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ \text{Row matrix} & \text{Column matrix} & \text{Square matrix} & 0 & I \end{array}$$

شکل ۱۹-۲: نمونه‌هایی از ماتریس

اعمال و روابط

در جبر خطی، رابطه تساوی و چهار عمل جمع، تفریق، ضرب و ضرب اسکالر برای ماتریس‌ها تعریف شده است.

تساوی

دو ماتریس برابرند، به شرطی که تعداد سطرها و ستون‌ها و عناصر متناظر هر سطر و ستون برابر باشند؛ به عبارت دیگر $A=B$ اگر شرط $a_{ij}=b_{ij}$ را برای تمام i ها و j ها داشته باشیم.

جمع و تفریق

اگر دو ماتریس دارای سطرها و ستون‌های هم اندازه باشند، می‌توان آن‌ها را با هم جمع کرد. جمع را به این شکل نشان می‌دهیم $C=A+B$. در این حالت، ماتریس به دست آمده C هم دارای همان تعداد سطر و ستون A, B است. هر عنصر از C مجموع دو عنصر متناظر A, B است: $C_{ij}=A_{ij}+B_{ij}$. تفریق هم همین گونه است، به جز این که هر عضو از B از عنصر متناظرش در A کم می‌شود: $D_{ij}=A_{ij}-B_{ij}$.

مثال ۲-۲۸

شکل ۲-۲۰ نمونه‌ای از جمع و تفریق را نشان می‌دهد.

$$\begin{bmatrix} 12 & 4 & 4 \\ 11 & 12 & 30 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} + \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix} \quad \begin{bmatrix} -2 & 0 & -2 \\ -5 & -8 & 10 \end{bmatrix} = \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 10 \end{bmatrix} - \begin{bmatrix} 7 & 2 & 3 \\ 8 & 10 & 20 \end{bmatrix}$$

$$C = A + B \quad D = A - B$$

شکل ۲۰-۲ جمع و تفریق ماتریس‌ها

ضرب

می‌توانیم دو ماتریس با اندازه‌های مختلف را در هم ضرب کنیم به شرطی که تعداد ستون‌های ماتریس نخست با تعداد سطرهای ماتریس دوم برابر باشد. اگر A ماتریس $L \times m$ و B ماتریس $m \times p$ باشد، حاصل ضرب آن‌ها ماتریس C با اندازه $L \times p$ می‌شود. اگر هر عنصر از ماتریس A را a_{ij} و هر عنصر از ماتریس B را b_{jk} بنامیم، در نتیجه هر عنصر از ماتریس C ، c_{ik} خواهد بود و می‌توان به شکل زیر آن را محاسبه نمود.

$$C_{ik} = \sum a_{ij} \times b_{jk} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + \dots + a_{im} \times b_{mj}$$

مثال ۲۹-۲

شکل ۲۱-۲ حاصل یک ماتریس سطری (1×3) در یک ماتریس ستونی (3×1) را نشان می‌دهد. نتیجه، ماتریسی با اندازه 1×1 است.

$$\begin{array}{ccc} C & A & B \\ \left[\begin{matrix} 53 \end{matrix} \right] & = \left[\begin{matrix} 5 & 2 & 1 \end{matrix} \right] \times \left[\begin{matrix} 7 \\ 8 \\ 2 \end{matrix} \right] \end{array} \quad \text{In which: } \boxed{53 = 5 \times 7 + 2 \times 8 + 1 \times 2}$$

شکل ۲۱-۲: ضرب یک ماتریس سطری در یک ماتریس ستونی

مثال ۳۰-۲

شکل ۲۲-۲ حاصل یک ماتریس 2×3 در یک ماتریس 3×4 را نشان می‌دهد. نتیجه یک ماتریس 2×4 است.

$$\begin{array}{ccc} C & A & B \\ \left[\begin{matrix} 52 & 18 & 14 & 9 \\ 41 & 21 & 22 & 7 \end{matrix} \right] & = \left[\begin{matrix} 5 & 2 & 1 \\ 3 & 2 & 4 \end{matrix} \right] \times \left[\begin{matrix} 7 & 3 & 2 & 1 \\ 8 & 0 & 0 & 2 \\ 1 & 3 & 4 & 0 \end{matrix} \right] \end{array}$$

شکل ۲۲-۲: ضرب یک ماتریس 2×3 در یک ماتریس 3×4

ضرب اسکالر

ماتریس را می‌توان در یک عدد (که به آن اسکالر می‌گویند) ضرب کرد. اگر $C = XA$ باشد، حاصل ماتریسی است با اندازه $L \times m$ که $C_{ij} = x \times a_{ij}$ است.

$$\begin{matrix} & \mathbf{B} & & \mathbf{A} \\ \begin{bmatrix} 15 & 6 & 3 \\ 9 & 6 & 12 \end{bmatrix} & = 3 \times & \begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 4 \end{bmatrix} \end{matrix}$$

شکل ۲-۲۳: ضرب اسکالر

مثال ۲-۳۱: شکل ۲-۲۳ نمونه‌ای از ضرب اسکالر را نشان می‌دهد.

دترمینان

دترمینان ماتریس مربع A با اندازه $m \times m$ که به صورت $\det(A)$ نشان داده می‌شود، عدد اسکالری است که به صورت بازگشتی و به شکل زیر محاسبه می‌شود:

(۱) اگر $m=1$ باشد پس $\det(A) = a_{11}$.

(۲) اگر $m > 1$ باشد $\det(A) = \sum_{i=1}^m (-1)^{i+j} \times a_{ij} \times \det(A_{ij})$

به گونه‌ای که A_{ij} ماتریسی است که با حذف سطر i و ستون j از ماتریس A به دست آمده است.

دترمینان فقط برای ماتریس مربع قابل محاسبه است.

مثال ۲-۳۲

شکل ۲-۲۴ نشان می‌دهد که چگونه با استفاده از تعریف بازگشتی بالا می‌توانیم دترمینان ماتریس به اندازه 2×2 را بر مبنای دترمینان ماتریس 1×1 محاسبه کنیم. مثال زیر نشان می‌دهد، وقتی که m برابر ۱ یا ۲ باشد؛ یافتن دترمینان یک ماتریس بسیار ساده است.

$$\det \begin{bmatrix} 5 & 2 \\ 3 & 4 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det[4] + (-1)^{1+2} \times 2 \times \det[3] \rightarrow 5 \times 4 - 2 \times 3 = 14$$

$$\text{or } \det \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = a_{11} \times a_{22} - a_{12} \times a_{21}$$

شکل ۲-۲۴: محاسبه‌ی دترمینان ماتریس 2×2

مثال ۲-۳۳

شکل ۲-۲۵ محاسبه‌ی دترمینان یک ماتریس 3×3 را نشان می‌دهد.

$$\det \begin{bmatrix} 5 & 2 & 1 \\ 3 & 0 & -4 \\ 2 & 1 & 6 \end{bmatrix} = (-1)^{1+1} \times 5 \times \det \begin{bmatrix} 0 & -4 \\ 1 & 6 \end{bmatrix} + (-1)^{1+2} \times 2 \times \det \begin{bmatrix} 3 & -4 \\ 2 & 6 \end{bmatrix} + (-1)^{1+3} \times 1 \times \det \begin{bmatrix} 3 & 0 \\ 2 & 1 \end{bmatrix}$$

$$= (+1) \times 5 \times (+4) + (-1) \times 2 \times (24) + (+1) \times 1 \times (3) = -25$$

شکل ۲-۲۵ محاسبه‌ی دترمینان یک ماتریس ۳×۳

وارون ماتریس‌ها

ماتریس‌ها، هم وارون نسبت به جمع و هم وارون نسبت به ضرب دارند.

وارون نسبت به جمع

وارون نسبت به جمع ماتریس A ماتریس دیگری مانند B است، به طوری که $A+B=0$ باشد؛ به عبارت دیگر برای تمام مقادیر i و j داریم $B_{ij} = -A_{ij}$. معمولاً وارون نسبت به جمع A را با $-A$ نشان می‌دهیم.

وارون نسبت به ضرب

وارون نسبت به ضرب فقط برای ماتریس‌های مربع تعریف شده است. وارون ضربی ماتریس مربع A ماتریس مربع B است، به طوری که $A \times B = B \times A = I$ باشد. معمولاً وارون ضربی A را با A^{-1} نشان می‌دهیم. وارون نسبت به ضرب تنها زمانی وجود دارد که $\det(A)$ وارون ضربی در مجموعه مربوطه داشته باشد. از آنجا که هیچ عدد صحیحی وارون ضربی در Z ندارد، در نتیجه ماتریس‌های موجود در Z همگی فاقد وارون ضربی هستند؛ با این حال، ماتریس‌های دارای عناصر مجموعه‌ی اعداد حقیقی^۱ وارون دارند به شرطی که $\det(A) \neq 0$ باشد.

وارون نسبت به عمل ضرب فقط برای ماتریس مربع تعریف شده است.

ماتریس باقیمانده‌ها^۲

در رمزنگاری بیشتر از ماتریس باقیمانده‌ها استفاده می‌شود؛ ماتریس‌هایی که تمام عناصر آن متعلق به Z_n می‌باشد. تمام اعمالی که روی ماتریس باقیمانده‌ها انجام می‌شود، دقیقاً مانند اعمالی است که بر روی ماتریس‌های عدد صحیح انجام می‌شود به جز اینکه تمام اعمال در حساب پیمانه‌ای انجام می‌شود. یک نتیجه جالب این است که ماتریس باقیمانده، دارای وارون ضرب است به شرطی که دترمینان ماتریس

^۱ Real

^۲ Residue Matrices

دارای وارون نسبت به ضرب در Z_n باشد؛ به عبارت دیگر، یک ماتریس باقیمانده دارای وارون ضربی است، اگر $\gcd(\det(A), n) = 1$ باشد.

مثال ۲-۳۴

شکل ۲-۲۶ ماتریس باقیمانده A در Z_{26} و وارون ضربی (A^{-1}) را نشان می‌دهد. $\det(A) = 21$ است پس دارای وارون ضربی ۵ در Z_{26} می‌باشد. توجه داشته باشید، وقتی که دو ماتریس را در هم ضرب می‌کنیم، حاصل، ماتریس یکه نسبت به ضرب در Z_{26} می‌باشد.

$$A = \begin{bmatrix} 3 & 5 & 7 & 2 \\ 1 & 4 & 7 & 2 \\ 6 & 3 & 9 & 17 \\ 13 & 5 & 4 & 16 \end{bmatrix} \quad A^{-1} = \begin{bmatrix} 15 & 21 & 0 & 15 \\ 23 & 9 & 0 & 22 \\ 15 & 16 & 18 & 3 \\ 24 & 7 & 15 & 3 \end{bmatrix}$$

$$\det(A) = 21 \quad \det(A^{-1}) = 5$$

۲-۲۶: ماتریس باقیمانده و وارون ضربی آن

تجانس

دو ماتریس به پیمانه n متجانس‌اند و به صورت $A \equiv B \pmod{n}$ نوشته می‌شوند، به شرطی که تعداد سطرها و ستون‌های آن‌ها یکسان باشد و تمام عناصر متناظر به پیمانه n متجانس باشند؛ به عبارت دیگر $A \equiv B \pmod{n}$ اگر برای تمام i ها و j ها $a_{ij} \equiv b_{ij} \pmod{n}$ باشد.

۲-۴- تجانس خطی^۱

معمولاً رمزنگاری، شامل حل یک معادله یا مجموعه‌ای از معادلات یک یا چند متغیره با ضرایبی از مجموعه Z_n می‌باشد. در این بخش چگونگی حل معادلاتی را نشان می‌دهیم که توان تمام متغیرهای آن یک است (معادله‌ی خطی).

معادلات خطی تک متغیره^۲

اجازه دهید بررسی کنیم که چگونه معادلات تک متغیری یعنی معادلاتی به شکل $ax \equiv b \pmod{n}$ را می‌توانیم حل کنیم. معادلاتی از این نوع ممکن است یا راه‌حل نداشته باشند یا تعداد محدودی راه‌حل

^۱ Linear Congruence

^۲ Single-Variable Linear Equations

داشته باشند. فرض کنید $\gcd(a, n) = d$ باشد. اگر $d \nmid b$ باشد پس راه‌حلی وجود ندارد ولی اگر $d \mid b$ یک راه‌حل داریم.

اگر $d \mid b$ از راهکار زیر برای یافتن راه‌حل‌های مربوطه استفاده می‌کنیم.

(۱) با تقسیم هر دو طرف معادله و خود پیمانه بر d معادله را کاهش می‌دهیم.

(۲) برای یافتن راه‌حل ویژه x_0 ، هر دو طرف معادله کاهش یافته را در وارون ضربی، ضرب می‌کنیم.

(۳) راه‌حل‌های کلی عبارت‌اند از: $x = x_0 + k(n/d)$ for $k = 0, 1, \dots, (d-1)$

مثال ۲-۳۵

معادله $10x \equiv 2 \pmod{15}$ را حل کنید:

حل:

نخست $\gcd(15, 10) = 5$ را به دست می‌آوریم. از آنجا که ۵ بر ۲ قابل تقسیم نیست، این معادله راه‌حل ندارد.

مثال ۲-۳۶

پاسخ معادله $14x \equiv 12 \pmod{18}$ را به دست آورید.

حل:

توجه داشته باشید که $\gcd(18, 14) = 2$ است. از آنجا که ۱۲ بر ۲ بخش‌پذیر است، دقیقاً دو راه‌حل داریم، اما ابتدا معادله را کاهش می‌دهیم.

$$14x \equiv 12 \pmod{18} \rightarrow 7x \equiv 6 \pmod{9} \rightarrow x \equiv 6(7^{-1}) \pmod{9}$$

$$x_0 = (6 \times 7^{-1}) \pmod{9} = (6 \times 4) \pmod{9} = 6$$

$$x_1 = x_0 + 1 \times (18/2) = 15$$

هر دو پاسخ ۶ و ۱۵ در رابطه‌ی تجانس صدق می‌کند، زیرا $14 \times 16 \pmod{18} = 12$ و

$$(14 \times 15) \pmod{18} = 12$$

مثال ۲-۳۷

معادله $3x + 4 \equiv 6 \pmod{13}$ را حل کنید.

حل

نخست معادله را به شکل $ax \equiv b \pmod{n}$ تغییر می‌دهیم. ۴- (وارون جمعی ۴) را به هر دو طرف اضافه می‌کنیم، رابطه $3x \equiv 2 \pmod{13}$ به دست می‌آید. چون $\gcd(3, 13) = 1$ است، معادله فقط یک راه‌حل دارد و آن هم $5 = 18 \pmod{13} = (2 \times 3) \pmod{13} = x_0$ است. می‌بینیم که پاسخ، در معادله اصلی $3 \times 5 + 4 \equiv 6 \pmod{13}$ صدق می‌کند.

مجموعه معادلات خطی^۱

می‌توانیم مجموعه معادلات خطی با پیمانه یکسان را حل کنیم به شرطی که ماتریس شکل گرفته از ضرایب متغیرها، وارون‌پذیر باشد. برای این کار سه ماتریس می‌سازیم؛ نخستین ماتریس، ماتریس مربع متشکل از ضرایب متغیرها است. دومین ماتریس، ماتریس ستونی ساخته شده از خود متغیرهاست و سومین ماتریس، ماتریس ستونی ساخته شده از مقادیر سمت راست عملگر تجانس است. می‌توانیم مجموعه معادلات را به شکل ضرب ماتریس هم تفسیر کنیم. اگر هر دو طرف تجانس را در وارون ضربی اولین ماتریس ضرب کنیم، نتیجه‌ی آن، باقی ماندن ماتریس متغیرها در سمت چپ است و این بدین معناست که مسئله را می‌توان به وسیله ضرب ماتریس، همان‌گونه که در شکل ۲-۲۷ نشان داده شده است حل کرد.

$$\begin{array}{ccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \dots & + & a_{1n}x_n & \equiv & b_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \dots & + & a_{2n}x_n & \equiv & b_2 \\ \vdots & & \vdots & & & & \vdots & & \vdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \dots & + & a_{nn}x_n & \equiv & b_n \end{array}$$

a. Equations

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \equiv \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \equiv \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}^{-1} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

b. Interpretation

c. Solution

شکل ۲-۲۷: مجموعه معادلات خطی

مثال ۲-۳۸

مجموعه معادلات سه مجهولی زیر را حل کنید.

$$3x + 5y + 7z \equiv 3 \pmod{16}$$

=====

^۱ Set of Linear Equations

$$x + 4y + 13z \equiv 5 \pmod{16}$$

$$2x + 7y + 3z \equiv 4 \pmod{16}$$

حل:

در اینجا z, y, x نقش x_3, x_2, x_1 را بازی می‌کنند. ماتریس تشکیل شده از سه معادله‌ی سه مجهولی، وارون‌پذیر است. وارون ضربی ماتریس را به دست آورده و آن را در ماتریس ستونی شکل گرفته از ۳، ۴ و ۵ (حاصل سمت راست رابطه) ضرب می‌کنیم. نتیجه $x \equiv 15 \pmod{16}$ ، $y \equiv 4 \pmod{16}$ و $z \equiv 14 \pmod{16}$ است. می‌توانیم با جاگذاری این مقادیر در معادله، پاسخ را به دست آوریم.

۲-۵- وب سایت‌ها پیشنهادی

سایت‌های زیر اطلاعات بیشتری در زمینه مباحث بیان شده در این فصل در اختیار شما قرار می‌دهند.

http://en.wikipedia.org/wiki/Euclidean_algorithm
http://en.wikipedia.org/wiki/Multiplicative_inverse
http://en.wikipedia.org/wiki/Additive_inverse

۶-۲ چکیده

* مجموعه اعداد صحیح که با Z نشان داده می‌شود، شامل تمام اعداد صحیح از مثبت بینهایت تا منفی بینهایت می‌باشد. سه عمل دودویی پرکاربردی که برای اعداد صحیح تعریف شده‌اند عبارت‌اند از: جمع، تفریق و ضرب. تقسیم در این مقوله جای نمی‌گیرد زیرا حاصل عمل تقسیم به جای یک خروجی، دو خروجی تولید می‌کند.

* در حساب عدد صحیح اگر a را بر n تقسیم کنیم، می‌توانیم q و r را به دست آوریم. رابطه‌ی بین این چهار عدد صحیح را می‌توان به شکل $a = q \times n + r$ نشان داد. $a|b$ به شرطی که $a = q \times n$ باشد. در این فصل به چهار ویژگی بخش‌پذیری اشاره کردیم.

* دو عدد صحیح مثبت می‌توانند بیش از یک مقسوم‌علیه مشترک داشته باشند؛ اما معمولاً بزرگ‌ترین مقسوم‌علیه مشترک برای ما اهمیت دارد. الگوریتم اقلیدسی، روش موثر و سیستماتیکی برای محاسبه‌ی بزرگ‌ترین مقسوم‌علیه مشترک دو عدد صحیح را در اختیار ما قرار می‌دهد.

* الگوریتم اقلیدسی بسط‌یافته می‌تواند $\gcd(a, b)$ و درعین حال مقادیر t, s که در معادله‌ی $as + bt = \gcd(a, b)$ صدق می‌کنند را محاسبه نماید.

* معادله‌ی خطی سیال دو مجهوله به شکل $ax + by = c$ است. این معادله، هم پاسخ ویژه و هم پاسخ کلی دارد.

* در حساب پیمانه‌ای، فقط باقیمانده‌ها برای ما اهمیت دارند، زیرا می‌خواهیم تنها مقدار r را وقتی که a را بر n تقسیم می‌کنیم بدانیم. از یک عملگر جدید موسوم به عملگر پیمانه‌ای استفاده می‌کنیم؛ بنابراین $a \bmod n = r$ است. n را پیمانه و r را باقیمانده‌ی عملگر می‌نامیم.

* نتیجه‌ی عمل به پیمانه n همیشه یک عدد صحیح بین 0 و $n-1$ است. پس عملگر پیمانه‌ای، مجموعه‌ای را ایجاد می‌کند که در حساب پیمانه‌ای از آن به عنوان مجموعه‌ی کوچک‌ترین باقیمانده به پیمانه n یا Z_n یاد می‌شود.

* نگاشت مجموعه‌ی Z به مجموعه‌ی Z_n ، یک به یک نیست. اعضای نامحدودی از Z را می‌توان به یک عضو Z_n متجانس به پیمانه n نگاشت نمود. برای نشان دادن این که دو عدد صحیح متجانس هستند، از عملگر تجانس \equiv استفاده می‌کنیم.

- * کلاس باقیمانده $[a]$ مجموعه‌ای از اعداد صحیح متجانس به پیمانه n است. این کلاس از مجموعه‌ی کل اعداد صحیح x تشکیل شده به شرطی که $x = a \bmod n$ باشد.
- * سه عملگر دودویی (جمع، تفریق و ضرب) که برای مجموعه Z تعریف شده است را می‌توان برای مجموعه Z_n نیز تعریف کرد. شاید لازم باشد با استفاده از عملگر پیمانه، نتیجه را به Z_n نگاشت کرد.
- * در Z_n دو عدد a, b وارون جمعی یکدیگرند به شرطی که $a+b \equiv 0 \pmod{n}$ باشد و وارون ضربی یکدیگرند، به شرطی که $bx \equiv 1 \pmod{n}$ باشد. عدد صحیح a دارای وارون ضربی در Z_n است، اگر و تنها اگر $\gcd(n, b) = 1$ باشد (a, n نسبت به هم اول باشند).
- * الگوریتم اقلیدسی بسط‌یافته، وارون‌های ضربی b در Z_n را زمانی پیدا می‌کند که b, n معلوم و $\gcd(n, b) = 1$ باشد.
- * ماتریس، آرایه‌ای مستطیل شکل از $m \times n$ عنصر است که i تعداد سطرها و m تعداد ستون‌هاست. ماتریس را با یک حرف بزرگ و توپر نظیر A نشان می‌دهیم. عنصر a_{ij} در سطر i و ستون j قرار دارد.
- * دو ماتریس برابرند اگر تعداد سطر و ستون برابر و عناصر نظیر هم برابر باشد.
- * جمع و تفریق تنها روی ماتریس‌های هم اندازه انجام می‌شود. می‌توانیم دو ماتریس با اندازه‌های مختلف را در هم ضرب کنیم، به شرطی که تعداد ستون‌های ماتریس اول با تعداد سطرهاى دوم ماتریس برابر باشد.
- * در ماتریس‌های باقیمانده، تمام عناصر در مجموعه Z_n هستند. تمام اعمال روی ماتریس‌های باقیمانده، در چارچوب حساب پیمانه‌ای انجام می‌شود. ماتریس باقیمانده وارون دارد به شرطی که ماتریس وارون دترمینان داشته باشد.
- * معادله‌ای به شکل $ax \equiv b \pmod{n}$ ممکن است راه‌حلی نداشته باشد یا فقط تعداد معدودی راه‌حل داشته باشد. اگر $\gcd(a, n) \mid b$ برقرار باشد فقط تعداد معدودی راه‌حل وجود دارد.
- * مجموعه‌ای از معادلات خطی با پیمانه یکسان را می‌توان حل کرد، به شرطی که ماتریس متشکل از ضرایب و متغیرها، وارون داشته باشد.

۷-۲ مجموعه تمرین‌ها

پرسش‌های دوره‌ای

(۱) تفاوت بین Z, Z_n را بیان کنید. کدام مجموعه می‌تواند اعداد صحیح منفی داشته باشد؟

چگونه می‌توانیم یک عدد صحیح در Z را به یک عدد صحیح در Z_n نگاشت کنیم؟

(۲) چهار ویژگی بخش‌پذیری که در این فصل مورد بحث قرار گرفت را نام ببرید. یک عدد

صحیح که فقط یک مقسوم‌علیه داشته باشد را نام ببرید. یک عدد صحیح با دو مقسوم‌علیه را

نام ببرید. یک عدد صحیح با بیش از دو مقسوم‌علیه را نام ببرید.

(۳) بزرگ‌ترین مقسوم‌علیه مشترک دو عدد صحیح را تعریف کنید. با کدام الگوریتم می‌توان

بزرگ‌ترین مقسوم‌علیه مشترک را پیدا کرد؟

(۴) معادله خطی سیال دو مجهوله چیست؟ چنین معادله‌ای می‌تواند چند راه‌حل داشته باشد؟

راه‌حل‌ها را چگونه می‌توان پیدا کرد؟

(۵) عملگر پیمانه‌ای چیست و کاربرد آن کدام است؟ تمام ویژگی‌هایی را که در این فصل برای

عملگر پیمانه‌ای برشمردیم نام ببرید.

(۶) تجانس را تعریف کرده و آن را با تساوی مقایسه کنید.

(۷) دسته باقیمانده و کوچک‌ترین باقیمانده را تعریف کنید.

(۸) تفاوت بین مجموعه Z_n و مجموعه Z_n^* چیست؟ در کدام مجموعه، هر عنصر یک وارون

نسبت به جمع دارد؟ در کدام مجموعه، هر عنصر یک وارون ضربی دارد؟ برای یافتن وارون

ضربی یک عدد صحیح در Z_n چه الگوریتمی مورد استفاده قرار می‌گیرد؟

(۹) ماتریس را تعریف کنید. ماتریس سطری چیست؟ ماتریس ستونی چیست؟ ماتریس مربع

چیست؟ چه نوع ماتریسی دترمینان دارد؟ چه نوع ماتریسی می‌تواند وارون داشته باشد؟

(۱۰) تجانس خطی را تعریف کنید. برای یافتن راه‌حل معادله‌ای از نوع $ax \equiv b \pmod{n}$ چه

نوع الگوریتمی به کار می‌رود؟ چگونه می‌توان مجموعه معادلات خطی را حل کرد؟

تمرین‌ها

(۱۱) کدام یک از روابط زیر صحیح و کدام یک نادرست می‌باشد؟

۵|۲۶ ۳|۱۲۳ ۲۷|۱۲۷ ۱۵|۲۱ ۲۳|۹۶ ۵|۸

۱۲) با استفاده از الگوریتم اقلیدسی بزرگ‌ترین مقسوم‌علیه مشترک زوج عدد صحیح‌های زیر را به دست آورید.

- a) (۸۸، ۲۲۰)
- b) (۲۰۰، ۴۲)
- c) (۲۴، ۳۲۰)
- d) (۴۰۱، ۷۰۰)

۱۳) پاسخ پرسش‌های زیر را پیدا کنید.

a) فرض کنید $\gcd(a, b) = 24$ باشد، $\gcd(a, b, 16)$ را به دست آورید.

b) فرض کنید $\gcd(a, b, c) = 24$ باشد، $\gcd(a, b, c, 16)$ را به دست آورید.

c) $\gcd(200, 180, 450)$ را به دست آورید.

d) $\gcd(200, 180, 450, 610)$ را به دست آورید.

۱۴) فرض کنید n عدد صحیح غیر منفی باشد.

a) $\gcd(2n+1, n)$ را به دست آورید.

b) با استفاده از نتیجه قسمت a، $\gcd(201, 301)$ ، $\gcd(40, 81)$ ، $\gcd(250, 501)$ را به دست آورید.

۱۵) فرض کنید n یک عدد صحیح غیر منفی است.

a) $\gcd(3n+2, 1n+1)$ را به دست آورید.

b) با استفاده از نتیجه قسمت قبل $\gcd(201, 301)$ و $\gcd(81, 121)$ را به دست آورید.

۱۶) با استفاده از الگوریتم اقلیدسی بسط‌یافته، بزرگ‌ترین مقسوم‌علیه مشترک زوج‌های زیر و مقادیر t, s را به دست آورید.

- a) (۴، ۷)
- b) (۲۹۱، ۴۲)
- c) (۸۴، ۳۲۰)
- d) (۴۰۰، ۶۰)

(۱۷) پاسخ پرسش‌های زیر را به دست آورید.

- a) $۲۲ \bmod ۷$
- b) $۱۴۰ \bmod ۱۰$
- c) $-۷۸ \bmod ۱۳$
- d) $۰ \bmod ۱۵$

(۱۸) با استفاده از مراحل کاهش، اعمال زیر را انجام دهید.

- a) $(۲۷۳+۱۴۷) \bmod ۱۰$
- b) $(۴۲۲۳+۱۷۳۲۳) \bmod ۱۰$
- c) $(۱۴۸+۱۴۴۲۳) \bmod ۱۲$
- d) $(۲۴۶۷+۴۶۱) \bmod ۱۲$

(۱۹) با استفاده از کاهش، اعمال زیر را انجام دهید.

- a) $(۱۲۵ \times ۴۵) \bmod ۱۰$
- b) $(۴۲۴ \times ۳۲) \bmod ۱۰$
- c) $(۱۴۴ \times ۳۴) \bmod ۱۲$
- d) $(۲۲۱ \times ۲۳) \bmod ۱۲$

(۲۰) از ویژگی‌های عملگر \bmod برای اثبات موارد زیر استفاده کنید.

(a) باقیمانده‌ی هر عدد صحیح، وقتی که تقسیم بر ۱۰ شود، سمت راست‌ترین رقم خود عدد است.

(b) باقیمانده‌ی هر عدد صحیح وقتی که بر ۱۰۰ تقسیم شود، عدد صحیحی است که از دو رقم سمت راست خود عدد تشکیل شده است.

(c) باقیمانده‌ی هر عدد صحیح وقتی که بر ۱۰۰۰ تقسیم شود، عدد صحیحی است از سه رقم سمت راست خود عدد تشکیل شده است.

(۲۱) در حساب به ما گفته‌اند که باقیمانده‌ی تقسیم یک عدد صحیح بر ۵ مانند باقیمانده‌ی تقسیم سمت راست‌ترین رقم آن عدد بر ۵ است. از ویژگی‌های عملگر \bmod برای اثبات این ادعا استفاده کنید.

(۲۲) در حساب به ما گفته‌اند که باقیمانده‌ی تقسیم یک عدد صحیح بر ۲ مانند باقیمانده‌ی تقسیم سمت راست‌ترین رقم آن عدد بر ۲ است. از ویژگی‌های عملگر mod برای اثبات این ادعا استفاده کنید.

(۲۳) در حساب به ما گفته‌اند که باقیمانده‌ی تقسیم یک عدد صحیح بر ۴ همانند باقیمانده‌ی تقسیم دو رقم سمت راست آن عدد بر ۴ است. با استفاده از ویژگی‌های عملگر mod این ادعا را ثابت کنید.

(۲۴) در حساب به ما گفته‌اند که باقیمانده‌ی تقسیم یک عدد صحیح بر ۸ همانند باقیمانده‌ی تقسیم سه رقم سمت راست آن عدد بر ۸ می‌باشد. با استفاده از ویژگی‌های عملگر mod این ادعا را ثابت کنید.

(۲۵) در حساب به ما گفته‌اند که باقیمانده‌ی تقسیم یک عدد صحیح بر ۹ همانند باقیمانده‌ی تقسیم مجموعه ارقام آن عدد بر ۹ است. به عبارت دیگر، باقیمانده‌ی تقسیم ۶۳۷۱ بر ۹ با باقیمانده‌ی تقسیم ۱۷ بر ۹ یکی است، چرا که $۱۷ = ۱ + ۷ + ۳ + ۶$. با استفاده از ویژگی‌های عملگر mod این ادعا را ثابت کنید.

(۲۶) موارد زیر، باقیمانده‌ی تقسیم توان‌های ۱۰ بر ۷ را نشان می‌دهد. می‌توانیم ثابت کنیم که این الگو را می‌توان برای توان‌های بالاتر نیز تکرار کرد.

$$\begin{aligned} 10^0 \bmod 7 &= 1 & 10^1 \bmod 7 &= 3 & 10^2 \bmod 7 &= 2 \\ 10^3 \bmod 7 &= -1 & 10^4 \bmod 7 &= -3 & 10^5 \bmod 7 &= -2 \end{aligned}$$

با استفاده از این اطلاعات، باقیمانده‌ی تقسیم یک عدد صحیح بر ۷ را پیدا کنید. درستی روش خود را با عدد ۶۳۱۴۵۳۶۷۲ آزمایش کنید.

(۲۷) موارد زیر باقیمانده‌ی تقسیم توان‌های ۱۰ بر عدد ۱۱ را نشان می‌دهد. می‌توانیم ثابت کنیم که این الگو را می‌توان برای توان‌های بالاتر نیز تکرار کرد.

$$10^0 \bmod 11 = 1 \quad 10^1 \bmod 11 = -1 \quad 10^2 \bmod 11 = 1 \quad 10^3 \bmod 11 = -1$$

با استفاده از این اطلاعات، باقیمانده‌ی تقسیم یک عدد صحیح بر ۱۱ را به دست آورید. درستی روش خود را با عدد ۶۳۱۴۵۳۶۷۲ آزمایش کنید.

(۲۸) محاسبات زیر باقیمانده‌ی تقسیم توان‌های ۱۰ بر ۱۳ را نشان می‌دهد. می‌توانیم ثابت کنیم که این الگو را می‌توان برای توان‌های بالاتر نیز تکرار کرد.

$$10^0 \bmod 13 = 1 \quad 10^1 \bmod 13 = -3 \quad 10^2 \bmod 13 = -4$$

$$10^3 \bmod 13 = -1 \quad 10^4 \bmod 13 = 3 \quad 10^5 \bmod 13 = 4$$

با استفاده از این اطلاعات، باقیمانده‌ی تقسیم یک عدد صحیح بر ۱۳ را پیدا کنید. روش خود را با ۶۳۱۴۵۳۶۷۲ امتحان کنید.

(۲۹) اجازه دهید برای حروف بزرگ، مقادیر عددی تعیین کنیم ($A=0, B=1, \dots, Z=25$)؛ اکنون می‌توانیم با استفاده از پیمانه ۲۶، حساب پیمانه‌ای را روی سیستم زیر انجام دهیم.

(a) حاصل $(A+N) \bmod 26$ در این سیستم چیست؟

(b) حاصل $(A+6) \bmod 26$ در این سیستم چیست؟

(c) حاصل $(Y+5) \bmod 26$ در این سیستم چیست؟

(d) حاصل $(C+10) \bmod 26$ در این سیستم چیست؟

(۳۰) تمام زوج‌های وارون‌پذیر در پیمانه ۲۰ را بنویسید.

(۳۱) تمام زوج‌های وارون‌پذیر نسبت به ضرب در پیمانه ۲۰ را بنویسید.

(۳۲) با استفاده از الگوریتم اقلیدسی بسط‌یافته، وارون ضربی هر یک از اعداد صحیح زیر در Z_{180} را به دست آورید.

a) ۳۸

b) ۷

c) ۱۳۲

d) ۲۴

(۳۳) پاسخ‌های ویژه و کلی برای معادلات خطی سیال زیر را به دست آورید.

a) $25x + 10y = 15$

b) $19x + 13y = 20$

c) $14x + 21y = 77$

d) $40x + 16y = 88$

(۳۴) نشان دهید که برای معادلات خطی سیال زیر، راه حل وجود ندارد.

a) $15x + 12y = 13$

b) $18x + 30y = 20$

c) $15x + 25y = 69$

d) $40x + 30y = 98$

(۳۵) یک دفتر پستی، تنها تمبر ۳۹ سستی و ۱۵ سستی می فروشد. مشتری برای ارسال بسته پستی

که باید ۲۰۷۰ دلار تمبر داشته باشد، چه تعداد تمبر باید بخرد. چند راه حل پیدا کنید.

(۳۶) تمام راه حل های هر یک از معادلات خطی زیر را پیدا کنید.

a) $3x \equiv 4 \pmod{5}$

b) $4x \equiv 4 \pmod{6}$

c) $9x \equiv 12 \pmod{7}$

d) $256x \equiv 422 \pmod{60}$

(۳۷) تمام راه حل های هر یک از معادلات خطی زیر را پیدا کنید.

a) $3x + 5 \equiv 4 \pmod{5}$

b) $4x + 6 \equiv 4 \pmod{6}$

c) $9x + 4 \equiv 12 \pmod{7}$

d) $132x + 42 \equiv 248 \pmod{50}$

(۳۸) با استفاده از ماتریس شکل ۲-۲۸، $(B \times A) \pmod{16}$ را به دست آورید.

$$\begin{array}{ccc} \begin{bmatrix} 3 & 7 & 10 \end{bmatrix} & \times & \begin{bmatrix} 2 \\ 4 \\ 12 \end{bmatrix} & \begin{bmatrix} 3 & 4 & 6 \\ 1 & 1 & 8 \\ 5 & 8 & 3 \end{bmatrix} & \times & \begin{bmatrix} 2 & 0 & 1 \\ 1 & 1 & 0 \\ 5 & 2 & 4 \end{bmatrix} \\ \mathbf{A} & & \mathbf{B} & \mathbf{A} & & \mathbf{B} \end{array}$$

شکل ۲-۲۸: ماتریس تمرین ۳۸

(۳۹) در شکل ۲-۲۹ وارون ضربی و دترمینان ماتریس باقیمانده حاصل از Z_{10} را به دست

آورید.

$$\begin{array}{ccc} \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix} & \begin{bmatrix} 4 & 2 \\ 1 & 1 \end{bmatrix} & \begin{bmatrix} 3 & 4 & 6 \\ 1 & 1 & 8 \\ 5 & 8 & 3 \end{bmatrix} \\ \mathbf{A} & \mathbf{B} & \mathbf{C} \end{array}$$

شکل ۲-۲۹: ماتریس تمرین ۳۹

۴۰) تمام راه‌حل‌های مجموعه‌های معادلات خطی زیر را به دست آورید.

a) $3x + 5y = 4 \pmod{5}$

$$2x + y = 3 \pmod{5}$$

b) $3x + 2y = 5 \pmod{7}$

$$2x + 6y = 4 \pmod{7}$$

c) $7x + 3y = 3 \pmod{7}$

$$4x + 2y = 5 \pmod{7}$$

d) $2x + 3y = 5 \pmod{8}$

$$x + 6y = 3 \pmod{8}$$

فصل ۳

رمزنگاری کلید متقارن سستی

چشم انداز

این فصل، نگاهی گذرا به رمزنگاری کلید متقارن که در گذشته به کار می‌رفته دارد و با توضیح اصول آماده می‌سازد. این فصل اهداف زیر را دنبال می‌کند:

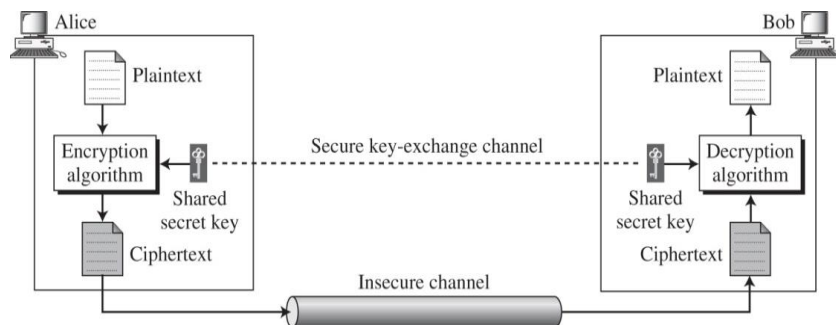
- ✱ تعریف اصطلاحات و مفاهیم رمز کلید متقارن.
- ✱ بررسی دو گونه رمز سستی: رمزنگاری جایگزین و رمزهای انتقال.
- ✱ توصیف موضوع‌های مرتبط با رمزگشایی که هدفش شکستن رمزهای کلید متقارن است.
- ✱ معرفی مفاهیم رمزنگاری جریان داده و رمزنگاری بلوکی.
- ✱ بررسی برخی از رمزنگاری‌های رایجی که در گذشته مورد استفاده قرار می‌گرفت، همچون ماشین انیگما.

در ادامه با استفاده از مثال‌هایی از رمزنویسی، ایده‌ی کلی که زمینه‌ساز ابداع رمزنگاری کلید متقارن بود، معرفی خواهد شد. اصطلاحات و تعاریف ارائه شده در این فصل، در فصول بعدی در زمینه رمزنگاری کلید متقارن مورد استفاده قرار می‌گیرد. پس از آن به بحث در مورد رمزنگاری کلید متقارن سستی می‌پردازیم. امروزه این رمزنگاری دیگر به کار نمی‌رود، اما بنا بر دلایل زیر آن‌ها را مطالعه می‌کنیم: نخست، این رمزها، ساده‌تر از رمزهای مدرن می‌باشند و درک آن‌ها آسان‌تر است. دوم شالوده اصلی رمزنویسی و رمزگشایی را نشان می‌دهند. از این شالوده می‌توان برای درک بهتر رمزنگاری مدرن استفاده

کرد. سوم، دلیل منطقی استفاده از رمزهای مدرن را هویدا می‌کنند، زیرا رمزهای سنتی را می‌توان با استفاده از رایانه به سادگی مورد حمله قرار داد. رمزهایی که در دوران گذشته مطمئن و خدشه‌ناپذیر بودند، امروزه و در عصر رایانه دیگر چندان مطمئن نیستند.

۱-۳ مقدمه

شکل ۱-۳ ایده کلی رمزنگاری کلید متقارن را نشان می‌دهد.



شکل ۱-۳ ایده کلی رمز کلید متقارن

در شکل ۱-۳ شخصی مثل آلیس می‌تواند پیامی را برای شخص دیگری مثلاً باب، از طریق یک کانال نامطمئن ارسال نماید. با این فرض که رقیبی مانند Eve، نمی‌تواند محتوا پیام را به سادگی به وسیله استراق سمع از کانال ارتباطی به دست آورد.

پیام اصلی ارسالی از سوی آلیس به باب را متن عادی و پیامی که از طریق کانال ارسال شده است را متن رمزی می‌نامیم. برای ساختن متن رمزی از متن عادی، آلیس از الگوریتم رمزنویسی و یک کلید رمز به اشتراک گذاشته شده با باب استفاده می‌کند. برای بازسازی متن عادی از متن رمز، باب از الگوریتم رمزگشایی و همان کلید رمز مشترک استفاده می‌کند. الگوریتم‌های رمزنگاری و رمزگشایی را رمزنویسی می‌نامیم. کلید عبارت است از مجموعه‌ی مقادیری که الگوریتم رمزنویسی بر پایه‌ی آن عمل می‌کند.

توجه داشته باشید که رمزنویسی کلید متقارن، برای هر دو عمل رمزنگاری و رمزگشایی از یک کلید یگانه استفاده می‌کند (خود کلید ممکن است مجموعه‌ای از مقادیر باشد). به علاوه، الگوریتم‌های رمزنگاری و رمزگشایی وارون یکدیگرند. اگر «P» متن عادی، «C» متن رمز و «K» کلید باشد، الگوریتم رمزگشایی $E_k(x)$ و وارون یکدیگرند؛ در نتیجه تأثیر یکدیگر را خنثی می‌کنند، به شرطی که یکی پس از دیگری بر یک ورودی یکتا اعمال شوند. به عبارت دیگر داریم:

$$\text{رمزنگاری: } C = E_k(P) \quad \text{رمزگشایی: } P = D_k(C)$$

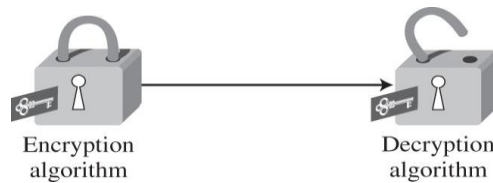
$$D_k(E_k(x)) = E_k(D_k(x)) = x$$

می‌توانیم ثابت کنیم که پیام عادی که باب بازسازی کرده است، دقیقاً همان پیامی است که آلیس ارسال نموده بود. فرض می‌کنیم باب متن p_1 را از پیام اصلی بازسازی کرده باشد، پس ثابت می‌کنیم که $p_1 = p$ است:

$$C = E_k(P) \text{ فرستنده} \quad P_1 = D_k(C) = D_k(E_k(P)) = P \text{ گیرنده}$$

شایان ذکر است که طبق اصل کریشف (که بعداً توضیح داده خواهد شد) بهتر است روش رمزگشایی و روش رمزنگاری را در دسترس عموم قرار دهیم ولی کلید رمز مشترک را پنهان نگه داریم. این بدان معناست که آلیس و باب برای تبادل کلید رمز نیاز به کانال دیگری دارند که مطمئن و امن باشد. مثلاً آلیس و باب می‌توانند یک بار یکدیگر را ملاقات نموده و کلید را شخصاً مبادله نمایند. کانال مطمئن در این مثال، تبادل رودرروی کلید است. همچنین می‌توانند برای تبادل، به شخص ثالث نیز اعتماد کنند. آلیس و باب می‌توانند با استفاده از نوع دیگری از رمزنگاری - کلید رمز نامتقارن - که بعداً در این فصل به آن خواهیم پرداخت، یک کلید رمز موقت ایجاد کنند. در فصول بعدی در مورد این شیوهی ارتباط مفصلاً بحث خواهیم کرد. در این فصل فرض می‌کنیم که کلید رمز مشترک بین آلیس و باب وجود دارد. با استفاده از رمزنویسی کلید متقارن، آلیس و باب می‌توانند از همان کلید برای ارتباط از سوی باب به آلیس هم استفاده کنند، به همین دلیل است که به این روش، کلید رمز متقارن می‌گویند. عنصر دیگری که در کلید رمز متقارن مطرح است، تعداد کلیدها می‌باشد. آلیس برای ارتباط با شخص دیگری مثلاً دیوید، باید از کلید رمز دیگری استفاده کند. اگر m تعداد افراد یک گروه باشند که می‌خواهند با یکدیگر ارتباط برقرار کنند، چه تعداد کلید لازم است؟ پاسخ $m(m-1)/2$ است؛ زیرا هر فرد برای ارتباط با سایر افراد گروه به $m-1$ کلید نیاز دارد. اما کلید بین A, B را می‌توان در هر دو جهت از A به B و از B به A استفاده کرد. در فصول بعد خواهیم دید که چگونه می‌توان بر این مشکل چیره شد.

رمزنگاری را می‌توان به منزله‌ی گذاشتن پیام در جعبه و قفل کردن آن؛ و رمزگشایی را می‌توان باز کردن جعبه و بیرون آوردن پیام در نظر گرفت. همان‌گونه که در شکل ۲-۳ نشان داده شده است، در رمزنگاری کلید متقارن، یک کلید یکتا عمل قفل کردن و باز کردن جعبه را انجام می‌دهد. فصول بعدی نشان می‌دهد که رمزنویسی کلید نامتقارن مستلزم داشتن دو کلید می‌باشد، یکی برای قفل کردن و یکی برای باز کردن جعبه.



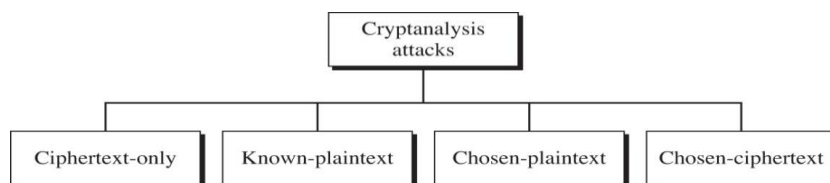
شکل ۳-۲: در رمزنویسی کلید متقارن، قفل کردن و باز کردن با یک کلید یگانه انجام می‌شود.

اصل کریشهف^۱

اگر چه ممکن است چنین به نظر بیاید که اگر الگوریتم رمزنگاری/رمزگشایی و کلید رمز را پنهان نگه داریم، رمز بسیار مطمئن‌تر خواهد بود، اما این کار توصیه نمی‌شود. بر اساس اصل کریشهف باید همیشه فرض کرد که رقیب، الگوریتم رمزنویسی/رمزگشایی را می‌داند؛ مقاومت رمز در برابر شکسته شدن فقط باید مبتنی بر سری بودن کلید باشد، به عبارت دیگر، کلید باید به قدری مشکل باشد که نیازی به پنهان کردن الگوریتم رمزنگاری/رمزگشایی نباشد. هنگام مطالعه رمزهای جدید این اصل بیشتر خود را نمایان می‌کند. امروزه تنها چند الگوریتم انگشت شمار برای رمزنگاری نوین وجود دارد، با این وجود دامنه‌ی کلید برای هر الگوریتم چنان گسترده است که یافتن کلید را برای رقیب بسیار مشکل می‌کند.

کشف رمز^۲

از آنجا که رمزنویسی علم و هنر ایجاد کدهای سری است، کشف رمز علم و هنر شکستن این کدها است. علاوه بر مطالعه‌ی روش‌های رمزنگار، لازم است به مطالعه‌ی روش‌های کشف رمز نیز پردازیم. مطالعه‌ی روش‌های کشف رمز به ما کمک می‌کند تا کدهای سری بهتری ایجاد کنیم. همان‌گونه که در شکل ۳-۳ نشان داده شده است، چهار روش رایج کشف رمز وجود دارد. در این فصل و فصول بعدی برخی از این روش‌ها و رمزهای خاص را مطالعه خواهیم کرد.



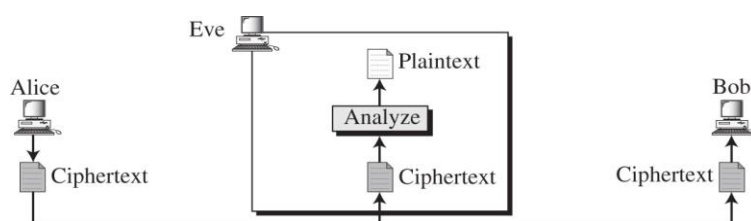
شکل ۳-۳: روش‌های کشف رمز

^۱ Kerckhoff's Principle

^۲ Cryptanalysis

روش کشف رمز تنها با معلوم بودن متن رمز^۱

در این روش، Eve تنها به متن رمز شده دسترسی دارد. او سعی می‌کند کلید مناسب و متن اولیه را پیدا کند. فرض آبنمک است که Eve الگوریتم را می‌داند و می‌تواند متن ارسالی را استراق سمع کند. این روش دسترسی به متن رمز، محتمل‌ترین روش است، زیرا Eve برای این دسترسی تنها متن رمز را لازم دارد. جهت خنثی کردن کشف رمز پیام به وسیله رقیب، رمز می‌بایست در برابر این نوع حملات بسیار مقاوم باشد. شکل ۳-۴ این مراحل را نشان می‌دهد.



۳-۴: حمله فقط با در اختیار داشتن متن رمز

روش‌های گوناگونی را می‌توان در کشف رمز با این روش به کار برد. در اینجا به چند مورد رایج آن اشاره می‌کنیم.

روش آزمودن تمام کلیدها یا روش جستجوی جامع کلید^۲: Eve سعی می‌کند تمام کلیدهای

ممکن را به کار ببرد. فرض می‌کنیم که Eve الگوریتم و دامنه کلید (فهرست تمام کلیدهای ممکن) را می‌داند و با استفاده از متن رمزنگاری شده و با آزمودن همه کلیدهای ممکن، به کشف رمز متن رمزی ادامه می‌دهد تا متن عادی قابل فهمی به دست آورد.

در گذشته استفاده از این روش کار سختی بود، اما امروزه با استفاده از رایانه ساده شده است. برای ناکام گذاشتن این روش، تعداد حالت‌های کلیدهای ممکن باید بسیار زیاد باشد.

کشف رمز به روش تحلیل آماری^۳

برای کشف رمز می‌توان از برخی از ویژگی‌های ذاتی زبانی که متن عادی با استفاده از آن نوشته شده است و روش‌های آماری استفاده کرد. به عنوان مثال، می‌دانیم که حرف E بیشترین کاربرد را در متون انگلیسی دارد. برای کشف رمز، یک شاخص با نرخ استفاده زیاد در متن رمز را پیدا کرده و فرض می‌کنیم که حرف متناظر با آن در متن عادی، حرف E است. پس از یافتن تعدادی از این حروف، تحلیل‌گر

^۱ Ciphertext-Only Attack

^۲ Brute-Force Attack

^۳ Statistical Attack

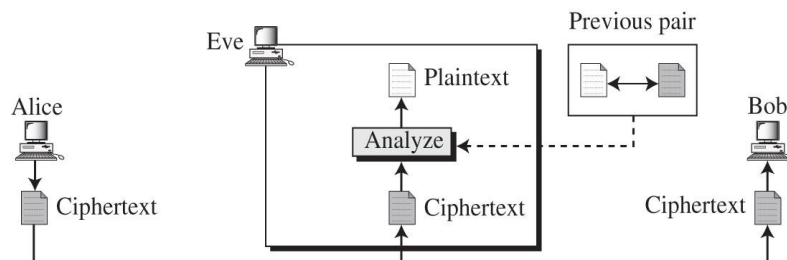
می‌تواند کلید را پیدا کرده و از آن برای رمزگشایی استفاده نماید. برای جلوگیری از این نوع حملات، روش رمزنگاری باید ویژگی‌های ذاتی زبان را پنهان کند.

کشف رمز با استفاده از بررسی الگو^۱

ممکن است برخی از رمزها ویژگی‌های زبانی را پنهان کنند، اما در عوض الگوهایی در متن رمز تولید کند. کاشف رمز ممکن است از این الگوها برای شکستن رمز استفاده کند. بنابراین، استفاده از رمزهایی که متن رمز را تا حد ممکن تصادفی جلوه دهد، با اهمیت است.

حمله با استفاده از متن عادی شناخته‌شده^۲

همان‌گونه که در شکل ۳-۵ نشان داده شده است؛ در حمله از گونه متن اولیه شناخته شده، Eve علاوه بر متن رمز استراق سمع شده، به قسمت‌هایی از متن اولیه / متن رمز نیز دسترسی دارد.



شکل ۳-۵: حمله به روش متن اولیه شناخته‌شده

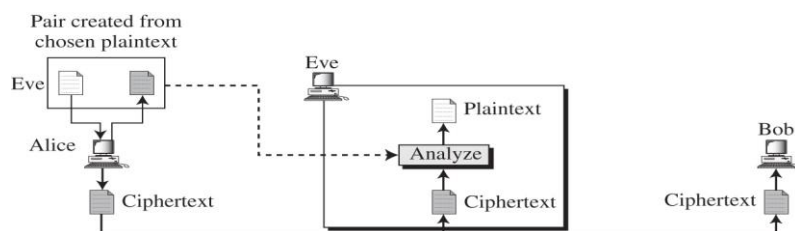
Eve پیشتر متن عادی رمز شده را گردآوری نموده است. به عنوان مثال، آلیس پیام سری برای باب می‌فرستد، اما وی بعداً محتوای پیام را برای عموم منتشر می‌کند. با فرض این که آلیس کلیدش را عوض نمی‌کند، Eve اقدام به نگهداری متن اولیه و متن رمز جهت استفاده از آن‌ها برای شکستن رمز پیام سری بعدی آلیس به باب می‌کند. Eve از رابطه بین دو متن قبلی برای تحلیل پیام رمز فعلی استفاده می‌کند. روش‌های استفاده شده در کشف رمز پیشین را می‌توان در اینجا نیز به کار برد. اجرای این حمله بسیار ساده‌تر است، زیرا Eve اطلاعات زیادی برای تحلیل دارد؛ با این وجود احتمال وقوع آن بسیار کمتر است، زیرا آلیس ممکن است کلیدش را عوض کرده باشد یا این که محتوای هیچ یک از پیام‌های پیشین را افشا نکرده باشد.

^۱ Pattern Attack

^۲ Known-Plaintext Attack

حمله با استفاده از متن عادی انتخابی^۱

حمله از گونه متن عادی انتخابی، شبیه حمله به روش متن عادی شناخته شده است؛ با این تفاوت که زوج متن اولیه/ متن رمز توسط خود حمله کننده انتخاب می‌شود. شکل ۳-۶ این مراحل را نشان می‌دهد.

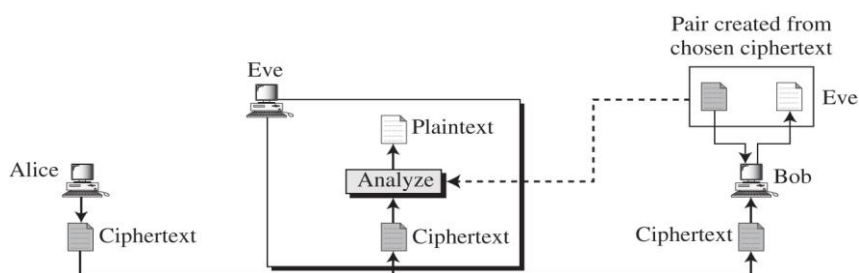


شکل ۳-۶: حمله به روش متن عادی انتخاب شده

به عنوان مثال، حمله از این گونه به شرطی اتفاق می‌افتد که Eve به رایانه آلیس دسترسی داشته باشد پس او می‌تواند متن عادی‌ای را انتخاب کند و متن رمز مربوطه را ایجاد نماید. البته Eve کلید را ندارد زیرا معمولاً کلید در نرم‌افزاری که فرستنده از آن استفاده می‌کند، ذخیره شده است. انجام این نوع حمله بسیار ساده‌تر و در عین حال بسیار غیر محتمل‌تر است.

حمله با استفاده از متن رمز انتخابی^۲

حمله از گونه‌ی متن رمز انتخابی، شبیه به حمله از گونه‌ی متن عادی انتخابی است، با این تفاوت که Eve متن رمزی را انتخاب می‌کند و به منظور ایجاد جفت متن رمز/ متن عادی، آن را رمزگشایی می‌کند. این اتفاق زمانی رخ می‌دهد که Eve به رایانه باب دسترسی داشته باشد. شکل ۳-۷ این مراحل را نشان می‌دهد.



شکل ۳-۷: حمله به روش متن رمز انتخابی

^۱ Chosen-Plaintext Attack

^۲ Chosen-Ciphertext Attack

رسته رمزهای سنتی^۱

رمزهای کلید متقارن سنتی را به دو دسته تقسیم می‌کنیم: رمزهای جایگزین و رمزهای جایگشتی. در رمز جایگزین، متن رمز حرف دیگری را، جایگزین حرفی در متن عادی می‌نماییم. در این جایگشتی، موقعیت و جایگاه حروف را در متن عادی تغییر می‌دهیم.

۲-۳ رمزهای جایگزین^۲

رمز جایگزین، نشانه‌ای را جایگزین نشانه دیگر می‌کند. اگر نشانه‌ها در متن عادی حروف الفبایی باشند، حرفی را جایگزین حرف دیگر می‌کنیم. به عنوان مثال می‌توانیم D را جایگزین A و Z را جایگزین T کنیم. چنانچه نشانه‌ها رقم باشند (۰ تا ۹)، می‌توانیم ۷ را جایگزین ۳ و ۶ را جایگزین ۲ کنیم. رمز جایگزین را می‌توان به دو دسته‌ی رمزهای تک الفبایی و رمزهای چند الفبایی تقسیم کرد.

رمز جایگزین، نشانه‌ای را جایگزین نشانه دیگر می‌کند

رمزهای تک الفبایی^۳

ابتدا در مورد گروهی از رمزهای جایگزین به نام «رمزهای تک الفبایی» صحبت می‌کنیم. در جایگزین تک الفبایی، یک حرف در متن عادی همیشه بدون توجه به موقعیت و جایگاه آن در متن، به حرف دیگری در متن رمزی تغییر می‌یابد. به عنوان مثال، چنانچه طبق الگوریتم باید حرف A به حرف D تغییر یابد، تمام حروف A به حرف D تبدیل می‌شوند؛ به عبارت دیگر، رابطه بین حروف در متن عادی و متن رمز، یک به یک است.

در جایگزین تک الفبایی، رابطه بین نشانه در متن اولیه با نشانه در متن رمزی همیشه یک به یک است.

¹ Categories of Traditional Ciphers

² Substitution Ciphers

³ Monoalphabetic Ciphers

مثال ۱-۳

در رمزنگاری زیر، متن عادی و متن رمز شده آن را آورده‌ایم. از حروف کوچک برای نشان دادن متن عادی و از حروف بزرگ برای نشان دادن متن رمز استفاده کرده‌ایم. احتمالاً رمز تک الفبایی است، زیرا هر دو L متن ساده به شکل O رمزنویسی شده است.

Plaintext: hello

Ciphertext: KHOOR

مثال ۲-۳

در رمز زیر، متن عادی و متن رمزی آن را نشان داده‌ایم. رمز، تک الفبایی نیست زیرا شاخص‌های مختلفی جایگزین هر حرف L شده‌اند.

Plaintext: hello

Ciphertext: ABNZF

رمز افزایشی^۱

ساده‌ترین رمز تک الفبایی، رمز افزایشی است. این رمز را گاهی رمز انتقالی و گاهی رمز سزاری هم می‌نامند، اما عبارت «رمز افزایشی» شالوده ریاضیاتی آن را بهتر آشکار می‌سازد. فرض کنید متن عادی شامل حروف کوچک (از a تا z) و متن رمز شامل حروف بزرگ (از A تا Z) است. همان‌گونه که در شکل ۳-۸ نشان داده شده است، برای این که بتوانیم اعمال ریاضی را روی متن عادی و متن رمز انجام دهیم، برای هر حرف (بزرگ یا کوچک) مقدار عددی را در نظر می‌گیریم.

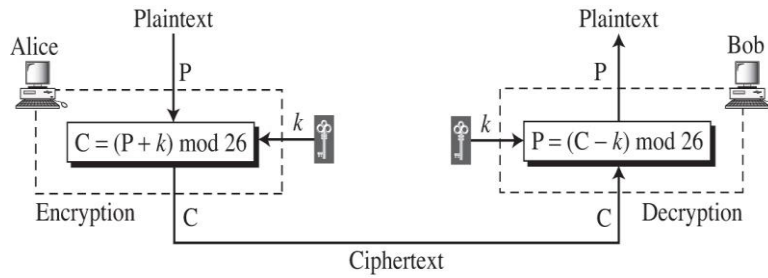
Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

شکل ۳-۸: نمایش شاخص‌های متن عادی و متن رمز در Z_{26}

در شکل ۳-۸، به هر حرف (بزرگ یا کوچک) یک عدد صحیح در Z_{26} اختصاص داده شده است. کلید سری بین آلیس و باب هم یک عدد صحیح در Z_{26} است. الگوریتم رمزگشایی کلید را از شاخص‌های متن رمز کم می‌کند. توجه کنید تمام اعمال در مجموعه‌ی Z_{26} انجام می‌شود. شکل ۳-۹ مراحل کار را نشان می‌دهد.

=====

^۱ Additive Cipher



شکل ۳-۹: مراحل کار در رمز افزایشی

به سادگی می‌توانیم ثابت کنیم که رمزنگاری و رمزگشایی وارون یکدیگرند، زیرا متن عادی ایجاد شده به وسیله‌ی باب (P_1) همان متن فرستاده شده‌ی آلیس (P) است. زمانی که رمزنگاری به روش افزایشی باشد، متن عادی، متن رمز و حتی کلید هم، اعداد صحیحی در مجموعه‌ی Z_{26} هستند.

مثال ۳-۳

از رمز افزایشی با کلید مساوی ۱۵ جهت رمزنگاری **hello** استفاده کنید.

حل:

الگوریتم رمزنگاری را شاخص به شاخص، روی متن عادی اعمال می‌کنیم.

Plaintext: h \rightarrow ۰۷	Encryption: $(07+15) \bmod 26$	Ciphertext: ۲۲ \rightarrow W
Plaintext: e \rightarrow ۰۴	Encryption: $(04+15) \bmod 26$	Ciphertext: ۲۲ \rightarrow T
Plaintext: l \rightarrow ۱۱	Encryption: $(11+15) \bmod 26$	Ciphertext: ۲۲ \rightarrow A
Plaintext: l \rightarrow ۱۱	Encryption: $(11+15) \bmod 26$	Ciphertext: ۲۲ \rightarrow A
Plaintext: o \rightarrow ۱۴	Encryption: $(14+15) \bmod 26$	Ciphertext: ۲۲ \rightarrow D

نتیجه «WTAAD» است. توجه داشته باشید، رمز، تک الفبایی است، زیرا حروف یکسان متن عادی (مثل L) به شکل حروف یکسان (مثل A) رمزنگاری شده‌اند.

مثال ۳-۴

از رمز افزایشی با کلید مساوی ۱۵ جهت رمزگشایی پیام «WTAAD» استفاده کنید.

حل:

الگوریتم رمزگشایی را حرف به حرف روی متن رمز اعمال می‌کنیم.

Ciphertext: W → ۲۲	Encryption: $(۲۲-۱۵) \bmod ۲۶$	Plaintext: ۰۷ → h
Ciphertext: T → ۱۹	Encryption: $(۱۹-۱۵) \bmod ۲۶$	Plaintext: ۰۴ → e
Ciphertext: A → ۰۰	Encryption: $(۰۰-۱۵) \bmod ۲۶$	Plaintext: ۱۱ → l
Ciphertext: A → ۰۰	Encryption: $(۰۰-۱۵) \bmod ۲۶$	Plaintext: ۱۱ → l
Ciphertext: D → ۰۳	Encryption: $(۰۳-۱۵) \bmod ۲۶$	Plaintext: ۱۴ → o

نتیجه «hello» است. توجه داشته باشید که عمل به پیمانه ۲۶ می‌باشد (به فصل ۲ مراجعه نمایید)،

این بدان معنی است که پاسخ منفی را باید به Z_{26} نگاشت کنیم (مثلاً ۱۵- می‌شود ۱۱).

رمز انتقالی^۱

از نگاه تاریخی، رمزهای افزایشی را رمزهای انتقالی می‌نامند. علت آن است که الگوریتم رمزنگاری را می‌توان به عنوان کلید انتقال شاخص‌ها به سمت پایین و الگوریتم رمزگشایی را به عنوان کلید تغییر شاخص‌ها به سمت بالا در ترتیب حروف الفبا تفسیر کرد. به عنوان مثال اگر کلید مساوی ۱۵ باشد، الگوریتم رمزگذاری، ۱۵ شاخص به سمت پایین (به سمت انتهای الفبا) انتقال می‌یابد. الگوریتم رمزگشایی هم ۱۵ شاخص به سمت بالا (به سمت ابتدای الفبا) حرکت می‌کند. البته وقتی به انتها یا ابتدای الفبا برسیم برمی‌گردیم.

رمز سزار^۲

ژولیوس سزار از رمز افزایشی برای برقراری ارتباط با افسران استفاده می‌کرده، به همین دلیل برخی اوقات رمز افزایشی را رمز سزار نیز می‌نامند. سزار از یک کلید ۳ تایی برای ارتباطاتش استفاده می‌کرد.

رمزهای افزایشی را برخی اوقات رمز انتقالی یا رمز سزار می‌نامند.

^۱ Shift Cipher

^۲ Caesar Cipher

کشف رمز^۱

رمزهای افزایشی در برابر حملات به متن رمز که با استفاده از جستجوی جامع کلیدها (حملات آزمودن کلمه کلیدها) انجام می‌شود، آسیب‌پذیرند. دامنه‌ی کلید رمز افزایشی بسیار محدود است زیرا فقط ۲۶ کلید وجود دارد، با این حال، یکی از کلیدها - صفر - بدون استفاده است (با کلید صفر متن رمز دقیقاً مانند متن عادی است). این امر باعث می‌شود فقط ۲۵ کلید باقی بماند. رمزشکن می‌تواند به سادگی اقدام به رمزگشایی با استفاده از آزمودن تمام کلیدهای ممکن نماید.

مثال ۳-۵

رمزشکن پیام رمزی UVACLYFZLJBYL را دریافت کرده است. نشان دهید که چگونه می‌تواند از روش آزمودن کلمه کلیدها برای شکستن رمز استفاده کند.

حل:

رمزشکن کلیدها را از ۱ تا ۷ امتحان می‌کند. با کلید ۷ متن عادی معنادار به دست می‌آید.

متن رمز: UVACLYFZLJBYL

K = 1	→	Plaintext: tuzbkxeykiaxk
K = 2	→	Plaintext: styajwdxjhzwj
K = 3	→	Plaintext: rsxzivcwigyvi
K = 4	→	Plaintext: qrwyhubvhfxuh
K = 5	→	Plaintext: pqvxgtaugewtg
K = 6	→	Plaintext: opuwfsztfdvsv
K = 7	→	Plaintext: notverysecure

رمزهای افزایشی ممکن است در معرض حملات آماری نیز قرار گیرد. وقتی که رقیب متن رمز طولانی در اختیار داشته باشد، این روش حمله محتمل‌تر است. رقیب می‌تواند از تعداد تکرار حروف در یک زبان خاص استفاده کند. جدول ۳-۱ تعداد تکرار برای یک متن انگلیسی ۱۰۰ حرفی را نشان می‌دهد.

=====

¹ Cryptanalysis

جدول ۳-۱: تعداد تکرار حروف در متن انگلیسی

Letter	Frequency	Letter	Frequency	Letter	Frequency	Letter	Frequency
E	12.7	H	6.1	W	2.3	K	0.08
T	9.1	R	6.0	F	2.2	J	0.02
A	8.2	D	4.3	G	2.0	Q	0.01
O	7.5	L	4.0	Y	2.0	X	0.01
I	7.0	C	2.8	P	1.9	Z	0.01
N	6.7	U	2.8	B	1.5		
S	6.3	M	2.4	V	1.0		

با این وجود، گاهی اوقات تحلیل متن رمز تنها بر مبنای اطلاعاتی در مورد تعداد تکرار یک حرف خاص مشکل است. ممکن است تعداد تکرار ترکیبات حروفی خاص را نیز لازم داشته باشیم. باید تعداد تکرار رشته‌های دو حرفی یا سه حرفی در متن رمز را بدانیم و آن‌ها را با تعداد تکرار رشته‌های دو حرفی و سه حرفی در زبان مورد استفاده در متن عادی مقایسه کنیم.

در جدول ۳-۲ رایج‌ترین گروه‌های دو حرفی و سه حرفی برای متن انگلیسی نشان داده شده است.

جدول ۳-۲: گروه‌بندی بر مبنای تعداد تکرار گروه‌های دو حرفی و سه حرفی در زبان انگلیسی

Digram	TH, HE, IN, ER, AN, RE, ED, ON, ES, ST, EN, AT, TO, NT, HA, ND, OU, EA, NG, AS, OR, TI, IS, ET, IT, AR, TE, SE, HI, OF
Trigram	THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR, DTH

مثال ۳-۶

رمزشکن متن رمز شده زیر را دریافت کرده است. با استفاده از روش حمله آماری، متن اولیه را پیدا کنید.

XLILSYWIMWRSJVSWEPIJSVJSYVQMPPMSRHSPPPEVWMXMXMWASVX-
LQSVILY -VVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIV

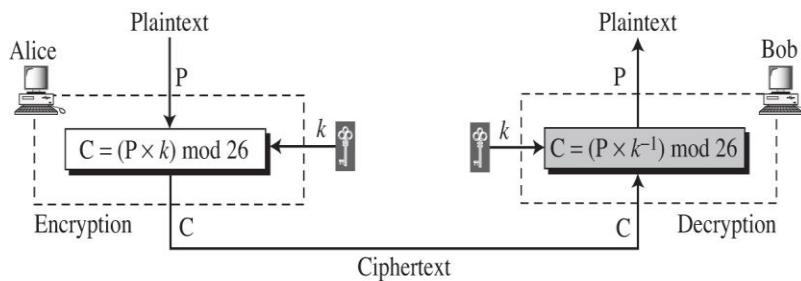
حل:

وقتی که رمزشکن تعداد تکرار حروف در متن رمز شده را به شکل جدول در آورد، حاصل چنین است: $S=1$, $V=13$, $I=14$ و الی آخر. پرکاربردترین حرف I با ۱۴ مورد تکرار است. این نشان می‌دهد که شاخص I در متن رمز احتمالاً حرف e در متن عادی است؛ یعنی کلید مساوی E است. رمزشکن متن را رمزگشایی می‌کند و متن زیر به دست می‌آید:

The house is now for sale for four million dollars it is worth more hurry before the seller receives more offers

رمزهای افزاینده^۱

همان گونه که در شکل ۳-۱۰ نشان داده شده است، در رمز افزاینده، الگوریتم رمزنگاری متن عادی را در کلید ضرب می کند و الگوریتم رمزگشایی، متن رمز را بر کلید تقسیم می کند؛ با این حال، از آنجا که این اعمال در مجموعه Z_{26} است، رمزگشایی به معنی ضرب در وارون ضربی کلید می باشد. توجه داشته باشید برای این که رمزنگاری و رمزگشایی وارون یکدیگر باشند، باید کلید از مجموعه Z_{26} انتخاب شود.



شکل ۳-۱۰: رمزنگاری مبتنی بر ضرب

در رمز افزاینده، متن عادی و متن رمز اعداد صحیحی در Z_{26} هستند، کلید عدد صحیحی در Z_{26}^* می باشد.

مثال ۳-۷

دامنه کلید برای یک رمز افزاینده چقدر است؟

حل:

کلید باید در مجموعه Z_{26}^* باشد. این مجموعه فقط ۱۲ عضو دارد:

۱، ۳، ۵، ۹، ۱۱، ۱۵، ۱۷، ۱۹، ۲۱، ۲۳، ۲۵

مثال ۳-۸

از رمز افزاینده برای رمزنگاری پیام «hello» با کلید ۷ استفاده می کنیم. متن رمز حاصل «XCZZU» می شود.

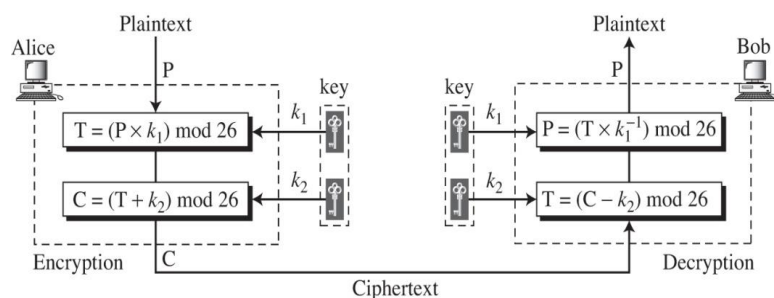
Plaintext: h \rightarrow ۰۷ Encryption: $(۰۷ \times ۰۷) \bmod ۲۶$ Ciphertext: ۲۳ \rightarrow X

^۱ Multiplicative Ciphers

Plaintext: e \rightarrow ۰۴	Encryption: $(04 \times 7) \bmod 26$	Ciphertext: ۰۲ \rightarrow C
Plaintext: l \rightarrow ۱۱	Encryption: $(11 \times 7) \bmod 26$	Ciphertext: ۲۵ \rightarrow Z
Plaintext: l \rightarrow ۱۱	Encryption: $(11 \times 7) \bmod 26$	Ciphertext: ۲۵ \rightarrow Z
Plaintext: o \rightarrow ۱۴	Encryption: $(14 \times 7) \bmod 26$	Ciphertext: ۲۰ \rightarrow U

رمز تناظری^۱

می‌توانیم رمزهای افزایشی و افزاینده را برای به دست آوردن آنچه که رمز تناظری نامیده می‌شود، با هم ترکیب کنیم. رمز تناظری، ترکیبی از هر دو رمز با یک جفت کلید می‌باشد. اولین کلید با رمز افزاینده و دومین کلید با رمز افزایشی به کار می‌رود. شکل ۳-۱۱ نشان می‌دهد که رمز تناظری در واقع دو رمز است که یکی پس از دیگری اعمال می‌شود. برای ما تنها عمل پیچیده‌ی رمزنگاری یا رمزگشایی ۲۶ وجود، از یک نتیجه موقت (T) استفاده کرده‌ایم و برای نشان دادن این که هر وقت از ترکیب رمزها استفاده می‌کنیم، باید مطمئن باشیم که هر یک از آن‌ها، وارون‌پذیرند و به ترتیب وارون هم در رمزنگاری و رمزگشایی به کار می‌روند، پس آن‌ها را به شکل دو عمل مجزا نشان داده‌ایم. اگر آخرین عمل در رمزنگاری جمع باشد، اولین عمل در رمزگشایی باید تفریق باشد.



شکل ۳-۱۱: رمز تناظری

در رمز تناظری، رابطه بین متن عادی P و متن رمز C به شرح زیر می‌باشد:

$$C = (P \times k_1 + k_2) \bmod 26 \quad P = ((C - k_2) \times k_1^{-1}) \bmod 26$$

¹ Affine Cipher

مثال ۳-۹

رمز تناظری از یک جفت کلید استفاده می‌کند که کلید اول از Z_{26}^* و کلید دوم از Z_{26} است. اندازه دامنه کلید، $۲۶ \times ۱۲ = ۳۱۲$ می‌باشد.

مثال ۳-۱۰

از رمز تناظری برای رمزنگاری پیام «hello» با کلیدهای (۷ و ۲) استفاده نمایید.

حل:

از ۷ برای کلید ضربی و از ۲ برای کلید افزایشی استفاده می‌کنیم. «ZEBBW» به دست می‌آید.

P: h \rightarrow ۰۷	Encryption: $(۰۷ \times ۷ + ۲) \bmod ۲۶$	C: ۲۵ \rightarrow Z
P: e \rightarrow ۰۴	Encryption: $(۰۴ \times ۷ + ۲) \bmod ۲۶$	C: ۰۴ \rightarrow E
P: l \rightarrow ۱۱	Encryption: $(۱۱ \times ۷ + ۲) \bmod ۲۶$	C: ۰۱ \rightarrow B
P: l \rightarrow ۱۱	Encryption: $(۱۱ \times ۷ + ۲) \bmod ۲۶$	C: ۰۱ \rightarrow B
P: o \rightarrow ۱۴	Encryption: $(۱۴ \times ۷ + ۲) \bmod ۲۶$	C: ۲۲ \rightarrow W

مثال ۳-۱۱

از رمز تناظری برای رمزگشایی پیام «ZEBBW» با کلیدهای (۷ و ۲) در پیمانه ۲۶ استفاده نمایید.

حل:

وارون نسبت به جمع $(۲۶ \bmod) ۲۴ \equiv -۲$ را به متن رمز دریافتی اضافه می‌کنیم؛ سپس جهت یافتن شاخص‌های متن عادی، نتیجه را در وارون ضربی $(۲۶ \bmod) ۱۵ \equiv ۷^{-۱}$ ضرب می‌کنیم. از آنجا که ۲ وارون نسبت به جمع در Z_{26} و وارون نسبت به ضرب در Z_{26}^* دارد، متن عادی به شکل زیر به دست می‌آید که از آن در مثال ۳-۱۰ استفاده کردیم.

C: Z \rightarrow ۲۵	Decryption: $((۲۵ - ۲) \times ۷^{-۱}) \bmod ۲۶$	P: ۰۷ \rightarrow h
C: E \rightarrow ۰۴	Decryption: $((۰۴ - ۲) \times ۷^{-۱}) \bmod ۲۶$	P: ۰۴ \rightarrow e
C: B \rightarrow ۰۱	Decryption: $((۰۱ - ۲) \times ۷^{-۱}) \bmod ۲۶$	P: ۱۱ \rightarrow l
C: B \rightarrow ۰۱	Decryption: $((۰۱ - ۲) \times ۷^{-۱}) \bmod ۲۶$	P: ۱۱ \rightarrow l
C: W \rightarrow ۲۲	Decryption: $((۲۲ - ۲) \times ۷^{-۱}) \bmod ۲۶$	P: ۱۴ \rightarrow o

مثال ۳-۱۲

رمز افزایشی، حالت خاصی از رمز تناظری است که در آن $K_1=1$ است. رمز افزایشی حالت خاصی از رمز تناظری است که در آن $K_2=0$ است.

کشف رمز در رمز تناظری^۱

اگرچه می‌توان از روش‌های آماری و آزمودن کلیه کلیدهای ممکن و حمله به متن رمز استفاده کرد اما اجازه دهید حمله به متن عادی انتخاب شده را آزمایش کنیم. فرض کنید رمز شکن متن رمز زیر را دریافت کرده است.

PWUFFOGWCHFDWIWEJOUUNJORSMDWRHVCMWJUPVCCG

همچنین او توانسته است مدت زمان کوتاهی به رایانه فرستنده دسترسی پیدا کند. این زمان تنها برای نوشتن یک متن ساده دو حرفی «ep» کافی بوده است؛ سپس سعی می‌کند متن کوتاه را با استفاده از دو الگوریتم متفاوت رمزنگاری کند. او از دو الگوریتم استفاده می‌کند زیرا مطمئن نیست که کدام یک، رمز تناظری باشد.

Algorithm 1: Plaintext: et ciphertext: \rightarrow WC

Algorithm 2: Plaintext: et ciphertext: \rightarrow WF

برای یافتن کلید، رمز شکن از راه کار زیر استفاده می‌کند.

(a) او می‌داند که اگر اولین الگوریتم یک الگوریتم تناظری باشد، می‌تواند دو معادله زیر را بر مبنای اولین مجموعه داده‌ها بسازد.

$$e \rightarrow W \quad 04 \rightarrow 22 \quad (04 \times K_1 + K_2) \equiv 22 \pmod{26}$$

$$e \rightarrow C \quad 19 \rightarrow 02 \quad (19 \times K_1 + K_2) \equiv 02 \pmod{26}$$

همان‌گونه که در فصل ۲ یاد گرفتیم، می‌توان دو معادله تجانس را حل کرد و مقادیر K_1 ، K_2 را به دست آورد. با این وجود، پاسخ قابل قبول نیست زیرا $K_1=16$ نمی‌تواند اولین قسمت کلید باشد چون ۱۶ وارون نسبت به ضرب در Z_{26}^* ندارد.

$$\begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} 4 & 1 \\ 19 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 22 \\ 2 \end{bmatrix} = \begin{bmatrix} 19 & 7 \\ 3 & 24 \end{bmatrix} \begin{bmatrix} 22 \\ 2 \end{bmatrix} = \begin{bmatrix} 16 \\ 10 \end{bmatrix} \rightarrow k_1 = 16 \quad k_2 = 10$$

=====

¹ Cryptanalysis of Affine Cipher

(b) اکنون Eve نتیجه دومین مجموعه از داده‌ها را آزمایش می‌کند.

$$e \rightarrow W \quad 04 \rightarrow 22 \quad (04 \times K_1 + K_2) \equiv 22 \pmod{26}$$

$$e \rightarrow F \quad 19 \rightarrow 05 \quad (19 \times K_1 + K_2) \equiv 05 \pmod{26}$$

ماتریس مربع و وارون آن یکی است. اکنون او $K_1 = 11$ و $K_2 = 4$ را دارد. این زوج قابل قبول‌اند زیرا K_1 وارون ضربی در Z_{26}^* دارد. رمزشکن برای رمزگشایی پیام، کلیدهای (۱۹ و ۲۲) را که وارون (۱۱ و ۴) است، آزمایش می‌کند.

متن عادی عبارت است از:

Best time of the year is spring when flowers bloom

رمز جایگزین تک الفبایی^۱

از آنجا که رمزهای افزایشی، افزایشده و تناظری، دامنه کلید کوچکی دارند، در برابر حمله‌ی آزمودن کلیدها، بسیار آسیب پذیرند. پس از توافق آلیس و باب بر سر یک کلید یکتا، از آن کلید برای رمزنگاری نامه در قالب متن رمزی یا رمزگشایی نامه در قالب متن عادی استفاده می‌کنند، به عبارت دیگر کلید، مستقل از پیام‌هایی است که منتقل می‌شود.

راه‌حل بهتر نگاشت بین هر شاخص در متن عادی و شاخص مربوطه در متن رمز می‌باشد. آلیس و باب می‌توانند روی جدولی که نشان دهنده‌ی نگاشت بین شاخص‌ها است توافق کنند. شکل ۳-۱۲ چنین نگاشتی را نشان می‌دهد.

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	N	O	A	T	R	B	E	C	F	U	X	D	Q	G	Y	L	K	H	V	I	J	M	P	Z	S	W

شکل ۳-۱۲: یک کلید نمونه برای رمز جایگزین تک الفبایی

مثال ۳-۱۳

می‌توانیم از کلید شکل ۳-۱۲ برای رمزنگاری پیام زیر استفاده کنیم.

متن رمز به شکل زیر است:

ICFVQRVVNEFVRNVSIYRGASHLIOJICNHTIYBFGTICRXR

=====

^۱ Monoalphabetic Substitution Cipher

کشف رمز

اندازه‌ی فضای کلید برای رمز جایگزین $26!$ است (تقریباً 4×10^{26}). این امر باعث می‌شود که حمله‌ی آزمودن کلیه کلیدها برای رمزشکن بینهایت مشکل شود، حتی اگر از یک رایانه قوی هم استفاده کند. با این وجود او می‌تواند بر اساس تعداد تکرار حروف، از حمله آماری استفاده کند.

رمزهای تک الفبایی، تعداد تکرار حروف را در متن رمز تغییر نمی‌دهد و این باعث می‌شود که رمز در برابر حمله آماری آسیب‌پذیر باشد.

رمزهای چند الفبایی^۱

در جایگزینی چند الفبایی، هر تکرار از وقوع نویسه ممکن است یک جایگزین متفاوت داشته باشد. رابطه‌ی بین شاخص در متن عادی با یک شاخص در متن رمز، یک به چند است؛ به عنوان مثال «a» را می‌توان در ابتدای متن به صورت «D» و در وسط متن به صورت «N» رمزنویسی کرد. مزیت رمز چند الفبایی این است که به وسیله‌ی آن می‌توان تعداد تکرار یک حرف در زبان مبدأ را پنهان کرد. رمزشکن نمی‌تواند از تعداد تکرار تنها یک حرف برای شکستن متن رمز استفاده کند.

برای ایجاد یک رمز چند الفبایی، باید هر شاخص متن رمز را هم به شاخص متناظر آن در متن عادی و هم به موقعیت شاخص در پیام رمزی مرتبط ساخت. این امر حاکی از آن است که کلید ما باید جریانی از کلیدهای فرعی باشد که در آن هر کلید فرعی به نوعی بستگی به موقعیت شاخص متن عادی-ای دارد که از آن برای رمزنویسی استفاده می‌کند؛ به عبارت دیگر، ما به یک رشته کلید $K = (K_1, K_2, K_3, \dots)$ نیاز داریم که در آن K_i برای رمزنویسی i امین شاخص در متن عادی جهت ایجاد i امین شاخص در متن رمز به کار می‌رود.

رمز کلید خودکار^۲

برای مشاهده‌ی وابستگی موقعیتی کلید، اجازه دهید در مورد یک رمز چند الفبایی ساده که به آن رمز کلید خودکار می‌گویند بحث کنیم. در این رمز، کلید، جریانی از کلیدهای فرعی است که هر کلید فرعی برای رمزنگاری شاخص مربوطه در متن عادی به کار می‌رود. اولین کلید فرعی، مقدار از پیش تعیین‌شده

¹ Polyalphabetic Ciphers

² Autokey Cipher

ای است که آلیس و باب به صورت سری بر سر آن توافق کرده‌اند؛ دومین کلید فرعی، مقدار اولین شاخص متن عادی (بین ۰ و ۲۵) می‌باشد؛ سومین کلید فرعی، مقدار دومین شاخص متن عادی است و الی آخر.

$$P=P_1P_2P_3\dots \quad C=C_1C_2C_3\dots \quad k=(K_1, P_1, P_2, \dots)$$

$$\text{Encryption: } C_i=(P_i+K_i) \bmod 26 \quad \text{Decryption: } P_i=(C_i-K_i) \bmod 26$$

نام کلید رمز -کلید خودکار- حاکی از این است که کلیدهای فرعی به صورت خودکار در طول فرآیند رمزنگاری از شاخص‌های رمزی متن عادی تولید می‌شوند.

مثال ۳-۱۴

فرض کنید آلیس و باب بر سر استفاده از رمز کلید خودکار با مقدار کلید عادی $K_1=12$ توافق کرده‌اند. اکنون آلیس می‌خواهد پیام را برای باب بفرستد. فرآیند رمزنگاری، شاخص به شاخص انجام می‌شود. نخست برابر شکل ۳-۸ عدد صحیح هر شاخص، جایگزین خود شاخص در متن عادی می‌شود. به منظور تولید نخستین شاخص متن رمز، اولین کلید فرعی به آن افزوده می‌شود. هنگامی که شاخص -های متن عادی خوانده می‌شود، بقیه کلید ساخته می‌شود. توجه داشته باشید که رمز، چند الفبایی است؛ زیرا حرف «a» که در متن سه بار تکرار شده به صورت متفاوتی رمزنگاری شده است. حرف «t» که سه بار در متن عادی آمده است، نیز به صورت‌های متفاوتی رمزنگاری شده است.

Plaintext:	a	t	t	a	c	k	i	s	t	o	d	a	y
P's Values:	00	19	19	00	02	10	08	18	19	14	03	00	24
Key stream:	12	00	19	19	00	02	10	08	18	19	14	03	00
C's Values:	12	19	12	19	02	12	18	00	11	7	17	03	24
Ciphertext:	M	T	M	T	C	M	S	A	L	H	R	D	Y

کشف رمز (تجزیه و تحلیل رمز)

رمز کلید خودکار، قطعاً آمار تعداد تکرار یک حرف واحد در متن عادی را پنهان می‌کند، با این حال، این رمز نیز مانند رمز افزایشی در برابر حمله آزمودن تمام کلیدها آسیب‌پذیر است. اولین کلید فرعی تنها می‌تواند یکی از ۲۵ مقدار (۱ تا ۲۵) باشد؛ بنابراین رمزهای چند الفبایی می‌خواهیم که نه تنها ویژگی‌های زبانی را پنهان کنند، بلکه دامنه کلیدی گسترده نیز داشته باشند.

رمز playfair

نمونه دیگری از رمز چند الفبایی، رمزی است که ارتش بریتانیا در طول جنگ جهانی اول از آن استفاده می‌کرد. کلید سری در این رمز متشکل از ۲۵ حرف الفبایی مرتب شده در یک ماتریس ۵×۵ می‌باشد (در رمزنگاری، حروف **I** و **J** یکی در نظر گرفته می‌شدند). چیدمان متفاوت حروف در ماتریس می‌تواند کلیدهای سری متفاوت زیادی را بسازد. یکی از چیدمان‌های احتمالی، در شکل ۳-۱۳ نشان داده شده است. ما حروف را در ماتریس به صورت مورب، که از گوشه بالا سمت راست شروع می‌شود قرار داده-ایم.

Secret Key =

L	G	D	B	A
Q	M	H	E	C
U	R	N	I/J	F
X	V	S	O	K
Z	Y	W	T	P

شکل ۳-۱۳: نمونه‌ای از کلید سری در رمز

پیش از رمزنگاری، اگر دو حرف در کنار هم یکسان باشند، یک حرف اضافی برای جدا کردن آن دو، بین آن‌ها قرار می‌گیرد. پس از گذاشتن حروف اضافی، اگر تعداد کارکترها در متن عادی فرد بود، یک حرف اضافی دیگر نیز به آخر متن اضافه می‌کنیم تا تعداد شاخص‌های متن زوج گردد. این رمز از سه قاعده زیر برای رمزنگاری استفاده می‌کند:

(a) اگر دو حرف از یک زوج در ردیف واحدی از کلید سری قرار گرفته باشند، شاخص رمزی متناظر برای دو حرف، حرف کناری سمت راست در همان ردیف می‌باشد (با حرکت به سمت آغاز ردیف، اگر حرف متن عادی آخرین شاخص در ردیف باشد).

(b) اگر دو حرف یک زوج در ستون واحدی از کلید سری قرار داشته باشند، شاخص رمز شده‌ی متناسب برای هر حرف، حرف زیر آن حرف در همان ستون است (با حرکت به سمت آغاز ردیف، اگر حرف متن عادی، آخرین شاخص در آن ستون باشد).

(c) اگر دو حرف یک زوج در ستون یا ردیف واحدی از کلید سری نباشند، شاخص رمز شده-ی متناسب برای هر حرف، حرفی است که در ردیف خودش، اما در همان ستون حرف دیگر باشد.

این رمز، مطابق معیارهای ما برای رمز چند الفبایی است. کلید، جریانی از کلیدهای فرعی است که کلیدهای فرعی به طور همزمان دو کلید ایجاد می‌کنند. در رمزنگاری **playfair** جریان کلید و جریان رمز یکی است. این بدین معنی است که قواعد بالا را می‌توان به عنوان قواعدی برای ایجاد جریان کلید نیز تلقی کرد. الگوریتم رمزنگاری، یک زوج از حروف متن عادی را گرفته و با استفاده از قواعد بالا، یک زوج کلید فرعی ایجاد می‌کند. می‌توانیم ادعا کنیم که جریان کلید به موقعیت شاخص حروف در متن عادی بستگی دارد. در اینجا وابستگی موقعیت، تفسیر متفاوتی دارد. کلید فرعی برای هر شاخص متن عادی به همسایه پیشین یا پسین آن بستگی دارد. اگر به رمز به این شیوه نگاه کنیم، متن رمز در واقع جریان کلید است.

$$P=P_1P_2P_3... \quad C=C_1C_2C_3... \quad k=[(k_1,k_2),(k_3,k_4),...]$$

$$\text{Encrytion: } C_i=K_i \quad \text{Decryption: } P_i=K_i$$

مثال ۳-۱۵

اجازه دهید با استفاده از کلید در شکل ۳-۱۳ متن ساده «hello» را رمزنگاری کنیم. وقتی حروف را به گروه‌های دو حرفی زوج تقسیم کنیم، «he, ll, o» به دست می‌آید. باید یک X بین دو حرف «L» قرار دهیم، در این صورت «he, lx, lo» حاصل می‌شود. در نتیجه پیام، به شکل زیر در می‌آید.

$$\begin{array}{lll} \text{he} \rightarrow \text{EC} & \text{lx} \rightarrow \text{QZ} & \text{lo} \rightarrow \text{BX} \\ \text{plaintext: hello} & & \text{Ciphertext: ECQZBX} \end{array}$$

از این مثال می‌توان دریافت که رمز در واقع رمز چند الفبایی است: دو حرف «l» به صورت «Q» و «B» رمزنگاری شده‌اند.

کشف رمز در رمز **playfair**

به طور قطع حمله‌ی آزمودن تمام کلیدهای ممکن در این رمز بسیار مشکل است. اندازه‌ی دامنه‌ی کلید ۲۵! است. علاوه بر این، این روش رمزنویسی تعداد تکرار تک حرفی شاخص‌ها را نیز پنهان می‌کند. با این وجود، تعداد تکرار ترکیبی از چند (تا حدی به علت درج شاخص اضافی) محفوظ است، بنابراین کاشف رمز می‌تواند با استفاده از حمله به روش دسترسی فقط به متن رمز و بر اساس آزمایش تعداد تکرار چند حرفی، کلید را پیدا کند.

Vigener رمز

ریاضیدان فرانسوی قرن شانزدهم، یک رمز جالب چند الفبایی طراحی کرد. این رمز از راهبرد متفاوتی برای ایجاد جریان کلید استفاده می‌کند. جریان کلید، تکرار جریان کلید سری عادی با طول m می‌باشد، طوری که $1 < m < 26$. وقتی (K_1, K_2, \dots, K_m) کلید سری عادی‌ای است که آلیس و باب بر آن توافق کرده باشند، می‌توان رمز را به شکل زیر توصیف کرد.

$$P=P_1P_2P_3\dots \quad C=C_1C_2C_3\dots \quad k=[(k_1,k_2,\dots,k_m),(k_1,k_2,\dots,k_m),\dots]$$

$$\text{Encryption: } C_i=P_i+K_i \quad \text{Decryption: } C_i-k_i$$

یک تفاوت عمده بین رمز تک الفبایی و دو رمز چند الفبایی دیگری که در مورد آن صحبت کردیم، این است که جریان کلید به شاخص‌های متن عادی بستگی نداشته و فقط به موقعیت شاخص در متن عادی بستگی دارد؛ به عبارت دیگر، جریان کلید را می‌توان بدون دانستن متن عادی ایجاد کرد.

مثال ۳-۱۶

اگر موافق باشید بررسی کنیم که چگونه می‌توانیم با استفاده از کلمه‌ی کلید ۶ حرفی «PASCAL»، پیام را رمزنگاری کنیم. جریان کلید عادی (۱۱ و ۰ و ۲ و ۱۸ و ۰ و ۱۵) می‌باشد. جریان کلید عبارت است از تکرار کلید عادی (هر چند بار که لازم باشد).

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	15	00	18	02	00	11	15	00	18	02	00	11	15	00
C's values:	07	07	22	10	18	22	23	18	11	6	13	19	02	06
Ciphertext:	H	H	W	K	S	W	X	S	L	G	N	T	C	G

مثال ۳-۱۷

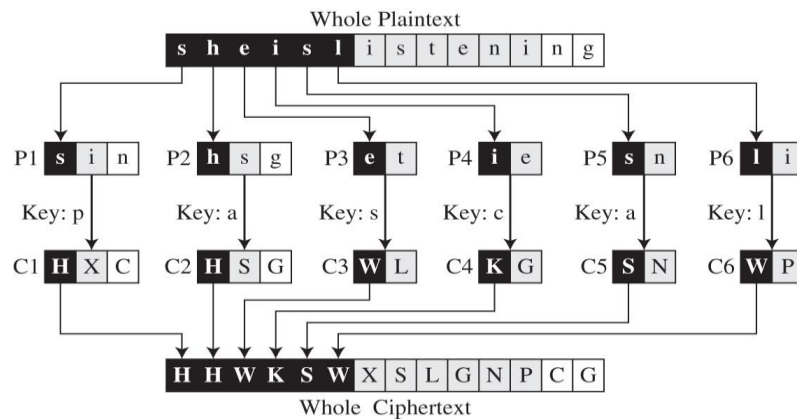
رمز را می‌توان ترکیبی از رمزهای افزایشی m دانست. شکل ۳-۱۴ نشان می‌دهد که چگونه متن اولیه مثال پیشین را به عنوان شش قطعه متفاوت که هر کدام جداگانه رمزنگاری می‌شود تلقی کرد. شکل به ما کمک می‌کند که بعداً روش کشف رمز را درک کنیم. m قطعه از متن عادی وجود دارد که برای ایجاد m قطعه متن رمز، هر قطعه به صورت جداگانه رمزنگاری می‌شود.

مثال ۳-۱۸

با استفاده از مثال ۳-۱۸ می‌توانیم بگوییم که رمز افزایشی، نوع خاصی از رمز Vigenere است که در آن $m=1$ می‌باشد.

جدول Vigenere

با نگاه از زاویه دیگری به جدول ۳-۳ می‌توان، به شیوه‌ی دیگری به رمزها نگاه کرد.



شکل ۳-۱۴ رمز به شکل ترکیبی از رمزهای افزایشی m

جدول ۳-۳: جدول رمز Vigenere

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	v	w	x	y	z	
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

اولین ردیف، کارکترهای متن عادی که باید رمزنگاری شوند را نشان می‌دهد. ستون اول حاوی شاخص‌هایی است که کلید باید از آن‌ها استفاده کند. بقیه جدول، شاخص‌های متن رمز را نشان می‌دهد. برای یافتن متن رمزی در متن عادی «she is listening» با استفاده از کلمه «PASCAL» به عنوان کلید، حرف «S» را در ردیف اول و «P» را در ستون اول پیدا می‌کنیم، محل تقاطع، شاخص متن رمز، حرف «H» است. برای رمزگشایی می‌توانیم «A» را در ردیف اول در ستون دوم پیدا کنیم، محل تقاطع شاخص متن رمز «H» است. تا وقتی که تمام شاخص‌های متن رمز پیدا شوند، به این کار ادامه می‌دهیم.

کشف رمز در رمز Vigenere

تمام رمزهای چند الفبایی، اثری از نرخ تکرار شاخص‌ها باقی نمی‌گذارد؛ با این وجود، رمزشکن هنوز می‌تواند از روش‌هایی برای کشف رمز پیام رمزنگاری شده استفاده کند. در اینجا کشف رمز دارای دو قسمت است: یافتن طول کلید و یافتن خود کلید.

(۱) روش‌های گوناگونی برای یافتن طول کلید طراحی شده است. در مورد یکی از روش‌ها در اینجا بحث می‌کنیم. در روش «Kasiski test» کاشف رمز به دنبال قسمت‌هایی از متن رمز می‌گردد که تکرار شده باشد و دستکم سه شاخص داشته باشد. فرض کنید دو قسمت پیدا شده و فاصله بین آن‌ها «d» است. کاشف رمز $d|m$ را در نظر می‌گیرد، طوری که m طول کلید است. اگر قسمت‌های تکراری بیشتری با فاصله d_1, d_2, \dots, d_n یافت شود، پس $\gcd(d_1, d_2, \dots, d_n) | m$ می‌باشد. این فرض منطقی است زیرا اگر دو شاخص یکی باشند و شاخص‌های $k \times m$ ($k=1, 2, \dots$) در متن عادی پخش شده باشند، پس آن‌ها یکی هستند و شاخص‌های $K \times m$ در متن رمز پخش شده‌اند. کاشف رمز برای پیگیری مواردی که در شاخص‌های موجود در کلید هویدا نیست، از قسمت‌هایی استفاده می‌کند که دستکم سه شاخص داشته باشد. ممکن است مثال ۳-۲۰ در درک بهتر دلیل به ما کمک کند.

(۲) پس از پیدا شدن طول کلید، کاشف رمز از ایده‌ای که در مثال ۳-۱۸ نشان داده شده است استفاده می‌کند. او متن رمز را به m تکه مجزا تقسیم می‌کند و روش استفاده شده برای کشف رمز افزایشی، از جمله حمله از گونه‌ی نرخ تکرار را به کار می‌برد. هر تکه از متن رمز را می-

توان رمزگشایی کرد و برای به دست آوردن کل متن عادی، آن‌ها را کنار هم گذاشت؛ به عبارت دیگر، کل متن رمز تعداد تکرار یک حرف از متن عادی را حفظ نمی‌کند، اما هر تکه این کار را انجام می‌دهد.

مثال ۳-۱۹

اجازه دهید فرض کنیم متن رمزی زیر را داریم.

LIOMWGFEGGDVWGHHCQUCRHRWAGWIOWQLKGZETKKMEVLWPCZVGT
HVTSGXQOVGCSVETQLTJSUMVWVEUVLXEWSLGFZMVVWLGYHCUSWXQH
KVGSHHEEVFLCFDGVSUMPHKIRZDMPHBBVWVWJWIXGFWLTSHGJOUEEHH
VUCFVGOWICQLTJSUXGLW

حاصل آزمون برای تکرار بخش‌های سه شاخصی و نتایج حاصل جدول ۳-۴ است.

جدول ۳-۴: تست مثال ۳-۱۹

String	First Index	Second Index	Difference
JSU	68	168	100
SUM	69	117	48
VWV	72	132	60
MPH	119	127	8

تفاوت بزرگ‌ترین مقسوم‌علیه‌های مشترک ۴ است و به این معنی است که طول کلید، مضربی از ۴ است. متن رمز را به چهار تکه تقسیم می‌کنیم. تکه C_1 متشکل از شاخص‌های ۱، ۵، ۹، ...، تکه C_2 متشکل از شاخص‌های ۲، ۶، ۱۰، ... و الی آخر. هر تکه را به طور جداگانه تحلیل آماری می‌کنیم. برای به دست آوردن کل متن عادی، یک شاخص را بین تکه‌های رمزگشایی قرار می‌دهیم. اگر متن عادی معنی نداشت، یک m دیگر را امتحان می‌کنیم.

C1: LWGWCRAOKTEPGTQCTJVUEGVGUQGECVPRPVJGTJEUGCJG
P1: jueuapymircneroarhtsthihytrahcieixsthcarrehe
C2: IGGGQHGWGKVCTSSOSQSWVWFVYSHSVFSSHZHWFSOHCOQSL
P2: usssctsiswhofeaeceihcetesoecatnpntherhctecex
C3: OFDHURWQZKLZHGVVLUVLSZWHWKHFDDUKDHVIWUHFWLWU
P3: lcaerotnwhiwedssirsiirhketehretltiideatrairt
C4: MEVHCWILEMWVXGETMEXLMLCXVELGMIMBWXLGEVVITX
P4: iardysehaisrrtcapiafpwtethecarhaesfterectpt

در این مورد، متن حاصل معنا دارد.

ژولیوس سزار^۱ در جنگ‌هایش از یک سیستم رمزی که امروزه به آن رمز سزار می‌گویند استفاده می‌کرد. این روش رمزنگاری، رمز افزایشی با دو کلید مساوی هم است و برای به دست آوردن متن رمز هر حرف در متن عادی، سه حرف در لیست الفبا جابجا می‌شود.

نمونه جالب دیگری از رمز چند الفبایی، رمز Hill است که شخصی بنام لیستر هیل^۲ مبدع آن بوده است. برعکس رمزهای چند الفبایی دیگری که تاکنون در مورد آن‌ها صحبت کردیم، متن عادی به بلوک‌های مساوی تقسیم می‌شود و در هر زمان یک بلوک از آن رمزنگاری می‌شود. این رمزنگاری به شیوه‌ای است که هر شاخص در بلوک، در رمزنگاری سایر شاخص‌های موجود در بلوک سهیم است. به این دلیل رمز Hill جزو آن دسته از رمزهایی است که به آن‌ها رمزهای بلوکی گفته می‌شود. سایر رمزهایی که تاکنون مطالعه کرده‌ایم، جزء آن دسته از رمزهایی هستند که به آن‌ها رمزهای جریانی گفته می‌شود. در پایان این فصل به تفاوت بین رمزهای بلوکی و جریانی می‌پردازیم.

در رمز Hill کلید، ماتریس مربع با اندازه $m \times m$ می‌باشد که m اندازه بلوک است. همان‌گونه که در شکل ۳-۱۵ نشان داده شده است، اگر کلید را ماتریس K بنامیم، هر عضو از ماتریس K_{ij} است.

$$K = \begin{bmatrix} k_{11} & k_{12} & \dots & k_{1m} \\ k_{21} & k_{22} & \dots & k_{2m} \\ \vdots & \vdots & & \vdots \\ k_{m1} & k_{m2} & \dots & k_{mm} \end{bmatrix}$$

شکل ۳-۱۵: کلید در رمز Hill

اجازه دهید نشان دهیم که چگونه بلوکی از متن رمزنگاری می‌شود. اگر m شاخص بلوک متن عادی را P_1, P_2, \dots, P_n بنامیم، شاخص‌های متناظر در بلوک متن رمز، C_1, C_2, \dots, C_m هستند. پس داریم:

$$C_1 = P_1 k_{11} + P_2 k_{21} + \dots + P_m k_{m1}$$

$$C_2 = P_1 k_{12} + P_2 k_{22} + \dots + P_m k_{m2}$$

...

$$C_m = P_1 k_{1m} + P_2 k_{2m} + \dots + P_m k_{mm}$$

¹ Julius Caesar

² Lester s.Hill

معادله نشان می‌دهد که چگونه هر شاخص متن رمز نظیر C_1 به تمام شاخص‌های متن عادی در بلوک (P_1, P_2, \dots, P_m) بستگی دارد. با این وجود، باید هوشیار باشیم که تمام ماتریس‌های مربع دارای وارون نسبت به ضرب در Z_{26} نمی‌باشند؛ بنابراین آلیس و باب باید در انتخاب کلید دقت کنند. اگر ماتریس وارون نسبت به ضرب نداشته باشد، باب نمی‌تواند پیام رمزی آلیس را رمزگشایی کند.

ماتریس کلید در رمز Hill، باید وارون ضربی داشته باشند.

مثال ۳-۲۰

استفاده از ماتریس، امکان رمزنگاری کل متن عادی را برای آلیس فراهم می‌کند. در این حالت، متن عادی یک ماتریس $m \times l$ است که l تعداد بلوک‌ها است. به عنوان مثال، متن اولیه code is ready وقتی که نویسه اضافی Z را به آخرین بلوک اضافه کنیم و فاصله‌ها را حذف کنیم، می‌تواند به صورت ماتریس 4×3 درآید. متن رمز حاصل OHKNIHGKLISS می‌شود. باب می‌تواند با استفاده از وارون ماتریس کلید، پیام را رمزگشایی کند. در شکل ۳-۱۶ رمزنگاری و رمزگشایی را مشاهده می‌کنید.

$$\begin{matrix} & \mathbf{C} & \\ \begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix} & = & \begin{matrix} & \mathbf{P} & \\ \begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix} & \end{matrix} & \begin{matrix} & \mathbf{K} & \\ \begin{bmatrix} 09 & 07 & 11 & 13 \\ 04 & 07 & 05 & 06 \\ 02 & 21 & 14 & 09 \\ 03 & 23 & 21 & 08 \end{bmatrix} & \end{matrix} \end{matrix}$$

a. Encryption

$$\begin{matrix} & \mathbf{P} & \\ \begin{bmatrix} 02 & 14 & 03 & 04 \\ 08 & 18 & 17 & 04 \\ 00 & 03 & 24 & 25 \end{bmatrix} & = & \begin{matrix} & \mathbf{C} & \\ \begin{bmatrix} 14 & 07 & 10 & 13 \\ 08 & 07 & 06 & 11 \\ 11 & 08 & 18 & 18 \end{bmatrix} & \end{matrix} & \begin{matrix} & \mathbf{K}^{-1} & \\ \begin{bmatrix} 02 & 15 & 22 & 03 \\ 15 & 00 & 19 & 03 \\ 09 & 09 & 03 & 11 \\ 17 & 00 & 04 & 07 \end{bmatrix} & \end{matrix} \end{matrix}$$

b. Decryption

شکل ۳-۱۶: مثال ۳-۲۰

کشف رمز در رمزهای Hill

رمزگشایی مبتنی بر متن رمزی، در این روش رمزنگاری مشکل است. نخست، حمله با استفاده از روش آزمودن تمام کلیدها به رمز Hill بی‌نهایت مشکل است زیرا کلید ماتریس $m \times m$ است. هر ورودی به ماتریس می‌تواند یکی از ۲۶ مقدار را داشته باشد. در نگاه اول معنایش این است که دامنه کلید $26^{m \times m}$ است. با این وجود، تمام ماتریس‌ها وارون ضربی ندارند، پس دامنه‌ی کلید از مقدار کوچک‌تر، اما هنوز گسترده است.

دوم، رمزهای Hill مشخصه‌های آماری متن عادی را حفظ نمی‌کند. رمزشکن نمی‌تواند تحلیل تعداد تکرار را روی حروف، دیاگرام‌هایی با یک حرف، دو حرف و سه حرفی در متن انجام دهد. فقط می‌توان تحلیل تعداد تکرار را روی کلمات با اندازه m که عین هم باشند انجام دهد.

با این حال، اگر رمزشکن مقدار m را بداند و زوج متن عادی/ متن رمز را دست‌کم برای m بلوک بداند، می‌تواند حمله به روش متن عادی شناخته شده را انجام دهد. این بلوک‌ها می‌توانند متعلق به یک پیام واحد یا پیام‌های مختلف باشند، اما در هر حال باید معلوم باشند. او می‌تواند دو ماتریس $m \times m$ تولید کند. P متن عادی و C متن رمز؛ ردیف‌های متناظر نشان دهنده‌ی زوج متن عادی/ متن رمز شناخته شده هستند. چون $C = PK$ است پس می‌تواند با استفاده از رابطه $K = CP^{-1}$ کلید را بیابد، البته به شرطی که P وارون‌پذیر باشد. اگر P وارون‌پذیر نباشد باید از مجموعه‌ی متفاوتی از زوج‌های متن عادی/ متن رمز استفاده کند.

اگر رمزشکن مقدار m را نداند، می‌تواند مقادیر مختلفی را آزمایش نماید، به شرطی که m خیلی بزرگ نباشد.

مثال ۳-۱۲

فرض کنید رمزشکن می‌داند $m = 3$ است. همان‌گونه که در شکل ۳-۱۷ نشان داده شده است او سه بلوک زوج متن عادی/ متن رمز را (که الزاماً از یک متن واحد نیستند) در نظر می‌گیرد.

$$\begin{array}{ccc} \begin{bmatrix} 05 & 07 & 10 \end{bmatrix} & \longleftrightarrow & \begin{bmatrix} 03 & 06 & 00 \end{bmatrix} \\ \begin{bmatrix} 13 & 17 & 07 \end{bmatrix} & \longleftrightarrow & \begin{bmatrix} 14 & 16 & 09 \end{bmatrix} \\ \begin{bmatrix} 00 & 05 & 04 \end{bmatrix} & \longleftrightarrow & \begin{bmatrix} 03 & 17 & 11 \end{bmatrix} \\ P & & C \end{array}$$

شکل ۳-۱۷: مثال ۳-۲۲ شکل‌گیری متن رمزی

رمزشکن ماتریس P و C را از این زوج‌ها می‌سازد. از آنجا که P وارون‌پذیر است ماتریس P را وارون کرده و همان‌گونه که در شکل ۳-۱۸ نشان داده شده است، برای به دست آوردن ماتریس K آن را در C ضرب می‌کند.

$$\begin{array}{ccc} \begin{bmatrix} 02 & 03 & 07 \\ 05 & 07 & 09 \\ 01 & 02 & 11 \end{bmatrix} & = & \begin{bmatrix} 21 & 14 & 01 \\ 00 & 08 & 25 \\ 13 & 03 & 08 \end{bmatrix} \begin{bmatrix} 03 & 06 & 00 \\ 14 & 16 & 09 \\ 03 & 17 & 11 \end{bmatrix} \\ K & P^{-1} & C \end{array}$$

شکل ۳-۱۸: مثال ۳-۲۲، یافتن کلید

رمزشکن اکنون کلید را دارد و می‌تواند هر متن رمزی که با آن کلید رمزنگاری شده است را رمزگشایی نماید.

رمز یک‌بار مصرف^۱

یکی از اهداف رمزنگاری، سری بودن در حد اعلاست. مطالعات شان نشان داده است که اگر هر سمبل متن عادی با یک کلید تصادفی انتخاب شده از دامنه کلید رمزنگاری شود، می‌توان به سری بودن در حد اعلا دست یافت. به عنوان مثال، یک رمز افزایشی را می‌توان به سادگی شکست، زیرا یک کلید یکتا برای رمزنگاری تمام نویسه‌ها به کار می‌رود. با این وجود، حتی این رمز ساده می‌تواند رمز در حد اعلا شود، به شرطی که کلیدی که برای رمزنگاری هر نویسه مورد استفاده قرار می‌گیرد به صورت تصادفی از دامنه کلیدها (۰۰،۰۱،۰۲،۰۳،۰۴،۰۵) انتخاب شده باشد؛ یعنی اگر اولین نویسه با استفاده از کلید ۰۴، نویسه دوم با کلید ۰۲، نویسه سوم با کلید ۲۱ و الی آخر رمزنگاری شود، حمله به روش استفاده از متن رمز غیرممکن است. اگر فرستنده هر بار که پیامی را می‌فرستد کلید را عوض کند و از توالی تصادفی دیگری از اعداد صحیح استفاده نماید، سایر انواع حملات نیز غیرممکن می‌شود.

شخصی به نام ورنام^۲ از این ایده در رمز موسوم به رمز یک‌بار مصرف استفاده کرد. در این رمز، طول کلید با طول متن عادی برابر است و کلید کاملاً تصادفی انتخاب می‌شود.

رمز یک‌بار مصرف، رمز کاملی است، اما اجرای تجاری آن تقریباً غیرممکن است. اگر کلید باید هر بار مجدداً ایجاد شود، آلیس هنگام ارسال پیام به باب چگونه کلید جدید را هر بار به او بگوید؟ با این وجود، مواردی وجود دارد که می‌توان از رمز یک‌بار مصرف استفاده کرد. به عنوان مثال، اگر رئیس جمهور کشوری بخواهد پیام کاملاً سری به رئیس جمهور کشور دیگری بفرستد، می‌تواند پیش از ارسال پیام، کلید تصادفی را توسط فرستاده قابل اطمینان به کشور مورد نظر بفرستد.

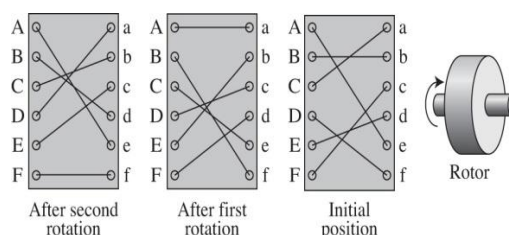
در فصول آخر کتاب در بخش‌هایی که کاربرد مدرن رمزنویسی معرفی می‌شود، در مورد برخی از تغییرات رمز یک‌بار مصرف بحث خواهیم کرد.

^۱ One-Time Pad

^۲ Vernam

رمز روتور^۱

اگرچه رمزهای یک‌بار مصرف عملی نیستند، اما رمز روتور گامی به سوی رمزنویسی مطمئن‌تر می‌باشد. این رمز از ایده جایگزینی تک الفبایی استفاده می‌کند اما نداشت بین نویسه‌های متن رمز و متن عادی برای هر نویسه متن عادی را تغییر می‌دهد. شکل ۳-۱۹ نمونه‌ی ساده شده‌ای از یک رمز روتر را نشان می‌دهد.



شکل ۳-۱۹: یک رمز روتور

روتور نشان داده شده در شکل ۳-۱۹ فقط از شش حرف استفاده می‌کند اما روتور واقعی از ۲۶ حرف استفاده می‌کند. روتور همیشه به صورت سخت‌افزاری سیم‌کشی شده است و ارتباط بین نویسه‌های رمزنگاری/ رمزگشایی به وسیله‌ی چرخ‌دنده‌ها^۲ تأمین می‌شود. توجه داشته باشید که سیم‌کشی طوری نشان داده شده است که انگار روتور شفاف است و می‌توان داخل آن را دید.

پیکربندی وضعیت اولیه روتور، کلید سری بین آلیس و باب است. با استفاده از تنظیم اولیه، نخستین نویسه متن عادی رمزنگاری می‌شود و نویسه دوم پس از اولین چرخش، رمزگذاری می‌شود و الی آخر (در شکل ۳-۱۹ چرخش ۱/۶ است، اما در دنیای واقعی ۱/۲۶ چرخش است).

اگر روتور بی‌حرکت باشد، یک کلمه سه حرفی نظیر **bee** به صورت **"BAA"** رمزنگاری می‌شود (رمز جایگزینی تک الفبایی). اما اگر در حال چرخش باشد، به صورت **"BCA"** رمزنگاری می‌شود (رمز روتور). بنابراین رمز روتور رمز چند الفبایی است، زیرا یک نویسه متن عادی که دوبار تکرار شده، به صورت دو نویسه متفاوت رمزنگاری شده است.

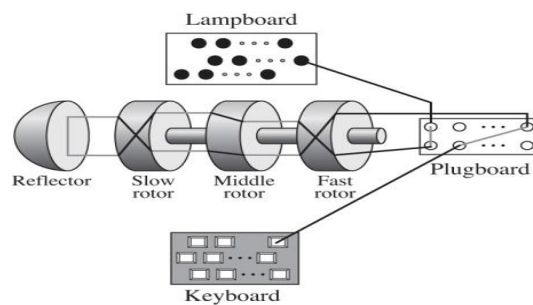
رمز روتور مانند رمز جایگزینی تک الفبایی در برابر حمله‌ی آماری مقاوم است، زیرا تعداد تکرار حروف را حفظ نمی‌کند.

¹ Rotor Cipher

² Brushes

ماشین انگیما

ماشین انگیما در اصل به وسیله‌ی **Sherbius** اختراع شد، اما ارتش آلمان آن را تغییر داده و در خلال جنگ جهانی دوم به صورت گسترده از آن استفاده کرد. این ماشین بر پایه‌ی اصول رمزهای روتور است. شکل ۳-۲۰ دیاگرام شماتیک ساده‌ای از این ماشین را نشان می‌دهد.



شکل ۳-۲۰: طرحی از ماشین انگیما

موارد زیر مؤلفه‌های اصلی این ماشین هستند:

- (۱) صفحه کلیدی با ۲۶ کلید، برای وارد کردن متن عادی در هنگام رمزنگاری و وارد کردن پیام رمز در هنگام رمزگشایی.
- (۲) صفحه کلید با ۲۶ لامپ که نویسه‌های متن رمز را در رمزنگاری و نویسه‌های متن عادی را در رمزگشایی نشان می‌دهد.
- (۳) صفحه سیم‌بندی با ۲۶ فیش که از طریق ۱۳ سیم به صورت دستی برای تولید جایگشت‌های متفاوت متصل شده‌اند و هر روز پیکربندی آن عوض می‌شود.
- (۴) سه روتور سیم‌کشی شده مانند آنچه که در بخش پیش توضیح داده شد. هر بار سه روتور از میان پنج روتور موجود انتخاب می‌شود. برای هر نویسه وارد شده به وسیله صفحه کلید، روتور سریع، ۱/۲۶ از یک چرخش کامل می‌چرخد. روتور میانی به ازای هر چرخش کامل روتور سریع، ۱/۲۶ چرخش می‌چرخد. روتور کند به ازای هر چرخش کامل روتور میانی، ۱/۲۶ می‌چرخد.
- (۵) یک بازتابنده^۱ که ثابت و از پیش سیم‌کشی شده است.

=====

¹ Reflector

کتاب کد

برای استفاده از ماشین انگیم، کتاب کد منتشر شد که در این کتاب، تنظیمات گوناگونی به شرح زیر برای هر در نظر گرفته شده بود:

(a) سه روتوری که باید از میان پنج روتور موجود انتخاب کرد.

(b) ترتیب نصب روتورها.

(c) تنظیمات صفحه Plugboard.

(d) کد یک حرفی روز.

مراحل رمزنگاری پیام

برای رمزنگاری پیام، کاربران مراحل زیر را انجام می‌دادند:

(۱) تنظیم نقطه شروع کار روتورها برحسب کد روز. به عنوان مثال، اگر کد «HUA» بود، مقدار

شروع به کار روتور به ترتیب، «H»، «U» و «A» خواهد بود.

(۲) انتخاب یک سه حرفی تصادفی، نظیر «ACF» و رمزنگاری رشته «ACFACF» (تکرار کد) با

استفاده از تنظیم عادی روتورها بر اساس گام ۱. به عنوان مثال، فرض کنید که رمزنگاری

«OPNABT» باشد.

(۳) تنظیم موقعیت‌های آغازین روتورها بر اساس OPN (نصف کد رمزنگاری شده).

(۴) افزودن ۶ حرف رمزنگاری شده حاصل از مرحله دوم (OPNABT) به ابتدای پیام.

(۵) رمزنگاری پیام حاوی کد ۶ حرفی و ارسال پیام رمزنگاری شده.

مراحل رمزگشایی پیام

برای رمزگشایی پیام کاربران مراحل زیر را انجام می‌دادند:

(۱) دریافت پیام و جدا کردن ۶ حرف اول آن.

(۲) تنظیم موقعیت آغازین روتورها بر اساس کد روز.

(۳) رمزگشایی ۶ حرف اول با استفاده از تنظیم عادی در گام ۲.

(۴) تنظیم موقعیت روتورها بر اساس نیمه اول کد رمزگشایی شده.

(۵) رمزگشایی پیام (بدون ۶ حرف اول).

کشف رمز

همان‌گونه که می‌دانیم با وجود این که ارتش آلمان و بقیه دنیا تا چند دهه بعد در مورد شکسته شدن رمز ماشین انگیم چیزی نشنیده بودند، اما رمز این ماشین در خلال جنگ جهانی دوم شکسته شد. پرسش این است که چگونه چنین رمز پیچیده‌ای شکسته شد؟ اگرچه ارتش آلمان می‌کوشید سیم‌کشی روتورها را پنهان نگه دارد، اما متفقیین توانستند تعدادی از این ماشین‌ها را به چنگ آورند. گام بعدی یافتن پیکربندی هر روزه و کد مورد استفاده در بکار اندازی روتورها برای هر پیام بود. اختراع کامپیوتر به متفقیین کمک کرد تا بر این مشکل چیره شوند. تصویر کامل ماشین و روش کشف رمز آن را می‌توان در برخی از وب سایت‌های انگیم یافت.

۳-۳- رمزهای مبتنی بر ترانهش (پس و پیش سازی)^۱

این‌گونه رمزها، سمبلی را جایگزین سمبل دیگر نمی‌کنند، بلکه مکان سمبل‌ها را در متن عوض می‌کنند. ممکن است اولین سمبل متن عادی را در دهمین جایگاه متن رمز یافت. یک سمبل در هشتمین جایگاه متن عادی ممکن است در اولین جایگاه متن رمز ظاهر شود؛ به عبارت دیگر، سمبل‌ها را دوباره چینش می‌کند (جابجا می‌کند).

در رمز ترانهش، سمبل‌ها دوباره چینش می‌شوند.

رمزهای ترانهش بدون کلید

رمزهای ترانهش ساده که در گذشته مورد استفاده قرار می‌گرفت، کلید ندارند. دو روش برای جابجا کردن نویسه‌ها وجود دارد. در روش اول، متن در یک جدول به شکل ستون به ستون نوشته می‌شود سپس ردیف به ردیف منتقل می‌شود. در روش دوم، متن در جدول ردیف به ردیف نوشته می‌شود و سپس ستون به ستون منتقل می‌گردد.

=====

¹ Transposition ciphers

مثال ۲۲-۳

یک نمونه‌ی خوب از رمز بدون کلید با استفاده از روش اول، «رمز نرده‌های راه‌آهن» است. در این رمز، متن عادی در دو خط با الگوی زیگزاگ (به روش ستون به ستون است) مرتب می‌شود، آلیس برای این که پیام به روش سطر به سطر را برای باب بفرستد، چنین می‌نویسد:

m e t m e a t t h e p a r k

سپس با ارسال ردیف اول و به دنبال آن ردیف دوم، متن رمز “MEMATEAKETETHPR” را تولید می‌کند. باب متن رمز را دریافت نموده و آن را نصف می‌کند (در این مورد نیمه‌ی دوم یک نویسه کمتر از نیمه‌ی اول دارد). اولین نیمه، ردیف اول و نیمه‌ی دوم ردیف دوم را تشکیل می‌دهد. باب نتیجه را به صورت زیگزاگ می‌خواند، چون کلیدی وجود ندارد و تعداد ردیف‌ها (برابر ۲) ثابت است. رمزگشایی متن رمز برای رمزشکن ساده خواهد بود. تنها چیزی که او باید بداند این است که رمز نرده‌های راه‌آهن به کار رفته است.

مثال ۲۳-۳

آلیس و باب می‌توانند بر سر تعداد ستون‌ها توافق نموده و از روش دوم استفاده کنند. آلیس، همان متن عادی را ردیف به ردیف در یک جدول چهار ستونی می‌نویسد.

m	e	e	t
m	e	a	t
t	h	e	p
a	r	k	

سپس وی با ترانهش ستون به ستون نویسه‌ها، متن رمز “MMTAEHREAEKTP” را ایجاد می‌کند. باب متن رمز را دریافت نموده و مراحل را به صورت وارون انجام می‌دهد. او پیام دریافتی را ستون به ستون می‌نویسد و ردیف به ردیف به صورت متن عادی می‌خواند. رمزشکن اگر تعداد ستون‌ها را بداند، به سادگی می‌تواند پیام را رمزگشایی کند.

مثال ۲۴-۳

در واقع، رمز و مثال ۲۳-۳ می‌تواند یک رمز ترانهش باشد. در شکل زیر، جابجایی هر نویسه در متن عادی به متن رمز بر مبنای موقعیت‌های هر نویسه نشان داده شده است.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
01	05	09	13	02	06	10	13	03	07	11	15	04	08	12

دومین نویسه‌ی متن عادی به جایگاه پنجم در متن رمز و سومین نویسه به نهمین جایگاه منتقل شده است و الی آخر. اگرچه نویسه‌ها جابجا شده‌اند، اما در این جابجایی الگوی زیر وجود دارد: (۰۲،۰۶،۱۰،۱۳)، (۰۳،۰۷،۱۱،۱۵) و (۰۸،۱۲) در هر قسمت تفاوت بین دو عدد مجاور ۴ است.

رمزهای ترانهش مبتنی بر کلید^۱

رمزهای بدون کلید با استفاده از نوشتن متن عادی به شیوه‌ای (به عنوان مثال ردیف به ردیف) و خواندن آن به شیوه‌ای دیگر (به عنوان مثال، ستون به ستون) نویسه‌ها را جابجا می‌کند. به منظور ایجاد متن رمز کامل، جابجایی را بر روی کل متن عادی انجام می‌دهند. روش دیگر تقسیم متن عادی به گروه‌هایی با اندازه‌های از پیش تعیین شده، که به آن‌ها بلوک گفته می‌شود و سپس استفاده از یک کلید برای جابجایی نویسه‌ها در هر بلوک به طور جداگانه می‌باشد.

مثال ۳-۲۵

آلیس می‌خواهد پیام "Enemy attacks tonight" را برای باب بفرستد. آلیس و باب توافق کرده‌اند که متن را به گروه‌های پنج نویسه‌ای تقسیم نموده و سپس نویسه‌های هر گروه را جابجا کنند. گروه‌بندی پس از افزودن یک نویسه‌ی اضافی به آخر پیام به منظور یک اندازه کردن گروه‌ها، به شکل زیر است.

Enemy attac kston ightz

کلید استفاده شده برای رمزنگاری و رمزگشایی یک کلید جابجایی است که نشان می‌دهد چگونه نویسه‌ها جابجا می‌شوند. برای این پیام، فرض کنید آلیس و باب از کلید زیر استفاده می‌کنند.

Encryption ↓	3	1	4	5	2	↑ Decryption
	1	2	3	4	5	

نویسه‌ی سوم در بلوک متن اولیه، نویسه‌ی اول در بلوک متن رمز می‌شود. نویسه‌ی اول بلوک متن اولیه، دومین نویسه‌ی بلوک متن رمز می‌شود و الی آخر. نتیجه‌ی جابجایی می‌شود:

EEMYN TAACT TKONS HITZG

=====

¹ Keyed Transposition Ciphers

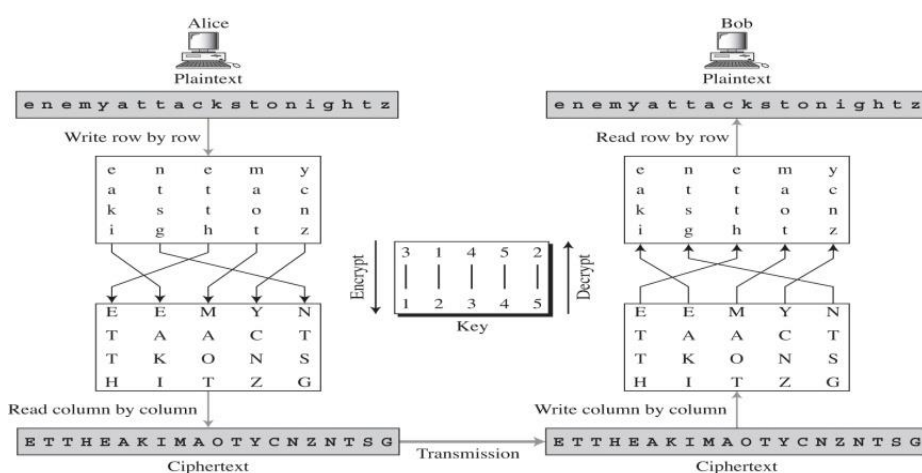
آلیس متن رمز "EEMYNTAACTTKONSHITZG" را برای باب ارسال می‌کند. باب متن رمز را به گروه‌های ۵ نویسه‌ای تقسیم می‌کند و با استفاده از ترتیب وارون، متن عادی را به دست می‌آورد.

ترکیب دو رویکرد

رمزهای ترانهش جدیدتر، برای دستیابی به پراکندگی بهتر حروف متن عادی، دو رویکرد را باهم ترکیب می‌کند. نخست، متن در یک جدول ردیف به ردیف نوشته می‌شود. دوم به وسیله مرتب نمودن مجدد ستون‌ها، جابجایی را انجام می‌دهند. سوم، جدول جدید، ستون به ستون خوانده می‌شود. مراحل اول و سوم، دوباره چینی کلی بدون کلید را فراهم می‌کند و مرحله دوم چینش مجدد بلوک کلیددار می‌باشد. معمولاً این‌گونه رمزها را رمزهای ترانهش ستونی مبتنی بر کلید^۱ یا فقط رمزهای ترانهش ستونی^۲ می‌نامند.

مثال ۳-۲۶

فرض کنید آلیس دوباره پیام مثال ۳-۲۵ را این بار با استفاده از رویکرد ترکیبی، رمزنگاری می‌نماید. در شکل ۳-۲۱، این رمزنگاری و رمزگشایی را مشاهده می‌کنید.



شکل ۳-۲۱: مثال ۳-۲۶

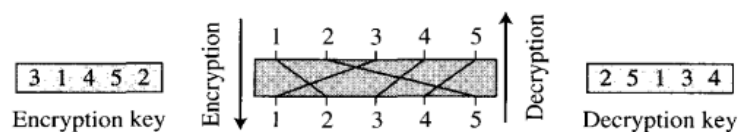
آلیس با نوشتن ردیف به ردیف متن عادی، اولین جدول را ایجاد می‌کند. مانند مثال پیش، با استفاده از همان کلید ستون‌ها جابجا می‌شوند. با خواندن جدول دوم به صورت ستون به ستون، متن رمز تولید می‌شود. باب هر سه مرحله را به صورت وارون انجام می‌دهد. او متن رمز را ستون به ستون در جدول اول می‌نویسد، ستون‌ها را جابجا می‌کند و سپس جدول دوم را ردیف به ردیف می‌خواند.

¹ Keyed Columnar Transposition Ciphers

² Columnar Transposition Ciphers

کلیدها

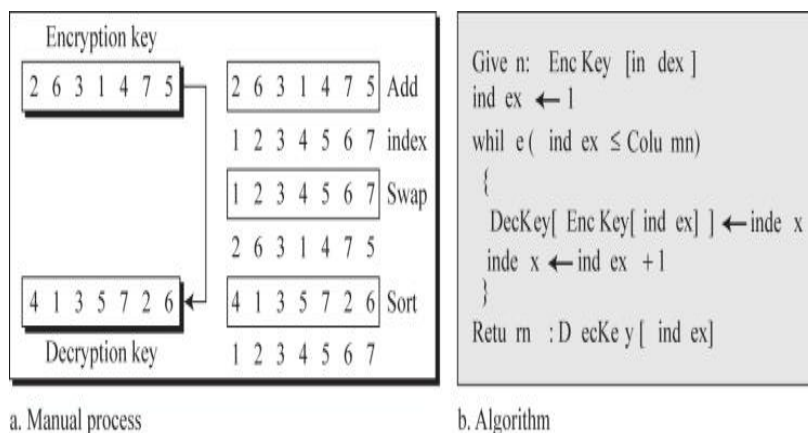
در مثال ۳-۲۷ تنها یک کلید در رمزنگاری و رمزگشایی برای تغییر ستون بکار رفته است. حرکت به سمت پایین برای رمزنگاری و حرکت به بالا برای رمزگشایی. ایجاد دو کلید از این نوع برای نمایش گرافیکی رایج است: یکی برای رمزنگاری و دیگری برای هدایت مسیر (جهت حرکت). کلیدها در جدولی با یک ورودی برای هر ستون، ذخیره می‌شوند. ورودی، شماره‌ی ستون منبع را نشان می‌دهد، شماره ستون مقصد را می‌توان از موقعیت (جایگاه) ورودی متوجه شد. شکل ۳-۲۲ نشان می‌دهد که چگونه دو جدول را می‌توان از ترسیم گرافیکی کلید ساخت.



شکل ۳-۲۲: کلید رمزنگاری / رمزگشایی در رمزهای ترانهش

کلید رمزنگاری (۳۱۴۵۲) است. ورودی اول نشان می‌دهد که ستون ۳ منبع و ستون ۱ موقعیت شاخص ورودی در مقصد است. کلید رمزگشایی (۲۵۱۳۴) است. اولین ورودی نشان می‌دهد که ستون ۲ در منبع، ستون ۱ در مقصد خواهد بود.

چگونه می‌توان کلید رمزگشایی را در صورتی که کلید رمزنگاری مشخص است، تولید نمود یا برعکس؟ همان‌گونه که در شکل ۳-۲۳ نشان داده شده است، می‌توان این مراحل را در چند مرحله به صورت دستی انجام داد. ابتدا شاخص‌ها را به جدول کلید اضافه کنید، سپس محتوا و شاخص‌ها را مبادله نموده و بالآخره زوج‌ها را بر اساس شاخص منظم و دسته‌بندی کنید.



شکل ۳-۲۳: وارون کلید در رمز ترانهش

استفاده از ماتریس‌ها

برای نشان دادن مراحل رمزنگاری/رمزگشایی در رمز ترانهش، می‌توانیم از ماتریس‌ها استفاده کنیم. متن عادی و متن رمز، ماتریس‌های $l \times m$ هستند که مقادیر عددی نویسه‌ها را نشان می‌دهند. کلیدها هم ماتریس‌های مربع به اندازه $m \times m$ می‌باشند. در ماتریس جایگشتی، هر سطر و ستون دقیقاً یک مقدار ۱ و بقیه مقادیرشان صفر هستند. با ضرب ماتریس متن عادی در ماتریس کلید به منظور به دست آوردن ماتریس متن رمز، رمزنگاری انجام می‌شود و با ضرب متن رمز در وارون ماتریس کلید، ماتریس متن عادی به دست می‌آید و متن رمزگشایی می‌شود. نکته جالب این است که در این حالت، ماتریس رمزگشایی، وارون ماتریس رمزنگاری است. با این وجود لازم نیست ماتریس را وارون نمود، بلکه می‌توان ماتریس کلید رمزنگاری را به سادگی جابجا کرد (جابجا کردن سطرها و ستون‌ها) و از این راه ماتریس کلید رمزگشایی را به دست آورد.

مثال ۳-۲۷

شکل ۳-۲۴ مراحل رمزنگاری را نشان می‌دهد. با ضرب ماتریس متن عادی 4×5 در کلید رمزنگاری 5×5 ، ماتریس متن رمز 4×5 به دست می‌آید. استفاده درست از ماتریس مستلزم تغییر نویسه‌های مثال ۳-۲۷ به مقادیر عددی آن‌ها (از ۰ تا ۲۵) می‌باشد. توجه داشته باشید که ضرب ماتریس، فقط جابجایی ستونی ترانهش را مهیا می‌کند. برای خواندن و نوشتن در ماتریس، باید سایر الگوریتم‌ها را به کار برد.

$$\begin{array}{c}
 \begin{matrix} 3 & 1 & 4 & 5 & 2 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{matrix} \\
 \begin{bmatrix} 04 & 13 & 04 & 12 & 24 \\ 00 & 19 & 19 & 00 & 02 \\ 10 & 18 & 19 & 14 & 13 \\ 08 & 06 & 07 & 19 & 25 \end{bmatrix} \times \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 04 & 04 & 12 & 24 & 13 \\ 19 & 00 & 00 & 02 & 19 \\ 19 & 10 & 14 & 13 & 18 \\ 07 & 08 & 19 & 25 & 06 \end{bmatrix} \\
 \text{Plaintext} \qquad \qquad \text{Encryption key} \qquad \qquad \text{Ciphertext}
 \end{array}$$

شکل ۳-۲۴: نمایش کلید به عنوان ماتریسی در رمز ترانهش

کشف رمز در رمزهای ترانهش^۱

رمزهای ترانهش در برابر انواع گوناگونی از حملات به شیوه‌ی حمله به متن اولیه، آسیب‌پذیرند.

^۱ Cryptanalysis of Transposition Ciphers

حمله به شیوه آماری

رمز ترانهش، تعداد تکرار حروف در متن رمز را تغییر نمی‌دهد، فقط ترتیب حروف را عوض می‌کند؛ بنابراین، اولین حمله‌ای که می‌تواند انجام شود، تجزیه و تحلیل تعداد تکرار تک حرفی است. اگر متن رمز به اندازه‌ی کافی طولانی باشد، این روش می‌تواند مفید واقع شود. این نوع حمله را پیشتر دیده‌ایم. با این وجود، رمزهای ترانهش تعداد تکرار دیاگرامی دو حرفی و سه حرفی را حفظ نمی‌کند؛ به این معنی که رمزشکن نمی‌تواند از این ابزار برای رمزگشایی استفاده کند، در واقع اگر رمز تعداد تکرار دیاگرام دو حرفی و سه حرفی را حفظ نکند، اما تعداد تکرار تک حرفی را نگه دارد، احتمال دارد که رمز، یک رمز ترانهش باشد.

حمله به روش آزمودن کلیه کلیدها

رمزشکن می‌تواند تمام کلیدهای ممکن را برای رمزگشایی پیام آزمایش کند. با این وجود، تعداد کلیدها می‌تواند بسیار زیاد باشد. (۱!، ۲!، ۳!، ...، $n!$) طول متن رمز است. یک روش بهتر، حدس زدن تعداد ستون‌هاست. رمزشکن می‌داند که تعداد ستون‌ها، بر n بخش‌پذیر است. به عنوان مثال، اگر طول رمز ۲۰ نویسه باشد، در آن صورت $20 = 1 \times 2 \times 4 \times 5$ می‌باشد؛ بدین معنی که تعداد ستون‌ها می‌تواند ترکیبی از این فاکتورها باشد (۱، ۲، ۴، ۵، ۱۰، ۲۰). با این حال، اولین (تنها یک ستون) قابل توجه نیست و آخرین (تنها یک ردیف) غیر محتمل است.

مثال ۲۸-۳

فرض کنید رمزشکن متن رمز "EEMYNTAACTTKONSHITZG" را دریافت کرده است. طول پیام $20 = n$ بدین معناست که تعداد ستون‌ها می‌تواند ۱، ۲، ۴، ۵، ۱۰، ۲۰ باشد. او اولین مقدار را نادیده می‌گیرد، زیرا به این معنی است که فقط یک ستون وجود دارد و جابجایی صورت نگرفته است. (a) اگر تعداد ستون‌ها ۲ باشد. تنها دو جابجایی (۱، ۲) و (۲، ۱) امکان‌پذیر است. اولین مورد به معنی عدم وجود جابجایی است. او دومی را آزمایش می‌کند. او متن رمز را به واحدهای دو نویسه‌ای تقسیم می‌کند: "EE MY NT AA CT TK ON SH IT ZG"، سپس هر یک از دو

حرف را جابجا می‌کند. چیزی که به دست می‌آید این است: "EE YM TN AA TC KT NO HSTIGZ" که معنی ندارد.

(b) اگر تعداد ستون‌ها ۴ باشد، جابجایی $24=4!$ می‌باشد. اولین مورد (۴ ۳ ۲ ۱) یعنی هیچ جابجایی وجود ندارد. رمز شکن باید سایر احتمالات را آزمایش کند. پس از بررسی تمام ۲۳ احتمال دیگر، متن عادی قابل فهمی را به دست نمی‌آورد.

(c) اگر تعداد ستون‌ها ۵ باشد، جابجایی $120=5!$ می‌باشد. اولین مورد (۵ ۴ ۳ ۲ ۱) به معنی عدم وجود جابجایی است. رمز شکن باید بقیه احتمالات را آزمایش کند. حاصل جابجایی (۴ ۳ ۱ ۵ ۲)، متن عادی "enemyattackstonightz" می‌باشد که پس از حذف حرف اضافه شده Z و افزودن فاصله‌ها معنا پیدا می‌کند.

حمله‌ی مبتنی بر الگو

حمله‌ی دیگر به رمز ترانهش می‌تواند حمله مبتنی بر الگو باشد. متن رمز ایجاد شده در رمز ترانهش بر پایه‌ی کلید دارای الگوهای تکراری می‌باشد. در شکل زیر، مکانی که هر نویسه در متن رمز مثال ۲۸-۳ از آن سرچشمه گرفته را نشان می‌دهد.

۰۳ ۰۸ ۱۳ ۱۸ ۰۱ ۰۶ ۱۱ ۱۶ ۰۴ ۰۹ ۱۴ ۱۹ ۰۵ ۱۰ ۱۵ ۲۰ ۰۲ ۰۷ ۱۲ ۱۷

اولین نویسه‌ی متن رمز، از سومین نویسه‌ی متن عادی، دومین نویسه‌ی متن رمز از هشتمین نویسه‌ی متن عادی، بیستمین نویسه‌ی در متن رمز از هفدهمین نویسه‌ی در متن عادی آمده است و الی آخر. الگویی در لیست فوق وجود دارد. پنج گروه داریم: (۱۸، ۱۳، ۸، ۳)، (۱۶، ۱۱، ۶، ۱)، (۱۹، ۱۴، ۹، ۴)، (۲۰، ۱۵، ۱۰، ۵) و (۱۷، ۱۲، ۷، ۲). در تمام گروه‌ها تفاوت بین دو عدد کنار هم ۵ است. رمز شکن می‌تواند از این نظم برای شکستن رمز استفاده کند. اگر او تعداد ستون‌ها (که در این مورد ۵ است) را بداند یا بتواند آن را حدس بزند، می‌تواند متن رمز را در گروه‌های ۴ نویسه‌ای قرار داده و با جابجا کردن گروه‌ها، متن عادی را بیابد.

رمزهای ترانهش دوگانه^۱

=====

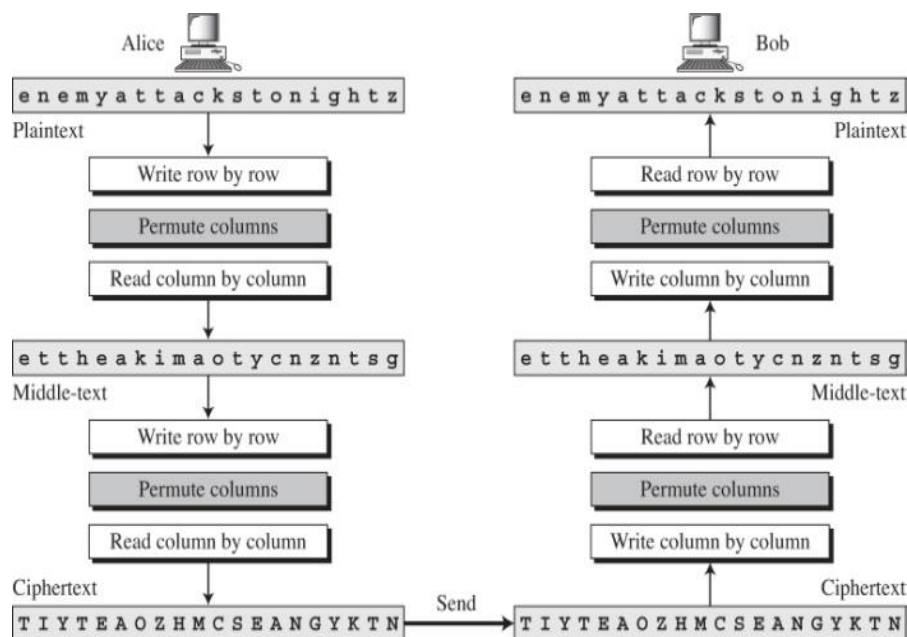
^۱ Double Transposition Ciphers

رمزهای ترانهش دوگانه می‌تواند کار رمزگشایی را مشکل کند. نمونه‌ای از چنین رمزی می‌تواند رمزی باشد که برای رمزنگاری و رمزگشایی در مثال ۳-۲۶ دو بار الگوریتم را تکرار کرده است. می‌توان در هر مرحله از یک کلید استفاده کرد، اما معمولاً در تمام مراحل یک کلید یگانه بکار می‌رود.

مثال ۳-۲۹

اجازه دهید مثال ۳-۲۶ را با استفاده از ترانهش دوگانه تکرار کنیم. شکل ۳-۲۵ فرایند کار را

نشان می‌دهد.



شکل ۳-۲۵: رمز ترانهش دوگانه

اگرچه رمزشکن هنوز هم می‌تواند از حمله به شیوه تعداد تکرار حروف تک حرفی استفاده کند، اما اکنون انجام حمله‌ی مبتنی بر الگو بسیار سخت‌تر است. تجزیه و تحلیل الگوی متن نشان می‌دهد:

۰۲ ۱۹ ۱۱ ۰۸ ۱۷ ۱۵ ۰۹ ۰۱ ۱۲ ۱۰ ۰۴ ۱۸ ۲۰ ۰۶ ۰۳ ۰۷ ۰۵ ۱۶ ۱۳

با مقایسه مجموعه فوق و نتیجه در مثال ۳-۲۸ مشاهده می‌کنیم که الگوی تکراری وجود ندارد.

ترانهش دوگانه، نظم و قاعده‌هایی که پیشتر دیدیم را برهم می‌زند.

۳-۴- رمزهای جریانی و بلوکی^۱

رمزهای متقارن به دو دسته‌ی گسترده تقسیم می‌شوند: رمزهای جریانی و رمزهای بلوکی. اگرچه معمولاً این دسته‌بندی برای رمزهای مدرن بکار می‌روند، اما برای رمزهای سنتی نیز کاربرد دارد.

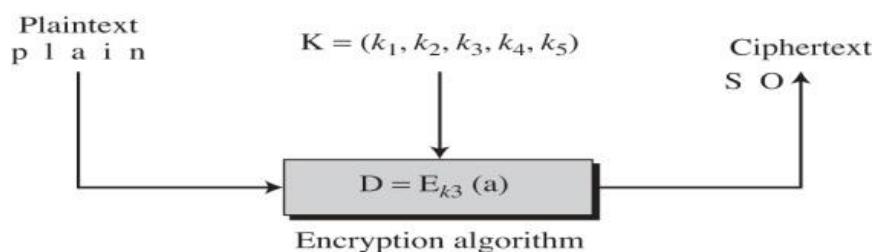
رمزهای جریانی^۲

در رمزهای جریانی، رمزگشایی یک سمبل (همچون یک نویسه یا یک بیت) تک‌تک انجام می‌شود. در این شیوه‌ی رمزنگاری یک جریان متن عادی، یک جریان متن رمز و یک جریان کلید داریم. جریان متن عادی را P جریان متن رمز را C و جریان کلید را K می‌نامیم.

$$P = P_1, P_2, P_3, \dots \quad C = C_1, C_2, C_3, \dots \quad K = (K_1, K_2, K_3, \dots)$$

$$C_1 = E_{K_1}(P_1) \quad C_2 = E_{K_2}(P_2) \quad C_3 = E_{K_3}(P_3)$$

شکل ۳-۲۶ ایده‌ی کلی رمز جریانی را نشان می‌دهد. نویسه‌های متن عادی تک‌تک به الگوریتم وارد می‌شوند و نویسه‌های متن رمز نیز تک‌تک تولید می‌شوند. می‌توان جریان کلید را به روش‌های گوناگونی تولید کرد و می‌تواند جریانی از مقادیر از پیش تعیین شده باشد یا اینکه آن را همزمان با رمزنگاری از الگوریتم ایجاد کرد. ممکن است این مقادیر به نویسه‌های متن عادی یا متن رمز بستگی داشته باشند. همچنین ممکن است مقادیر جریان کلید، به مقادیر کلید قبلی نیز بستگی داشته باشد.



شکل ۳-۲۶: رمز جریانی

شکل ۳-۲۶ لحظه‌ای را نشان می‌دهد که نویسه سوم در جریان متن عادی با استفاده از سومین مقدار جریان کلید رمزنگاری می‌شود. نتیجه، سومین نویسه‌ی جریان متن رمز تولید می‌شود.

مثال ۳-۳۰

¹ Stream And Block Ciphers

² Stream Ciphers

رمزهای افزایشی را می‌توان در دسته رمزهای جریانی دسته‌بندی کرد که در آن جریان کلید، مقادیر حاصل از تکرار کلید است؛ به عبارت دیگر، جریان کلید، جریان از پیش تعیین شده‌ی کلیدها یا $K = (k, k, \dots, k)$ تلقی می‌شود.

با این وجود از آنجا که در این رمز جریان کلید به صورت مستقل ایجاد می‌شود، هر نویسه در متن رمز فقط به نویسه‌ی متناظر آن در متن عادی بستگی دارد.

مثال ۳-۳۱

رمزهای جایگزین تک الفبایی - که در این فصل مورد بحث قرار گرفت - نیز رمز جریانی است. با این وجود، هر مقدار از جریان کلید در این حالت، عبارت است از نگاشت نویسه‌ی متن عادی با نویسه متن رمز متناظر با آن در جدول نگاشت.

مثال ۳-۳۲

بر اساس تعریف، رمزهای *vigenere* نیز رمزهای جریانی هستند. در این حالت، جریان کلید عبارت است از تکرار مقادیر m ، به گونه‌ای که m اندازه‌ی کلید باشد. به عبارت دیگر

$$K = (k_1, k_2, \dots, k_m, k_1, k_2, \dots, k_m, \dots)$$

مثال ۳-۳۳

می‌توانیم پیمانه‌ای برای تقسیم رمزهای جریانی بر اساس جریان کلیدشان وضع کنیم. می‌توانیم ادعا کنیم که رمز جریان، رمز تک الفبایی است به شرطی که مقدار K_i به جایگاه (موقعیت) نویسه‌ی متن عادی در جریان متن عادی بستگی نداشته باشد، در غیر این صورت رمز چند الفبایی است.

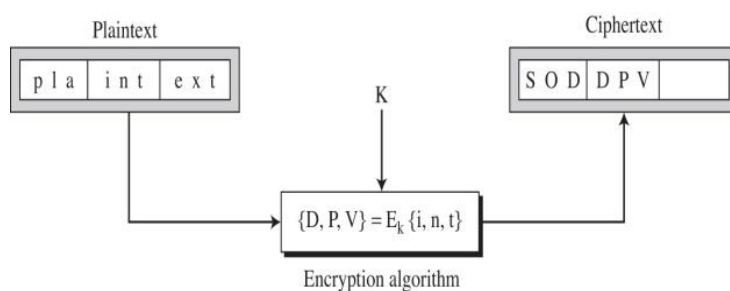
✱ رمزهای افزایشی قطعاً تک الفبایی هستند زیرا K_i در جریان کلید ثابت است و به جایگاه (موقعیت) نویسه در متن عادی بستگی ندارد.

✱ رمزهای جایگزینی تک الفبایی، قطعاً تک الفبایی‌اند زیرا K_i به جایگاه (موقعیت) نویسه‌ی متناظر آن در جریان متن عادی بستگی ندارد و فقط به مقدار نویسه‌ی متن عادی بستگی دارد.

✱ رمزهای *vigenere* رمزهای چند الفبایی‌اند زیرا کلید K_i قطعاً به جایگاه (موقعیت) نویسه متن عادی بستگی دارد. با این وجود، این وابستگی چرخه‌ای است. کلید دو نویسه‌ی m با جایگاه‌های (موقعیت‌های) جداگانه، یکسان است.

رمزهای بلوکی^۱

در رمزهای بلوکی، گروهی از سمبل‌های متن عادی به اندازه m ($m > 1$) که با همدیگر رمزنگاری شده‌اند، یک گروه متن رمز با همان اندازه را تولید می‌کنند. بر اساس این تعریف، در رمز بلوکی، تنها از یک کلید برای رمزنگاری کل بلوک استفاده می‌شود؛ حتی اگر کلید از مقادیر گوناگونی ساخته شده باشد. شکل ۳-۲۷ مفهوم رمز بلوکی را نشان می‌دهد.



شکل ۳-۲۷: رمز بلوکی

در رمز بلوکی، یک بلوک متن رمز به تمام بلوک متن عادی بستگی دارد.

مثال ۳-۳۴

رمزهای **playfair** رمزهای بلوکی هستند. اندازه بلوک $m = 2$ است و دو نویسه باهم رمزنگاری می‌شوند.

مثال ۳-۳۵

رمزهای **Hill** رمزهای بلوکی هستند. بلوکی از متن عادی به اندازه ۲ یا بیشتر با استفاده از یک کلید (یک ماتریس) باهم رمزنگاری می‌شود. در این رمزها، مقدار هر نویسه در متن رمز به مقادیر نویسه‌های متن عادی بستگی دارد. اگرچه کلید از مقادیر $m \times m$ ساخته شده است، اما به عنوان یک کلید یکتا در نظر گرفته می‌شود.

مثال ۳-۳۶

از تعریف رمز بلوکی چنین برمی‌آید که هر رمز بلوکی، رمز چند الفبایی است، زیرا هر نویسه در یک بلوک متن رمز به تمام نویسه‌های موجود در بلوک متن عادی بستگی دارد.

=====

¹ Block Ciphers

ترکیب

در عمل، بلوک‌های متن عادی تک‌تک رمزنگاری می‌شوند، اما از جریانی از کلیدها برای رمزنگاری کل پیام، بلوک به بلوک، استفاده می‌کنند؛ به عبارت دیگر، وقتی که به بلوک‌ها جداگانه نگاه می‌کنیم، رمز، یک رمز بلوکی است، اما وقتی به کل متن و به هر بلوک به عنوان یک واحد یگانه نگاه می‌کنیم، رمز، یک رمز جریانی است. هر بلوک از کلید مجزایی که ممکن است از پیش یا در خلال مراحل رمزنگاری ایجاد شده باشد استفاده می‌کند. مثال‌هایی در این مورد را می‌توان در فصول بعدی یافت.

۳-۵- وب سایت‌ها پیشنهادی

وب سایت‌های زیر اطلاعات بیشتری در مورد موضوعات بحث شده در این فصل را به شما ارائه می‌دهد.

<http://www.cryptogram.org>

<http://www.cdt.org/crypto/>

<http://www.cacr.math.uwaterloo.ca/>

<http://www.acc.stevens.edu/crypto.php>

<http://www.crypto.com/>

<http://theory.lcs.mit.edu/~rivest/crypto-security.html>

<http://www.trincoll.edu/depts/cpsc/cryptography/substitution.html>

<http://hem.passagen.se/tan01/transpo.html>

<http://www.strangehorizons.com/2001/20011008/steganography.shtml>

۳-۶- چکیده

- * رمزنویسی کلید متقارن، از یک کلید برای رمزگشایی و رمزنگاری استفاده می‌کند. علاوه بر این، الگوریتم‌های رمزنگاری و رمزگشایی وارون یکدیگرند.
- * متن اصلی را متن عادی و متنی را که از طریق کانال ارسال می‌کنیم متن رمز می‌نامند. به منظور ساختن متن رمز از متن عادی، الگوریتم رمزنگاری با کلید سری مشترک بکار می‌رود. برای بازسازی متن عادی از متن رمز، الگوریتم رمزگشایی و همان کلید سری مورد استفاده قرار می‌گیرد. به الگوریتم‌های رمزنگاری و رمزگشایی، رمزنویسی می‌گوییم.
- * بر اساس اصل کریشهف، همیشه باید فرض کرد که رقیب الگوریتم رمزنگاری/رمزگشایی را می‌داند. باید تنها از طریق سری بودن کلید در برابر حمله به رمز مقاومت کرد.
- * کشف رمز، علم و هنر شکستن رمز است. چهار نوع حمله کشف رمز رایج وجود دارد: دانستن فقط متن رمزی، متن عادی شناخته شده، متن عادی انتخاب شده و متن رمز انتخاب شده.
- * رمزهای کلید متقارن سستی را می‌توان به دو دسته گسترده تقسیم کرد: رمزهای جایگزین و رمزهای ترانهش. رمز جایگزین، نویسه‌ای را جایگزین نویسه‌ی دیگر می‌کند. رمز ترانهش، سمبل‌ها را دوباره چینش می‌کند.
- * رمزهای جایگزین را می‌توان به دو گروه بزرگ تقسیم کرد: رمزهای تک الفبایی و رمزهای چند الفبایی. رابطه‌ی بین یک نویسه در متن عادی و نویسه‌های متن رمز، یک به یک است. در جایگزینی چند الفبایی، رابطه‌ی بین یک نویسه در متن عادی و نویسه‌های متن رمز، یک به چند است.
- * رمزهای تک الفبایی شامل رمزهای: افزایشی، ضربی، تکراری و رمزهای جایگزین تک الفبایی می‌باشند.
- * رمزهای چند الفبایی شامل کلید خودکار، Hill، vigenere، playfair و رمز یک‌بار مصرف، روتور و رمزهای انگیم می‌باشد.

* رمزهای ترانهش شامل رمزهای ترانهش بدون کلید، کلیددار و دوگانه می‌باشد.

* رمزهای متقارن را می‌توان به دو دسته گسترده تقسیم کرد: رمزهای جریانی و رمزهای بلوکی. در رمزهای جریانی رمزنگاری و رمزگشایی سمبل‌ها تک‌تک انجام می‌شود. در رمز بلوکی، سمبل‌های موجود در یک بلوک با همدیگر رمزنگاری می‌شوند. در عمل، بلوک‌های متن عادی به صورت تکی رمزنگاری می‌شود اما در جریانی از کلیدها برای رمزنگاری کل پیام، بلوک به بلوک استفاده می‌کنند.

۳-۷- مجموعه تمرین‌ها

پرسش‌های دوره‌ای

- (۱) رمز کلید متقارن را تعریف کنید.
- (۲) تفاوت بین رمز جایگزین و رمز ترانهش را بیان کنید.
- (۳) تفاوت بین رمز تک الفبایی و رمز چند الفبایی را بیان کنید.
- (۴) تفاوت بین رمز جریانی و رمز بلوکی را بیان کنید.
- (۵) آیا تمام رمزهای جریانی، تک الفبایی‌اند؟ توضیح دهید.
- (۶) آیا تمام رمزهای بلوکی، چند الفبایی‌اند؟ توضیح دهید.
- (۷) سه رمز تک الفبایی را نام ببرید.
- (۸) سه رمز چند الفبایی را نام ببرید.
- (۹) دو رمز ترانهش را نام ببرید.
- (۱۰) چهار نوع حمله رمزگشایی را نام ببرید.

تمرینات

- (۱۱) یک باشگاه خصوصی کوچک فقط ۱۰۰ عضو دارد. به پرسش‌های زیر پاسخ دهید:
 - (a) اگر اعضای باشگاه بخواهند برای یکدیگر پیام‌های سری بفرستند، چه تعداد کلید سری لازم است؟
 - (b) اگر همه به رئیس باشگاه اعتماد داشته باشند، چه تعداد کلید سری لازم است؟ اگر شخصی بخواهد پیامی را برای عضو دیگری ارسال کند، ابتدا آن را برای رئیس می‌فرستند، سپس رئیس آن را برای سایر اعضا ارسال می‌کند.
 - (c) اگر رئیس تصمیم بگیرد که هر دو نفری که می‌خواهند باهم ارتباط داشته باشند باید ابتدا با او ارتباط پیدا کنند، چه تعداد کلید لازم است؟ سپس رئیس کلید موقتی را برای استفاده بین آن دو ایجاد می‌کند. رمزنگاری شده و برای هر دو عضو ارسال می‌گردد.

(۱۲) چند باستان‌شناس، نسخه‌های نوشته شده به زبانی ناآشنا را یافتند. آن‌ها بعداً جدول کوچکی را در همان مکان یافتند که روی آن جمله‌ای به همان زبان به همراه ترجمه‌ی یونانی آن نوشته شده بود. با استفاده از جدول، آن‌ها می‌توانستند نسخه اصلی را بخوانند. باستان‌شناسان از چه نوع رمزنگاری استفاده کردند؟

(۱۳) آلیس فقط می‌تواند روی رایانه‌اش از رمز افزایشی برای ارسال پیام به دوستش استفاده کند. او فکر می‌کند که اگر پیام را دو بار رمزنگاری کند و هر بار از کلید متفاوتی استفاده نماید، پیام مطمئن‌تر خواهد بود. آیا حق با اوست؟ پاسخ خود را توضیح دهید.

(۱۴) آلیس پیام طولانی برای ارسال دارد و از رمز جایگزینی تک الفبایی استفاده می‌کند. او فکر می‌کند اگر پیام را فشرده کند، ممکن است متن در برابر حمله تعداد تکرار تک حرفی که توسط رمزشکن انجام می‌شود محفوظ بماند. آیا فشرده‌سازی کمک می‌کند؟ باید متن را پیش از رمزنگاری فشرده نماید یا پس از آن؟ پاسخ خود را توضیح دهید.

(۱۵) معمولاً آلیس برای رمزنگاری متن عادی هم از حروف (a تا z) و هم از اعداد (۰ تا ۹) استفاده می‌کند.

(a) اگر از رمز افزایشی استفاده کند دامنه کلید چقدر است؟ پیمانه چقدر است؟

(b) اگر از رمز ضربی استفاده کند دامنه کلید چقدر است؟ پیمانه چقدر است؟

(c) اگر از رمز تکراری استفاده کند دامنه کلید چقدر است؟ پیمانه چقدر است؟

(۱۶) فرض کنید فاصله، نقطه و علامت سوال به منظور افزایش دامنه کلید رمزهای اولیه به متن اولیه اضافه شود.

(a) اگر رمز افزایشی استفاده شود دامنه کلید چقدر است؟

(b) اگر رمز ضربی استفاده شود دامنه کلید چقدر است؟

(c) اگر رمز تکراری استفاده شود دامنه کلید چقدر است؟

(۱۷) آلیس و باب تصمیم گرفته‌اند اصل کریشف را نادیده بگیرند و نوع رمزی را که استفاده می‌کنند پنهان نمایند.

(a) رمزشکن چگونه می‌تواند تصمیم بگیرد که آیا رمز جایگزین مورد استفاده

قرارگرفته یا رمز ترانهش؟

(b) اگر رمز شکن بداند که رمز مورد استفاده رمز جایگزین است، چگونه می تواند

تصمیم بگیرد که این رمز، رمز افزایشی، ضربی یا تکراری است؟

(c) اگر رمز شکن بداند که رمز مورد استفاده رمز ترانهش است، چگونه می تواند

اندازه ی قسمت m را بداند؟

(۱۸) در هر یک از رمزهای زیر، اگر فقط یک نویسه در متن عادی عوض شود، بیش ترین تعداد

نویسه هایی که در متن رمز تغییر خواهند کرد چقدر است؟

(a) افزایشی

(b) ضربی

(c) تکراری

(d) Vigenere

(e) کلید خودکار

(f) رمز یک بار مصرف

(g) روتور

(h) انگیما

(۱۹) در هر یک از رمزهای زیر، اگر فقط یک نویسه در متن عادی عوض شود، حداکثر تعداد

نویسه هایی که در متن رمز تغییر خواهند کرد چقدر است؟

(a) ترانهش

(b) ترانهش دوگانه

(c) Playfair

(۲۰) هر یک از رمزهای زیر جریانی است یا بلوکی؟ پاسخ خود را توضیح دهید.

(a) Playfair

(b) کلید خودکار

(c) رمز یک بار مصرف

(d) روتور

(e) انگیما

(۲۱) با استفاده از رمزهای زیر پیام “this is an exercise” را رمزنگاری کنید. فاصله بین کلمات را نادیده بگیرید. برای به دست آوردن متن عادی، پیام را رمزگشایی کنید.

(a) رمز افزایشی با کلید مساوی ۲۰.

(b) رمز ضربی با کلید مساوی ۱۵.

(c) رمز تکراری با کلید مساوی (۱۵، ۲۰).

(۲۲) با استفاده از رمزهای زیر پیام “that house is being sold tonight” را رمزنگاری کنید. فاصله بین کلمات را نادیده بگیرید. برای به دست آوردن متن عادی پیام را رمزگشایی کنید.

(a) رمز vigenere با کلید “dollars”.

(b) رمز کلید خودکار با کلید مساوی ۷.

(c) رمز playfair با کلید ایجاد شده از متن (به شکل ۳-۱۳ نگاه کنید).

(۲۳) برای رمزنگاری پیام “life is full of surprises” از رمز vigenere با کلمه کلید “HEALTH” استفاده کنید.

(۲۴) از رمز playfair برای رمزنویسی پیام “the key is hidden under the door pad” استفاده کنید. کلمه رمز را می‌توان با پرکردن ردیف اول و قسمتی از ردیف دوم با کلمه “GUIDANCE” و پرکردن بقیه ماتریس با سایر حروف الفبا به دست آورد.

(۲۵) از رمز Hill برای رمزنویسی پیام “we live in an insecure world” استفاده کنید. کلید زیر را به کار ببرید.

$$K = \begin{bmatrix} 03 & 02 \\ 05 & 07 \end{bmatrix}$$

(۲۶) آقای جان در حال مطالعه‌ی کتاب اسرارآمیزی حاوی رمزنگاری است. در قسمتی از این کتاب، نویسنده‌ی متن رمز “CIW” را عنوان کرده و دو پاراگراف بعد به خواننده می‌گوید که این رمز یک رمز ترانهش است و متن عادی آن “yes” می‌باشد. در فصل بعد قهرمان جدولی را در غار پیدا می‌کند که روی آن “XVIEWYWI” حک شده است. جان بلافاصله معنی متن رمز را می‌فهمد. در اینجا جان چه شیوه از رمزگشایی را به کار برده است؟ متن عادی چیست؟

(۲۷) رمزشکن به صورت مخفیانه به کامپیوتر آلیس دسترسی پیدا می‌کند و با استفاده از رمز او "abcdefghij" را تایپ می‌کند. صفحه نمایش، "CABDEHFGIJ" را نشان می‌دهد. اگر رمزشکن بداند که آلیس از رمز ترانهش مبتنی بر کلید استفاده می‌کند، به پرسش‌های زیر پاسخ دهید:

(a) رمزشکن چه نوع حمله‌ای را انجام داده است؟

(b) اندازه‌ی کلید جابجایی چقدر است؟

(۲۸) از حمله‌ی آزمودن کلمه‌ی کلیدها برای رمزگشایی پیام زیر که آلیس با استفاده از رمز افزایشی آن را رمزنویسی کرده است استفاده کنید. فرض کنید آلیس همیشه از کلیدی که به تولدش، یعنی سیزدهم نزدیک است استفاده می‌کند.

NCJAEZRCLASJLYODEPRLYZRCLASJLCPEHZDZQLNZTY

(۲۹) از حمله‌ی آزمودن کلمه‌ی کلیدها برای رمزگشایی پیام زیر استفاده کنید. فرض کنید می‌دانید که رمزنگاری به شیوه‌ی رمز تکراری است و اینکه متن عادی "ab" به صورت "GL" رمزنویسی می‌شود.

XPALASXYFGFUKPXUSOGEUTKCDGFXANMGNVS

(۳۰) از شیوه رمزشکنی تعداد تکرار یک حرف برای رمزگشایی پیام زیر استفاده کنید. فرض کنید می‌دانید که این پیام به روش رمز جایگزینی تک الفبایی رمزنویسی شده است.

ONHOVEJHWOBEVGVOCBWHNUGBLHGBGR

(۳۱) فرض کنید که علائم نگارشی (نقطه، علامت سوال و فاصله) به الفبای رمزنگاری رمز Hill اضافه می‌شوند. پس، از ماتریس کلید 2×2 در Z_{26} می‌توان برای رمزنگاری و رمزگشایی استفاده کرد.

(a) کل تعداد ماتریس‌های ممکن را پیدا کنید.

(b) ثابت شده است که تعداد کل ماتریس‌های برگشت‌پذیر که N ، تعداد اندازه‌ی الفبا است برابر $(N^2 - N)(N^2 - 1)$ می‌باشد. با استفاده از این الفبا دامنه رمز Hill را به دست آورید.

(۳۲) از حمله تعداد تکرار یک حرف برای شکستن متن رمز زیر استفاده کنید. می‌دانید که این متن در رمز افزایشی تولید شده است.

OTWEWNGWCBPQABIZVQAPMLJGZWTTQVOBQUMAPMIDGZCABEQVB
MZLZIXMLAXZQVOQVLMMXAVWEIVLLZSNZWABJQZLWNLMTQOPBVIUMLG
WCBPAEQNBGTMTNBBPMVMABITIAKWCTLVBBQUMQBEPQTMQBIEIAQVUG
BZCAB

(۳۳) از تست Kasiski و حمله‌ی تعداد تکرار تک حرفی برای شکستن متن رمز زیر استفاده کنید. می‌دانید که این متن رمز با رمز **vigenere** تولید شده است.

MPYIGOB SRMIDBSYRDIKATXAILFDFKXTPPSNTTJIGTHDELTTXAI REIHSVO
BSMLUCFIOEPZIWACRFXICUVXVTOPXDLWPENDHPTSIDDBXWWTZPHNSOCL
OUMSNRCCVUUXZHHNWSVXAUHIKLTIMOICHTYPBHMHXGXHOLWPEWWW
WDALOCTSQZELT

(۳۴) کلید رمزنگاری در رمز ترانهش (۴، ۵، ۱، ۶، ۲، ۳) است. کلید رمزگشایی را پیدا کنید.

(۳۵) ماتریس کلید رمزنگاری رمز ترانهش با کلید (۴، ۵، ۱، ۶، ۲، ۳) را به دست آورید. ماتریس کلید رمزگشایی را نیز پیدا کنید.

(۳۶) متن عادی “letusmeetnow” و متن رمز متناظر آن “HBCDFNOPIKLB” است. می‌دانیم که الگوریتم رمز Hill می‌باشد، اما اندازه‌ی کلید را نمی‌دانیم. ماتریس کلید را به دست آورید.

(۳۷) رمزهای Hill و رمزهای ضربی بسیار شبیه هم هستند. رمزهای Hill، رمزهای بلوکی هستند که از ضرب ماتریس‌ها استفاده می‌کنند. رمزهای ضربی، رمزهای جریانی هستند که از ضرب اسکالر استفاده می‌کنند.

- (a) با استفاده از جمع ماتریس‌ها، رمز بلوکی تعریف کنید که شبیه به رمز افزایشی باشد.
- (b) با استفاده از ضرب و جمع ماتریس‌ها، رمز بلوکی تعریف کنید که شبیه به رمز تکراری باشد.

(۳۸) اجازه دهید رمز جریانی جدیدی تعریف کنیم. این رمز به شیوه‌ی تکرار است اما کلید به موقعیت نویسه‌ها در متن عادی بستگی دارد. اگر نویسه‌ی متن عادی‌ای که باید رمزنگاری شود، در موقعیت i باشد، می‌توانیم کلید را به شرح زیر پیدا کنیم:

(a) کلید ضربی عضو $(1 \bmod 12)$ در Z_{26}^* است.

(b) کلید افزایشی عضو $(1 \bmod 26)$ در Z_{26} است.

پیام cryptography is fun را با این روش رمزنگاری کنید.

(۳۹) فرض کنید برای رمز Hill متن عادی یک ماتریس همسانی ضربی (I) است. رابطه‌ی بین کلید و متن رمز را پیدا کنید. از نتیجه به دست آمده برای انجام حمله‌ی متن عادی انتخاب شده به رمز Hill استفاده کنید.

(۴۰) Abthash رمز رایج بین نویسندگان کتاب مقدس بود. در Abtash، "A" به صورت "Z" و "B" به صورت "Y" و الی آخر رمزنگاری می‌شد. فرض کنید حروف الفبا به دو نیمه تقسیم شده است و حروف موجود در نیمه اول به وسیله حروف موجود در نیمه دوم و برعکس رمزنگاری می‌شوند. نوع رمز و کلید را پیدا کنید.

(۴۱) در رمز playbius هر حرف به صورت دو عدد صحیح رمزنویسی می‌شود. کلید، مانند رمز playfair ماتریس 5×5 از نویسه‌هاست. متن عادی به شکل نویسه‌هایی در قالب ماتریس است. متن رمز، دو عدد صحیح (هر کدام بین ۱ و ۵) است که عدد سطر و ستون را نشان می‌دهد. پیام "an exercise" را با استفاده از رمز playbius و با کلید زیر رمزنویسی نمایید.

	1	2	3	4	5
1	z	q	p	f	e
2	y	r	o	g	d
3	x	s	n	h	c
4	w	t	m	i/j	b
5	v	u	l	k	a

فصل ۴

ریاضیات رمزنگاری

بخش دو: ساختارهای جبری

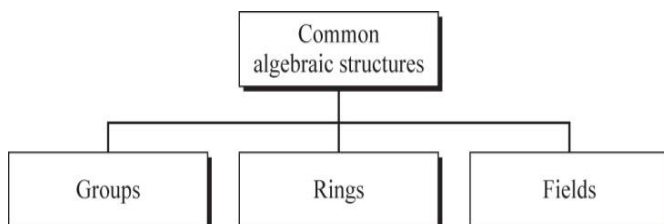
چشم انداز

هدف این فصل، آماده سازی شما برای چند فصل بعدی در زمینه ی رمزهای کلید متقارن مبتنی بر ساختارهای جبری است. این فصل اهداف زیر را دنبال می کند:

- ✱ مروری بر مفهوم ساختارهای جبری.
 - ✱ تعریف و ارائه ی نمونه هایی از گروه ها.
 - ✱ تعریف و ارائه ی نمونه هایی از حلقه ها.
 - ✱ تعریف و ارائه ی نمونه هایی از میدان ها.
 - ✱ پافشاری بر استفاده از میدان های متناهی از نوع $GF(2^n)$ ، چون که انجام اعمالی چون جمع، تفریق، ضرب و تقسیم را روی کلمات n بیتی در رمزهای پیشرفته امکان پذیر می کند.
- چند فصل بعدی در مورد رمزهای بلوکی کلید متقارن پیشرفته، که برخی از اعمال را روی کلمات n بیتی انجام می دهد، بحث می کنیم. درک، تجزیه و تحلیل این رمزها، نیازمند دانشی در زمینه جبر پیشرفته - که به آن ساختارهای جبری می گویند - می باشد. این فصل، نخست مروری بر موضوعات مرتبط با ساختارهای جبری دارد و سپس نشان می دهد که چگونه باید اعمالی نظیر جمع یا ضرب را روی کلمات n بیتی انجام داد.

۴-۱- ساختارهای جبری^۱

در فصل ۲ در مورد مجموعه‌هایی از اعداد، نظیر $Z, Z_n, Z_n^*, Z_p, Z_p^*$ صحبت کردیم. در رمزنگاری به مجموعه‌هایی از اعداد صحیح و اعمال خاصی که برای آن مجموعه‌ها تعریف شده است، نیاز داریم. ترکیب مجموعه و اعمالی که بر اعضای این مجموعه انجام می‌شود را «ساختار جبری» می‌نامند. در این فصل، سه ساختار رایج جبری را بررسی می‌کنیم: گروه‌ها، حلقه‌ها و میدان‌ها (شکل ۴-۱).



شکل ۴-۱: ساختارهای رایج جبری

گروه (G) مجموعه‌ای از اعضا با یک عمل دودویی « \bullet » است که چهار ویژگی را داراست. به گروه با ویژگی جابجایی، گروه آبدی گفته می‌شود؛ عملگر این گروه، علاوه بر چهار ویژگی لازم برای گروه‌ها، یک ویژگی دیگر، یعنی خاصیت جابجایی را هم دارد. چهار ویژگی برای گروه‌ها به علاوه خاصیت جابجایی، به صورت زیر تعریف می‌شوند:

* بسته بودن^۲: اگر a, b اعضای G باشند پس $c = a \bullet b$ نیز عضو G است؛ یعنی، نتیجه‌ی انجام

عملیات روی هر دو عضو مجموعه، عضو دیگری از آن مجموعه است.

* توزیع‌پذیری^۳: اگر a, b, c اعضای G باشند $(a \bullet b) \bullet c = a \bullet (b \bullet c)$. به عبارت دیگر، مهم

نیست که عملیات را به چه ترتیبی روی بیش از دو عضو انجام دهیم.

* جابجا پذیری^۴: برای تمام a, b در G داریم $a \bullet b = b \bullet a$. توجه داشته باشید که این ویژگی فقط

باید برای گروه با خاصیت جابجایی اعمال شود.

* عضو خنثی^۵: برای تمام اعضای مجموعه در G ، یک e وجود دارد که به آن عضو خنثی می-

گویند و داریم: $e \bullet a = a \bullet e = a$.

* وجود وارون^۱: برای هر a در G یک عضو a ، که به آن a' یا «وارون a » گفته می‌شود وجود دارد

به گونه‌ای که $a \bullet a' = a' \bullet a = e$.

^۱ Algebraic Structures

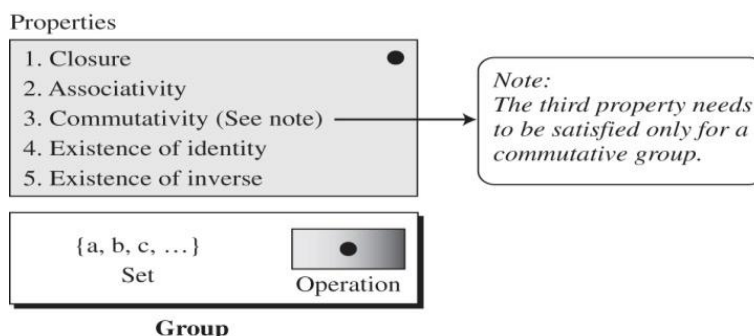
^۲ Closure

^۳ Associativity

^۴ Commutativity

^۵ Existence of identity

شکل ۴-۲ مفهوم گروه را نشان می‌دهد.



شکل ۴-۲: گروه

کاربرد:

اگرچه گروه شامل یک عملگر است، اما ویژگی‌های تعیین شده برای عملگر، امکان استفاده از اعمال دیگر را نیز می‌دهد؛ البته تا زمانی که این اعمال وارون یکدیگر باشند. به عنوان مثال، اگر عمل تعیین شده جمع باشد، گروه از جمع و تفریق پشتیبانی می‌کند، زیرا تفریق، جمعی است که از وارون نسبت به جمع استفاده می‌کند. این مسئله در مورد ضرب و تقسیم نیز صدق می‌کند. با این حال، گروه در یک زمان می‌تواند فقط از جمع / تفریق یا ضرب / تقسیم پشتیبانی کند، نه هر دوی آن‌ها.

مثال ۴-۱

مجموعه اعداد صحیح باقیمانده با عمل جمع $G = \langle \mathbb{Z}_n, + \rangle$ گروه جابجاپذیر می‌باشد. می‌توانیم بدون خارج شدن از مجموعه، اعمال جمع و تفریق را روی اعضای این مجموعه انجام دهیم. اجازه دهید پنج ویژگی را تک‌تک بررسی کنیم.

(۱) بسته بودن: این خاصیت وجود دارد. نتیجه‌ی جمع دو عدد صحیح در \mathbb{Z}_n ، عدد صحیح دیگری در \mathbb{Z}_n است.

(۲) توزیع‌پذیری: این خاصیت وجود دارد. نتیجه $4 + (3 + 2)$ با نتیجه $(4 + 3) + 2$ یکی است.

(۳) جابجایی: این خاصیت وجود دارد. $3 + 5 = 5 + 3$ درست است.

(۴) عضو خنثی که ۰ است. $3 + 0 = 0 + 3 = 3$ درست است.

¹ Existence of Inverse

(۵) هر عضو، یک وارون نسبت به جمع دارد. وارون هر عضو، متمم آن است. مثلاً وارون ۳، ۳- است (n-3 در Z_n) و وارون ۳- عدد ۳ است. وارون، به ما امکان انجام تفریق بر روی مجموعه را می‌دهد.

مثال ۴-۲

مجموعه Z_n^* با عمل ضرب $G = \langle Z_n^*, \times \rangle$ نیز یک گروه آبلی است. می‌توانیم بدون خارج شدن از مجموعه، اعمال ضرب و تقسیم را روی اعضای آن انجام دهیم. آزمایش سه ویژگی اول ساده است. عضو خنثی، یک است. هر عضو، وارون دارد که می‌توان آن را بر اساس الگوریتم اقلیدسی بسطیافته به دست آورد.

مثال ۴-۳

اگرچه معمولاً گروه را مجموعه‌ای از اعداد با اعمال عادی چون جمع و تفریق در نظر می‌گیریم، اما تعریف گروه به ما این اجازه را می‌دهد که هر گروه از اشیاء و عملی که ویژگی‌های فوق‌الذکر را برآورده کند، مجموعه بنامیم. اجازه دهید مجموعه $G = \langle \{a, b, c, d\}, \bullet \rangle$ و عمل \bullet را، به شکلی که در جدول ۴-۱ نشان داده شده است تعریف کنیم.

جدول ۴-۱: جدول عمل برای مثال ۴-۳

\bullet	a	b	c	d
a	a	b	c	d
b	b	c	d	a
c	c	d	a	b
d	d	a	b	c

حاصل یک گروه آبلی است که تمام پنج ویژگی لازم را دارد.

(۱) ویژگی بسته بودن وجود دارد. نتیجه اعمال عمل \bullet روی هر زوج از اعضا، با اعضای دیگری در آن مجموعه مشخص می‌شود.

(۲) ویژگی توزیع‌پذیری نیز وجود دارد. برای اثبات این ویژگی باید آن را برای هر ترکیبی از سه عضو آزمایش کنیم. مثلاً $(a \bullet b) \bullet c = a \bullet (b \bullet c) = d$.

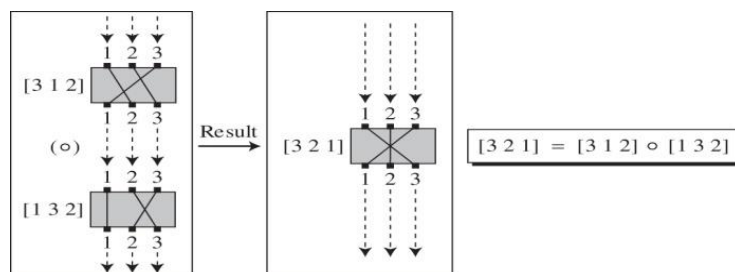
(۳) عمل \bullet دارای خاصیت جابجایی است. داریم $a \bullet b = b \bullet a$.

(۴) گروه یک عضو خنثی دارد که a است.

(۵) هر عضو یک وارون دارد. می‌توان با یافتن عضو خشتی در هر ردیف (سایه‌دار)، زوج‌های وارون را به دست آورد. زوج‌ها عبارت‌اند از: $(a,a), (b,d), (c,c)$.

مثال ۴-۴

در یک گروه، اعضای مجموعه الزاماً نباید عدد یا اشیاء باشند بلکه می‌توانند قواعد، نگاشت‌ها، توابع، یا حتی اعمال باشند. یک گروه بسیار جالب، گروه جایگشت است. مجموعه‌ی آن، مجموعه‌ی تمام جایگشت‌هاست و عملیاتش ترکیب است، یعنی به کار بردن یک جایگشت پس از جایگشت دیگر. شکل ۳-۴ ترکیب دو جایگشتی را نشان می‌دهد که سه ورودی را با هم جابجا می‌کند تا سه خروجی به دست آورد.



شکل ۳-۴: ترکیب جایگشت‌ها (مثال ۴-۴)

وروددی‌ها و خروجی‌ها می‌توانند نویسه (موضوع فصل ۲) یا بیت (موضوع فصل ۵) باشند. ما هر جایگشت را در جدولی نشان داده‌ایم. این جداول دو مشخصه دارند: ۱- محتوا جدول نشان می‌دهد که ورودی از کجا آمده است ۲- شاخصی که خروجی را تعیین می‌کند (این مورد در جدول نشان داده نشده است). ترکیب شامل اجرای دو جایگشت، یکی پس از دیگری است. توجه داشته باشید که عبارت در شکل ۳-۴ از راست به چپ خوانده می‌شود. جایگشت اول، $[1\ 3\ 2]$ و پس از آن $[3\ 1\ 2]$ است، نتیجه $[3\ 2\ 1]$ می‌باشد. با سه ورودی و سه خروجی، می‌تواند ۳ یا ۶ جایگشت متفاوت وجود داشته باشد. جدول ۲-۴ نشان می‌دهد که چگونه عملیات تعیین می‌شود. ردیف اول بیانگر جایگشت اول و ستون اول بیانگر جایگشت دوم می‌باشد. نتیجه، عضو محل تقاطع آن‌ها در جدول است.

در این مورد فقط چهار ویژگی از پنج ویژگی وجود دارد و گروه غیر آبدلی^۱ است.

(^۱) ویژگی بسته بودن وجود دارد.

=====

¹ Non-Abelian

(۲) ویژگی توزیع پذیری وجود دارد. برای اثبات این مسئله باید آن ویژگی را برای هر ترکیب از سه عنصر چک کنیم.

(۳) ویژگی جابجایی وجود ندارد. می توان آن را به سادگی چک کرد، اما این کار را به عنوان تمرین به عهده ی شما می گذاریم.

جدول ۴-۲: جدول عملیاتی برای گروه جایگشت

°	[1 2 3]	[1 3 2]	[2 1 3]	[2 3 1]	[3 1 2]	[3 2 1]
[1 2 3]	[1 2 3]	[1 3 2]	[2 1 3]	[2 3 1]	[3 1 2]	[3 2 1]
[1 3 2]	[1 3 2]	[1 2 3]	[2 3 1]	[2 1 3]	[3 2 1]	[3 1 2]
[2 1 3]	[2 1 3]	[3 1 2]	[1 2 3]	[3 2 1]	[1 3 2]	[2 3 1]
[2 3 1]	[2 3 1]	[3 2 1]	[1 3 2]	[3 1 2]	[1 2 3]	[2 1 3]
[3 1 2]	[3 1 2]	[2 1 3]	[3 2 1]	[1 2 3]	[2 3 1]	[1 3 2]
[3 2 1]	[3 2 1]	[2 3 1]	[3 1 2]	[1 3 2]	[2 1 3]	[1 2 3]

(۱) مجموعه یک عضو خنثی دارد که [۱ ۲ ۳] (بدون جایگشت است) می باشد. این عناصر در سطر اول نشان داده شده است.

(۲) هر عضو یک وارون دارد. با استفاده از عناصر خنثی می توان زوج عضو وارون را یافت.

مثال ۴-۵

در مثال پیش نشان دادیم که مجموعه ای از جایگشت ها با عملیات ترکیبی، گروه محسوب می شود. این مسئله حاکی از آن است که با استفاده از دو جایگشت یکی پس از دیگری، امنیت یک شیوه از رمزنگاری را نمی توان تقویت کرد زیرا به علت ویژگی بسته بودن می توان همیشه جایگشتی را یافت که همان کار را انجام دهد.

گروه متناهی^۱

اگر مجموعه دارای تعداد متناهی از اعضا باشد، به آن گروه متناهی، و در غیر این صورت گروه نامتناهی^۲ نامیده می شود.

ترتیب یک گروه^۳

ترتیب یک گروه $|G|$ عبارت است از تعداد اعضای موجود در آن گروه. اگر گروه متناهی نباشد، ترتیب نامتناهی است و اگر گروه متناهی باشد، ترتیب متناهی است.

^۱ Finite Group

^۲ Infinite Group

^۳ Order of a Group

زیرگروه‌ها^۱

زیرمجموعه H از یک گروه G ، یک زیرگروه G است به شرطی که خود H با توجه به عملیات تعریف شده بر روی G گروه محسوب شود. به عبارت دیگر، اگر $G = \langle s, \bullet \rangle$ گروه باشد $H = \langle T, \bullet \rangle$ نیز تحت همان عملیات گروه محسوب می‌شود. وقتی T زیرمجموعه‌ی غیر تهی S باشد، پس H زیرگروه G است. تعریف فوق حاکی از این است که:

- (۱) اگر a, b اعضای هر دو گروه باشند پس $C = a \bullet b$ نیز عضوی از هر دو گروه می‌باشد.
- (۲) گروه‌ها در عضو خنثی با هم مشترک‌اند.
- (۳) اگر a, a' عضو هر دو گروه باشند و a و همچنین a عضو هر دو گروه هستند.
- (۴) گروه ساخته شده از عضو خنثی G ، \bullet ، $\{e\}$ ، $H = \langle \bullet, e \rangle$ زیرگروهی از G است.
- (۵) هر گروه، زیر گروه خود نیز می‌باشد.

مثال ۴-۶

آیا گروه $H = \langle Z_{10}, + \rangle$ زیرگروهی از $G = \langle Z_{12}, + \rangle$ می‌باشد.

حل

پاسخ منفی است. اگرچه H زیرمجموعه G است، اما عملیات تعیین شده برای این دو گروه متفاوت است. عملیات H جمع به پیمانه ۱۰ است، در حالی که عملیات G جمع به پیمانه ۱۲ است.

زیرگروه‌های دوار^۲

اگر زیرگروهی از یک گروه را بتوان با استفاده از توان‌هایی از یک عضو ایجاد کرد، این زیرگروه را زیرگروه دوار می‌نامند. در اینجا واژه‌ی «توان» به معنی اعمال مکرر عملیات گروه بر روی یک عضو است:

$$a^n = \underbrace{a.a.a \dots a}_{n \text{ بار}}$$

مجموعه‌ی ساخته شده از این مراحل را به صورت $\langle a \rangle$ نشان می‌دهیم. توجه داشته باشید که اعضای تکراری باید حذف شوند و $a^0 = e$ است.

=====

¹ Subgroups

² Cyclic Subgroups

مثال ۷-۴

از گروه $G = \langle \mathbb{Z}_6, + \rangle$ می‌توان چهار زیرگروه چرخه‌ای (دوار) ساخت. این زیرگروه‌ها عبارت‌اند

از:

$$H_1 = \langle \{0\}, + \rangle, H_2 = \langle \{0, 2, 4\}, + \rangle, H_3 = \langle \{0, 3\}, + \rangle, H_4 = G$$

توجه داشته باشید که وقتی که عملیات جمع باشد، a^n یعنی a را n بار ضرب کنیم. همچنین شایان توجه است که در تمام این گروه‌ها، عملیات جمع به پیمانه ۶ است. موارد زیر نشان می‌دهند که چگونه اعضای این زیرگروه چرخه‌ای را پیدا کردیم.

(a) زیرگروه چرخه‌ای ایجاد شده از ۰، H_1 است که فقط یک عضو دارد و آن هم عضو خنثی است.
 $0 \bmod 6 = 0$

(توجه: مراحل مجدداً تکرار خواهد شد.)

(b) زیرگروه چرخه‌ای (دوار) ایجاد شده از ۱، H_4 است، که خود G می‌باشد.

$$1^0 \bmod 6 = 0$$

$$1^1 \bmod 6 = 1$$

$$1^2 \bmod 6 = (1+1) \bmod 6 = 2$$

$$1^3 \bmod 6 = (1+1+1) \bmod 6 = 3$$

$$1^4 \bmod 6 = (1+1+1+1) \bmod 6 = 4$$

$$1^5 \bmod 6 = (1+1+1+1+1) \bmod 6 = 5$$

(توجه: مراحل دوباره تکرار خواهد شد.)

(c) زیرگروه چرخه‌ای ایجاد شده از \mathbb{Z}_2 ، H_2 است، که دارای سه عضو می‌باشد: ۰، ۲، ۴.

$$2^0 \bmod 6 = 0$$

$$2^1 \bmod 6 = 2$$

$$2^2 \bmod 6 = (2+2) \bmod 6 = 4$$

(توجه: مراحل دوباره تکرار خواهد شد.)

(d) زیرگروه چرخه‌ای ایجاد شده از ۳، H_3 است، که دارای دو عضو می‌باشد: ۰ و ۳.

$$3^0 \bmod 6 = 0$$

$$3^1 \bmod 6 = 3$$

(توجه: مراحل دوباره تکرار خواهد شد.)

(e) زیرگروه چرخه‌ای ایجاد شده از ۴، H_2 است که زیرگروه جدیدی نیست.

$$4^0 \bmod 6 = 0$$

$$4^1 \bmod 6 = 4$$

$$4^2 \bmod 6 = (4+4) \bmod 6 = 2$$

(توجه: مراحل دوباره تکرار خواهد شد.)

(f) زیرگروه چرخه‌ای ایجاد شده از ۵، H_4 است که خود G می‌باشد.

$$5^0 \bmod 6 = 0$$

$$5^1 \bmod 6 = 5$$

$$5^2 \bmod 6 = 4$$

$$5^3 \bmod 6 = 3$$

$$5^4 \bmod 6 = 2$$

$$5^5 \bmod 6 = 1$$

(توجه: مراحل دوباره تکرار خواهد شد.)

مثال ۴-۸

از گروه $G = \langle \mathbb{Z}_{10}^*, \times \rangle$ می‌توان سه زیرگروه چرخه‌ای ساخت. G فقط چهار عضو دارد: ۱، ۳، ۷، ۹.

۹. زیرگروه‌های چرخه‌ای عبارت‌اند از: $H_3 = G$ ، $\langle *, 9 \rangle$ ، $H_1 = \{1, 9\}$ موارد زیر نشان می‌دهند که چگونه عناصر این زیرگروه‌ها را پیدا کرده‌ایم.

(a) زیرگروه چرخه‌ای ایجاد شده از ۱، H_1 است. این زیرگروه فقط یک عضو دارد و آن عضو خنثی است.

$$1^0 \bmod 10 = 1$$

(توجه: مراحل دوباره تکرار خواهد شد.)

(b) زیرگروه چرخه‌ای ایجاد شده از ۳، H_3 است و آن خود G است.

$$3^0 \bmod 10 = 1$$

$$3^1 \bmod 10 = 3$$

$$3^2 \bmod 10 = 9$$

$$3^3 \bmod 10 = 7$$

(توجه: مراحل دوباره تکرار خواهد شد.)

(c) زیرگروه ایجاد شده از H_3 ، 7 است، که خود G است.

$$7^1 \bmod 10 = 7$$

$$7^2 \bmod 10 = 9$$

$$7^3 \bmod 10 = 3$$

(توجه: مراحل دوباره تکرار خواهد شد.)

(d) زیرگروه چرخه‌ای ایجاد شده از H_2 ، 9 است. این زیرگروه فقط دو عضو دارد.

$$9^1 \bmod 10 = 9$$

$$9^2 \bmod 10 = 1$$

(توجه: مراحل دوباره تکرار خواهد شد.)

گروه‌های دوار

گروه چرخه‌ای دوار گروهی است که خود زیرمجموعه‌ی گروهی می‌باشد. در مثال ۷-۴ گروه G یک زیرگروه دوار $H_5 = G$ دارد، یعنی گروه G گروه دوار است. در این حالت، عضوی که زیرگروه دوار را می‌سازد، می‌تواند خود گروه را نیز ایجاد کند. به این عضو، مولد می‌گویند. اگر g یک مولد باشد، اعضای موجود در گروه دوار متناهی را می‌توان به صورت زیر نوشت:

$$\{e, g, g^2, \dots, g^n\} \text{ به گونه‌ای که } g^n = e \text{ است.}$$

توجه داشته باشید که گروه دوار می‌تواند مولدهای زیادی داشته باشد.

مثال ۹-۴

(۱) گروه $G = \langle Z_6, + \rangle$ گروه دوار با دو مولد $g=1$ و $g=5$ است.

(۲) گروه $G = \langle Z_{10}^*, \times \rangle$ گروه دوار با دو مولد $g=3$ و $g=7$ می‌باشد.

نظریه لاگرانژ^۱

در نظریه لاگرانژ می‌توان ترتیب یک گروه را با ترتیب به زیرگروهش مرتبط ساخت. فرض کنید G گروه و H یک زیرگروه از G باشد. اگر ترتیب H به ترتیب $|G|$ و $|H|$ باشد، پس بر اساس این نظریه $|G|$ بر $|H|$ بخش‌پذیر است. در مثال $4-7$ ، $|G|=6$ است. ترتیب زیرمجموعه‌ها به این شکل است: $|H_1|=1$ ، $|H_2|=3$ ، $|H_3|=2$ ، $|H_4|=6$. آشکار است که تمام این ترتیب‌ها بر ۶ بخش‌پذیرند.

نظریه لاگرانژ کاربرد بسیار جالبی دارد. با فرض داشتن گره G با ترتیب $|G|$ ، ترتیب زیرگروه‌های بالقوه را می‌توان به سادگی تعیین کرد به شرطی که مقسوم‌علیه‌های G را بتوان پیدا کرد. به عنوان مثال، ترتیب گروه $G = \langle Z_{17}, + \rangle$ ، 17 است. تنها مقسوم‌علیه ۱۷ اعداد ۱ و ۱۷ است؛ این بدین معنی است که این گروه فقط می‌تواند دو زیرگروه داشته باشد، H_1 با عضو خنثی و $H_2=G$.

ترتیب یک عضو^۲

ترتیب یک عضو a در یک گروه $\text{ord}(a)$ ، کوچک‌ترین عدد صحیح n مانند $a^n = e$ می‌باشد. این تعریف را می‌توان چنین بازگو کرد: ترتیب یک عضو عبارت است از ترتیب گروه دواری که از آن عضو ساخته می‌شود.

مثال ۴-۱۰

- (a) در یک گروه $G = \langle Z_6, + \rangle$ ، ترتیب اعضا عبارت‌اند از:
 $\text{ord}(0) = 1$ ، $\text{ord}(1) = 6$ ، $\text{ord}(2) = 3$ ، $\text{ord}(3) = 2$ ، $\text{ord}(4) = 3$ ، $\text{ord}(5) = 6$
- (b) در گروه $G = \langle Z_{10}^*, \times \rangle$ ، ترتیب اعضا عبارت‌اند از:
 $\text{ord}(1) = 1$ ، $\text{ord}(3) = 4$ ، $\text{ord}(7) = 4$ ، $\text{ord}(9) = 2$

حلقه

حلقه، که به صورت $R = \langle \{ \dots \}, \cdot, \square \rangle$ ، نشان داده می‌شود، یک ساختار جبری با دو عمل است که عمل اول باید تمام پنج ویژگی اشاره شده برای گروه آبدی را داشته باشد و عمل دوم باید فقط دو ویژگی اول را برآورده سازد؛ به علاوه، عمل دوم بر پایه عمل اول توزیع‌پذیر باشند. توزیع‌پذیری بدین معنی است که برای تمام اعضای a, b, c از R ، $a \square (b \cdot c) = (a \square b) \cdot (a \square c)$ و $(a \cdot b) \square c = (a \square c) \cdot (b \square c)$.

=====

¹ Lagrange's Theorem

² Order of an Element

را داشته باشیم. حلقه جابجایی، حلقه‌ای است که در آن خاصیت جابجایی برای دومین عملیات نیز وجود دارد. شکل ۴-۴ یک حلقه و یک حلقه جابجایی نمونه را نشان می‌دهد.

کاربرد

حلقه شامل دو عمل است، با این حال می‌توان به منظور برآورده کردن ویژگی‌های سوم و چهارم، عمل دوم را انجام نداد؛ به عبارت دیگر، عمل اول در واقع عمل بر روی یک زوج، نظیر جمع و تفریق می‌باشد و عمل دوم، یک عمل تکی مانند ضرب است و نمی‌تواند تقسیم باشد.

مثال ۴-۱۱

مجموعه Z با دو عمل جمع و ضرب، یک حلقه جابجایی است و آن را با $R = \langle Z, +, \times \rangle$ نشان می‌دهیم. مثلاً $5 \times (3 + 2) = (5 \times 3) + (5 \times 2) = 25$ نسبت به جمع توزیع پذیر است. اگرچه می‌توانیم جمع و تفریق را بر روی این مجموعه انجام دهیم، اما فقط می‌توانیم ضرب را انجام دهیم، نه تقسیم را. در این ساختار، تقسیم مجاز نیست زیرا حاصل آن عضوی خارج از مجموعه است. حاصل تقسیم ۱۲ بر ۵، $2/4$ است که در مجموعه وجود ندارد.

Distribution of ☐ over ☒

1. Closure <input checked="" type="radio"/> 2. Associativity 3. Commutativity 4. Existence of identity 5. Existence of inverse	1. Closure <input type="checkbox"/> 2. Associativity 3. Commutativity <input type="checkbox"/>	Note: The third property is only satisfied for a commutative ring.
{a, b, c, ...} Set	<input checked="" type="checkbox"/> <input type="checkbox"/> Operations	

شکل ۴-۴: حلقه

میدان

میدان، که با $F = \langle \{...\}, \cdot, \square \rangle$ نشان داده می‌شود، یک حلقه‌ی جابجایی است که در آن، عملیات دوم تمام ویژگی‌های تعیین شده برای عملیات اول را داراست به استثنای این که عضو خنثای عملیات اول، که گاهی به آن عضو صفر می‌گویند، وارون ندارد. شکل ۴-۵ میدان را نشان می‌دهد.

Distribution of ☐ over ☒

1. Closure	<input checked="" type="radio"/>	1. Closure	<input type="checkbox"/>
2. Associativity	<input type="checkbox"/>	2. Associativity	<input type="checkbox"/>
3. Commutativity	<input type="checkbox"/>	3. Commutativity	<input type="checkbox"/>
4. Existence of identity	<input type="checkbox"/>	4. Existence of identity	<input type="checkbox"/>
5. Existence of inverse	<input type="checkbox"/>	5. Existence of inverse	<input type="checkbox"/>

Set
 ☒ ☐ Operations

Note:
The identity element of the first operation has no inverse with respect to the second operation.

شکل ۴-۵: میدان

کاربرد

میدان، ساختاری است که یک زوج از عملیاتی را که در ریاضی استفاده می‌کنیم، پشتیبانی می‌کند: جمع / تفریق و ضرب / تقسیم. فقط یک استثنا وجود دارد: تقسیم بر صفر مجاز نیست.

میدان‌های متناهی

اگر چه میدان‌هایی با ترکیب نامتناهی داریم اما میدان‌های متناهی در رمزنگاری کاربرد گسترده‌ای دارد. میدان متناهی - میدانی با تعدادی متناهی از اعضا، ساختار بسیار مهمی در رمزنگاری محسوب می‌شود. گالوا^۱ نشان داد که برای این که یک میدان متناهی باشد، باید تعداد اعضایش p^n باشد، به گونه‌ای که p عدد اول و n عدد صحیح مثبت است. معمولاً میدان متناهی را میدان‌های گالوا^۲ می‌نامند و آن را به صورت $GF(p^n)$ نشان می‌دهند.

میدان گالوا - $GF(pn)$ - میدان متناهی با pn عضو هستند.

میدان‌های $GF(p)$

وقتی که $n=1$ باشد، میدان $GF(p)$ داریم. این میدان می‌تواند مجموعه‌ای از $\{0, 1, \dots, p-1\}$ و Z_p با دو عملگر محاسباتی^۳ (جمع و ضرب) باشد. شایان یادآوری است که در این مجموعه هر عضو، یک وارون نسبت به جمع دارد و اعضای غیر صفر نیز دارای یک وارون نسبت به جمع می‌باشند (وارون ضرب برای ۰ وجود ندارد).

=====

¹ Galois

² Galois Fields

³ Arithmetic Operation

مثال ۴-۱۲

همان‌گونه که در شکل ۴-۶ نشان داده شده است، یک میدان بسیار رایج در این دسته $GF(2)$ با مجموعه $\{0, 1\}$ و دو عملیات جمع و ضرب می‌باشد.

$GF(2)$			
$\{0, 1\}$	$\begin{array}{ c c c } \hline + & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & 0 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline \times & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & 0 & 1 \\ \hline \end{array}$	$\begin{array}{ c c c } \hline a & 0 & 0 \\ \hline -a & 1 & 1 \\ \hline \end{array} \quad \begin{array}{ c c c } \hline a & 0 & 1 \\ \hline a^{-1} & -1 & -1 \\ \hline \end{array}$
	Addition	Multiplication	Inverses

شکل ۴-۶: میدان $GF(2)$

نکاتی در مورد این دامنه وجود دارد که باید به آن‌ها توجه کرد. نخست، این مجموعه فقط دو عضو دارد که ارقام یا بیت‌های دودویی $(0, 1)$ هستند. دوم، عملیات جمع در واقع XOR است که بر روی دو رقم دودویی اعمال می‌کنیم. سوم، عملیات ضرب عمل AND است که بر روی دو رقم دودویی اعمال می‌کنیم. چهارم، عملیات جمع یا تفریق^۱ یکسان‌اند. پنجم، عملیات ضرب و تقسیم^۲ یکسان‌اند.

نتیجه‌ی جمع/تفریق در $GF(2)$ مانند نتیجه‌ی عملیات XOR می‌باشد.

نتیجه‌ی ضرب/تقسیم مانند نتیجه‌ی عملیات AND است.

مثال ۴-۱۳

همان‌گونه که در شکل ۴-۷ نشان داده شده است، می‌توانیم $GF(5)$ را روی مجموعه Z_5 (۵ عدد اول است) با اعمال جمع و ضرب تعریف کنیم.

اگر چه می‌توانیم از الگوریتم اقلیدسی بسط‌یافته برای پیدا کردن وارون‌های ضربی اعضای $GF(5)$ استفاده کنیم، اما نگاه کردن به جدول ضرب و یافتن هر زوج با حاصل ضرب برابر با ۱ ساده‌تر است. حاصل، زوج‌های $(1, 1)$ ، $(2, 3)$ ، $(3, 2)$ ، $(4, 4)$ می‌باشند. توجه داشته باشید که به استثنای تقسیم بر صفر که مجاز نیست، می‌توانیم جمع/تفریق و ضرب/تقسیم را بر روی این مجموعه انجام دهیم.

=====

^۱ عملیات XOR

^۲ عملیات AND

GF(5)

0

1

2

3

4

+

0

1

2

3

4

0

1

2

3

4

×

0

1

2

3

4

0

1

2

3

4

+

0

1

2

3

4

0

1

2

3

4

×

0

1

2

3

4

Addition

Multiplication

Additive inverse

Multiplicative inverse

شکل ۴-۷: میدان GF(5)

میدانهای $GF(p^n)$

علاوه بر میدانهای $GF(p)$ ، در رمزنگاری، میدانهای $GF(p^n)$ نیز برای ما حائز اهمیت است؛ با این وجود، مجموعه Z, Z_n, Z_n^*, Z_p ، که پیشتر با عملیاتی نظیر جمع و ضرب به کار می‌رفت، نمی‌تواند ویژگی‌های میدان را برآورده سازد بلکه باید چند مجموعه و عملیات جدید برای آن مجموعه‌ها تعیین گردد. در بخش بعدی نشان خواهیم داد که چگونه $GF(p^n)$ ، میدان بسیار مفیدی در رمزنگاری محسوب می‌شود.

خلاصه

مطالعه ساختارهای جبری، امکان استفاده از مجموعه‌هایی را که در آن‌ها عملیات‌های شبیه به جمع/تفریق و ضرب/تقسیم وجود دارد را فراهم می‌سازد. باید بین سه ساختار، تفاوت قائل شویم. اولین ساختار، گروه، یک زوج مرتبط از عملیات‌ها؛ ساختار دوم، حلقه، یک جفت مرتبط از عملیات‌ها و یک عملیات تکی و ساختار سوم؛ میدان‌ها، دو جفت از عملیات ما را پشتیبانی می‌کند. جدول ۴-۳ می‌تواند در زمینه مشاهده تفاوت‌ها به ما کمک کند.

جدول ۴-۳: خلاصه ساختارهای جبری

Algebraic Structure	Supported Typical Operations	Supported Typical Sets of Integers
Group	(+ -) or ($\times \div$)	Z_n or Z_n^*
Ring	(+ -) and (\times)	Z
Field	(+ -) and ($\times \div$)	Z_p

۴-۲- میدانهای $GF(2^n)$

در رمزنگاری، معمولاً می‌بایست از چهار عمل (جمع، تفریق، ضرب و تقسیم) استفاده کنیم؛ به عبارت دیگر باید از میدان استفاده کنیم. با این حال، وقتی با کامپیوتر کار می‌کنیم، اعداد صحیح مثبت به شکل کلمات n بیتی، که در آن n معمولاً ۸، ۱۶، ۳۲، ۶۴ و الی آخر می‌باشد، ذخیره می‌شود. این بدین معنی است که دامنه اعداد صحیح از ۰ تا 2^{n-1} است.

پیمانه^۱ برابر 2^n است؛ بنابراین اگر بخواهیم از میدان استفاده کنیم، دو انتخاب داریم:

(۱) می‌توان از $GF(p)$ با مجموعه Z_p استفاده کنیم، به گونه‌ای که p بزرگ‌ترین عدد اول کمتر از 2^n است. اگر چه این طرح کار می‌کند، اما ناکافی است، زیرا نمی‌توانیم از اعداد صحیح از p تا 2^n-1 استفاده کنیم. مثلاً اگر $n=4$ باشد بزرگ‌ترین عدد اول کمتر از 2^4 ، ۱۳ است. این بدین معنی است که نمی‌توانیم از اعداد صحیح ۱۳، ۱۴، ۱۵ استفاده کنیم. اگر $n=8$ باشد، بزرگ‌ترین عدد اول کوچک‌تر از 2^8 ، ۲۵۱ است، بنابراین نمی‌توانیم از ۲۵۱، ۲۵۲، ۲۵۳، ۲۵۴، ۲۵۵ استفاده کنیم.

(۲) می‌توانیم در $GF(2^n)$ کار کنیم و از مجموعه‌ی اعضای 2^n استفاده کنیم. اعضای این مجموعه، کلمات n بیتی هستند. به عنوان مثال، اگر $n=3$ باشد، مجموعه چنین است:

$$\{000, 001, 010, 011, 100, 101, 110, 111\}$$

با این وجود، نمی‌توانیم هر عضو را عدد صحیحی بین ۰ تا ۷ تلقی کنیم زیرا چهار عمل اصلی را نمی‌توان انجام داد (پیمانه 2^n اول نیست). برای این که ویژگی‌های تعیین شده برای میدان تحقق یابد، لازم است مجموعه‌ای از کلمات n بیتی و دو عملیات جدید تعیین کنیم.

مثال ۴-۱۴

فرض کنید یک میدان $GF(2^2)$ که در آن، مجموعه‌ی چهار کلمه‌ی دو بیتی دارد، را تعیین کنیم: $\{00, 01, 10, 11\}$. همان‌گونه که در شکل ۴-۸ نشان داده شده است می‌توانیم جمع و ضرب را برای این میدان دوباره تعریف کنیم، به گونه‌ای که تمام ویژگی‌های این عملیات برآورده شود.

=====

¹ Modulus

Addition					Multiplication				
\oplus	00	01	10	11	\otimes	00	01	10	11
00	00	01	10	11	00	00	00	00	00
01	01	00	11	10	01	00	01	10	11
10	10	11	00	01	10	00	10	11	01
11	11	10	01	00	11	00	11	01	10
Identity: 00					Identity: 01				

شکل ۴-۸: مثالی از میدان $GF(2^2)$

هر کلمه، وارون جمعی خودش می‌باشد. هر کلمه (به غیر از ۰۰)، یک وارون نسبت به ضرب دارد. زوج‌های دارای وارون ضربی عبارت‌اند از: (۰۱، ۰۱)، (۱۰، ۱۱). جمع و ضرب، به شکل چندجمله‌ای تعریف می‌شوند.

چندجمله‌ای‌ها^۱

اگرچه می‌توانیم مستقیماً قوانین اعمال جمع و ضرب روی کلمات n بیتی که ویژگی‌های $GF(2^n)$ را برآورد می‌کنند، تعریف کنیم، اما کار کردن با نگارش در قالب کلمات n بیتی، به صورت چندجمله‌ای با درجه $n-1$ ساده‌تر است. چندجمله‌ای با درجه $n-1$ عبارتی به شکل زیر می‌باشد:

$$F(x) = a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_1 x^1 + a_0 x^0$$

طوری که X^i ، i امین عبارت و a_i ضریب i^2 امین عبارت می‌باشد. اگر چه با چندجمله‌ای‌ها در جبر آشنا هستیم، اما برای نشان دادن یک کلمه n بیتی در قالب چندجمله‌ای، باید از قوانین زیر پیروی کنیم:

(۱) توان X ، جایگاه بیت در کلمه n بیتی را مشخص می‌کند. یعنی بیت سمت راست در جایگاه صفر است (مربوط به X^0)؛ بیت سمت چپ در جایگاه $n-1$ است (مربوط به X^{n-1}).

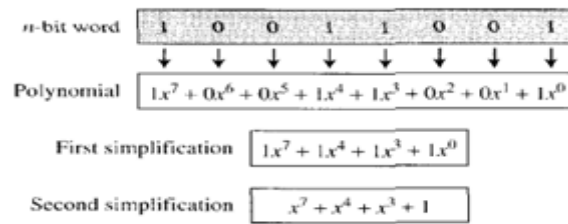
(۲) ضریب عبارات، مقدار بیت‌ها را تعیین می‌کند. از آن جا که یک بیت می‌تواند فقط مقدار ۰ یا ۱ را داشته باشد، ضریب چندجمله‌ای ما می‌تواند ۰ یا ۱ باشد.

مثال ۴-۱۵

شکل ۴-۱۹ نشان می‌دهد که چطور می‌توانیم یک کلمه n بیتی (۱۰۰۱۱۰۰۱) را با استفاده از چندجمله‌ای نشان دهیم.

¹ Polynomials

² Coefficient



شکل ۴-۹: نمایش یک کلمه ۸ بیتی به وسیله یک چندجمله‌ای

توجه داشته باشید که اگر ضریب صفر باشد، عبارت به طور کلی حذف می‌شود و اگر ۱ باشد، ضریب حذف می‌شود. همچنین توجه داشته باشید که X^0 برابر ۱ است.

مثال ۴-۱۶

برای یافتن یک کلمه ۸ بیتی مربوط به چندجمله‌ای $X^0 + X^2 + X^5$ نخست جملات حذف شده را می‌نویسیم. از آنجا که $n=8$ است، چندجمله‌ای از درجه ۷ است. چندجمله‌ای گسترش یافته به شکل زیر می‌باشد.

$$0 \cdot X^7 + 0 \cdot X^6 + 1 \cdot X^5 + 0 \cdot X^4 + 0 \cdot X^3 + 1 \cdot X^2 + 1 \cdot X^1 + 0 \cdot X^0$$

حاصل کلمه ۸ بیتی ۰۰۱۰۰۱۱۰ می‌باشد.

عملگر در چندجمله‌ای‌ها

توجه کنید که هر گونه عملیات روی چندجمله‌ای‌ها، در واقع شامل دو عملیات است: عملیات روی ضریب‌ها و عملیات روی دو چندجمله‌ای؛ به عبارت دیگر لازم است دو میدان تعیین کنیم، یکی برای ضریب‌ها و دیگری برای چندجمله‌ای‌ها. ضریب‌ها از ۰ و ۱ تشکیل شده است. برای این منظور می‌توانیم از دامنه $GF(2)$ استفاده کنیم. پیشتر (در مثال ۴-۱۴) در مورد این میدان صحبت کرده‌ایم. برای چندجمله‌ای‌ها، میدان $GF(2^n)$ را لازم داریم، که به طور مختصر آن را توضیح خواهیم داد.

چندجمله‌ای‌های نشان دهنده‌ی کلمات ۸ بیتی از دو میدان استفاده می‌کنند: $GF(2)$ و $GF(2^n)$.

پیمانه‌ها

پیشتر عملیات روی چند جمله‌ای‌ها را تعریف کردیم؛ اکنون لازم است در مورد چندجمله‌ای‌های پیمانه-ای^۱ صحبت کنیم. جمع دو چندجمله‌ای، هیچ‌گاه یک چندجمله‌ای خارج از مجموعه را ایجاد نمی‌کند، با این حال ممکن است حاصل ضرب دو چندجمله‌ای را بر یک پیمانه تقسیم کنیم و تنها باقیمانده را نگه

¹ Modulus Polynomials

جدول ۴-۴: لیست چند جمله‌ای کاهش نایذیر

Degree	Irreducible Polynomials
1	$(x + 1), (x)$
2	$(x^2 + x + 1)$
3	$(x^3 + x^2 + 1), (x^3 + x + 1)$
4	$(x^4 + x^3 + x^2 + x + 1), (x^4 + x^3 + 1), (x^4 + x + 1)$
5	$(x^5 + x^2 + 1), (x^5 + x^3 + x^2 + x + 1), (x^5 + x^4 + x^3 + x + 1),$ $(x^5 + x^4 + x^3 + x^2 + 1), (x^5 + x^4 + x^2 + x + 1)$

اکنون اجازه دهید عملیات جمع را برای چندجمله‌ای‌های با ضرایبی در $\text{GF}(2)$ تعریف کنیم. جمع، بسیار ساده است؛ بدین گونه که ضریب عبارات هم درجه در $\text{GF}(2)$ را با هم جمع می‌کنیم. توجه داشته باشید که جمع دو چندجمله‌ای با درجه $n-1$ ، همیشه یک چندجمله‌ای با درجه $n-1$ است و به این معنی است که لازم نیست با استفاده از پیمانانه، نتیجه را کاهش دهیم.

اجازه دهید $(x^3 + x^2 + 1) \oplus (x^5 + x^2 + x)$ را در $GF(2^4)$ انجام دهیم، از علامت \oplus استفاده می‌کنیم تا نشان دهیم که منظور ما جمع چندجمله‌ای است. در زیر مراحل کار نشان داده شده است:

$$\bullet X^V + \bullet X^7 + 1X^0 + \bullet X^{\xi} + \bullet X^{\eta} + 1X^{\gamma} + 1X^1 + \bullet X^{\cdot} \quad \oplus$$

$$\bullet X^V + \bullet X^7 + \bullet X^0 + \bullet X^{\xi} + 1X^{\eta} + 1X^{\gamma} + \bullet X^1 + 1X^{\cdot}$$

$$\bullet X^V + \bullet X^7 + 1X^0 + \bullet X^{\xi} + 1X^r + \bullet X^{\bar{r}} + 1X^1 + 1X^{\cdot} \longrightarrow X^0 + X^r + X + 1$$

¹ Modulus Arithmetic

یک راه میانبر برای این کار به این صورت است که: عبارات غیر مشترک را نگه دارید و عبارات مشترک را حذف کنید؛ به عبارت دیگر X^0, X^3, X^5 را نگه می‌داریم و X^2 را که در هر دو چندجمله‌ای مشترک است، حذف می‌کنیم.

مثال ۴-۱۸

یک راه میانبر دیگر نیز وجود دارد، زیرا جمع در $GF(2)$ به معنی انجام عمل XOR می‌باشد؛ بنابراین می‌توانیم XOR را بر دو کلمه - بیت به بیت - اعمال کنیم تا نتیجه را به دست آوریم. در مثال پیشین $X^0 + X^2 + 2$ می‌شود 00100110 و $X^3 + X^2 + 1$ می‌شود 00101011 ، یا به شکل چندجمله‌ای می‌شود $X^0 + X^3 + X + 1$.

چندجمله‌ای همانی نسبت به جمع^۱: حاصل چندجمله‌ای خنثی نسبت به جمع، یک چندجمله‌ای صفر است (چندجمله‌ای صفر چند جمله‌ایست که تمام ضرایبش صفر است)، زیرا حاصل جمع یک چندجمله‌ای با خودش یک چندجمله‌ای صفر است.

وارون نسبت به جمع در یک چندجمله‌ای^۲: وارون نسبت به جمع در یک چندجمله‌ای با ضرایبی در $GF(2)$ ، خود یک چندجمله‌ای است. این بدان معناست که عمل تفریق مانند عمل جمع است. اعمال جمع و تفریق روی چندجمله‌ایها، یک عمل یگانه محسوب می‌شود.

ضرب

ضرب در چندجمله‌ای‌ها عبارت است از مجموع حاصل ضرب هر عبارت چندجمله‌ای اول با تمام عبارات چندجمله‌ای دوم. ولی باید سه نکته را به خاطر داشته باشیم: نخست، ضرب ضرایب در $GF(2)$ انجام می‌شود؛ دوم، نتیجه ضرب X^i در X^j می‌شود X^{i+j} ؛ سوم، ممکن است در ضرب، عباراتی با درجه بیش از $n-1$ تولید شود و این بدان معناست که لازم است با استفاده از یک چندجمله‌ای پیمانه‌ای نتیجه را کاهش دهید. نخست نشان می‌دهیم که چگونه با استفاده از تعریف بالا می‌توان چندجمله‌ای‌ها را ضرب کرد، سپس الگوریتم کارآمدتری را خواهیم دید که از آن می‌توان در برنامه‌ی کامپیوتری استفاده کرد.

¹ Additive Identity

² Additive Inverse

مثال ۴-۱۹

حاصل $(x^0 + x^2 + x) \otimes (x^0 + x^2 + x^3 + x^5 + x)$ در $GF(2^8)$ با چندجمله‌ای کاهش‌ناپذیر $(x^8 + x^4 + x^3 + x + 1)$ را به دست آورید. توجه داشته باشید که از علامت \otimes برای نشان دادن ضرب دو چندجمله‌ای استفاده می‌کنیم.

حل

مانند آنچه که در جبر آموختیم، ابتدا دو چندجمله‌ای را ضرب می‌کنیم. توجه داشته باشید در این مراحل، یک زوج عبارت با توان‌های مساوی x حذف می‌شوند. مثلاً $x^9 + x^9$ به طور کلی حذف می‌شود. زیرا، همان‌گونه که در بالا توضیح دادیم، حاصل آن‌ها یک چندجمله‌ای صفر است.

$$P_1 \otimes P_2 = x^0(x^0 + x^2 + x^3 + x^5 + x) + x^2(x^0 + x^2 + x^3 + x^5 + x) + x(x^0 + x^2 + x^3 + x^5 + x)$$

$$P_1 \otimes P_2 = x^{12} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x^0 + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2$$

$$P_1 \otimes P_2 = (x^{12} + x^9 + x^2) \bmod (x^8 + x^4 + x^3 + x + 1) = x^0 + x^3 + x^2 + x + 1$$

برای یافتن پاسخ نهایی، چندجمله‌ای درجه ۱۲ را بر چندجمله‌ای پیمانه تقسیم می‌کنیم و فقط باقیمانده را نگه می‌داریم. مراحل کار مانند همان چیزی است که در جبر یاد گرفتیم، اما باید به خاطر داشته باشیم که در اینجا، تفریق مانند جمع است. شکل ۴-۱۰ مراحل تقسیم را نشان می‌دهد.

$$\begin{array}{r}
 x^4 + 1 \\
 \hline
 x^8 + x^4 + x^3 + x + 1 \quad \left| \begin{array}{l} x^{12} + x^7 + x^2 \\ x^{12} + x^8 + x^7 + x^5 + x^4 \\ \hline x^8 + x^5 + x^4 + x^2 \\ x^8 + x^4 + x^3 + x + 1 \\ \hline \end{array} \right. \\
 \hline
 \text{Remainder } \boxed{x^5 + x^3 + x^2 + x + 1}
 \end{array}$$

شکل ۴-۱۰: تقسیم چندجمله‌ای با ضرایبی در $GF(2)$

عنصر خنثی در ضرب^۱: عنصر خنثی در ضرب همیشه ۱ است. به عنوان مثال در $GF(2^8)$ ، وارون ۱ در ضرب، الگوی بیتی ۰۰۰۰۰۰۰۱ می‌باشد.

وارون نسبت به ضرب^۲: یافتن وارون نسبت به ضرب، کمی پیچیده‌تر است. الگوریتم اقلیدسی بسط‌یافته را باید بر پیمانه و چندجمله‌ای اعمال کرد. این مراحل دقیقاً مانند مراحل اعداد صحیح است.

¹ Multiplicative Identity

² Multiplicative Inverse

مثال ۴-۲۰

در $GF(2^4)$ ، وارون (x^2+1) با پیمانه (x^4+x+1) را پیدا کنید.

حل

مانند آنچه که در جدول ۴-۵ نشان داده شده است از الگوریتم اقلیدسی بسط یافته استفاده می کنیم.

جدول ۴-۵: الگوریتم اقلیدسی مثال ۴-۲۰

q	r_1	r_2	r	t_1	t_2	t
(x^2+1)	(x^4+x+1)	(x^2+1)	(x)	(0)	(1)	(x^2+1)
(x)	(x^2+1)	(x)	(1)	(1)	(x^2+1)	(x^3+x+1)
(x)	(x)	(1)	(0)	(x^2+1)	(x^3+x+1)	(0)
	(1)	(0)		(x^3+x+1)	(0)	

این بدین معناست که $(x^2+1)^{-1}$ به پیمانه (x^4+x+1) می شود (x^3+x+1) . با ضرب دو چندجمله ای و یافتن باقیمانده، در صورتی که نتیجه بر پیمانه بخش پذیر باشد، می توان پاسخ را به سادگی اثبات کرد.

$$[(x^2+1) \otimes (x^3+x+1)] \bmod (x^4+x+1) = 1$$

مثال ۴-۲۱

در $GF(2^8)$ وارون (x^6) به پیمانه $(x^8+x^4+x^3+x+1)$ را پیدا کنید.

حل

از الگوریتم اقلیدسی به صورتی که در جدول ۴-۶ نشان داده شده است، استفاده کنید.

q	r_1	r_2	r	t_1	t_2	t
(x^6)	$(x^8+x^4+x^3+x+1)$	(x^5)	(x^4+x^3+x+1)	(0)	(1)	(x^6)
$(x+1)$	(x^5)	(x^4+x^3+x+1)	(x^3+x^2+1)	(1)	(x^3)	(x^4+x^3+1)
(x)	(x^4+x^3+x+1)	(x^3+x^2+1)	(1)	(x^3)	(x^4+x^3+1)	$(x^5+x^4+x^3+x)$
(x^2+x^2+1)	(x^3+x^2+1)	(1)	(0)	(x^4+x^3+1)	$(x^5+x^4+x^3+x)$	(0)
	(1)	(0)		$(x^6+x^4+x^3+x)$	(0)	

جدول ۴-۶: الگوریتم اقلیدسی مثال ۴-۲۱.

یعنی $(x^6)^{-1}$ به پیمانه $(x^8+x^4+x^3+x+1)$ می شود (x^6+x^3+x+1) . با ضرب دو چندجمله ای و یافتن باقیمانده، در صورتی که نتیجه به پیمانه بخش پذیر باشد، می توان پاسخ را به سادگی اثبات کرد.

$$[(x^6) \otimes (x^6+x^3+x+1)] \bmod (x^8+x^4+x^3+x+1) = 1$$

ضرب با استفاده از کامپیوتر

به خاطر وجود عمل تقسیم، در نوشتن برنامه برای ضرب دو چندجمله‌ای مشکل بزرگی وجود دارد. پس در پیاده‌سازی از الگوریتم بهتری به این صورت استفاده می‌کنیم: مکرراً یک چندجمله‌ای کاهش یافته را در x ضرب می‌کنیم، مثلاً به جای یافتن نتیجه $(X^2 \otimes P_2)$ ، برنامه نتیجه $(X \otimes (X \times P_2))$ را به دست می‌آورد. به اختصار در مورد مزیت این راهکار صحبت خواهیم کرد، نخست اجازه دهید برای نشان دادن مراحل، مثالی را مطرح کنیم.

مثال ۴-۲۲

با استفاده از الگوریتم بالا، نتیجه ضرب $P_1 = (x^0 + x^2 + x)$ در $P_2 = (x^0 + x^2 + x^3 + x^4 + x^5)$ در $GF(2^8)$ با چندجمله‌ای کاهش‌ناپذیر $(x^8 + x^4 + x^3 + x + 1)$ را به دست آورید.

حل

مراحل در جدول ۴-۷ نشان داده شده است. نخست نتیجه‌ی جزئی ضرب $x^0, x^1, x^2, x^3, x^4, x^5$ در P_2 را به دست می‌آوریم. توجه داشته باشید که اگر چه فقط سه عبارت را لازم داریم، حاصل ضرب $x^m \otimes p_2$ برای m از ۰ تا ۵ را به دست آورده‌ایم، زیرا هر محاسبه به نتیجه‌ی قبلی بستگی دارد.

جدول ۴-۷: یک الگوریتم کارآمد برای ضرب با استفاده از چندجمله‌ای‌ها (مثال ۴-۲۲)

Powers	Operation	New Result	Reduction
$x^0 \otimes P_2$		$x^7 + x^4 + x^3 + x^2 + x$	No
$x^1 \otimes P_2$	$x \otimes (x^7 + x^4 + x^3 + x^2 + x)$	$x^5 + x^2 + x + 1$	Yes
$x^2 \otimes P_2$	$x \otimes (x^5 + x^2 + x + 1)$	$x^6 + x^3 + x^2 + x$	No
$x^3 \otimes P_2$	$x \otimes (x^6 + x^3 + x^2 + x)$	$x^7 + x^4 + x^3 + x^2$	No
$x^4 \otimes P_2$	$x \otimes (x^7 + x^4 + x^3 + x^2)$	$x^5 + x + 1$	Yes
$x^5 \otimes P_2$	$x \otimes (x^5 + x + 1)$	$x^6 + x^2 + x$	No
$P_1 \times P_2 = (x^6 + x^2 + x) + (x^6 + x^3 + x^2 + x) + (x^5 + x^2 + x + 1) = x^5 + x^3 + x^2 + x + 1$			

الگوریتم بالا دو مزیت دارد. نخست با تغییر یک بیتی کلمه n بیتی می‌توان به سادگی به ضرب یک چندجمله‌ای در x - عملی که به وسیله زبان‌های کاربردی رایج به راحتی اجرا می‌شود - دست یافت. دوم، فقط اگر حداکثر توان چندجمله‌ای $n-1$ باشد، لازم است کاهش انجام شود؛ در این حالت با عمل XOR در پیمانه می‌توان کاهش را به سادگی انجام داد زیرا بالاترین توان در نتیجه، حداکثر ۸

است؛ سپس می‌توانیم با طراحی یک الگوریتم ساده به شکل زیر هر نتیجه‌ی میانی^۱ (جزئی) را به دست آوریم:

(۱) اگر پراهمیت‌ترین بیت نتیجه‌ی قبلی صفر باشد، فقط نتیجه‌ی قبلی را یک بیت به سمت چپ جابجا می‌کنیم.

(۲) اگر پراهمیت‌ترین بیت نتیجه قبلی ۱ باشد:

(a) آن را یک بیت به سمت چپ جابجا می‌کنیم

(b) سپس آن را با پراهمیت‌ترین بیت پیمانه‌ی XOR می‌کنیم.

مثال ۴-۲۳

با استفاده از الگویی به اندازه ۸ مثال ۴-۲۲ را تکرار کنید.

حل

$P_1 = 000100110$, $P_2 = 10011110$ و پیمانه برابر 100011010 (۹ بیت) است. علامت \oplus نشان

دهنده XOR می‌باشد. حاصل را در جدول ۴-۸ مشاهده نمایید.

جدول ۴-۸: الگوریتم کارآمد برای ضرب با استفاده از کلمات n بیتی.

Powers	Shift-Left Operation	Exclusive-Or
$x^0 \otimes P_2$		10011110
$x^1 \otimes P_2$	00111100	$(00111100) \oplus (00011010) = 00100111$
$x^2 \otimes P_2$	01001110	01001110
$x^3 \otimes P_2$	10011100	10011100
$x^4 \otimes P_2$	00111000	$(00111000) \oplus (00011010) = 00100011$
$x^5 \otimes P_2$	01000110	01000110
$P_1 \otimes P_2 = (00100111) \oplus (01001110) \oplus (01000110) = 00101111$		

در این حالت برای ضرب دو چندجمله‌ای فقط پنج عمل جابجایی به چپ و چهار عمل XOR را

لازم داریم. به طور کلی برای ضرب دو چندجمله‌ای با درجه $n-1$ ، حداکثر به $n-1$ عمل جابجایی به

چپ و $2n$ عمل XOR نیاز است.

می‌توان با استفاده از جابجایی به چپ و عملیات XOR
به حاصل ضرب چندجمله‌ای در $GF(2^n)$ دست یافت.

Partial Result¹

مثال ۴-۲۴

میدان $GF(2^3)$ دارای ۸ عضو است. از چندجمله‌ای کاهش‌ناپذیر (x^3+x^2+1) استفاده می‌کنیم و جدول‌های جمع و ضرب را برای این میدان تشکیل می‌دهیم. توجه داشته باشید که دو چندجمله‌ای کاهش‌ناپذیر درجه ۳ وجود دارد. چندجمله‌ای دوم (x^3+x+1) می‌باشد که حاصل آن یک جدول کاملاً متفاوت برای ضرب است. جدول ۴-۹ جمع را نشان می‌دهد. خانه‌های سایه‌دار به سادگی زوج‌های وارون جمعی را برای ما مشخص می‌کنند.

استفاده از مولد

گاهی اوقات تعیین اعضای میدان $GF(2^n)$ با استفاده از مولد ساده‌تر است. در این میدان چندجمله‌ای کاهش‌ناپذیر $P(x)$ و عضوی از میدان، مثل a ، باید در رابطه $f(a)=0$ صدق کند، به ویژه اگر g مولد میدان باشد، در آن صورت $f(g)=0$. این نکته می‌تواند کمک کند که اعضای میدان را می‌توان به صورت زیر تولید کرد:

$\{0, g, g^2, \dots, g^{n-1}\}$ به قسمی که: $N=2^n-2$ است.

جدول ۴-۹: جدول جمع برای $GF(2^3)$

\oplus	000 (0)	001 (1)	010 (x)	011 (x+1)	100 (x ²)	101 (x ² +1)	110 (x ² +x)	111 (x ² +x+1)
000 (0)	000 (0)	001 (1)	010 (x)	011 (x+1)	100 (x ²)	101 (x ² +1)	110 (x ² +x)	111 (x ² +x+1)
001 (1)	001 (1)	000 (0)	011 (x+1)	010 (x ²)	101 (x ² +1)	100 (x ² +x)	111 (x ² +x+1)	110 (x ² +x)
010 (x)	010 (x)	011 (x+1)	000 (0)	001 (1)	110 (x ² +x)	111 (x ² +x+1)	100 (x ² +x)	101 (x ² +1)
011 (x+1)	011 (x+1)	010 (x)	001 (1)	000 (0)	111 (x ² +x+1)	110 (x ² +x)	101 (x ² +1)	100 (x ²)
100 (x ²)	100 (x ²)	101 (x ² +1)	110 (x ² +x)	111 (x ² +x+1)	000 (0)	001 (1)	010 (x)	011 (x+1)
101 (x ² +1)	101 (x ² +1)	100 (x ²)	111 (x ² +x+1)	110 (x ² +x)	001 (1)	000 (0)	011 (x+1)	010 (x)
110 (x ² +x)	110 (x ² +x)	111 (x ² +x+1)	100 (x ²)	101 (x ² +1)	010 (x)	011 (x+1)	000 (0)	001 (1)
111 (x ² +x+1)	111 (x ² +x+1)	110 (x ² +x)	101 (x ² +1)	100 (x ²)	011 (x+1)	010 (x)	001 (1)	000 (0)

جدول ۴-۱۰: جدول ضرب برای $GF(2^3)$ با چندجمله‌ای کاهش‌ناپذیر (x^3+x^2+1)

\otimes	(0)	(1)	(x)	(x+1)	(x ²)	(x ² +1)	(x ² +x)	(x ² +x+1)
000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)	000 (0)
001 (1)	000 (0)	001 (1)	010 (x)	011 (x+1)	100 (x ²)	101 (x ² +1)	110 (x ² +x)	111 (x ² +x+1)
010 (x)	000 (0)	010 (x)	100 (x)	110 (x ² +x)	101 (x ² +1)	111 (x ² +x+1)	001 (1)	011 (x+1)
011 (x+1)	000 (0)	011 (x+1)	110 (x ² +x)	101 (x ² +1)	001 (1)	010 (x)	111 (x ² +x+1)	100 (x)
100 (x ²)	000 (0)	100 (x ²)	101 (x ² +1)	001 (1)	111 (x ² +x+1)	011 (x+1)	010 (x)	110 (x ² +x)
101 (x ² +1)	000 (0)	101 (x ² +1)	111 (x ² +x+1)	010 (x)	011 (x+1)	110 (x ² +x)	100 (x ²)	001 (1)
110 (x ² +x)	000 (0)	110 (x ² +x)	001 (1)	111 (x ² +x+1)	010 (x)	100 (x ²)	011 (x+1)	101 (x ² +1)
111 (x ² +x+1)	000 (0)	111 (x ² +x+1)	011 (x+1)	100 (x ²)	110 (x ² +x)	001 (1)	101 (x ² +1)	010 (x)

مثال ۴-۲۵

با استفاده از چندجمله‌ای کاهش ناپذیر $f(x)=x^4+x+1$ اعضای دامنه $GF(2^4)$ را تولید کنید.

حل

اعضای $g^0, g^1, g^2, g^3, g^4, g^5, g^6, g^7, g^8, g^9, g^{10}, g^{11}, g^{12}, g^{13}, g^{14}$ به سادگی ایجاد می‌شوند، زیرا آن‌ها نمایش چهار بیتی به شکل $x^3, x^2, x, 1$ هستند (در اینجا انجام تقسیم چندجمله‌ای لازم نیست). اعضای g^4 تا g^{14} را، که x^4 تا x^{14} را نشان می‌دهند، باید بر چندجمله‌ای کاهش ناپذیر تقسیم کرد. برای گریز از تقسیم چندجمله‌ای می‌توان از رابطه $f(g)=g^4+g+1=0$ استفاده کرد. با استفاده از این رابطه، $g^4=-g-1$ را داریم. از آن جا که در این میدان جمع و تفریق عمل یکسانی است، $g^4=g+1$ است. برای یافتن مقدار تمام اعضا به عنوان کلمات چهار بیتی، از این رابطه استفاده می‌کنیم.

$0 = 0$	$= 0$	$= 0$	\longrightarrow	$0 = (0000)$
$g^0 = g^0$	$= g^0$	$= g^0$	\longrightarrow	$g^0 = (0001)$
$g^1 = g^1$	$= g^1$	$= g^1$	\longrightarrow	$g^1 = (0010)$
$g^2 = g^2$	$= g^2$	$= g^2$	\longrightarrow	$g^2 = (0100)$
$g^3 = g^3$	$= g^3$	$= g^3$	\longrightarrow	$g^3 = (1000)$
$g^4 = g^4$	$= g^4$	$= g+1$	\longrightarrow	$g^4 = (0011)$
$g^5 = g(g^4)$	$= g(g+1)$	$= g^2+g$	\longrightarrow	$g^5 = (0110)$
$g^6 = g(g^5)$	$= g(g^2+g)$	$= g^3+g^2$	\longrightarrow	$g^6 = (1100)$
$g^7 = g(g^6)$	$= g(g^3+g)$	$= g^3+g+1$	\longrightarrow	$g^7 = (1011)$
$g^8 = g(g^7)$	$= g(g^3+g+1)$	$= g^2+1$	\longrightarrow	$g^8 = (0101)$
$g^9 = g(g^8)$	$= g(g^2+1)$	$= g^3+g$	\longrightarrow	$g^9 = (1010)$
$g^{10} = g(g^9)$	$= g(g^3+g)$	$= g^2+g+1$	\longrightarrow	$g^{10} = (0111)$
$g^{11} = g(g^{10})$	$= g(g^2+g+1)$	$= g^3+g^2+g$	\longrightarrow	$g^{11} = (1110)$
$g^{12} = g(g^{11})$	$= g(g^3+g^2+g)$	$= g^3+g^2+g+1$	\longrightarrow	$g^{12} = (1111)$
$g^{13} = g(g^{12})$	$= g(g^3+g^2+g+1)$	$= g^3+g^2+1$	\longrightarrow	$g^{13} = (1101)$
$g^{14} = g(g^{13})$	$= g(g^3+g^2+1)$	$= g^3+1$	\longrightarrow	$g^{14} = (1001)$

هدف اصلی کاهش عبارت g^1 تا g^4 جهت ترکیب عبارات g^3, g^2, g ، ۱ با استفاده از رابطه $g^4 = g + 1$ است. به عنوان مثال،

$$g^{12} = g(g^{11}) = g(g^3 + g^2 + 1) = g^4 + g^3 + g^2 = g^3 + g^2 + g + 1$$

پس از کاهش، انتقال توان‌ها به یک کلمه n بیتی ساده است. به عنوان مثال $g^3 + 1$ برابر است با ۱۰۰۱ زیرا فقط توان‌های ۰ و ۳ وجود دارند. توجه داشته باشید که در این مراحل، دو عبارت مساوی یکدیگر را حذف می‌کنند. به عنوان مثال، $g^2 + g^2 = 0$.

وارون‌ها^۱

یافتن وارون‌ها با استفاده از این روش ساده است.

وارون نسبت به جمع^۲

وارون نسبت به جمع هر عضو، خود آن عضو می‌باشد، زیرا در این میدان جمع و تفریق یکی است: $-g^3 = g^3$.

وارون نسبت به ضرب^۳

یافتن وارون نسبت به ضرب هر عضو نیز بسیار ساده است. مثلاً می‌توانیم به شیوه‌ی زیر وارون عددی را نسبت به ضرب به دست آوریم:

$$(g^3)^{-1} = g^{-3} = g^{12} = g^3 + g^2 + g + 1 \rightarrow (1111)$$

توجه داشته باشید که توان‌ها به پیمانه $2^n - 1$ محاسبه شده‌اند که در این مثال ۱۵ است؛ بنابراین ۱۵

$mod \ 15 = 12 \ mod \ 3$ به سادگی می‌توان ثابت کرد که g^3 و g^{12} وارون یکدیگرند، زیرا

$$g^3 * g^{12} = g^{15} = g^0 = 1$$

عملیات‌ها^۴

با استفاده از این شیوه می‌توان چهار عمل تعریف شده برای این میدان انجام داد.

=====

¹ Inverses

² Additive Inverses

³ Multiplicative Inverses

⁴ Operation

جمع و تفریق

جمع و تفریق یک عمل یگانه محسوب می‌شوند. همان‌گونه که در مثال زیر نشان داده شده است، نتایج واسطه^۱ را می‌توان ساده کرد.

مثال ۴-۲۶

نتایج اعمال جمع و تفریق در زیر نشان داده شده است:

$$g^3 + g^{12} + g^9 = g^3 + (g^3 + g^2 + g + 1) + (g^3 + g + 1) = g^3 + g^2 \rightarrow (1100) \quad (a)$$

$$g^3 - g^6 = g^3 + (g^3 + g^2) = g^2 \rightarrow (0100) \quad (b)$$

ضرب و تقسیم

ضرب، جمع توان‌های پیمانه $2^n - 1$ می‌باشد. تقسیم، همان ضرب با استفاده از وارون نسبت به ضرب است.

مثال ۴-۲۷

نتیجه اعمال ضرب و تقسیم در زیر نشان داده شده است:

$$g^9 * g^{11} = g^{20} = g^{20 \bmod 15} = g^5 = g^2 + g \rightarrow (01101) \quad (a)$$

$$g^3 / g^8 = g^3 * g^9 = g^{12} = g^2 + g + 1 \rightarrow (0111) \quad (b)$$

خلاصه

از میدان متناهی $GF(2^n)$ می‌توان برای تعیین چهار عمل جمع، تفریق، ضرب و تقسیم بر روی کلمات n بیتی استفاده کرد. تنها محدودیت این است که تقسیم بر صفر تعریف نشده است. هر کلمه‌ی n بیتی را می‌توان به صورت یک چندجمله‌ای با درجه‌ی $n-1$ و ضریب‌هایی در $GF(2)$ نشان داد و این به معنی این است که انجام عمل بر روی کلمات n بیتی با اعمال آن بر روی چندجمله‌ای یکسان است. برای اعمال پیمانه می‌بایست وقتی که دو چندجمله‌ای را در هم ضرب کرده و یک چندجمله‌ای کاهش‌ناپذیر با درجه n تعیین کنیم. برای یافتن وارون‌های ضربی می‌توانیم الگوریتم اقلیدسی بسط‌یافته را روی چندجمله‌ای‌ها اعمال کنیم.

=====

¹ Intermediate Results

۴-۳- چکیده

- * رمزنگاری متکی به مجموعه‌ها و اعمال خاص تعریف شده برای آن‌ها می‌باشد. ترکیب مجموعه و عملگرهای اعمال شده بر اعضای مجموعه را یک ساختار جبری می‌نامند. در این فصل، سه ساختار جبری معرفی شده است: گروه‌ها، حلقه‌ها و میدان‌ها.
- * گروه عبارت است از یک ساختار جبری با یک عمل دودویی به طوری که چهار ویژگی داشته باشد: ویژگی بسته بودن، ویژگی شرکت‌پذیری، وجود عضو خنثی، وجود وارون. گروه جابجاپذیر، که به آن گروه آبدی می‌گویند، گروهی است که در آن عملگر تعیین شده، یک ویژگی دیگر را با عنوان خاصیت جابجایی برآورده می‌سازد.
- * یک زیرمجموعه H از یک گروه G ، یک زیرگروه G محسوب می‌شود به شرطی که خود H با توجه به عملگر تعریف شده در G ، یک گروه محسوب شود. اگر با استفاده از توان‌های یک عضو از زیرگروه بتوان گروهی را ایجاد کرد، به این زیرگروه، زیرگروه دوار می‌گویند. یک گروه دوار گروهی است که زیرگروه دوار خود باشد.
- * قضیه لاگرانژ، ترتیب یک گروه را با ترتیب زیرگروهش مرتبط می‌سازد. اگر ترتیب G و H به ترتیب $|G|$ و $|H|$ باشند در آن صورت $|G|$ بر $|H|$ بخش‌پذیر است.
- * ترتیب یک عضو a در گروه عبارت است از کوچک‌ترین عدد صحیح مثبت n که $a^n = e$.
- * حلقه یک ساختار جبری با دو عملگر می‌باشد. اولین عملگر باید تمام پنج ویژگی اشاره شده برای گروه آبدی را برآورده نماید. ویژگی دوم فقط باید دو ویژگی اول را داشته باشد. علاوه بر این عملگر دوم باید بر عملگر اول توزیع‌پذیر باشد. حلقه‌ی جابجایی، حلقه‌ای است که در آن ویژگی جابجایی برای عمل دوم نیز وجود دارد.
- * میدان، یک حلقه جابجایی است که در آن عمل دوم تمام پنج ویژگی تعیین شده برای عمل اول را به استثنای این که عضو خنثای عمل اول فاقد وارون است را دارد. میدان متناهی که به آن میدان گالوا هم می‌گویند میدانی با p^n عضو است، به طوری که p عدد اول و n یک عدد صحیح است. میدان‌های $GF(p^n)$ در رمزنگاری برای انجام اعمال ریاضی بر روی کلمات n بیتی مورد استفاده قرار می‌گیرد.
- * چندجمله‌ای‌ها با ضرایبی در $GF(2)$ برای نشان دادن کلمات n بیتی به کار می‌رود. جمع و ضرب روی کلمات n بیتی را می‌توان به عنوان جمع و ضرب چندجمله‌ای تعریف کرد.

* گاهی اوقات تعیین اعضای دامنه $GF(2^n)$ با استفاده از مولد ساده‌تر است. اگر g یک مولد میدان باشد پس $f(g)=0$ است. یافتن وارون‌ها و انجام اعمال روی اعضای میدان زمانی ساده‌تر می‌شود که اعضا به شکل توان‌های مولد نشان داده شده باشند.

۴-۴- مجموعه تمرین‌ها

پرسش‌های دوره‌ای

- (۱) ساختارهای جبری را تعریف کرده و سه ساختار اشاره شده در این فصل را نام ببرید.
- (۲) گروه را تعریف کرده و تفاوت بین گروه و گروه جابجایی را بیان کنید.
- (۳) حلقه را تعریف کرده و تفاوت بین حلقه و حلقه جابجایی را بیان کنید.
- (۴) میدان را تعریف کرده و تفاوت بین میدان متناهی و میدان نامتناهی را بیان کنید.
- (۵) با استفاده از مجموعه باقیمانده‌ها، مثالی از گروه ارائه دهید.
- (۶) با استفاده از مجموعه باقیمانده‌ها، مثالی از حلقه ارائه دهید.
- (۷) با استفاده از مجموعه باقیمانده‌ها، مثالی از میدان ارائه دهید.
- (۸) نشان دهید که چگونه یک چندجمله‌ای می‌تواند یک کلمه n بیتی را نشان دهد.
- (۹) چندجمله‌ای کاهش‌ناپذیر را تعریف کنید.
- (۱۰) چندجمله‌ای کاهش‌ناپذیر را تعریف کنید.

تمرین‌ها

- (۱۱) برای گروه $G = \langle \mathbb{Z}_4, + \rangle$:
 - (a) ثابت کنید که این گروه، یک گروه آبلی است.
 - (b) نتیجه $3+2$ و $3-2$ را نشان دهید.
- (۱۲) برای گروه $G = \langle \mathbb{Z}^*, \times \rangle$:
 - (a) ثابت کنید که این گروه، یک گروه آبلی است.
 - (b) نتیجه 5×1 و $5 \div 1$ را نشان دهید.
 - (c) نشان دهید که چرا نباید نگران تقسیم بر صفر در این گروه باشیم.
- (۱۳) تنها یک عمل برای گروه موجود در جدول ۴-۱ تعیین شده است. فرض کنید این عمل جمع است. جدول عملیات تفریق (عمل وارون جمع) را نشان دهید.
- (۱۴) ثابت کنید که گروه جایگشت در جدول ۴-۲ مبادله‌ای نیست.

(۱۵) ثابت کنید که گروه جایگشت در جدول ۲-۴ با افزودن چند عضو جدید ویژگی جابجایی پیدا می‌کند.

(۱۶) یک جدول جایگشت شبیه به جدول ۲-۴ برای دو ورودی و دو خروجی ایجاد کنید.

(۱۷) آلیس از سه جایگشت متوالی [۱۲۳] و [۳۲۴] و [۲۱۳] استفاده می‌کند. نشان دهید که چگونه باب می‌تواند فقط از یک جایگشت برای آشکار شدن فرآیند استفاده کند. از جدول ۲-۴ استفاده نمایید.

(۱۸) تمام زیرگروه‌های، گروه‌های زیر را پیدا کنید.

- a) $G = \langle Z_{16}, + \rangle$
- b) $G = \langle Z_{23}, + \rangle$
- c) $G = \langle Z_{16}^*, \times \rangle$
- d) $G = \langle Z_{17}^*, \times \rangle$

(۱۹) با استفاده از قضیه‌ی لاگرانژ، ترتیب تمام زیرگروه‌های بالقوه گروه‌های زیر را پیدا کنید.

- a) $G = \langle Z_{18}, + \rangle$
- b) $G = \langle Z_{29}, + \rangle$
- c) $G = \langle Z_{12}^*, \times \rangle$
- d) $G = \langle Z_{19}^*, \times \rangle$

(۲۰) ترتیب تمام اعضای موجود در گروه‌های زیر را پیدا کنید.

- a) $G = \langle Z_8, + \rangle$
- b) $G = \langle Z_7, + \rangle$
- c) $G = \langle Z_9^*, \times \rangle$
- d) $G = \langle Z_{19}^*, \times \rangle$

(۲۱) مثال ۲۵-۴ را با استفاده از چندجمله‌ای $f(x) = x^4 + x^3 + 1$ دوباره انجام دهید.

(۲۲) مثال ۲۶-۴ را با استفاده از چندجمله‌ای $f(x) = x^4 + x^3 + 1$ دوباره انجام دهید.

(۲۳) مثال ۲۷-۴ را با استفاده از چندجمله‌ای $f(x) = x^4 + x^3 + 1$ دوباره انجام دهید.

(۲۴) کدام یک از موارد زیر میدان معتبر گالوا محسوب می‌شود؟

- a) $GF(12)$
- b) $GF(13)$
- c) $GF(16)$

d) $GF(17)$

(۲۵) برای هر یک از کلمات n بیتی زیر، چندجمله‌ای را پیدا کنید که بیانگر آن کلمه باشد.

a) 10010

b) 10

c) 100001

d) 00011

(۲۶) هر یک از چندجمله‌ای‌های زیر معرف چه کلمه n بیتی هستند؟

a) $X^{2^4} + 1$ in $GF(2^4)$

b) $X^{2^4} + 1$ in $GF(2^6)$

c) $X + 1$ in $GF(2^2)$

d) X^2 in $GF(2^8)$

(۲۷) در دامنه $GF(7)$ نتیجه موارد زیر را به دست آورید.

a) $5+3$

b) $5-4$

c) 3×5

d) $5 \div 3$

(۲۸) ثابت کنید که (x) و $(x+1)$ چندجمله‌ای کاهش ناپذیر درجه ۱ هستند.

(۲۹) ثابت کنید که (x^2+x+1) یک چندجمله‌ای کاهش ناپذیر درجه ۲ است.

(۳۰) ثابت کنید که (x^3+x^2+1) یک چندجمله‌ای کاهش ناپذیر درجه ۳ است.

(۳۱) با استفاده از چندجمله‌ای‌ها، کلمات n بیتی زیر را ضرب کنید.

a) $(10) \times (11)$

b) $(1000) \times (1010)$

c) $(10000) \times (11100)$

(۳۲) وارون نسبت به ضرب چندجمله‌ای‌های زیر در $GF(2^2)$ را به دست آورید؛ توجه داشته باشید

که فقط یک پیمانه برای این میدان وجود دارد.

a) ۱

b) X c) $X+1$

(۳۳) از الگوریتم اقلیدسی بسط یافته برای یافتن وارون (X^4+X^3+1) در $GF(2^5)$ با استفاده از پیمانه‌ی (X^5+X^2+1) استفاده کنید.

(۳۴) با استفاده از (X^4+X^3+1) به عنوان پیمانه، جدول جمع و ضربی در $GF(2^4)$ ایجاد کنید.

(۳۵) با استفاده از جدول ۴-۱۰ اعمال زیر را انجام دهید:

a) $(100) \div (010)$ b) $(100) \div (000)$ c) $(101) \div (011)$ d) $(000) \div (111)$

(۳۶) نشان دهید که چگونه می‌توان با استفاده از الگوریتم موجود در جدول ۴-۷ ضرب (X^3+X^2+X+1) در (X^2+1) در میدان $GF(2^4)$ را انجام داد. از (X^4+X^3+1) به عنوان پیمانه استفاده کنید.

(۳۷) نشان دهید که چگونه می‌توان با استفاده از الگوریتم موجود در جدول ۴-۸ ضرب (10101) در (10000) در $GF(5^5)$ را انجام داد. از (X^5+X^2+1) به عنوان پیمانه استفاده کنید.

فصل ۵

مقدمه‌ای بر رمزهای کلید متقارن پیشرفته

چشم‌انداز

این فصل، چشم‌اندازهای گوناگونی به شرح زیر دارد:

- ✱ مشخص کردن تفاوت بین رمزهای کلید متقارن پیشرفته و سنتی.
- ✱ معرفی رمزهای بلوکی پیشرفته و بحث و بررسی ویژگی‌های آن‌ها.
- ✱ توضیح این که چرا باید رمزهای بلوکی پیشرفته را همانند رمزهای جایگزینی طراحی کرد.
- ✱ معرفی اجزاء رمزهای بلوکی نظیر جعبه‌های P و جعبه‌های S.
- ✱ بحث و بررسی رمز فرآورده^۱ و تفاوت بین رسته‌های رمزهای فرآورده: رمزهای Feistel و غیر Feistel.
- ✱ بحث و بررسی دو نوع حمله‌ای که خاص رمزهای بلوکی پیشرفته طراحی شده است: تحلیل رمز دیفرانسیلی و خطی^۲.
- ✱ معرفی رمزهای جریانی و تفاوت بین رمزهای جریانی همگام^۳ و ناهمگام^۴.
- ✱ بحث و بررسی شیفت ریجسترهای پس‌خور^۵ برای اجرای رمزهای جریانی.

=====

¹ Product Cipher

² Differential And Linear Cryptanalysis

³ Synchronous

⁴ Non- Synchronous

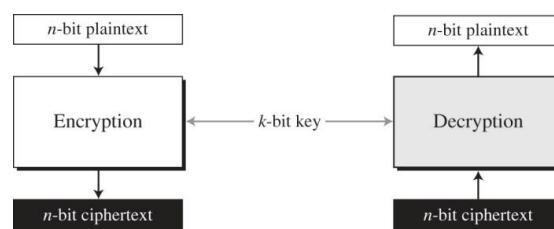
⁵ Feedback Shift Registers

رمزهای کلید متقارن سنتی که تاکنون مطالعه کردیم، رمزهای مبتنی بر نویسه^۱ هستند. با پیدایش کامپیوتر، به رمزهای مبتنی بر بیت^۲ نیاز داریم، چون اطلاعاتی که باید رمزگذاری شود فقط متن نیست. اطلاعات ممکن است شامل اعداد، گرافیک، فایل‌های صوتی و داده‌های تصویری باشد. تبدیل این نوع از داده‌ها به جریانی از بیت‌ها، رمزنگاری این جریان و سپس ارسال جریان رمزنگاری شده کار ساده‌ای است. علاوه بر این وقتی که با متن در سطح بیت سروکار داریم ۸ (یا ۱۶) بیت جایگزین هر نویسه می‌شود و این بدان معنی است که تعداد نشانه‌ها (سمبل‌ها) ۸ (یا ۱۶) برابر بزرگ‌تر می‌شود. گنجاندن تعداد بیشتر از نشانه‌ها (سمبل‌ها)، باعث بالا رفتن امنیت می‌شود.

این فصل پیش‌نیاز لازم برای مطالعه‌ی رمزهای جریانی و بلوکی پیشرفته که در سه فصل آینده مورد بحث و بررسی قرار خواهد گرفت را در اختیار شما قرار می‌دهد. بخش بزرگی از این فصل به بحث در خصوص پندارهای کلی درباره‌ی رمزهای بلوکی پیشرفته اختصاص داده شده است. بخش کوچکی نیز به بحث در مورد اصول رمزهای جریانی پیشرفته می‌پردازد.

۵-۱- رمزهای بلوکی پیشرفته^۳

رمز بلوکی پیشرفته‌ی کلید متقارن، متن عادی به اندازه n بیتی را با هم رمزنگاری یا رمزگشایی می‌نماید. الگوریتم رمزنگاری یا رمزگشایی از یک کلید k بیتی استفاده می‌کند. الگوریتم رمزگشایی باید وارون الگوریتم رمزنگاری باشد و هر دو باید از کلید سری یکسانی استفاده نمایند تا باب بتواند رمز ارسالی از سوی آلیس را بازیابی کند. شکل ۵-۱ ایده کلی رمزنگاری و رمزگشایی در رمز بلوکی پیشرفته را نشان می‌دهد.



شکل ۵-۱: رمز بلوکی پیشرفته

=====

¹ Character-oriented Ciphers

² Bit-Oriented Ciphers

³ Modern Block Ciphers

اگر پیام کمتر از n بیت داشته باشد، باید برای ایجاد یک بلوک n بیتی، تعدادی بیت padding به آخر بلوک اضافه شود. اگر پیام بیشتر از n بیت داشته باشد، باید به بلوک‌های n بیتی تقسیم شود و در صورت لزوم باید بیت padding به تعداد مورد نیاز به بلوک آخر اضافه گردد. مقادیر رایج n عبارتند از: ۶۴، ۱۲۸، ۲۵۶ و ۵۱۲ بیت.

مثال ۵-۱

چه تعداد padding باید به پیام ۱۰۰ نویسه‌ای اضافه شود که اگر از کد ASCII ۸ بیتی استفاده شود و رمز بلوکی، بلوک‌های به اندازه ۶۴ بیتی را پذیرش نماید؟

حل

حاصل رمزگذاری ۱۰۰ نویسه با استفاده از ASCII هشت بیتی، یک پیام ۸۰۰ بیتی است. متن اولیه باید بر ۶۴ بخش پذیر باشد. اگر $|M|$ طول پیام و $|pad|$ طول padding باشد،

$$|M| + |Pad| = 0 \pmod{64} \rightarrow |Pad| = -800 \pmod{64} \rightarrow 32 \pmod{64}$$

یعنی ۳۲ بیت padding (مثلاً ۳۲ بیت صفر) باید به پیام اضافه شود، پس متن اولیه متشکل از ۸۳۲ بیت یا سیزده بلوک ۶۴ بیتی می‌باشد. توجه داشته باشید که فقط بلوک آخر دارای padding است. جهت ایجاد سیزده بلوک متن رمزی، رمزنگار سیزده بار از الگوریتم رمزنگاری استفاده می‌کند.

رمز جایگزینی یا انتقال

رمز بلوکی پیشرفته را می‌توان طوری طراحی کرد که به عنوان رمز جایگزینی یا رمز انتقال عمل کند. این ایده در رمزهای سنتی نیز به کار رفته است، با این تفاوت که نشانه‌هایی (سمبل‌هایی) که می‌بایست جایگزین یا منتقل شوند، به جای بیت، نویسه بود. اگر رمزنویسی در قالب رمز جایگزینی طراحی شده باشد، بیت ۰ یا ۱ می‌تواند جایگزین یک بیت ۱ یا یک بیت ۰ در متن عادی شود. این بدین معناست که متن رمز می‌تواند از نظر تعداد ۱ها متفاوت باشند. یک بلوک متن اولیه ۶۴ بیت با ۱۲ صفر و ۵۲ یک را می‌توان به صورت متن رمز با ۳۴ صفر و ۳۰ یک رمزنگاری کرد. اگر رمزنویسی در قالب رمز انتقال طراحی شده باشد، فقط ترتیب بیت‌ها عوض می‌شود (جابجا می‌شوند)؛ تعداد ۱ها در متن عادی و متن رمز یکسان است. در هر یک از این موارد تعداد بیت‌های متن عادی یا متن‌های رمز احتمالی 2^n است، زیرا هر یک از بیت‌های در n بلوک می‌تواند یک از دو مقدار ۰ یا ۱ را داشته باشند.

رمزهای بلوکی پیشرفته به شکل رمزهای جایگزینی طراحی شده‌اند زیرا همان‌گونه که مثال بعدی نشان می‌دهد ویژگی‌های ذاتی رمز انتقالی (حفظ تعداد ۱ها و ۰ها) باعث آسیب پذیری رمز در برابر حملات از نوع جستجوی کلیدی حالات کلید^۱ می‌شود.

مثال ۵-۲

فرض کنید یک رمز بلوکی داریم که $n=64$ است. اگر ده عدد ۱ در متن رمز وجود داشته باشد، رمز شکن باید چند بار آزمون و خطا را انجام دهد تا متن عادی را از متن رمز استراق سمع شده در هر یک از حالات زیر بازیابی کند؟

(a) متن رمز به شکل رمز جایگزین طراحی شده است.

(b) متن رمز به شکل رمز انتقال طراحی شده است.

حل

(a) در حالت اول (جایگزینی) رمز شکن نظری در مورد تعداد ۱های موجود در متن عادی ندارد. رمز شکن باید تمام بلوک‌های شصت و چهار بیتی (2^{64}) احتمالی را آزمایش نماید تا بتواند حالتی را که معنی دارد پیدا کند. اگر رمز شکن بتواند در هر ثانیه یک میلیارد بلوک را آزمایش کند، باز هم به طور متوسط صدها سال طول می‌کشد تا موفق شود.

(b) در حالت دوم (انتقال) رمز شکن می‌داند که دقیقاً ده عدد ۱ در متن عادی وجود دارد، زیرا انتقال، تعداد ۱ها (یا صفر) را در متن رمز عوض نمی‌کند. او می‌تواند با استفاده از بلوک‌های ۶۴ بیتی‌ای که دقیقاً ده عدد ۱ دارند، اقدام به حمله از گونه‌ی جستجوی جامع کلیدی حالات کلید نماید. فقط ۸۱۶، ۲۱۴، ۴۷۳، ۱۵۱ = $[10! / 64!]$ از کلمات شصت و چهار بیتی 2^{64} وجود دارد که دقیقاً ده عدد یک دارند. اگر او بتواند در هر ثانیه یک میلیارد آزمایش انجام دهد، پس در ظرف کمتر ۳ دقیقه می‌تواند تمام آن‌ها را آزمایش کند.

برای پایداری رمز بلوکی پیشرفته در برابر حمله جستجوی جامع، این رمز می‌بایست به شکل رمز جایگزینی طراحی شود.

=====

¹ Exhaustive – Search Attacks

رمزهای بلوکی به شکل گروه‌های جایگشتی^۱

همان‌گونه که در فصول بعد خواهیم دید، لازم است بدانیم که آیا یک رمز بلوکی پیشرفته گروه است یا نه (برای یادآوری گروه به فصل ۴ رجوع کنید). برای پاسخ به این پرسش، نخست فرض کنید که کلید به اندازه کافی برای انتخاب هر گونه نگاشت^۲ ممکن از ورودی به خروجی طولانی باشد. این شکل کلید را کلید تمام سائز می‌نامیم. با این وجود، در عمل اندازه کلید کوچک‌تر است؛ فقط برخی از نگاشت از ورودی تا خروجی امکان‌پذیر است. اگر چه یک رمز بلوکی مستلزم داشتن کلیدی است که بین فرستنده و گیرنده سری است، اما اجزاء بدون کلید نیز وجود دارند که در رمزنگاری به کار می‌روند.

رمزهای کلید تمام سائز^۳

اگر چه رمزهای کلید تمام سائز در عمل مورد استفاده قرار نمی‌گیرند، اما برای درک مبحث رمزهای کلید با اندازه‌ی محدود^۴، نخست در مورد این مقوله صحبت می‌کنیم.

رمزهای بلوکی انتقال کلید تمام سائز^۵: یک رمز بلوکی انتقال کلید تمام سائز فقط بیت‌ها را بدون تغییر ارزش آن‌ها جابجا می‌کند، بنابراین می‌توان آن را به شکل یک جایگشت n شی با مجموعه‌ای از جدول‌های جایگشت $n!$ طراحی کرد، که در آن کلید تعیین می‌کند که باب و آلیس کدام جدول را استفاده نمایند. باید $n!$ کلید احتمالی داشته باشیم، بنابراین کلید باید $[\log_2 n!]$ بیت داشته باشد.

مثال ۵-۳

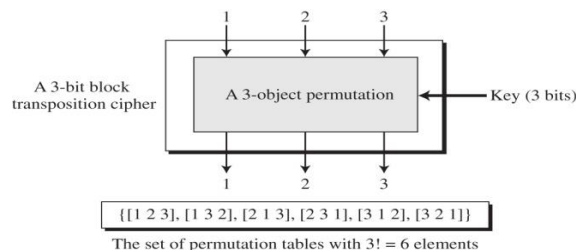
پیمانه و مجموعه‌ی جدول‌های جایگشت را برای یک رمز انتقال بلوکی ۳ بیتی نشان دهید، با فرض اینکه اندازه بلوک ۳ بیت باشد.

حل

همان‌گونه که در شکل ۵-۲ نشان داده شد، مجموعه‌ی جدول‌های جایگشت $3! = 6$ عضو دارد. طول کلید باید $[\log_2 6] = 3$ بیت باشد. توجه داشته باشید که اگر چه یک کلید ۳ بیتی می‌تواند $2^3 = 8$ نگاشت مختلف انتخاب کند، اما فقط از ۶ عدد از آن‌ها استفاده می‌کنیم.

=====

¹ Block Ciphers as Permutation Groups
² Mapping
³ Full-Size Key Ciphers
⁴ Partial-Size Key
⁵ Full-Size Key Transposition Block Ciphers



شکل ۵-۲: رمز بلوکی انتقال طراحی شده به شکل جایگشتی

رمزهای بلوکی جایگزینی کلید تمام سائز^۱: رمز جایگزینی کلید تمام سائز، بیت‌ها را جابجا نمی‌کند بلکه آن‌ها را جایگزین می‌کند. در نگاه اول چنین به نظر می‌رسد که رمز جایگزینی کلید تمام سائز را نمی‌توان به شکل جایگشت بسازیم، اما اگر بتوانیم کد ورودی و خروجی را بشکنیم، می‌توانیم رمز جایگزینی را به شکل یک جایگشت درست کنیم. شکستن کد یعنی انتقال یک عدد صحیح n بیتی به یک رشته 2^n بیتی با تنها یک عدد 1 و $2^n - 1$ عدد صفر. جایگاه تنها 1 در رشته n بیتی، مقدار عدد صحیح است که در جایگاهی از 0 تا $2^n - 1$ متغیر است. دیکدر، رویکرد وارون مراحل بالاست. از آنجا که ورودی و خروجی همیشه یک عدد 1 دارند، رمز را می‌توان به شکل یک جایگشت 2^n شیء ساخت.

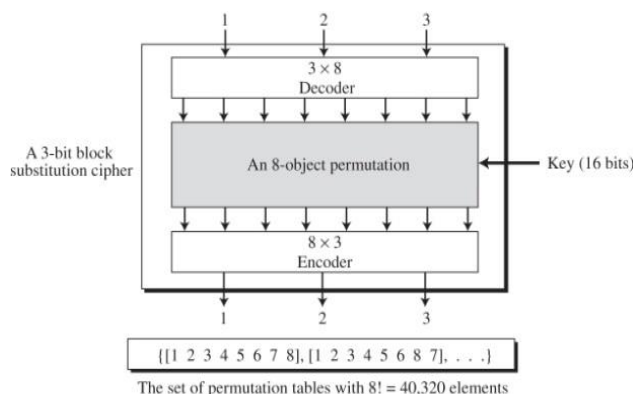
مثال ۵-۴

مدل و مجموعه جدول‌های جایگشت را برای یک رمز جایگزینی بلوکی ۳ بیتی نشان دهید.

حل

سه ورودی متن عادی می‌تواند عدد صحیحی بین 0 و 7 باشد. این را می‌توان به صورت یک رشته 8 بیتی با فقط یک عدد 1 کدگذاری کرد. به عنوان مثال 000 را می‌توان به شکل 00000001 و 101 را به صورت 00100000 کدگذاری کرد. شکل ۵-۳ مدل و مجموعه جدول‌های جایگشت را نشان می‌دهد. توجه داشته باشید که تعداد اعضای این مجموعه به ظاهر بزرگ‌تر از تعداد اعضای موجود در رمز انتقال است ($8! = 40320$). کلید هم بسیار طولانی‌تر است، $16 \text{ bit} = \lceil \log_2 40320 \rceil$ اگر چه یک کلید 16 بیتی می‌تواند 56536 نگاشت متفاوت را تولید کند، اما فقط 40320 عدد از آن‌ها مورد استفاده قرار می‌گیرد.

^۱ Full-Size Key Substitution Block Ciphers



شکل ۵-۳: یک مدل رمز بلوکی جایگزینی به شکل جایگشت

رمز انتقال n بیتی کلید تمام سائز با یک رمز بلوکی جایگزینی را می‌توان به صورت جایگشتی

ترسیم نمود که اندازه‌ی کلیدهایشان متفاوت است:

برای رمز انتقال طول کلید $\lceil \log_2 n! \rceil$ بیت است.

برای رمز جایگزین، طول کلید $\lceil \log_2 (2^n)! \rceil$ بیت است.

گروه جایگشت^۱: رمز جایگزین یا انتقال کلید تمام سائز، جایگشتی است که نشان می‌دهد اگر رمزنگاری (یا رمزگشایی) بیش از یک مرحله از هر یک از این رمزها استفاده کند، نتیجه با یک گروه جایگشت تحت عمل ترکیب برابر است. همان‌گونه که در فصل ۴ عنوان شد، یک جایگشت تکی را می‌توان همیشه جایگزین دو یا بیش از دو جایگشت آبخاری^۲ نمود. این بدان معنی است که داشتن بیش از یک مرحله در رمز کلید تمام سائز بی‌فایده است، زیرا تأثیر آن مانند داشتن یک مرحله تکی است.

رمزهای کلید اندازه‌ی محدود^۳

رمزهای واقعی نمی‌توانند از کلیدهای تمام سائز استفاده کنند زیرا اندازه‌ی کلید به ویژه برای یک رمز بلوکی جایگزین، بسیار بزرگ می‌شود. به عنوان مثال، اگر یک رمز جایگزینی عادی مثل DES، از کلید تمام سائز استفاده کرده باشد، کلیدش به اندازه $\log_2(2^{64})! \approx 2^{70}$ بیت خواهد بود. اندازه کلید برای DES فقط ۵۶ بیت است که کسر کوچکی از کلید تمام سائز می‌باشد. یعنی DES از میان $2^{(2^{70})}$ نگاشت احتمالی فقط از 2^{56} نگاشت استفاده می‌کند.

^۱Permutation Group

^۲Cascaded

^۳Partial- Size Key Cipher

گروه با ویژگی جایگشت^۱: اکنون پرسش این است که آیا جایگزینی یا انتقال کلید محدود چند مرحله-های یک گروه با خاصیت جایگشت تحت عمل ترکیب است یا نه؟ این پرسش بی‌نهایت حائز اهمیت است زیرا مشخص می‌کند که آیا نسخه‌ی چند مرحله‌ای از همان رمز را می‌توان برای دستیابی به امنیت بیشتر ایجاد کرد یا نه؟ (به مبحث DES چندگانه در فصل ۶ مراجعه کنید). رمز، کلید محدود زمانی یک گروه محسوب می‌شود که زیرگروه رمز، کلید تمام ساینز متناظر باشد؛ به عبارت دیگر، اگر رمز، کلید تمام ساینز گروه $G = \langle M, \circ \rangle$ را تولید نماید، طوری که M مجموعه‌ای از نگاشت‌ها و عمل ترکیب (\circ) باشد، در آن صورت رمز کلید محدود، باید زیرگروه $H = \langle N, \circ \rangle$ را بسازد، به طوری که N زیرمجموعه-ی M و عملگرش همان عملگر گروه باشد.

مثلاً ثابت شده است که DES چند مرحله‌ای، با کلید ۵۶ بیتی گروه نیست، زیرا هیچ زیرگروهی با نگاشت 2^{56} را نمی‌توان از گروه متناظر با نگاشت‌های 2^{64} ایجاد کرد.

رمز، کلید یک گروه تحت عمل ترکیب است به شرطی که این رمز، یک زیرگروه رمز کلید تمام ساینز متناظرش باشد.

رمزهای بدون کلید^۲

اگر چه عملاً رمز بدون کلید به تنهایی کاربرد ندارد، اما از آن‌ها به عنوان بخشی از رمزهای کلیددار استفاده می‌کنند.

رمزهای انتقال بدون کلید^۳: یک رمز (یا واحد) انتقال بدون کلید (یا کلید ثابت) را می‌توان به شکل پیاده‌سازی در سخت‌افزار به عنوان رمز انتقال از پیش سیم‌کشی شده^۴ در نظر گرفت. وقتی به شکل نرم-افزار به کار رود، کلید ثابت (قانون جایگشت یکتا^۵) را می‌توان به صورت جدول نشان داد. قسمت بعدی این فصل به بحث در مورد رمزهای انتقال بدون کلید می‌پردازد. این رمزها را جعبه‌های P ^۶ می‌نامند و از آن در ساخت بلوک‌های رمزهای بلوکی پیشرفته از آن استفاده می‌کنند.

=====

¹ Permutation Group

² Keyless Ciphers

³ Keyless Transposition Ciphers

⁴ Prewired

⁵ Single Permutation

⁶ P-Box

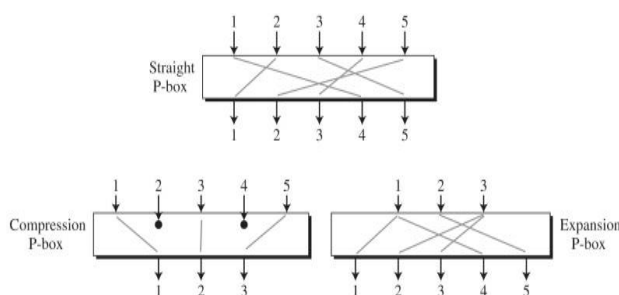
رمزهای جایگزین بدون کلید^۱: رمز جایگزین بدون کلید (با کلید ثابت) را می‌توان به عنوان یک نگاشت از پیش تعیین شده از ورودی به خروجی تلقی کرد. این نگاشت را می‌توان به شکل جدول، تابع ریاضی و غیره تعریف کرد. بخش بعدی این فصل به بحث در مورد رمزهای جایگزین بدون کلید می‌پردازد. این رمزها را جعبه‌های S^2 می‌نامند و در ساخت رمزهای بلوکی پیشرفته از آن استفاده می‌کنند.

اجرای یک رمز بلوکی پیشرفته

معمولاً رمزهای بلوکی پیشرفته، رمزهای جایگزینی کلیددار^۳ هستند که در آن کلید، فقط نگاشت محدودی از ورودی‌های احتمالی با خروجی‌های احتمالی را امکان‌پذیر می‌کند، با این وجود، رمزهای بلوکی پیشرفته معمولاً به صورت یک واحد جداگانه طراحی نمی‌شوند. برای ارائه‌ی ویژگی‌های مورد نیاز رمز بلوکی پیشرفته، چون ویژگی انتشار^۴ و بی‌نظمی^۵ (به اختصار توضیح داده خواهد شد)، یک رمز بلوکی که شامل از ترکیبی از واحدهای انتقالی (جعبه‌های P)، واحدهای جایگزین (جعبه‌های S) و تعدادی واحد دیگر (به اختصار توضیح داده خواهد شد) می‌باشد تشکیل شده است.

جعبه‌های P

جعبه‌های P (جعبه جایگشت) همانند رمز انتقال سنتی بر روی نویسه‌ها اعمال می‌شود. این جعبه بیت‌ها را جابجا می‌کند. در رمزهای بلوکی پیشرفته، سه نوع جعبه‌های P داریم: جعبه‌های P مستقیم، جعبه‌های P گسترشی و جعبه‌های P فشرده (شکل ۴-۵).



شکل ۴-۵: سه نوع جعبه P

شکل ۴-۵ جعبه P مستقیم 5×5 ، جعبه P فشرده 5×3 و جعبه گسترشی 3×5 را نشان می‌دهد. در

مورد هر یک از آن‌ها مفصل‌تر صحبت خواهیم کرد.

¹ Keyless Substitution Ciphers

² S-box

³ Keyed

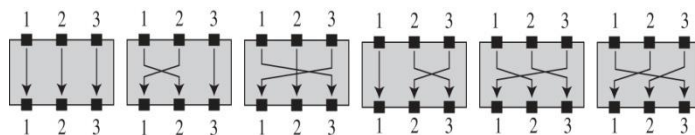
⁴ Diffusion

⁵ Confusion

جعبه‌های P مستقیم^۱: یک جعبه‌های P مستقیم با n ورودی و n خروجی، جایگشتی است با n نگاشت احتمالی.

مثال ۵-۵

شکل ۵-۵ تمام نگاشت‌های احتمالی 3×3 جعبه‌های P را نشان می‌دهد.



شکل ۵-۵ نگاشت‌های گوناگون جعبه P 3×3 احتمالی

اگرچه جعبه‌های P می‌توانند از یک کلید برای تعیین یکی از نگاشت‌های $n!$ استفاده کنند، اما معمولاً جعبه‌های P فاقد کلیداند، بدین معنی که نگاشت از پیش تعیین شده است. اگر جعبه‌های P در سخت‌افزار تعبیه شده باشد، از پیش سیم‌کشی شده‌اند و اگر در نرم‌افزار تعبیه شده باشد، قانون نگاشت با استفاده از یک جدول جایگشت نشان داده می‌شود. در حالت دوم، نویسه‌ها، ورودی جدول و موقعیت نویسه‌ها خروجی را مشخص می‌کند. جدول ۵-۱ نمونه‌ای از یک جدول جایگشت مستقیم در حالت $n=46$ را نشان می‌دهد.

جدول ۵-۱: مثال جایگشت برای جعبه‌های P مستقیم

58	50	42	34	26	18	10	02	60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06	64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01	59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05	63	55	47	39	31	23	15	07

جدول ۵-۱، ۶۴ نویسه معادل ۶۴ ورودی دارد. جایگاه (اندیس) ورودی مساوی خروجی است. از آنجا که اولین ورودی شامل عدد ۵۸ است، می‌دانیم که اولین خروجی از پنجاه و هشتمین ورودی می‌آید و چون آخرین ورودی ۷ است، می‌دانیم که شصت و چهارمین خروجی از هفتمین ورودی می‌آید و الی آخر.

مثال ۶-۵

=====

^۱ Straight P-Boxes

جدول جایگشت 8×8 برای جعبه‌های P مستقیم طراحی کنید طوری که دو بیت میانی بیت‌های (۵و۴) در کلمه ورودی به سمت دو بیت آخر (بیت‌های ۱ و ۸) در کلمه خروجی هدایت شده باشد. موقعیت‌های نسبی سایر بیت‌ها نباید تغییر کند.

حل

جعبه‌های P مستقیم با جدول [۴۱۲۳۶۷۸۵] لازم داریم. موقعیت‌های نسبی بیت‌های ورودی ۱، ۲، ۳، ۶، ۷، ۸ تغییر نمی‌کند، اما اولین خروجی، چهارمین ورودی و هشتمین خروجی، پنجمین ورودی را منتقل می‌کند.

جعبه‌های P متراکم^۱

جعبه‌های P متراکم یک جعبه P با n ورودی و m خروجی است طوری که $m < n$ است. برخی از ورودی‌ها بلوکه شده هستند و به خروجی نمی‌رسند (به شکل ۵-۴ نگاه کنید). معمولاً جعبه‌های P فشرده به کار رفته در رمزهای بلوکی پیشرفته بدون کلید و دارای یک جدول جایگشت می‌باشد. این جدول قانون جابجایی بیت‌ها را نشان می‌دهد. شایان یادآوری است که جدول جایگشت جعبه P فشرده m مدخل ورودی دارد، اما محتوای هر مدخل از ۱ تا n با تعدادی مقادیر نامشخص (ورودی‌هایی که بلوکه شده هستند) می‌باشد. جدول ۵-۲ نمونه‌ای از جدول جایگشت جعبه P فشرده به اندازه‌ی 32×24 است. توجه داشته باشید که ورودی‌های ۷، ۸، ۹، ۱۵، ۲۳، ۲۴، ۲۵ بلوکه شده‌اند.

جدول ۵-۲: نمونه‌ای از یک جدول جایگشت 32×24

01	02	03	21	22	26	27	28	29	13	14	17
18	19	20	04	05	06	10	11	12	30	31	32

وقتی از جعبه‌های P فشرده استفاده می‌کنیم که لازم است بیت‌ها را جابجا کنیم و در همان زمان تعداد بیت‌ها برای مرحله‌ی بعدی کاهش یابد.

جعبه‌های P گسترشی^۲: جعبه P گسترشی یک جعبه P با n ورودی و m خروجی است به طوری که $m > n$ می‌باشد. برخی از ورودی‌ها به بیش از یک خروجی متصل‌اند (به شکل ۵-۴ نگاه کنید). معمولاً جعبه‌های P گسترشی استفاده شده در رمزهای بلوکی پیشرفته بدون کلید و بر پایه‌ی جدول جایگشتی هستند که قانون جابجا شدن بیت‌ها را نشان می‌دهد. توجه کنید که جدول جایگشت

^۱ Compression P-Boxes

^۲ Expansion P-Boxes

برای یک جعبه P گسترشی دارای m مدخل ورودی نگاشت شده‌اند، اما $m-n$ مدخل تکرار می‌شوند (برخی ورودی‌ها به بیش از یک خروجی نگاشت شده‌اند). جدول ۳-۵ نمونه‌ای از جدول جایگشت جعبه P گسترشی به اندازه 12×16 را نشان می‌دهد. توجه داشته باشید که هر یک از ورودی‌ها ۱، ۳، ۹، ۱۲ به دو خروجی نگاشت می‌شوند.

جدول ۳-۵: نمونه‌ای از یک جدول جایگشت 12×16

01	09	10	11	12	01	02	03	03	04	05	06	07	08	09	12
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----

زمانی از جعبه‌های P گسترشی استفاده می‌کنیم که لازم است بیت‌ها را جابجا کنیم و در عین حال تعداد بیت‌ها برای مرحله‌ی بعدی افزایش یابد.

وارون‌پذیری: جعبه P مستقیم وارون‌پذیرند؛ یعنی می‌توانیم از جعبه P مستقیم در رمزنویسی و از وارون آن در رمز رمزگشایی استفاده کنیم. جدول‌های جایگشت هم باید وارون همدیگر باشند. در فصل ۳ دیدیم که چگونه می‌توان وارون جدول جایگشت را به دست آورد.

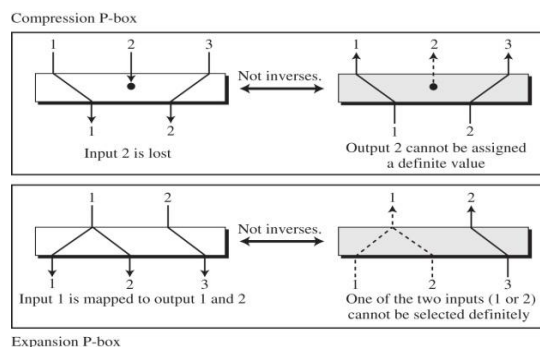
مثال ۷-۵

شکل ۶-۵ نشان می‌دهد که چگونه جدول جایگشت تک بعدی را می‌توان وارون کرد.

GF(2)			
$\{0, 1\}$	$+$	\times	
	+	\times	
	0	1	
	0	0	1
	1	1	0
	\times	0	1
	0	0	0
	1	0	1
	a	0	0
	$-a$	1	1
	a^{-1}	0	1
	a^{-1}	-1	-1

شکل ۶-۵ وارون یک جدول جایگشت

جعبه‌های فشرده و گسترشی وارون ندارند. در جعبه p فشرده، می‌توان یک ورودی را در خلال رمزنگاری حذف کرد. الگوریتم رمزگشایی راهی برای فهمیدن چگونگی جایگزینی بیت حذف شده ندارد (انتخابی بین یک بیت ۰ و یک بیت ۱). در جعبه p گسترشی، ممکن است در خلال رمزنگاری یک ورودی با بیش از چند خروجی نگاشت شده باشد؛ الگوریتم رمزگشایی راهنمایی در این زمینه که کدام یک از ورودی‌ها در خروجی تکرار شده‌اند ندارند. شکل ۷-۵ هر دو حالت را نشان می‌دهد.



شکل ۷-۵: جعبه‌های p گسترشی و متراکم به عنوان اجزای وارون ناپذیر.

شکل ۷-۵ همچنین نشان می‌دهد که جعبه p فشرده وارون جعبه p گسترشی نیست و بالعکس. یعنی در رمزگشایی نمی‌توانیم از یک جعبه P گسترشی استفاده کنیم یا بالعکس. با این وجود، همان‌گونه که بعداً در این فصل نشان داده خواهد شد، رمزهایی وجود دارند که در رمزنگاری از جعبه‌های P فشرده یا گسترشی استفاده می‌کنند، تأثیر آن‌ها به شیوه‌های دیگر در رمزگشایی از بین می‌رود. جعبه P مستقیم وارون‌پذیر است در حالی که جعبه‌های P فشرده و گسترشی وارون‌پذیر نیستند.

جعبه‌های S

جعبه S (جعبه جایگزین) را می‌توان یک رمز جایگزین کوچک (مینیاور) تلقی کرد؛ با این وجود، جعبه S می‌تواند تعداد متفاوتی ورودی و خروجی داشته باشد؛ به عبارت دیگر، ورودی یک جعبه S می‌تواند یک کلمه n بیتی، اما خروجی یک کلمه m بیتی باشد؛ طوری که m, n الزاماً یکسان نیستند. اگر چه یک جعبه S می‌تواند کلیددار یا بدون کلید باشد، اما معمولاً رمزهای بلوکی پیشرفته از جعبه‌های S بدون کلید استفاده می‌کنند، یعنی جایی که نگاشت ورودی‌ها به خروجی‌ها از پیش تعیین شده است. جعبه S یک واحد جایگزینی $m \times n$ است، به طوری که m و n الزاماً یکسان نیستند.

جعبه‌های S خطی در مقابل غیرخطی^۱: در جعبه S با n ورودی و m خروجی، ورودی‌ها را x_1, x_2, \dots, x_n و خروجی‌ها را y_1, \dots, y_m می‌نامیم. رابطه‌ی بین ورودی‌ها و خروجی‌ها را می‌توان به صورت مجموعه‌ای از معادلات به شکل زیر نشان داد.

$$y_1 = f_1(x_1, x_2, \dots, x_n)$$

^۱ Linear Versus Nonlinear S-Boxes

$$y_2 = f_2(x_1, x_2, \dots, x_n)$$

...

$$y_m = f_m(x_1, x_2, \dots, x_n)$$

در جعبه S خطی، رابطه فوق را می‌توان به این صورت بیان کرد:

$$y_1 = a_{1,1} x_1 \oplus a_{1,2} x_2 \oplus \dots \oplus a_{1,n} x_n$$

$$y_2 = a_{2,1} x_1 \oplus a_{2,2} x_2 \oplus \dots \oplus a_{2,n} x_n$$

...

$$y_m = a_{m,1} x_1 \oplus a_{m,2} x_2 \oplus \dots \oplus a_{m,n} x_n$$

در جعبه S غیرخطی نمی‌توانیم روابط فوق را برای هر خروجی داشته باشیم.

مثال ۵-۸

در جعبه S با سه ورودی و دو خروجی داریم:

$$y_1 = x_1 \oplus x_2 \oplus x_3 \quad y_2 = x_1$$

جعبه S خطی است زیرا $a_{1,1}=a_{1,2}=a_{1,3}=a_{2,1}=1$ و $a_{2,2}=a_{2,3}=0$ است. همان‌گونه که مشاهده می‌-

نمایید، این رابطه را می‌توان به صورت ماتریس به شکل زیر نشان داد.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

مثال ۵-۹

در یک جعبه S با سه ورودی و خروجی، داریم:

$$y_1 = (x_1)^2 + x_2 \quad y_2 = (x_1)^2 + x_1 x_2 + x_3$$

طوری که ضرب و جمع در $GF(2)$ است. این جعبه S غیرخطی است، زیرا رابطه‌ی خطی بین

ورودی‌ها و خروجی‌ها وجود ندارد.

مثال ۵-۱۰

جدول زیر رابطه‌ی ورودی/خروجی برای جعبه S به اندازه 3×2 را مشخص می‌کند. بیت سمت

چپ، سطر و دو بیت سمت راست، ستون را تعیین می‌کنند. دو بیت خروجی مقادیر محل تلاقی سطر و

ستون هستند.

	00	01	10	11	
Leftmost bit					Rightmost bits
0	00	10	01	11	
1	10	00	11	01	
	Output bits				

بر اساس این جدول، نتیجه‌ی ورودی ۰۱۰ خروجی ۰۱ و نتیجه‌ی ورودی ۱۰۱ خروجی ۰۰ است.

وارون‌پذیری: جعبه‌های S ، رمزهای جایگزینی هستند که رابطه‌ی بین ورودی و خروجی به وسیله‌ی جدول یا رابطه‌ی ریاضی تعیین می‌گردد. ممکن است جعبه S وارون‌پذیر باشد یا نباشد. در جعبه‌ی S وارون‌پذیر، باید تعداد بیت‌های ورودی و خروجی یکسان باشد.

مثال ۵-۱۱

شکل ۵-۸ نمونه‌ای از جعبه‌ی S وارون‌پذیر را نشان می‌دهد. یکی از جدول‌ها در الگوریتم رمزنگاری و جدول دیگر در الگوریتم رمزگشایی مورد استفاده قرار گرفته است. در هر جدول، بیت سمت چپ ورودی، بیانگر سطر و دو بیت کناری آن بیانگر ستون هستند. محل تلاقی سطر و ستون ورودی، مقدار خروجی را مشخص می‌کند.

به عنوان مثال، اگر ورودی جعبه سمت چپ ۰۰۱ باشد، خروجی ۱۰۱ است. ورودی ۱۰۱ در جدول سمت راست، خروجی ۰۰۱ را ایجاد می‌کند و این نشان می‌دهد که دو جدول وارون یکدیگرند.

'Xor

یک مؤلفه مهم در اکثر رمزهای بلوکی، عمل XOR است. همان‌گونه که در فصل ۴ گفتیم، اعمال جمع و تفریق در میدان $GF(2^n)$ به وسیله عملگر یگانه‌ی XOR انجام می‌شود.

The diagram illustrates the encryption and decryption process using a 3-bit S-box. The encryption table maps 3-bit inputs to 3-bit outputs, and the decryption table maps 3-bit inputs to 3-bit outputs. Both tables are 2x5 grids of 3-bit values.

Table used for encryption:

	00	01	10	11
0	011	101	111	100
1	000	010	001	110

Table used for decryption:

	00	01	10	11
0	100	110	101	000
1	011	001	111	010

شکل ۵-۸: جدول‌های جعبه S برای مثال ۵-۱۱

ویژگی‌ها: پنج ویژگی عمل XOR در میدان $GF(2^n)$ این عمل را به مؤلفه‌ای بسیار جالب برای استفاده در رمز بلوکی تبدیل کرده است.

=====

¹ Exclusive-Or

(۱) بسته بودن^۱: این ویژگی تضمین می‌کند که حاصل عمل XOR دو کلمه n بیتی، کلمه n بیتی دیگری داخل آن مجموعه است.

(۲) شرکت‌پذیری^۲: این ویژگی استفاده از بیش از یک XOR به هر ترتیبی را برای ما امکان‌پذیر می‌سازد.

$$x \oplus (y \oplus z) \leftrightarrow (x \oplus y) \oplus z$$

(۳) جابجایی: این ویژگی امکان جابجا کردن ورودی‌ها بدون تأثیر بر خروجی را برای ما فراهم می‌سازد.

$$x \oplus y \leftrightarrow y \oplus x$$

(۴) وجود عضو همانی: عضو همانی عمل XOR یک کلمه n بیتی است که تمام عناصر آن 0 یا (00...0) می‌باشد؛ بدین معنی که اعمال عمل XOR بر یک کلمه x و عضو همانی، ارزش x را تغییر نمی‌دهد.

$$x \oplus (0 \dots 0) = x$$

در رمز Feistel که در همین فصل آن را توضیح خواهیم داد، از این ویژگی استفاده می‌کنیم.

(۵) وجود وارون: در میدان $GF(2^n)$ هر کلمه، وارون افزایشی خود می‌باشد؛ بدین معنی که حاصل عمل XOR یک کلمه با خودش، عضو همانی است.

$$x \oplus x = (0 \dots 0)$$

در رمز Feistel از این ویژگی نیز استفاده می‌کنیم.

مکمل^۳: مکمل یک عمل یگانی (با یک ورودی و یک خروجی است) که ارزش هر بیت در کلمه را تغییر می‌دهد. بیت ۰ را به بیت ۱ و بیت ۱ به بیت ۰ تبدیل می‌شود. عمل مکمل در رابطه با عمل XOR برای ما جالب است. اگر \bar{x} مکمل x باشد، رابطه زیر را داریم:

$$x \oplus \bar{x} = (11 \dots 1) \text{ و } x \oplus (11 \dots 1) = \bar{x}$$

بعداً هنگام بحث در مورد امنیت برخی از رمزها از این ویژگی استفاده می‌کنیم.

وارون: وارون مؤلفه در رمزنگاری، زمانی مفهوم دارد که آن مؤلفه بیانگر یک عمل یگانی^۴ (با یک ورودی و یک خروجی) باشد. مثلاً یک جعبه P بدون کلید یا یک جعبه S بدون کلید را می‌توان وارون-

=====

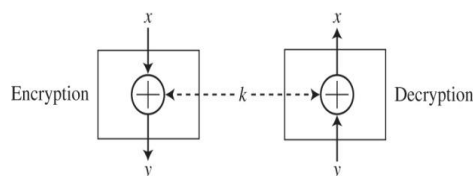
¹ Closure

² Associativity

³ Complement

⁴ Unary

پذیر نمود، زیرا یک ورودی و یک خروجی دارند. XOR یک عمل دودویی است و وارون آن تنها زمانی مفهوم دارد که یکی از ورودی‌ها ثابت باشد (در رمزنگاری^۱ و رمزگشایی یکسان باشد). به عنوان مثال، اگر یکی از ورودی‌ها کلید باشد، که معمولاً در رمزنگاری و رمزگشایی یکسان است. پس همان‌گونه که در شکل ۵-۹ نشان داده شده است، عمل XOR وارون خود می‌باشد.



شکل ۵-۹: وارون‌پذیری عمل XOR

در شکل ۵-۹ ویژگی وارون افزایش حاکی از این است که:

$$y = x \oplus k \rightarrow x = k \oplus y$$

در این فصل، هنگام توضیح ساختار رمزهای بلوکی از این ویژگی استفاده خواهیم کرد.

چرخش دایره‌ای^۲

مؤلفه‌ی دیگری که در برخی از رمزهای بلوکی پیشرفته یافت می‌شود، چرخش دایره‌ای است. چرخش می‌تواند به سمت چپ یا راست باشد. در چرخش دایره‌ای به سمت چپ، مکان هر بیت در یک کلمه n بیتی k موقعیت به سمت چپ تغییر می‌کند؛ سمت چپ‌ترین k بیت از سمت چپ برداشته و به سمت راست منتقل می‌شود.

در چرخش دایره‌ای به سمت راست^۳، مکان هر بیت در یک کلمه n بیتی k موقعیت به سمت راست تغییر می‌نماید؛ سمت راست‌ترین k بیت از سمت راست برداشته شده و به سمت چپ منتقل می‌شوند. شکل ۵-۱۰ اعمال چرخش به چپ و راست را در حالتی که $n=8$ و $K=3$ است، نشان می‌دهد. عمل چرخش دایره‌ای، بیت‌های موجود در یک کلمه را جابجا کرده و به پنهان کردن الگوهای کلمه اصلی کمک می‌نماید. اگر چه تعداد موقعیت‌هایی که باید تغییر یابد را می‌توان به عنوان کلید استفاده نمود، اما معمولاً عمل چرخش دایره‌ای بدون کلید بوده و مقدار K ثابت و از پیش تعیین شده است.

=====

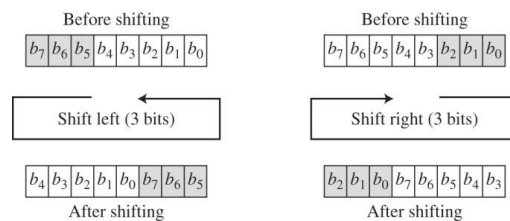
¹ Encryption

² Circular Shift

³ Circular Right Shift

وارون‌پذیری: اگر چرخش دایره‌ای به سمت چپ باشد، وارون عمل چرخش دایره‌ای به راست است. اگر یکی از آن‌ها در رمزنگاری مورد استفاده قرار گیرد، دیگری را می‌توان در رمزگشایی به کار برد.

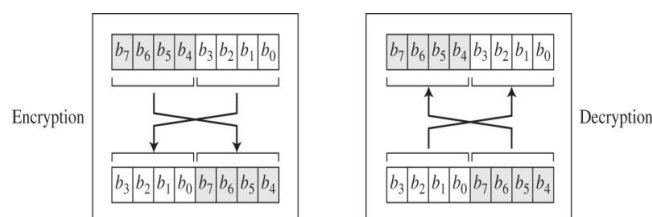
ویژگی‌ها: چرخش دایره‌ای دو ویژگی دارد که باید از وجود آن‌ها آگاه باشیم. نخست، چرخش به پیمانه n است؛ به عبارت دیگر اگر $K=0$ یا $K=n$ باشد، هیچ چرخشی وجود ندارد. اگر K بزرگ‌تر از n باشد، در آن صورت ورودی به اندازه $K \bmod n$ بیت تغییر می‌کند. دوم، چرخش دایره‌ای تحت عمل ترکیب، یک گروه محسوب می‌شود؛ یعنی چرخش یک کلمه بیش از یک بار با چرخش یک بار یک کلمه یکی است.



شکل ۵-۱۰: چرخش دایره‌ای یک کلمه ۸ بیتی به سمت چپ و راست

مبادله^۱

مبادله حالت خاصی از عمل چرخش دایره‌ای می‌باشد، به طوری که $K=n/2$ است؛ بدین معنی که این عمل تنها زمانی معتبر است که n عدد زوج باشد. از آنجا که چرخش به سمت چپ $n/2$ بیت با چرخش به راست $n/2$ یکسان است، این مؤلفه، وارون خود است. عملگر مبادله در رمزگشایی می‌تواند نتیجه‌ی یک عمل مبادله‌ای در رمزنگاری را کاملاً بی‌اثر کند. در شکل ۵-۱۱ عمل مبادله‌ای برای یک کلمه ۸ بیتی را نشان می‌دهد.



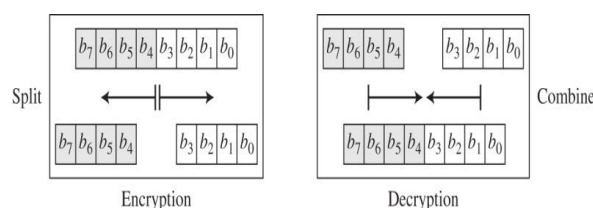
شکل ۵-۱۱: عمل مبادله روی یک کلمه ۸ بیتی

=====

¹ Swap

تفکیک (دو نیم کردن) و ترکیب^۱

دو عمل دیگری که در برخی از رمزهای بلوکی به کار رفته است، تفکیک و ترکیب است. معمولاً عمل تفکیک یک کلمه n بیتی را از وسط دو نیم می‌کند و دو کلمه با طول برابر می‌سازد. بیشتر عمل ترکیب، دو کلمه با طول مساوی را به منظور ایجاد یک کلمه n بیتی به هم متصل می‌کند. این دو عمل وارون یکدیگرند و می‌توان از آن‌ها به عنوان زوج حذف کننده اثر یکدیگر استفاده کرد. اگر یکی از آن‌ها در رمزنگاری به کار رفته باشد، دیگری در رمزگشایی مورد استفاده قرار می‌گیرد. شکل ۵-۱۲ این دو عمل را در حالتی نشان می‌دهد که $n=8$ است.



شکل ۵-۱۲: اعمال تفکیک و ترکیب روی یک کلمه ۸ بیتی

رمزهای فرآورده^۲

شانن مفهوم رمز فرآورده را معرفی کرد. رمز فرآورده، رمز پیچیده‌ای است که از ترکیب جایگزینی، جایگشت و سایر مؤلفه‌های عنوان شده در قسمت‌های پیشین ایجاد شده است.

پخش و آشفته‌گی^۳

هدف شانن از معرفی رمز فرآورده این بود که رمزهای بلوکی، این دو ویژگی مهم را داشته باشند: پخش و آشفته‌گی. هدف از پخش، پنهان کردن رابطه بین متن رمز و متن عادی است. این امر موجب ناکامی کسی می‌شود که می‌خواهد با استفاده از ویژگی‌های آماری متن رمز به متن عادی برسد. پخش بدین معنی است که هر نشانه (نویسه یا بیت) در متن رمز به برخی یا تمام نشانه‌های موجود در متن عادی بستگی داشته باشد. به عبارت دیگر، اگر تنها یک نشانه در متن عادی عوض شود، چندین یا تمام نشانه‌های متن رمز نیز عوض خواهند شد.

پخش، رابطه بین متن عادی و متن رمز را پنهان می‌کند.

=====

¹ Split And Combine

² Product Ciphers

³ Diffusion and Confusion

هدف از آشفتگی، پنهان کردن رابطه بین متن رمز و کلید است. این امر موجب ناکامی کسی می-شود که می‌کوشد با استفاده از متن رمز، کلید را پیدا کند. به عبارت دیگر، اگر فقط یک بیت در کلید عوض شود، چندین یا تمام نشانه‌های متن رمز نیز عوض خواهد شد.

آشفتگی، رابطه بین متن رمز و کلید را پنهان می‌کند.

گردش^۱

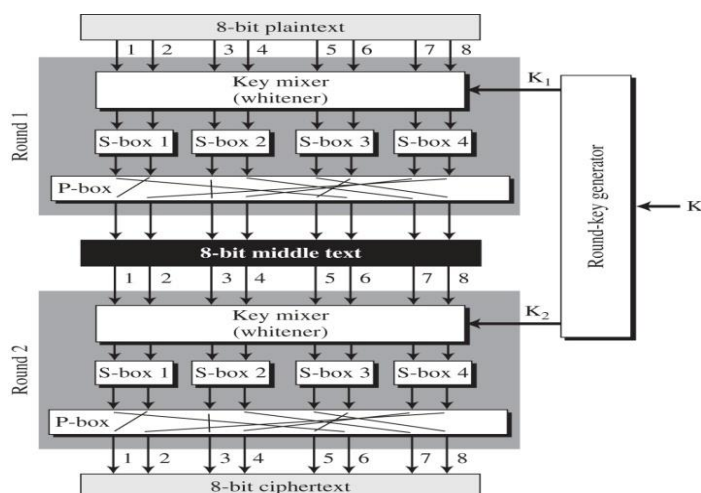
با استفاده از رمزهای فرآورده تکرار شونده، هر تکرار، ترکیبی از جعبه‌های S ، جعبه‌های P و سایر مؤلفه-هاست که می‌توان به پخش و آشفتگی دست یافت. هر تکرار را یک گردش می‌گویند. هر رمز بلوکی از یک «برنامه کلید» یا «مولد کلید»^۲ استفاده می‌کند. این برنامه یا مولد کلید با استفاده از کلید رمز، برای هر گردش یک کلید مجزا ایجاد می‌کند. در یک رمزنگاری N گردشی، برای ایجاد متن رمز، متن عادی، N بار رمزنگاری می‌شود و برای ایجاد متن عادی، متن رمز، N بار رمزگشایی می‌شود. متن ایجاد شده در سطوح میانی (بین دو گردش) را متن میانی^۳ می‌نامیم. شکل ۵-۱۳ یک رمز فرآورده با دو گردش را نشان می‌دهد. رمزهای فرآورده، بیش از دو گردش دارند.

در شکل ۵-۱۳ در هر روند سه تغییر اتفاق می‌افتد:

- (a) برای شفاف‌سازی^۴ متن (پنهان کردن بیت‌ها با استفاده از کلید)، متن ۸ بیتی با کلید ترکیب می‌شود. معمولاً این کار از طریق اعمال XOR کلمه‌ی ۸ بیتی با کلید ۸ بیتی انجام می‌شود.
- (b) خروجی‌های سپیدگر^۵ در چهار گروه دو بیتی سازمان‌دهی می‌شوند و سپس در ورودی چهار جعبه S قرار می‌گیرند. در این دگرگونی، مقادیر بیت‌ها بر اساس ساختار جعبه-های S تغییر می‌کند.
- (c) جهت جایگشت بیت‌ها، خروجی‌های جعبه‌های S از طریق یک جعبه P منتقل می‌شوند، به طوری که در گردش بعدی هر جعبه ورودی‌های متفاوتی را دریافت می‌کند.

=====

¹ Rounds
² Key Generator
³ Middle Text
⁴ Whiten
⁵ Whitener



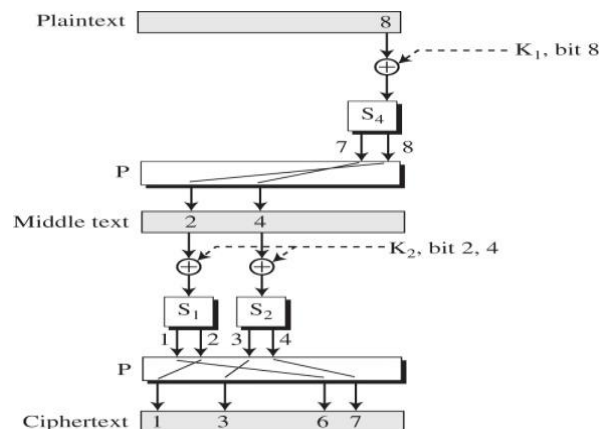
شکل ۵-۱۳: رمز فرآورده متشکل از دو گردش

پخش: طراحی ساده شکل ۵-۱۳ نشان می‌دهد که چگونه رمز فرآورده با ترکیب جعبه‌های S و جعبه‌های P می‌تواند وظیفه پخش را به عهده بگیرد. شکل ۵-۱۴ نشان می‌دهد که چگونه تغییر تنها یک بیت در متن عادی بر بیت‌های زیادی در متن رمز تأثیر می‌گذارد.

(a) در گردش اول، بیت ۸ پس از XOR با بیت K_1 مربوطه به وسیله‌ی S BOX4 بر دو بیت (۷ و ۸) اثر می‌گذارد. بیت ۷ پس و پیش شده و تبدیل به بیت ۲ می‌شود؛ بیت ۸ با بیت ۴ نیز پس و پیش می‌شود. پس از گردش اول، بیت ۸ بر بیت‌های ۲ و ۴ اثر می‌گذارد. در گردش دوم، بیت ۲، پس از XOR با بیت K_2 مربوطه به وسیله‌ی SBOX1 بر دو بیت (بیت‌های ۱ و ۲) اثر می‌گذارد. بیت ۴ پس از XOR با بیت مربوطه در K_2 ، روی بیت‌های ۳ و ۴ اثر می‌گذارد. بیت ۳ بدون تغییر می‌ماند؛ بیت ۴ پس و پیش شده و تبدیل به بیت ۷ می‌شود. پس از گردش دوم، بیت ۸ بر بیت‌های ۱، ۳، ۶ و ۷ تأثیر می‌گذارد.

(b) پیمودن این مراحل در جهت وارون (از متن رمز به متن عادی) نشان می‌دهد که هر بیت در متن رمز تحت تأثیر بیت‌های متعددی در متن عادی می‌باشد.

همچنین شکل ۵-۱۴ نشان می‌دهد که چطور می‌توان از طریق استفاده از رمز فرآورده به ویژگی آشفتگی دست یافت. چهار بیت متن رمز، بیت‌های ۱، ۳، ۶ و ۷ تحت تأثیر سه بیت از کلید (بیت ۸ در K_1 و بیت‌های ۲ و ۴ در K_2) قرار می‌گیرد. پیمودن این مراحل در جهت وارون نشان می‌دهد که هر بیت در هر گردش کلید، بر چندین بیت در متن رمز تأثیر می‌گذارد. رابطه‌ی بین بیت‌های متن رمز و بیت‌های کلید مبهم است.



شکل ۵-۱۴: پخش و آشفتگی در یک رمز بلوکی

رمزهای کاربردی (عملی):^۱ برای افزایش پخش و آشفتگی، رمزهای کاربردی در عمل از بلوکهای داده‌ای بزرگ‌تر، جعبه‌های S بیشتر و گردش‌های بیشتر استفاده می‌کنند. با کمی دقت مشاهده می‌شود که افزایش تعداد گردش‌ها با استفاده از جعبه‌های S بیشتر، می‌تواند به ایجاد رمز بهتری منتهی شود که در این صورت متن رمز بیشتر شبیه به یک کلمه‌ی n بیتی تصادفی است. در این روش، رابطه‌ی بین متن رمز و متن عادی کاملاً پوشیده شده است. افزایش تعداد گردش‌ها، تعداد کلیدهای گردش را افزایش می‌دهد و این امر باعث می‌شود که رابطه‌ی بین متن رمز و کلید بهتر پنهان شود.

دو رسته از رمزهای فرآورده

تمام رمزهای بلوکی پیشرفته، رمزهای فرآورده هستند، اما به دو رسته تقسیم شده‌اند: رمزهای رسته اول هم از مؤلفه‌های وارون و هم از مؤلفه‌های غیر وارون استفاده می‌کنند. معمولاً رمزهای این رسته را رمزهای Feistel می‌نامند. رمز بلوکی DES که در فصل ۶ مورد بحث قرار می‌گیرد، نمونه خوبی از رمز Feistel می‌باشد. رمزهای رسته دوم فقط از مؤلفه‌های وارون‌پذیر استفاده می‌کنند. رمزهای این رسته را (به علت عدم وجود واژه‌ای دیگر) رمزهای غیر Feistel می‌نامیم. رمز بلوکی AES که در فصل ۶ مورد بحث قرار می‌گیرد، نمونه‌ی خوبی از رمز غیر Feistel است.

رمزهای Feistel

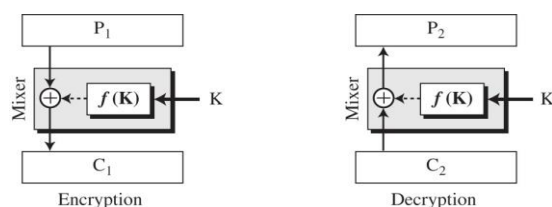
Feistel رمز بسیار جالب و هوشمندانه‌ای را طراحی کرد که دهه‌ها مورد استفاده قرار گرفته است. رمز Feistel می‌تواند سه نوع مؤلفه داشته باشد: خود وارون‌پذیر^۲، وارون‌پذیر^۱ و وارون‌ناپذیر^۲. رمز

^۱ Practical

^۲ Self-Invertible

Feistel تمام عناصر وارون‌ناپذیر را در یک واحد ترکیب می‌کند و از همان واحد در الگوریتم رمزنگاری و رمزگشایی استفاده می‌کند. پرسش اینجاست که چطور الگوریتم‌های رمزنگاری و رمزگشایی وارون یکدیگرند در حالی که هر یک از آن‌ها واحد وارون‌ناپذیر دارند. Feistel نشان داد که چگونه می‌توان این اثرها را خنثی نمود (تأثیرشان حذف شوند).

نگرش نخست: برای درک بهتر رمز Feistel اجازه دهید بررسی کنیم که چگونه می‌توانیم از همان مؤلفه وارون‌ناپذیر در الگوریتم‌های رمزنگاری و رمزگشایی استفاده کنیم. همان‌گونه که در شکل ۵-۱۵ نشان داده شده است، اگر از XOR استفاده کنیم، می‌توانیم تأثیرات یک مؤلفه وارون‌ناپذیر را در الگوریتم رمزنگاری حذف کنیم.



شکل ۵-۱۵: نگاه ساده از طراحی رمز Feistel

برای رمزنگاری، تابع وارون‌ناپذیر، کلید را به عنوان ورودی پذیرفته و خروجی آن با متن عادی XOR می‌شود. نتیجه، متن رمز است. به دلیل عدم وجود واژه‌ای دیگر، ترکیب تابع و عمل XOR را ترکیب کننده می‌نامیم. ترکیب کننده در پیشرفت‌های بعدی رمز Feistel نقش مهمی ایفا می‌کند. از آنجا که کلید رمزنگاری و رمزگشایی یکی است، می‌توانیم ثابت کنیم که دو الگوریتم وارون یکدیگرند؛ به عبارت دیگر اگر $C_2 = C_1$ باشد (در زمان انتقال، تغییری در متن رمز پیش نیاید) در نتیجه $P_1 = P_2$ می‌شود.

$$\text{رمزنگاری: } C_1 = P_1 \oplus f(K)$$

$$\text{رمزگشایی: } P_2 = C_2 \oplus f(K) = C_1 \oplus f(K) = P_1 \oplus f(K) \oplus f(K) = P_1 \oplus (0 \dots 0) = P_1$$

توجه داشته باشید که از دو ویژگی XOR (وجود وارون و وجود عضو همانی) استفاده شده است.

مباحث بالا ثابت می‌کند که اگرچه ترکیب کننده دارای یک عضو وارون‌ناپذیر است، خود وارون

همان ترکیب کننده است.

¹ Invertible

² Noninvertible

ترکیب کننده در طراحی Feistel خود وارون است.

مثال ۵-۱۲

با یک مثال ساده بحث را ادامه می‌دهیم. طول متن عادی و متن رمز هر کدام ۴ بیت و طول کلید ۳ بیت است. فرض کنید تابع، بیت‌های اول و سوم کلید را در نظر بگیرد؛ این دو بیت را که به صورت عدد ده دهی در نظر می‌گیرد به توان دو می‌رساند و نتیجه را به صورت الگوی دودویی ۴ بیتی تفسیر می‌کند. اگر متن عادی ۰۱۱۱ و کلید ۱۰۱ باشد، نتایج رمزنگاری و رمزگشایی را نشان دهید.

حل

تابع برای به دست آوردن ۱۱ به شکل دودویی و ۳ به شکل ده دهی، بیت‌های اول و دوم را انتخاب می‌کند. حاصل توان دوم ۹ است که معادل دودویی ۱۰۰۱ می‌شود.

$$\text{رمزنگاری: } C = P \oplus f(K) = 0111 \oplus 1001 = 1110$$

$$\text{رمزنگاری: } P = C \oplus f(K) = 1110 \oplus 1001 = 0111$$

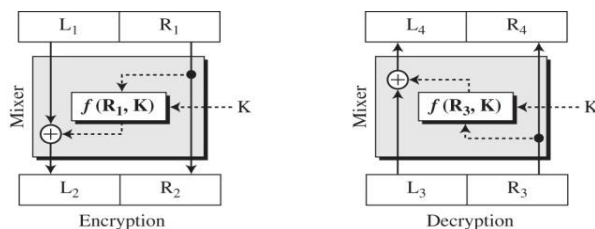
تابع $f(101) = 1001$ وارون‌ناپذیر است، اما عمل XOR استفاده از این تابع را، هم در الگوریتم رمزنگاری و هم در الگوریتم رمزگشایی امکان‌پذیر می‌سازد؛ به عبارت دیگر خود تابع وارون‌ناپذیر است، اما ترکیب‌کننده^۱، خود وارون است.

ارتقا: اجازه دهید برای نزدیک شدن به رمز (Feistel) ایده‌ی ساده‌ی خود را اصلاح کنیم. می‌دانیم که باید از همان ورودی برای عضو وارون‌ناپذیر (تابع) استفاده کنیم، اما نمی‌خواهیم فقط از کلید استفاده کنیم بلکه می‌خواهیم ورودی تابع، بخشی از متن عادی در رمزنگاری و بخشی از متن رمز در رمزگشایی نیز باشد. می‌توان از کلید به عنوان ورودی دوم تابع استفاده کرد. در این شیوه، تابع می‌تواند یک عضو مرکب^۲ با چند عضو بدون کلید و چند عضو دارای کلید باشد. برای دستیابی به این هدف، متن عادی و متن رمز را به دو بلوک با طول‌های مساوی چپ و راست تقسیم می‌کنیم. بلوک سمت چپ را L و بلوک سمت راست را R می‌نامیم. بلوک سمت راست را ورودی تابع و بلوک سمت چپ را XOR با خروجی تابع در مرحله‌ی پیش قرار دهید. یک نکته مهم را باید به خاطر داشت و آن این است که ورودی‌های تابع باید در رمزنگاری و رمزگشایی دقیقاً یکسان باشد. یعنی بخش سمت راست متن عادی در

¹ Mixer

² Complex

رمزنگاری و بخش سمت راست متن رمز در رمزگشایی باید یکسان باشد؛ به عبارت دیگر بخش سمت راست باید در خلال فرآیند رمزنگاری و رمزگشایی بدون تغییر باقی بماند. شکل ۵-۱۶ این مطلب را نشان می‌دهد.



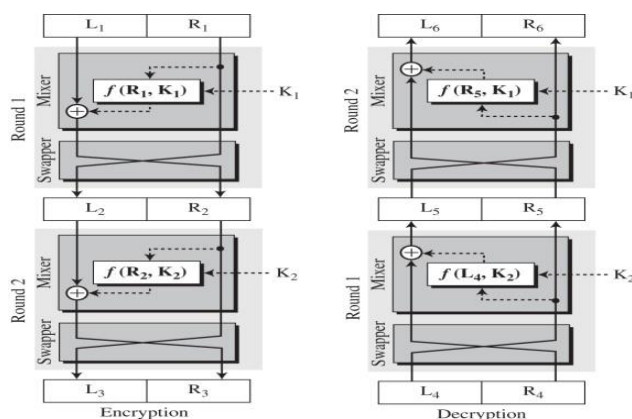
شکل ۵-۱۶: ارتقای طراحی قبلی Feistel

الگوریتم‌های رمزنگاری و رمزگشایی هنوز هم وارون یکدیگرند. فرض کنید $L_3=L_2$ و $R_3=R_2$ است (در طول انتقال، تغییری در متن رمز رخ نداده است).

$$R_4 = R_3 = R_2 = R_1$$

$$L_4 = L_3 \oplus f(R_3, K) = L_2 \oplus f(R_2, K) = L_1 \oplus f(R_1, K) \oplus f(R_1, K) = L_1$$

طرح نهایی: ارتقا (اصلاح) مرحله‌ی پیشین یک اشکال دارد. نیمه راست متن عادی هیچ‌وقت تغییر نمی‌کند. رمزشکن می‌تواند به وسیله‌ی استراق سمع متن رمز و استخراج نیمه‌ی راست آن، بلافاصله به نیمه راست دسترسی پیدا کند. این طراحی نیازمند اصلاحات بیشتری است؛ نخست افزایش تعداد گردش‌ها، دوم افزودن یک عضو جدید به نام مبادله‌گر به هر گردش. تأثیر مبادله‌گر در زمان رمزگشایی، تأثیر مبادله‌گر مرحله‌ی رمزنگاری از بین می‌برد. به هر حال، مبادله‌گر^۱ امکان جابجایی نیمه‌های راست و چپ را در دو گردش فراهم می‌کند. شکل ۵-۱۷ طراحی جدید را با دو گردش نشان می‌دهد.



شکل ۵-۱۷: طراحی نهایی رمز Feistel با دو گردش

=====

¹ swapper

شایان توجه است که کلیدهای دو گردش K_1 و K_2 موجودند. از این کلیدها به ترتیب وارون در رمزنگاری و رمزگشایی استفاده می‌شود.

از آنجا که ترکیب کننده‌ها وارون یکدیگرند و مبادله‌گرها نیز وارون یکدیگرند، آشکار است که رمزهای رمزنگاری و رمزگشایی نیز وارون یکدیگرند. با این وجود اجازه دهید بررسی کنیم که آیا می‌توانیم با استفاده از رابطه بین بخش‌های راست و چپ در هر رمز، این واقعیت را ثابت کنیم؟ به عبارت دیگر اجازه دهید ببینیم با فرض این که $L_4=L_2$ و $R_4=R_2$ است (در طول انتقال تغییری در طول متن رمز رخ نمی‌دهد)، آیا $L_1=L_3$ و $R_1=R_3$ است یا نه. ابتدا تساوی را برای متن میانی اثبات می‌کنیم.

$$L_0 = R_4 \oplus f(L_4, K_2) = R_2 \oplus f(R_2, K_1) = L_2 \oplus f(R_2, K_2) \oplus f(R_2, K_2) = L_2$$

$$R_0 = L_4 = L_2 = R_2$$

در نتیجه اثبات برقراری تساوی برای دو بلوک متن عادی، آسان است.

$$L_1 = R_0 \oplus f(L_0, K_1) = R_2 \oplus f(L_2, K_1) = L_1 \oplus f(R_1, K_1) \oplus f(R_1, K_1) = L_1$$

$$R_1 = L_4 = L_2 = R_1$$

رمزهای غیر Feistel

رمزهای غیر Feistel تنها از ابزار وارون‌پذیری استفاده می‌کند. هر جزء در متن عادی، جزء متناظری در متن رمز دارد. مثلاً جعبه‌های S برای این که هماهنگ باشند باید تعداد مساوی ورودی و خروجی داشته باشند. استفاده از جعبه‌های P متراکم یا گسترشی مجاز نیستند زیرا وارون‌ناپذیرند. در رمز غیر Feistel لازم نیست مانند رمزهای Feistel متن عادی را به دو نیمه تقسیم کنیم. شکل ۵-۱۳ را می‌توان یک رمز غیر Feistel تلقی کرد، زیرا مؤلفه‌های موجود در هر گردش عبارت‌اند از: XOR جعبه‌های S 2×2 که می‌توان آن‌ها را به صورت وارون‌پذیر طراحی کرد و یک جعبه P مستقیم که با استفاده از جدول جایگشت مناسب، وارون‌پذیر است. از آنجا که هر جزء وارون‌پذیر است، می‌توان نشان داد که هر گردش نیز وارون‌پذیر است. تنها لازم است کلیدهای گردش را به ترتیب وارون به کار ببریم. الگوریتم رمزگشایی از کلیدهای گردش K_1 و K_2 استفاده می‌کند.

حمله به رمزهای بلوکی

تمام حمله‌های عنوان شده برای رمزهای ستی را می‌توان روی رمزهای بلوکی پیشرفته نیز انجام داد، اما رمزهای بلوکی امروزه در برابر حمله‌های ذکر شده در فصل ۳ پایدارترند. به عنوان مثال، حمله‌ی آزمودن کلیدها^۱ معمولاً غیرقابل اجرا است، زیرا کلیدها غالباً کلیدهای بزرگ هستند. با این وجود، اخیراً حملات جدیدی به رمزهای بلوکی ابداع شده است که مبتنی بر ساختار رمزهای بلوکی پیشرفته می‌باشد. این حملات از تکنیک‌های تحلیل رمز خطی و دیفرانسیلی استفاده می‌کند.

تحلیل رمز دیفرانسیلی^۲

«الی بیهام»^۳ و «ادی شمیر»^۴ ایده تحلیل رمز دیفرانسیلی را معرفی کردند که از نوع حمله‌ی متن عادی منتخب است. رمز شکن می‌تواند به طریقی به کامپیوتر آلیس دسترسی پیدا کند، متن عادی منتخبی را در نظر بگیرد و متن رمز متناظر آن را تولید نماید. هدف، یافتن کلید رمز آلیس است.

تجزیه و تحلیل الگوریتم: پیش از این که رمز شکن از حمله به شیوه‌ی متن عادی منتخب استفاده کند باید الگوریتم رمزنگاری را مورد تجزیه و تحلیل قرار دهد تا اطلاعاتی در خصوص رابطه‌ی متن عادی و متن رمز به دست آورد. بدیهی است که رمز شکن کلید رمز را نمی‌داند؛ با وجود این، برخی از رمزها نقاط ضعف ساختاری دارند و این امر یافتن رابطه بین تفاوت‌های متن عادی و تفاوت‌های متن رمز را بدون دانستن کلید برای رمز شکن میسر می‌کند.

مثال ۵-۱۳

همان‌گونه که در شکل ۵-۱۸ نشان داده شده است، رمز تنها از یک عمل XOR تشکیل شده است. اگر منظور ما از تفاوت‌ها در متن عادی $P_1 \oplus P_2$ و تفاوت‌های متن رمز $C_1 \oplus C_2$ باشد، رمز شکن می‌تواند به راحتی رابطه‌ی بین تفاوت‌های متن عادی و تفاوت‌های متن رمز را بدون دانستن مقدار کلید پیدا کند. به شکل زیر ثابت می‌شود که $C_1 \oplus C_2 = P_1 \oplus P_2$ می‌باشد.

$$C = P_1 \oplus K \quad C_2 = P_2 \oplus K \rightarrow C_1 + C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

با این حال، این مثال بسیار غیر واقعی است؛ رمزهای بلوکی پیشرفته به این سادگی نیستند.

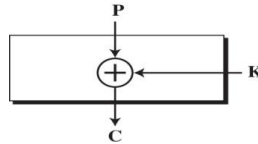
=====

¹ Brute- Force

² Differential Cryptanalysis

³ Eli Biham

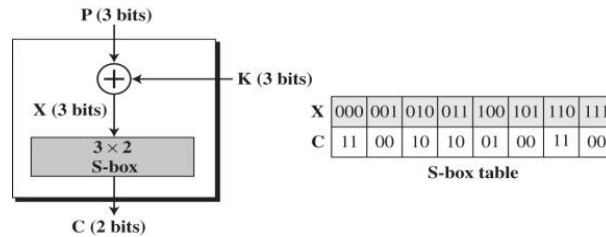
⁴ Adi Shamir



شکل ۵-۱۸: دیاگرام مثال ۵-۱۳

مثال ۵-۱۴

همان گونه که در شکل ۵-۱۹ نشان داده شده است، یک جعبه S به مثال ۵-۱۳ اضافه می‌کنیم.



شکل ۵-۱۹: دیاگرام مثال ۵-۱۴

اگرچه هنگام استفاده از تفاوت‌های بین X ها و P ها ($X_1 \oplus X_2 = P_1 \oplus P_2$) باز هم تأثیر کلید حذف می‌شود، اما وجود جعبه S مانع دستیابی رمزشکن به رابطه‌ی قطعی بین تفاوت‌های متن عادی و تفاوت‌های متن رمز می‌شود. رمزشکن می‌تواند جدول ۵-۴ را تشکیل دهد. این جدول نشان می‌دهد که برای هر تفاوت متن عادی در رمز، ممکن است چه تعداد تفاوت در متن رمز ایجاد کند. از آنجا که $X_1 \oplus X_2 = P_1 \oplus P_2$ است، این جدول طبق اطلاعات موجود در جدول ورودی/خروجی جعبه S در شکل ۵-۱۹ ترسیم شده است.

جدول ۵-۴: ورودی/خروجی دیفرانسیلی برای رمز مثال ۵-۱۴

		$C_1 \oplus C_2$			
		00	01	10	11
$P_1 \oplus P_2$	000	8			
	001	2	2		4
	010	2	2	4	
	011		4	2	2
	100	2	2	4	
	101		4	2	2
	110	4		2	2
	111			2	6

از آنجایی که اندازه‌ی کلید ۳ بیت است، هشت حالت گوناگون ورودی وجود دارد. این جدول نشان می‌دهد که اگر ورودی $(000)_2$ باشد، خروجی همیشه $(00)_2$ است؛ به عبارت دیگر جدول نشان می‌دهد که اگر ورودی $(100)_2$ باشد، دو حالت از اشکال خروجی به شکل $(00)_2$ ، دو حالت از اشکال خروجی به شکل $(01)_2$ و چهار حالت از اشکال خروجی به شکل $(01)_2$ وجود دارد.

مثال ۵-۱۵

همان‌گونه که در جدول ۵-۵ نشان داده شده است، نتیجه‌ی حاصل از مثال ۵-۱۴ می‌تواند اطلاعات احتمالی در اختیار رمزشکن قرار دهد. ورودی‌های این جدول، احتمال وقوع را نشان می‌دهد. ورودی‌هایی که احتمال صفر دارند هیچ‌وقت اتفاق نخواهد افتاد.

جدول ۵-۵: جدول توزیع دیفرانسیل برای مثال ۵-۱۵

		$C_1 \oplus C_2$			
		00	01	10	11
$P_1 \oplus P_2$	000	1	0	0	0
	001	0.25	0.25	0	0.50
	010	0.25	0.25	0.50	0
	011	0	0.50	0.25	0.25
	100	0.25	0.25	0.50	0
	101	0	0.50	0.25	0.25
	110	0.50	0	0.25	0.25
	111	0	0	0.25	0.75

همان‌گونه که بعداً خواهیم دید، رمزشکن اطلاعات زیادی برای آغاز حمله خود دارد. این جدول نشان می‌دهد که به علت ضعف ساختاری در جعبه S، احتمالات به صورت یکسان توزیع نشده است. گاهی اوقات جدول ۵-۵ را جدول توزیع دیفرانسیل یا پروفایل XOR می‌نامند.

حمله بر اساس متن عادی منتخب: پس از تجزیه و تحلیلی که می‌توان آن را فقط یک بار انجام داد و تا زمانی که ساختار رمز تغییر نکرده است از آن استفاده نمود؛ رمزشکن می‌تواند متن عادی‌ای را برای حمله انتخاب کند. جدول توزیع احتمال دیفرانسیل^۱ (جدول ۵-۵) به رمزشکن کمک می‌کند تا متون عادی‌ای را انتخاب کند که بالاترین احتمال را در جدول دارند.

تخمین مقدار کلید: پس، انجام حملاتی با متون عادی منتخب مناسب رمزشکن می‌تواند چند زوج متن عادی-متن رمز به دست آورد. این زوج‌ها حدس مقدار کلید را برای رمزشکن امکان‌پذیر می‌کند. این مرحله از C آغاز و به سمت P ادامه می‌یابد.

مثال ۵-۱۶

با نگاه به جدول ۵-۵ رمزشکن می‌داند که اگر $P_1 \oplus P_2 = 001$ باشد، پس با احتمال ۵۰ درصد $C_1 \oplus C_2 = 11$ است. رمزشکن $C_1 = 11$ را آزمایش می‌کند و $P_1 = 010$ را به دست می‌آورد (حمله به شیوه‌ی متن رمز منتخب). سپس $C_2 = 11$ را آزمایش می‌کند و $P_2 = 011$ را به دست می‌آورد (حمله‌ی دیگر

=====

^۱ Differential Distribution Table

به شیوه‌ی متن رمز منتخب). اکنون سعی می‌کند براساس اولین زوج C_1 و P_1 به شکل زیر برعکس عمل کند.

$$C_1=00 \rightarrow X_1=001 \text{ or } X_1=111$$

$$\text{if } X_1=001 \rightarrow K=X_1 \oplus P_1=011 \text{ if } X_1=111 \rightarrow K=X_1 \oplus P_1=101$$

با استفاده از زوج دوم، P_2 و C_2

$$C_2=11 \rightarrow X_2=000 \text{ or } X_2=110$$

$$\text{if } X_2=000 \rightarrow K=X_2 \oplus P_2=011 \text{ if } X_2=110 \rightarrow K=X_2 \oplus P_2=101$$

هر دو آزمایش، $K=011$ یا $K=101$ را تأیید می‌کند. اگرچه رمزشکن مقدار دقیق کلید را نمی‌داند، اما می‌داند بیت سمت راست ۱ است (بیت مشترک بین دو مقدار). حملات بیشتر، با این فرض که بیت سمت راست در هر کلید ۱ است، بیت‌های بیشتری را در هر کلید آشکار می‌کند.

مراحل کلی: رمزهای بلوکی پیشرفته بیش از آنچه که در این فصل مورد بحث قرار گرفت، پیچیده‌اند. علاوه بر این، این رمزها از گردش‌های مختلفی تشکیل شده‌اند. رمزشکن می‌تواند از راهکار زیر استفاده کند:

(۱) از آنجا که دو گردش یکسان است، رمزشکن می‌تواند برای هر جعبه S ، یک جدول توزیع دیفرانسیل (پرو فایل XOR) تشکیل دهد و به منظور ایجاد توزیع برای هر گردش، جدول‌ها را با هم ترکیب نماید.

(۲) با فرض این که هر گردش مستقل از هم هستند، رمزشکن می‌تواند به وسیله‌ی ضرب احتمالات متناظر، جدول توزیع برای کل رمز تشکیل دهد.

(۳) اکنون رمزشکن می‌تواند بر اساس جدول توزیع مرحله‌ی پیش، لیستی از متون عادی برای حمله تهیه کند. توجه داشته باشید که جدول مرحله ۲ فقط به رمزشکن در انتخاب تعداد کوچک‌تری از زوج‌های متن رمز-متن عادی کمک می‌کند.

(۴) رمزشکن متن رمزی را انتخاب کرده و متن عادی متناظر آن را می‌یابد، سپس برای پیدا کردن بیت‌هایی در کلید، نتیجه را تجزیه و تحلیل می‌کند.

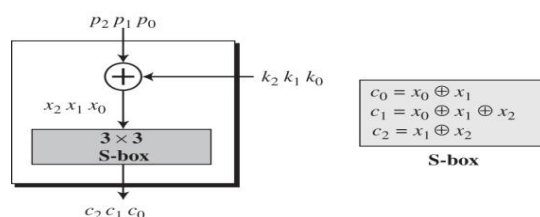
(۵) رمزشکن به منظور یافتن بیت‌های بیشتری در کلید، مرحله‌ی ۴ را تکرار می‌کند.

(۶) پس از پیدا کردن بیت‌های کافی از کلید، رمزشکن می‌تواند برای یافتن کل کلید، حمله از نوع جستجوی جامع کلیه کلیدها را انجام دهد.

تحلیل رمز دیفرانسیلی بر پایه‌ی جدول توزیع دیفرانسیل جعبه‌های نا هم‌شکل S در رمز بلوکی می‌باشد.

تحلیل رمز خطی

در سال ۱۹۹۳ میتسورو ماتسویی^۱ تحلیل رمز خطی را ارائه نمود. این تجزیه و تحلیل از حملات متن عادی شناخته شده (به جای حملات متن عادی منتخب در تحلیل رمز دیفرانسیلی) استفاده می‌کند. کل بحث این حمله مبتنی بر مفاهیم احتمالاتی است که فراتر از حوزه‌ی این کتاب است. برای مشاهده‌ی ایده‌ی اصلی پشت این حمله، فرض کنید که رمز تنها از یک گردش حاصل شده است. در شکل ۵-۲۰ C_0 ، C_1 و C_2 بیانگر سه بیت خروجی و X_0 ، X_1 و X_2 بیانگر سه بیت ورودی جعبه S می‌باشد. همان‌گونه که در ابتدای این فصل گفتیم، جعبه S یک دگرگونی خطی است که در آن، هر خروجی حاصل یک تابع خطی از ورودی است. با این مؤلفه‌ی خطی، همان‌گونه که در شکل زیر نشان داده شده است، می‌توانیم سه معادله‌ی خطی بین بیت‌های متن عادی و متن رمز ایجاد کنیم.



شکل ۵-۲۰: یک رمز عادی با یک جعبه S خطی

$$C_0 = P_0 \oplus K_0 \oplus P_1 \oplus K_1$$

$$C_1 = P_0 \oplus K_0 \oplus P_1 \oplus K_1 \oplus P_2 \oplus K_2$$

$$C_2 = P_1 \oplus K_1 \oplus P_2 \oplus K_2$$

با حل این معادلات، سه مجهول به دست می‌آید:

$$K_1 = (P_1) \oplus (C_0 \oplus C_1 \oplus C_2)$$

$$K_2 = (P_2) \oplus (C_1 \oplus C_2)$$

$$K_0 = (P_0) \oplus (C_1 \oplus C_2)$$

¹ Mitsuru Matsui

یعنی با سه حمله، متن عادی شناخته شده، می‌تواند مقادیر K_1 ، K_2 و K_3 را به دست آورد. با این وجود، رمزهای بلوکی واقعی به این سادگی نیستند، مؤلفه‌های بیشتری دارند و جعبه‌های S آن‌ها خطی نیست.

تقریب خطی^۱: در برخی از رمزهای بلوکی پیشرفته ممکن است برخی از جعبه‌های S به طور کامل غیرخطی نباشند. از نظر احتمالاتی، با استفاده از برخی توابع خطی می‌توان آن‌ها را تخمین زد. به طور کلی، با در نظر گرفتن یک رمز با متن عادی و متن رمز n بیتی و یک کلید m بیتی، به دنبال معادلاتی به شکل زیر می‌گردیم.

$$(K_1 \oplus K_2 \oplus \dots \oplus K_x) \oplus (P_1 \oplus P_2 \oplus \dots \oplus P_y) \oplus (C_1 \oplus C_2 \oplus \dots \oplus C_z)$$

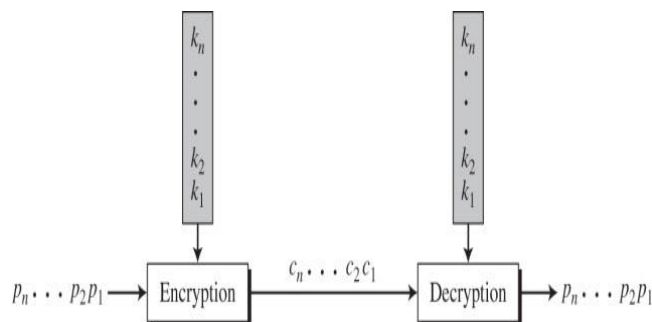
طوری که $1 \leq x \leq m$ ، $1 \leq y \leq n$ ، $1 \leq z \leq n$ باشد. می‌توان از بیت‌های موجود در متن عادی و متن رمز استراق سمع شده برای یافتن بیت‌های کلید استفاده کرد. برای مؤثر بودن این روش، هر معادله باید احتمالی به اندازه $1/2 + \epsilon$ داشته باشد، به ϵ گرایش (اریب)^۲ می‌گویند. معادله‌ای با ϵ بزرگ‌تر، مؤثرتر از معادله‌ای با ϵ کوچک‌تر می‌باشد.

۵-۲- رمزهای جریانی پیشرفته

در فصل ۳ تفاوت رمزهای جریانی سنتی و رمزهای بلوکی سنتی به اختصار بیان شد. تفاوت‌های مشابهی بین رمزهای جریانی پیشرفته و رمزهای بلوکی پیشرفته وجود دارد. در رمز جریانی پیشرفته، رمزنگاری و رمزگشایی ۲ بیت همزمان انجام می‌شود. جریان بیتی متن عادی به صورت $P = P_1 P_2 \dots P_n$ ، جریان بیتی متن رمز به شکل $C = C_1 C_2 \dots C_n$ و جریان بیتی کلید به شکل $K = K_1 K_2 \dots K_n$ مفروض‌اند که در آن P_i ، C_i و K_i کلمات ۲ بیتی هستند. همان‌گونه که در شکل ۵-۲۱ نشان داده شده است، رمزنگاری به صورت $P_i = D(K_i, C_i)$ و رمزگشایی به شکل $C_i = E(K_i, P_i)$ انجام می‌شود.

^۱ Linear Approximation

^۲ Bias



شکل ۵-۲۱: رمز جریانی

رمزهای جریانی سریع‌تر از رمزهای بلوکی هستند و پیاده سازی سخت‌افزاری رمز جریانی نیز ساده‌تر است. وقتی نیاز است جریان‌هایی به شکل بیت را رمزنگاری کرده و آن‌ها را در یک نرخ ثابت انتقال دهیم، استفاده از رمز جریانی گزینه‌ی بهتری است. همچنین رمزهای جریانی در برابر تحریف بیت‌ها در زمان انتقال، ایمن‌تر هستند.

در رمز جریانی پیشرفته، هر کلمه r بیتی در متن عادی با استفاده از یک کلمه r بیتی از جریان کلید رمزنویسی شده تا بدین ترتیب کلمه r بیتی متناظر آن به شکل جریان متن رمزی ایجاد شود. با نگاهی به شکل ۵-۲۱ می‌توان حدس زد که مسئله اصلی در رمزهای جریانی پیشرفته، چگونگی ایجاد جریان کلید $K = K_1 K_2 \dots K_n$ است. رمزهای جریانی پیشرفته به دو گروه گسترده تقسیم می‌شوند: همگام و ناهمگام.

رمزهای جریانی همگام^۱

در رمز جریانی همگام، جریان کلید، مستقل از جریان متن عادی یا متن رمز می‌باشد. کلید بدون هیچ گونه رابطه‌ای بین بیت‌های کلید و بیت‌های متن عادی یا متن رمز، ایجاد و استفاده می‌شود.

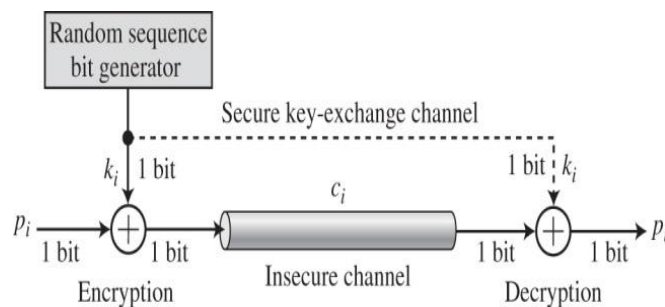
^۱ در رمز جریانی همگام، جریان کلید، مستقل از جریان متن عادی یا متن رمز است.

=====

^۱ Synchronous Stream Ciphers

رمز یک بار مصرف^۱

ساده‌ترین و مطمئن‌ترین نوع رمز جریانی همزمان، رمز یک بار مصرف است که شخصی به نام گیلبرت ورنام^۲ آن را ابداع و ثبت کرد. رمز یک بار مصرف از جریان کلیدی استفاده می‌کند که به طور تصادفی برای هر رمزنگاری انتخاب می‌شود. الگوریتم‌های رمزنگاری و رمزگشایی هر کدام از یک عمل XOR استفاده می‌کنند. بنا بر ویژگی‌های XOR که پیشتر در مورد آن‌ها صحبت کردیم، الگوریتم‌های رمزنگاری و رمزگشایی، وارون یکدیگرند. شایان توجه است که در این گونه رمزنگاری، عمل XOR هر بار بر روی یک بیت انجام می‌شود؛ به عبارت دیگر، عمل XOR بر روی کلمه ۱ بیتی و در دامنه $GF(2)$ است. همچنین توجه داشته باشید که باید کانال مطمئنی وجود داشته باشد به طوری که آلیس بتواند توالی جریان کلید را برای باب بفرستد.



شکل ۵-۲۲: رمز یک بار مصرف

رمز یک بار مصرف، رمزی ایده‌آل و کامل است. هیچ کس نمی‌تواند کلید یا ویژگی‌های آماری متن عادی یا متن رمز را حدس بزند. رابطه‌ای هم بین متن رمز و متن عادی وجود ندارد؛ به عبارت دیگر حتی اگر متن عادی الگوهایی داشته باشد، در واقع متن رمزی به یک جریان تصادفی از بیت‌ها تبدیل می‌شود. رمز شکن نمی‌تواند رمز را بشکند مگر این که تمام جریان‌های کلید تصادفی احتمالی را آزمایش کند و اگر اندازه‌ی متن عادی n بیت باشد، جریان‌های کلید تصادفی 2^n خواهد بود. ولی یک مسئله باقی است و آن این است که چگونه فرستنده و گیرنده، هر بار که می‌خواهند با هم ارتباط برقرار کنند یک کلید از رمز یک بار مصرف را به اشتراک بگذارند؟ آن‌ها باید به شکلی بر سر یک کلید تصادفی توافق کنند؛ بنابراین دستیابی به این رمز ایده‌آل و کامل بسیار مشکل است.

=====

¹ One- Time Pad

² Gilbert Vernam

مثال ۵-۱۷

الگو در متن رمز یک بار مصرف در هر یک از حالات زیر چیست؟

- (a) متن عادی از n عدد ۰ تشکیل شده است.
- (b) متن عادی از n عدد ۱ تشکیل شده است.
- (c) متن عادی از n عدد ۰ و ۱ متفاوت تشکیل شده است.
- (d) متن عادی توالی تصادفی از بیت‌هاست.

حل

(a) چون $K_i \oplus K_i = 0$ است، جریان متن رمز با جریان کلید یکسان است. اگر جریان کلید تصادفی باشد، متن رمز نیز تصادفی است. الگوهای متن عادی در متن رمز حفظ نمی‌شود.

(b) چون $K_i \oplus K_i = 1$ طوری که \bar{K}_i متمم K_i است، جریان متن رمز متمم جریان کلید می‌باشد. اگر جریان کلید تصادفی باشد، متن رمز نیز تصادفی است. باز هم الگوهای متن عادی در متن رمز محفوظ نمی‌ماند.

(c) در این حال، هر بیت در جریان متن رمز یا با بیت متناظر خود در کلید یکسان یا متمم آن است. بنابراین اگر جریان کلید تصادفی باشد، نتیجه هم یک توالی تصادفی است.

(d) در این حالت متن رمز قطعاً تصادفی است زیرا حاصل XOR دو بیت تصادفی، یک بیت تصادفی است.

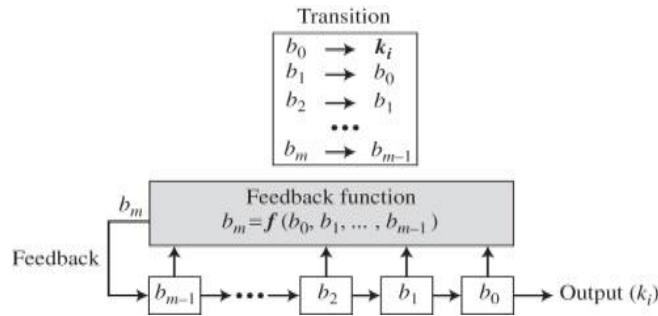
شیفت رجستر پس‌خور^۱

توافق بر سر رمز یک‌بار مصرف را می‌توان با استفاده از شیفت رجستر پس‌خور انجام داد. شیفت رجستر پس‌خور را می‌توان هم در نرم‌افزار و هم در سخت‌افزار پیاده‌سازی کرد، اما بررسی پیاده‌سازی سخت‌افزاری آن ساده‌تر است. همان‌گونه که در شکل ۵-۲۳ نشان داده شده است، شیفت رجستر پس‌خور از یک شیفت رجستر و یک تابع پس‌خور^۲ تشکیل شده است.

=====

^۱ Feedback Shift Register

^۲ Feedback Function



شکل ۵-۲۳: شیفت رجستر پس خور

شیفت رجستر، زنجیره‌ای از m سلول، b تا b_m است طوری که هر سلول تنها یک بیت را در خود نگه می‌دارد. مقدار اولیه‌ی سلول، یک کلمه m بیتی است که به آن مقدار اولیه یا دانه^۱ گفته می‌شود. هر وقت یک بیت خروجی مورد نیاز باشد، هر بیت، یک سلول به سمت راست جابجا می‌شود؛ این بدان معناست که هر سلول مقدارش را به خانه‌ی سمت راست خود می‌دهد و مقدار خود را از خانه‌ی سمت چپ دریافت می‌کند. اولین خانه‌ی سمت راست، b مقدارش را به عنوان خروجی به (K_i) می‌دهد. نخستین خانه سمت چپ، b_{m-1} مقدارش را از تابع پس‌خور می‌گیرد. خروجی تابع پس‌خور را b_m می‌نامیم. تابع پس‌خور نحوه‌ی ترکیب مقادیر سایر خانه‌ها برای محاسبه b_m را مشخص می‌کند. پس‌خور می‌تواند خطی یا غیرخطی باشد.

شیفت رجستر پس‌خور خطی^۲: در شیفت رجستر پس‌خور خطی، b_m یک تابع خطی از b_{m-1}, \dots, b_1, b به شکل زیر است.

$$b_m = c_{m-1}b_{m-1} + \dots + c_2b_2 + c_1b_1 + c.b. \quad (c_i \neq 0)$$

از آنجایی که با ارقام دودویی سروکار داریم و ضرب و جمع در میدان $GF(2)$ انجام می‌شود، پس مقدار c_i برابر ۱ یا ۰ است. اما c باید پیشتر برابر ۱ شود تا پس‌خوری از خروج به دست آید. عمل جمع هم XOR است. به عبارت دیگر:

$$b_m = c_{m-1}b_{m-1} + \dots + c_2b_2 + c_1b_1 + c.b. \quad (c_i \neq 0)$$

مثال ۵-۱۸

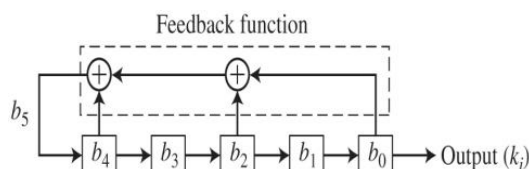
یک شیفت رجستر پس‌خور خطی با ۵ خانه ایجاد کنید که در آن $b_5 = b_4 \oplus b_2 \oplus b$ باشد.

¹ Seed

² Linear Feedback Shift Register

حل

اگر $c_i=0$ باشد، b_i نقشی در محاسبه b_m ندارد، یعنی b_i به تابع پس‌خورد متصل نیست. اگر $c_i=1$ باشد، b_i در محاسبه b_m نقش دارد. در این مثال c_1 و c_3 صفرند. این بدین معنی است که فقط سه اتصال داریم. در شکل ۵-۲۴ این طرح را ترسیم کرده‌ایم.



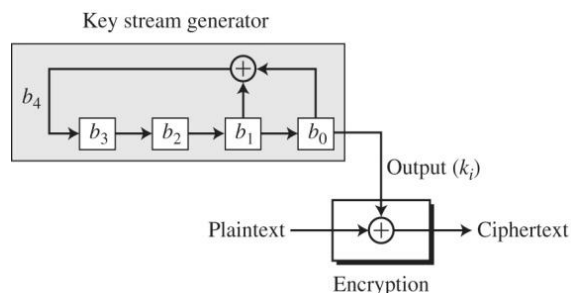
شکل ۵-۲۴: شیفت رجستر پس‌خور خطی برای مثال ۵-۱۸

مثال ۵-۱۹

یک شیفت رجستر پس‌خور خطی با ۴ خانه ایجاد کنید که در آن $b_4 = b_1 \oplus b_0$ باشد. اگر مقدار اولیه $(0001)_2$ باشد، مقدار خروجی برای ۲۰ انتقال را نشان دهید.

حل

شکل ۵-۲۵ طراحی و کاربرد شیفت رجستر پس‌خور خطی در رمزنگاری را نشان می‌دهد.



شکل ۵-۲۵: یک شیفت رجستر پس‌خور خطی برای مثال ۵-۱۹

جدول ۵-۶ مقادیر جریان کلید را نشان می‌دهد. برای هر انتقال، ابتدا مقدار b_4 محاسبه می‌شود و سپس هر بیت یک خانه به سمت راست جابجا می‌شوند.

جدول ۵-۶: مقادیر خانه و ترتیب کلید برای مثال ۵-۱۹

States	b_4	b_3	b_2	b_1	b_0	k_i
Initial	1	0	0	0	1	
1	0	1	0	0	0	1
2	0	0	1	0	0	0
3	1	0	0	1	0	0
4	1	1	0	0	1	0
5	0	1	1	0	0	1
6	1	0	1	1	0	0
7	0	1	0	1	1	0
8	1	0	1	0	1	1
9	1	1	0	1	0	1
10	1	1	1	0	1	0
11	1	1	1	1	0	1
12	0	1	1	1	1	0
13	0	0	1	1	1	1
14	0	0	0	1	1	1
15	1	0	0	0	1	1
16	0	1	0	0	0	1
17	0	0	1	0	0	0
18	1	0	0	1	0	0
19	1	1	0	0	1	0
20	1	1	1	0	0	1

توجه داشته باشید که جریان کلید ...۱۰۰۰۱ ۱۰۱۱۱۰۱۰۱۱۰۱۰۱ است. در نگاه اول این جریان مانند یک توالی تصادفی به نظر می‌رسد، اما اگر در انتقال‌ها دقیق‌تر شویم، می‌بینیم توالی متناوب است. همان‌گونه که در رشته‌ی زیر نشان داده شده است، این توالی تکرار ۱۵ بیت است.

۱۰۰۱۰۰۱۱۰۱۰۱۱۱ ۱۰۰۱۰۰۱۱۰۱۰۱۱۱ ۱۰۰۰۱۰۰۱۱۰۱۰۱۱ ۱۰۰۰۱۰۰۱۱۰۱۰۱۱...

جریان کلید حاصل از شیفت رجستر پس‌خور خطی توالی شبه تصادفی است که پس از n بیت تکرار می‌شود. این جریان یک دوره تناوب دارد، اما این تناوب برابر مقدار اولیه یعنی ۴ نیست. بر اساس شیوه‌ی طراحی و مقدار اولیه، تناوب می‌تواند تا $2^m - 1$ هم برسد. علت آن است که مقدار اولیه m بیتی می‌تواند تا 2^m الگوی متفاوت از تمام بیت‌های ۰ تا تمام بیت‌های ۱ را ایجاد کند؛ با این وجود اگر مقدار اولیه تماماً ۰ باشد بلااستفاده است. متن اولیه باید جریان مداومی از ۰ باشد، بنابراین این مقدار را در نظر نمی‌گیریم.

حداکثر دوره تناوب یک شیفت رجستر پس‌خور خطی m بیتی، برابر $2^m - 1$ است.

در مثال قبلی، دوره‌ی تناوب حداکثر (۱۵ = $2^4 - 1$) است. برای دستیابی به حداکثر دوره‌ی تناوب (تصادفی بودن بهتر) باید نخست تابع پس‌خور را به صورت یک چندجمله‌ای خاص با ضرایبی در میدان $GF(2)$ در نظر بگیریم.

$$b_m = c_{m-1}b_{m-1} + \dots + c_1b_1 + c_0b_0 \rightarrow x^m = c_{m-1}x^{m-1} + \dots + c_1x^1 + c_0x^0$$

از آنجا که در این میدان عمل جمع و تفریق یکسان است، تمام عبارات را می‌توان به یک طرف تساوی منتقل کرد و بدین ترتیب یک چندجمله‌ای با درجه m (که از آن به عنوان چندجمله‌ای خاص یاد می‌شود) به شکل زیر به دست آورد:

$$X^m + C_{m-1}X^{m-1} + \dots + C_1X + C_0X' = 0$$

شیفت رجستر پس‌خور خطی: این شیفت رجستردارای دوره تناوب حداکثر $2^m - 1$ است به شرطی که تعداد سلول‌های آن زوج و چندجمله‌ای خاص آن نیز یک چندجمله‌ای اولیه^۱ باشد. چندجمله‌ای اولیه، چندجمله‌ای غیرقابل کاهش است که بر $X^e + 1$ بخش‌پذیر است به قسمی که e کوچک‌ترین عدد صحیح به شکل $e = 2^k - 1$ و $k \geq 2$ باشد. تولید یک چندجمله‌ای اولیه کار ساده‌ای نیست پس یک چندجمله‌ای را نخست انتخاب، سپس نخستین بودن آن چک می‌شود. ولی، چندجمله‌ای‌های اولیه آزمایش شده‌ی بسیاری وجود دارد که می‌توان از میان آن‌ها، یکی را انتخاب کرد.

مثال ۵-۲۰

چندجمله‌ای خاص برای شیفت رجستر پس‌خور خطی در مثال ۱۹-۵ برابر $X^5 + X + 1$ می‌باشد که یک چندجمله‌ای اولیه است. جدول ۴-۴ (فصل ۴) نشان می‌دهد که این چندجمله‌ای، یک چندجمله‌ای کاهش‌ناپذیر است. همچنین این چندجمله‌ای بر $(X^3 + 1) = (X^2 + X + 1)(X + 1)$ بخش‌پذیر است که به معنی $e = 2^3 - 1 = 7$ می‌باشد.

حمله به شیفت رجستر پس‌خور خطی: شیفت رجستر پس‌خور خطی، ساختار بسیار ساده‌ای دارد و این سادگی باعث آسیب‌پذیری رمز در برابر حملات می‌شود. دو حمله‌ی رایج به حمله به شیفت رجستر پس‌خور خطی را بررسی می‌کنیم:

- (۱) اگر ساختار شیفت رجستر پس‌خور خطی شناخته شده باشد، رمزشکن می‌تواند پس از استراق سمع و تجزیه و تحلیل متن رمز n بیتی، تمام متون رمزی آتی را پیش‌بینی کند.
- (۲) اگر ساختار شیفت رجستر پس‌خور خطی شناخته شده نباشد، رمزشکن می‌تواند جهت شکستن رمز از حمله به شیوه‌ی متن عادی شناخته شده با $2n$ بیت استفاده کند.

=====

¹ Primitive Polynomial

شیفت ریجستر پس‌خور غیرخطی^۱: اساساً شیفت ریجستر پس‌خور خطی به علت خطی بودن، در برابر انواع حملات آسیب‌پذیر است. با استفاده از یک شیفت ریجستر پس‌خور غیرخطی می‌توان به رمز جریانی بهتری دست یافت. شیفت ریجستر پس‌خور غیرخطی دارای همان ساختار شیفت ریجستر پس-خور خطی است به جز اینکه b_m تابع غیرخطی از b_1, b_2, \dots, b_m است. مثلاً در شیفت ریجستر پس‌خور غیرخطی b_i بیتی، رابطه را می‌توان به شکل زیر نشان داد؛ طوری که AND و OR بیتی و متمم به شکل \bar{b} نشان داده شده است.

$$b_i = (b_3 \text{ AND } b_2) \text{ OR } (b_1 \text{ AND } \bar{b})$$

با این اوصاف، شیفت ریجستر پس‌خور غیرخطی مرسوم نیست، زیرا هیچ پایه و اساس ریاضی برای چگونگی ایجاد یک شیفت ریجستر پس‌خور غیرخطی با حداکثر دوره‌ی تفاوت وجود ندارد. ترکیب: رمز جریانی می‌تواند از ترکیب ساختار خطی و غیرخطی استفاده کند. برخی از شیفت ریجستر پس‌خور خطی را می‌توان با حداکثر دوره‌ی تناوب ساخته و سپس با یک تابع غیرخطی ترکیب نمود. رمزهای جریانی ناهمگام^۲:

در رمزهای جریانی ناهمگام، هر کلید در جریان کلید به متن عادی یا متن رمز قبلی بستگی دارد. در رمز جریانی ناهمگام، کلید به متن عادی یا به متن رمز بستگی دارد. در واقع دو روشی که برای ایجاد سبک‌های مختلف عمل برای رمزهای بلوکی مورد استفاده قرار می‌گرفت (سبک پس‌خور مبتنی بر خروجی و سبک شمارنده)، رمزهای جریانی را خلق کرده‌اند.

۳-۵ وب سایت‌های پیشنهادی

وب سایت‌های زیر اطلاعات بیشتری در زمینه‌ی موضوعات مطرح شده در این فصل را در اختیار شما قرار می‌دهد:

http://en.wikipedia.org/wiki/Feistel_cipher

<http://www.quadibloc.com/crypto/co040906.htm>

tiger.uic.edu/~jleon/mcs425-s05/handouts/feistel-diagram.pdf

=====

¹ Nonlinear Feedback Shift Register

² Nonsynchronous Stream Ciphers

۵-۴ چکیده

- * رمزهای کلید متقارن سنتی، رمزهای مبتنی بر نویسه هستند. با پیدایش کامپیوتر، نیازمند رمزهای مبتنی بر بیت هستیم.
- * رمز بلوکی پیشرفته کلید متقارن: یک بلوک n بیتی از متن عادی را رمزنگاری، یا یک بلوک n بیتی از متن رمز را رمزگشایی می‌کند. الگوریتم رمزنگاری یا رمزگشایی از یک کلید K بیتی یگانه استفاده می‌کند.
- * رمز بلوکی پیشرفته را می‌توان طوری طراحی کرد که مانند رمز جایگزین یا رمز انتقال عمل کند. با این وجود، برای مقاومت در برابر حمله‌ی آزمودن کلیدی کلیدها، رمز بلوکی پیشرفته باید همانند رمز جایگزین طراحی شده باشد.
- * معمولاً رمزهای بلوکی پیشرفته، رمزهای جایگزینی مبتنی بر کلید هستند که کلید فقط نگاشت بین ورودی‌های احتمالی را با خروجی‌های احتمالی امکان‌پذیر می‌کند.
- * رمز بلوکی پیشرفته، ترکیبی از جعبه‌های P ، واحدهای جایگزین، جعبه‌های S و چند واحد دیگر می‌باشد.
- * جعبه P (جعبه جایشگت)، رمز انتقال سنتی را به شکل نویسه‌ها، موازی سازی می‌نماید. سه نوع جعبه P وجود دارد: جعبه‌های P مستقیم، جعبه‌های P گسترشی و جعبه‌های P فشرده.
- * جعبه S (جایگزین) را می‌توان مینیاتوری از رمز جایگزین تلقی کرد؛ با این وجود، در جعبه S ممکن است تعداد مختلفی از ورودی و خروجی وجود داشته باشد.
- * عملگر چیره در بیشتر رمزهای بلوکی XOR است و آن را می‌توان به شکل جمع یا تفریق در میدان $GF(2^n)$ در نظر گرفت.
- * عملی که در برخی از رمزهای بلوکی پیشرفته یافت می‌شود، چرخش دایره‌ای است که این چرخش می‌تواند به سمت راست یا چپ صورت پذیرد. مبادله، حالت خاصی از چرخش دایره‌ای است به طوری که $K=n/2$ است. دو عملگر دیگری که می‌توان در برخی از رمزهای بلوکی یافت، تقسیم و ترکیب است.
- * شانن، مفهوم رمز فرآورده را ارائه نمود. رمز فرآورده، رمز پیچیده‌ای است که به منظور دستیابی به پخش و آشفتگی، جعبه‌های S ، جعبه‌های P و سایر مؤلفه‌ها را با هم ترکیب

می‌کند. پخش، رابطه‌ی بین متن عادی و متن رمز و آشفتگی و نیز کلید رمز و متن رمز را پنهان می‌کند.

* تمام رمزهای بلوکی پیشرفته، رمزهای فرآورده هستند، اما به دو رسته تقسیم می‌شوند: رمزهای Feistel و رمزهای غیر Feistel. رمزهای Fiestel هم از مؤلفه‌های وارون‌پذیر و هم از مؤلفه‌های وارون‌ناپذیر استفاده می‌کند. رمزهای غیر Feistel فقط مؤلفه‌های وارون‌پذیر را به کار می‌برد.

* برخی از حملات جدید به رمزهای بلوکی بر پایه‌ی ساختار رمزهای بلوکی بنا شده‌اند. این حملات از روش‌های تحلیل رمز خطی و دیفرانسیلی استفاده می‌کند.

* در رمز جریانی پیشرفته، هر کلمه ۲ بیتی در متن عادی با استفاده از یک کلمه ۲ بیتی در جریان کلید رمزنگاری می‌شود تا بدین ترتیب، کلمه ۲ بیتی متناظر در متن رمز به دست آید. رمزهای جریانی پیشرفته را می‌توان به دو دسته‌ی گسترده‌تر تقسیم کرد: رمزهای جریانی همزمان و رمزهای جریانی غیر همزمان. در رمز جریانی همزمان، کلید مستقل از متن عادی یا متن رمز است. در رمز جریانی غیر همزمان، کلید به متن عادی یا متن رمز بستگی دارد.

* ساده‌ترین و مطمئن‌ترین نوع رمز جریانی همزمان «رمز یک بار مصرف» است. رمز یک بار مصرف از کلیدی استفاده می‌کند که به طور تصادفی برای هر بار رمزنویسی تولید شده است. الگوریتم رمزنگاری و رمزگشایی هر کدام از عملگر XOR استفاده می‌کنند. رمز یک بار مصرف عملی نیست زیرا برای برقراری ارتباط می‌بایست تبادل کلید برای هر انتقال انجام شود. نمونه عملی برای رمز یک بار مصرف، شیفت ریجستر پس‌خور است، که می‌توان آن را به صورت سخت‌افزاری یا نرم‌افزاری پیاده‌سازی کرد.

۵-۵- مجموعه تمرین‌ها

پرسش‌ها دوره‌ای

- (۱) تفاوت بین رمز کلید متقارن سنتی و پیشرفته را بیان کنید.
- (۲) رمزهای بلوکی پیشرفته به جای اینکه به شکل رمزهای انتقال طراحی شوند، به صورت رمزهای جایگزین طراحی شده است. علت آن را بیان کنید.
- (۳) چرا رمزهای جایگزین و رمزهای انتقال را می‌توان جایگشت تلقی کرد؟
- (۴) چند مؤلفه‌ی رمز بلوکی پیشرفته را نام ببرید.
- (۵) جعبه P را تعریف کنید و سه نوع آن را نام ببرید. کدام نوع وارون‌پذیر است.
- (۶) جعبه S را تعریف کنید و شرایط لازم برای وارون‌پذیری جعبه S را ذکر نمایید.
- (۷) رمز فرآورده را تعریف کرده و دو رسته آن را نام ببرید.
- (۸) تفاوت بین پخش و آشفتگی را بیان کنید.
- (۹) تفاوت بین رمز Feistel و رمز غیر Feistel چیست؟
- (۱۰) تفاوت بین تحلیل رمز دیفرانسیلی و خطی را بیان کنید. کدام یک از آن‌ها حمله به شیوه‌ی متن عادی / منتخب و کدام یک حمله به شیوه‌ی متن عادی شناخته شده است؟
- (۱۱) تفاوت بین رمز جریانی همزمان و غیر همزمان را بیان کنید.
- (۱۲) شیفت ریجستر پس‌خور را تعریف کرده و موارد استفاده از آن را در رمزهای جریانی نام ببرید؟

تمرینات

- (۱۳) یک بلوک انتقال دارای ۱۰ ورودی و ۱۰ خروجی است. ترتیب گروه جایگشت چیست؟ اندازه کلید چند است؟
- (۱۴) یک بلوک جایگزین دارای ۱۰ ورودی و ۱۰ خروجی است. ترتیب گروه جایگشت چیست؟ اندازه کلید چند است؟
- (۱۵)
 - (a) حاصل چرخش به چپ دایره‌ای ۳ بیتی روی کلمه‌ی $(۱۰۰۱۱۰۱۱)_۲$ را نشان دهید.
 - (b) حاصل چرخش به راست دایره‌ای ۳ بیتی روی کلمه حاصل از قسمت پیش را نشان دهید.
 - (c) نتیجه‌ی قسمت b را با کلمه‌ی اصلی در قسمت a مقایسه کنید.
- (۱۶) (a) کلمه $(۱۰۰۱۱۰۱۱)_۲$ را مبادله کنید.

(b) کلمه‌ی حاصل از قسمت یک را مبادله کنید.

(c) با مقایسه‌ی نتیجه‌ی قسمت a و b نشان دهید که مبادله‌ی یک عمل خود وارون است.

(۱۷) حاصل اعمال زیر را به دست آورید:

a) $(01001101) \oplus (01001101)$

b) $(01001101) \oplus (10110010)$

c) $(01001101) \oplus (00000000)$

d) $(01001101) \oplus (11111111)$

(۱۸)

(a) با استفاده از یک دیکدر 8×3 ، کلمه‌ی ۰۱۰ را کدگذاری کنید.

(b) با استفاده از یک کدگذار 8×3 ، کلمه‌ی ۰۰۱۰۰۰۰۰ را کد نویسی کنید.

(۱۹) یک پیام ۲۰۰۰ نویسه دارد. اگر قرار باشد پیام را با استفاده از رمز بلوکی ۶۴ بیتی رمزنگاری کنیم، اندازه و تعداد بلوک‌ها را پیدا کنید.

(۲۰) جدول جایگشت حاصل از جعبه‌ی P مستقیم در شکل ۵-۴ را تشکیل دهید.

(۲۱) جدول جایگشت حاصل از جعبه‌ی P فشرده در شکل ۵-۴ را تشکیل دهید.

(۲۲) جدول جایگشت حاصل از جعبه‌ی P گسترشی در شکل ۵-۴ را تشکیل دهید.

(۲۳) جعبه P حاصل از جدول زیر را به دست آورید.

8	1	2	3	4	5	6	7
---	---	---	---	---	---	---	---

(۲۴) تعیین کنید که آیا جعبه P با جدول جایگشت زیر، یک جعبه‌ی P مستقیم، فشرده یا P گسترشی است؟

1	1	2	3	4	4
---	---	---	---	---	---

(۲۵) تعیین کنید که آیا جعبه‌ی P با جدول جایگشت زیر، یک جعبه‌ی P مستقیم، یک جعبه‌ی P فشرده یا یک جعبه‌ی P گسترشی است.

1	3	5	6	7
---	---	---	---	---

(۲۶) تعیین کنید که آیا جعبه‌ی P با جدول جایگشت زیر، یک جعبه‌ی P مستقیم، یک جعبه‌ی P فشرده یا یک جعبه‌ی P گسترشی است.

1	2	3	4	5	6
---	---	---	---	---	---

(۲۷) جدول زیر، رابطه‌ی ورودی/خروجی در یک جعبه‌ی S به اندازه 2×2 را نشان می‌دهد. جدول را برای جعبه‌ی S وارون نشان دهید.

	Input : right bit	
	۰	۱
Input : left bit	۰	۰۱
	۱	۱۰
		۰۰

(۲۸) یک شیفت ریجستر پس‌خور خطی با چندجمله‌ای خاص $x^0 + x^2 + 1$ را رسم کنید. دوره‌ی تناوب آن چیست؟

(۲۹) چندجمله‌ای خاص شیفت ریجستر پس‌خور خطی زیر چیست؟ حداکثر دوره‌ی تناوب چقدر است؟



(۳۰) اگر مقدار عادی برابر ۱۱۱۰ باشد، کلید ۲۰ بیتی ایجاد شده از شیفت ریجستر پس‌خور خطی در شکل ۵-۲۵ را به دست آورید؟

(۳۱) حداکثر طول دوره‌ی یک شیفت ریجستر پس‌خور خطی ۳۲ است. این شیفت ریجستر چند بیت دارد؟

(۳۲) در جعبه S به اندازه‌ی 2×6 برای به دست آوردن بیت چپ خروجی، عمل XOR را بر روی بیت‌های فرد، و برای به دست آوردن بیت سمت راست خروجی، عمل XOR را بر روی بیت‌های زوج انجام می‌دهد. اگر ورودی ۱۰۱۱۰۱ باشد، خروجی چیست؟

(۳۳) سمت چپ‌ترین بیت جعبه‌ی S به اندازه‌ی 3×8 ، سه بیت دیگر را به شرح زیر می‌چرخاند: اگر سمت چپ‌ترین بیت ۰ باشد، سه بیت دیگر یک بیت به سمت راست می‌روند.

اگر سمت چپ‌ترین بیت ۱ باشد، سه بیت دیگر یک بیت به سمت چپ می‌روند.

اگر ورودی ۱۰۱۱ باشد، خروجی چیست؟ اگر ورودی ۰۱۱۰ باشد، خروجی چیست؟

(۳۴) برنامه‌ای به صورت شبه کد بنویسید که یک کلمه n بیتی را به دو کلمه تقسیم کند به طوری که هر کلمه $n/2$ بیت داشته باشد.

(۳۵) برنامه‌ای به صورت شبه کد بنویسید که دو کلمه‌ی n بیتی را به یک کلمه‌ی $n/2$ بیتی تبدیل کند.

(۳۶) برنامه‌ای به صورت شبه کد بنویسید که نیمه‌های چپ و راست یک کلمه‌ی n بیتی را جابجا کند.

- (۳۷) برنامه‌ای به صورت شبه کد بنویسید که بر مبنای اولین پارامتری که به تابع داده می‌شود، یک کلمه‌ی n بیتی را K بیت به سمت راست یا چپ به صورت دایره‌ای بچرخاند.
- (۳۸) برنامه‌ای به صورت شبه کد برای جعبه‌ی P بنویسید که جایگشت‌هایش به وسیله‌ی جدول تعیین شود.
- (۳۹) برای جعبه‌ی S برنامه‌ای به صورت شبه کد بنویسید که ورودی/خروجی‌هایش به وسیله‌ی جدول تعیین شوند.
- (۴۰) برنامه‌ای به صورت شبه کد بنویسید که گردش‌های رمز غیر Feistel توضیح داده شده در شکل ۵-۱۳ را شبیه سازی کند.
- (۴۱) برنامه‌ای به صورت شبه کد بنویسید که گردش‌های رمز Feistel توضیح داده شده در شکل ۵-۱۷ را شبیه سازی کند.
- (۴۲) برنامه‌ای به صورت شبه کد بنویسید که شیفت رجیستر پس‌خور خطی n بیتی را شبیه سازی کند.

فصل ۶

استاندارد رمزنگاری داده‌ها^۱

چشم‌انداز

در این فصل در مورد استاندارد رمزنگاری داده‌ها، رمز بلوکی کلید متقارن پیشرفته صحبت می‌کنیم. موارد زیر چشم‌اندازهای اصلی این فصل هستند:

- ✱ مروری بر تاریخچه DES.
- ✱ تعریف ساختار اصلی DES.
- ✱ توصیف جزئیات چگونگی ساخت اجزاء DES.
- ✱ توصیف مراحل ایجاد کلیدهای گردشی.
- ✱ تجزیه و تحلیل DES.
- ✱ شناخت چگونگی استفاده DES از رمزنگاری Feistel برای دستیابی به پخش و آشفتگی بیت‌ها از متن عادی به متن رمز.

۶-۱- مقدمه

استاندارد رمزنگاری داده‌ها، DES، رمز بلوکی کلید متقارن می‌باشد که به وسیله‌ی موسسه‌ی ملی استاندارد و فناوری^۲ منتشر شده است.

=====

^۱ Data Encryption Standard (DES)

^۲ National Institute of Standards and Technology (NIST)

تاریخچه

در سال ۱۹۷۳ میلادی NIST فراخوانی برای طرح‌های پیشنهادی سیستم رمز کلید متقارن ملی منتشر کرد. طرحی از IBM، که تعدیلی از پروژه‌ای موسوم به Lucifer بود، به عنوان DES پذیرفته شد. در مارس ۱۹۷۵ میلادی به عنوان پیش‌نویسی از استاندارد پردازش اطلاعات فدرال^۱ در فدرال ریجستر منتشر شد.

پس از انتشار، پیش‌نویس به دو دلیل به شدت مورد انتقاد قرار گرفت. نخستین انتقاد در مورد طول کوتاه کلید (فقط ۵۶ بیت) بود که باعث می‌شد رمز در برابر حمله‌ی آزمودن کلیه کلیدها آسیب‌پذیر باشد. دومین انتقاد به برخی طراحی‌های مخفی پشت پرده در ساختار داخلی DES مربوط می‌شد، زیرا مشکوک بودند که برخی از بخش‌های این ساختار (جعبه‌های P) ممکن است دریچه‌های پنهانی داشته باشند که رمزگشایی پیام را بدون نیاز به کلید برای آژانس امنیت ملی^۲ میسر کند. در نتیجه طراحان IBM اطمینان خاطر دادند که ساختارهای درونی طوری طراحی شده‌اند که مانع تحلیل رمز به شیوه‌ی تحلیل دیفرانسیلی شوند.

سرانجام در ژانویه ۱۹۷۷ میلادی DES تحت عنوان FIPS در فدرال ریجستر منتشر شد. با این وجود NIST، DES را به عنوان استاندارد برای استفاده در کاربردهای طبقه‌بندی نشده پیشنهاد کرد. DES از زمان انتشارش تاکنون پرکاربردترین رمز بلوکی کلید متقارن می‌باشد. سپس NIST استاندارد جدیدی (FIPS 46-3) ارائه داد. این استاندارد جدید، استفاده از DES سه گانه^۳ (سه برابر تکرار رمز در DES) را برای کاربردهای آتی توصیه می‌کرد. همان‌گونه که در فصل ۷ خواهیم دید استاندارد جدید AES قرار است در بلندمدت جایگزین DES شود.

نمای کلی

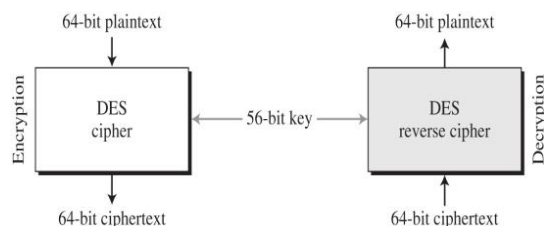
همان‌گونه که در شکل ۶-۱ نشان داده شده است، DES رمز بلوکی است.

=====

¹ Federal Information Processing Standard (FIPS)

² National Security Agency

³ Triple DES



شکل ۶-۱: رمزنگاری و رمزگشایی با DES

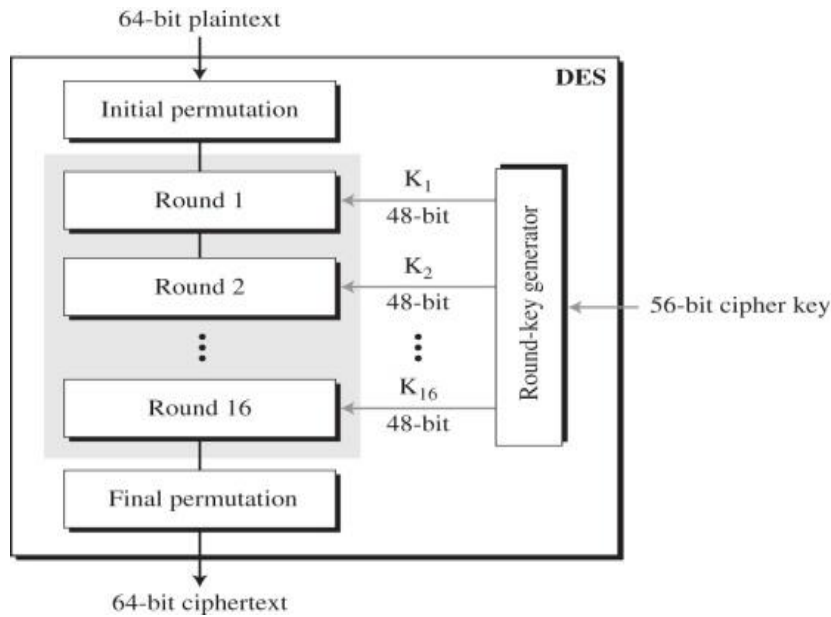
در بخش رمزنگاری، DES متن عادی ۶۴ بیتی را دریافت و آن را تبدیل به متن رمز ۶۴ بیتی می‌کند. در بخش رمزگشایی، DES، متن رمز ۶۴ بیتی را دریافت و بلوک ۶۴ بیتی از متن عادی را تولید می‌کند. هم برای رمزگشایی و هم برای رمزنگاری از کلید رمز ۵۶ بیتی یگانه استفاده می‌شود.

۲-۶ ساختار DES

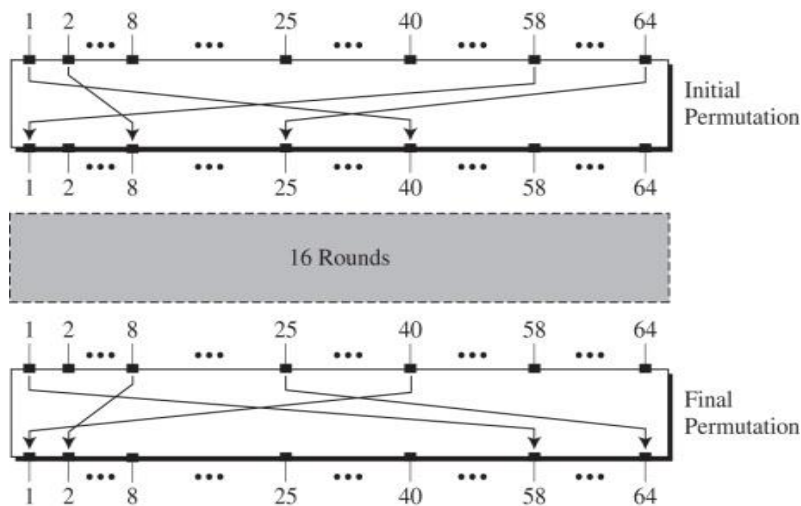
اجازه دهید بر روی رمزنگاری تمرکز کنیم، پس از آن در مورد رمزگشایی بحث خواهیم کرد. مراحل رمزنگاری از دو جایگشت (جعبه P) که ما آن را جایگشت نخستین و نهایی می‌نامیم و از شانزده گردش Feistel تشکیل شده است. هر گردش از کلید ۴۸ بیتی متفاوتی استفاده می‌کند. این کلید ۴۸ بیتی بر اساس الگوریتم از پیش تعیین شده‌ای که بعداً در مورد آن توضیح خواهیم داد، تولید می‌شود. شکل ۶-۲ اجزاء DES در بخش رمزنگاری را نشان می‌دهد.

جایگشت‌های نخستین و نهایی

شکل ۶-۳ جایگشت‌های نخستین و نهایی (جعبه‌های P) را نشان می‌دهد. هر یک از این جایگشت‌ها یک ورودی ۶۴ بیتی را دریافت و برابر قاعده‌ای از پیش تعیین شده‌ای آن‌ها را جابجا می‌کنند. در این شکل تنها تعداد محدودی درگاه ورودی و درگاه خروجی متناظر آن‌ها را نشان داده‌ایم. این جایگشت‌ها، جایگشت‌های مستقیم بدون کلیدی هستند که وارون یکدیگرند. مثلاً در جایگشت نخستین، پنجاه و هشتمین بیت ورودی، اولین بیت خروجی می‌شود. به همین ترتیب، در جایگشت نهایی، اولین بیت ورودی پنجاه و هشتمین بیت خروجی می‌شود؛ به عبارت دیگر، اگر بین این دو جایگشت گردش وجود نداشت، پنجاه و هشتمین بیت وارد شده به جایگشت نخستین با پنجاه و هشتمین بیت جایگشت نهایی یکسان است.



شکل ۶-۳: ساختار کلی DES



شکل ۶-۳: مراحل جایگشت نخستین و نهایی در DES

جدول ۶-۱ قوانین حاکم بر جایگشت جعبه‌های P را نشان می‌دهد. هر طرف جدول را می‌توان به عنوان یک آرایه‌ی ۶۴ عضوی در نظر گرفت. توجه داشته باشید که مانند هر جدول جایگشتی که تاکنون در مورد آن صحبت کرده‌ایم، مقدار هر عضو، شماره درگاه ورودی و ترتیب شاخص هر عضو، شماره درگاه خروجی را تعیین می‌کند.

جدول ۱-۶ جدول جایگشت عادی و نهایی

Initial Permutation	Final Permutation
58 50 42 34 26 18 10 02	40 08 48 16 56 24 64 32
60 52 44 36 28 20 12 04	39 07 47 15 55 23 63 31
62 54 46 38 30 22 14 06	38 06 46 14 54 22 62 30
64 56 48 40 32 24 16 08	37 05 45 13 53 21 61 29
57 49 41 33 25 17 09 01	36 04 44 12 52 20 60 28
59 51 43 35 27 19 11 03	35 03 43 11 51 19 59 27
61 53 45 37 29 21 13 05	34 02 42 10 50 18 58 26
63 55 47 39 31 23 15 07	33 01 41 09 49 17 57 25

این دو جایگشت اهمیتی رمزنگاری در DES ندارند. هر دو جایگشت، بدون کلید و از پیش تعیین شده‌اند. علت گنجانده شدن آن‌ها در DES هنوز مشخص نیست و طراحان DES نتوانسته‌اند علت را توضیح دهند. تنها تصور این است که DES برای اجرا در سخت‌افزار طراحی شده است و این دو جایگشت پیچیده ممکن است شبیه‌سازی نرم‌افزاری از این راهکار را ناکام بگذارد.

مثال ۱-۶

با در نظر گرفتن ورودی در مبنای ۱۶ به صورت زیر، خروجی جعبه‌ی جایگشت نخستین را به دست آورید.

۰X۰۰۰۲ ۰۰۰۰ ۰۰۰۰ ۰۰۰۱

حل

ورودی فقط دو عدد یک دارد (بیت ۱۵ و بیت ۶۴)؛ خروجی هم باید فقط دو عدد ۱ داشته باشد (ماهیت جایگشت مستقیم). با استفاده از جدول ۱-۶ می‌توانیم خروجی مرتبط با این دو بیت را به دست آوریم. بیت ۱۵ در ورودی به بیت ۶۳ در خروجی تبدیل می‌شود. بیت ۶۴ در ورودی به بیت ۲۵ در خروجی تبدیل می‌شود. بدین ترتیب، خروجی فقط دو تا یک دارد، بیت ۲۵ و بیت ۶۳. نتیجه در مبنای ۱۶ به این صورت است:

۰X۰۰۰۰ ۰۰۸۰ ۰۰۰۰ ۰۰۰۲

مثال ۲-۶

اگر ورودی به صورت زیر باشد، خروجی جایگشت نهایی را به دست آورده و با استفاده از آن ثابت کنید که جایگشت‌های نخستین و نهایی، وارون یکدیگرند.

۰X۰۰۰۰ ۰۰۸۰ ۰۰۰۰ ۰۰۰۲

حل

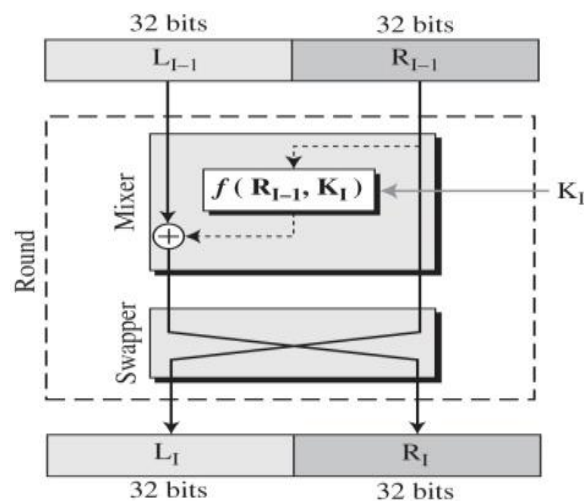
فقط بیت ۲۵ و ۶۴ یک هستند. سایر بیت‌ها، صفرند. در جایگشت نهایی، بیت ۲۵ به بیت ۶۴ و بیت ۶۳ به بیت ۱۵ منتقل می‌شود. نتیجه به این شرح است:

•X•••۲••••••••••

جایگشت‌های نخستین و نهایی جعبه‌های P مستقیمی هستند که وارون یکدیگرند و هیچ اهمیت ویژه‌ای در رمزنگاری DES ندارند.

گردش‌ها

DES از ۱۶ گردش استفاده می‌کند. همان‌گونه که در شکل ۶-۴ نشان داده شده است هر گردش در DES رمزی از نوع Feistel است.



شکل ۶-۴: یک گردش در DES (قسمت رمزنگاری)

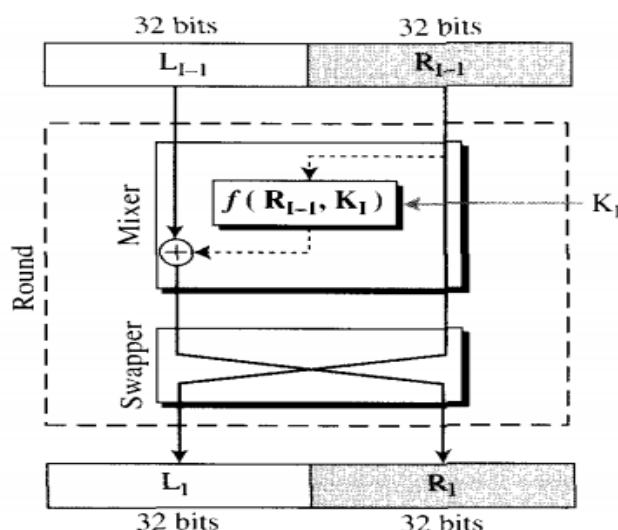
در این گردش، R_{1-1} و L_{1-1} را از گردش پیشین (یا از جعبه‌ی جایگشت نخستین) گرفته R_1 و L_1 را تولید می‌کند. R_1 و L_1 به گردش بعدی (جعبه‌ی جایگشت نهایی) می‌رود. همان‌گونه که در فصل ۵ اشاره کردیم، می‌توانیم فرض کنیم که هر گردش، دو جزء رمزنگاری (ترکیب‌کننده و مبادله‌گر) دارد. هر یک از این عناصر وارون‌پذیرند. مسلماً مبادله‌گر وارون‌پذیر است و نیمه چپ متن را با نیمه راست جابجا می‌کند. ترکیب‌کننده به خاطر استفاده از ویژگی عمل XOR وارون‌پذیر است. تمام عناصر وارون-ناپذیر درون تابع $f(R_{1-1}, K_1)$ گردآوری می‌شوند.

تابع DES

قلب رمزنگاری DES، تابع DES است. تابع DES برای تولید خروجی ۳۲ بیتی، کلید ۴۸ بیتی را بر سمت راست‌ترین ۳۲ بیت (R_{1-1}) اعمال می‌کند. این تابع از چهار قسمت تشکیل شده است: جعبه‌ی P گسترشی، سپیدگر^۱، گروهی از جعبه‌های S و جعبه‌ی P مستقیم (شکل ۵-۶).

جعبه‌ی P گسترشی: از آنجا که R_{1-1} ورودی ۳۲ بیتی و K_1 کلید ۴۸ بیتی است، نخست باید R_{1-1} را به ۴۸ بیت گسترش دهیم. R_{1-1} به هشت بخش ۴ بیتی تقسیم می‌شود، سپس هر بخش ۴ بیتی به ۶ بیت گسترش می‌یابد. این جایگشت گسترشی تابع یک قاعده‌ی از پیش تعیین شده است. برای هر بخش، بیت‌های ورودی ۱، ۲، ۳ و ۴ به ترتیب روی بیت‌های خروجی ۲، ۳، ۴ و ۵ کپی می‌شود. بیت خروجی یک از بیت ۴ بخش قبلی می‌آید و بیت خروجی ۶ از بیت ۱ بخش بعدی می‌آید. اگر بخش‌های ۸ و ۱ را بتوان بخش‌های مجاور در نظر گرفت، همان قانون بر بیت‌های ۱ و ۳۲ اعمال می‌شود. شکل ۶-۶ ورودی و خروجی را در جایگشت گسترشی نشان می‌دهد.

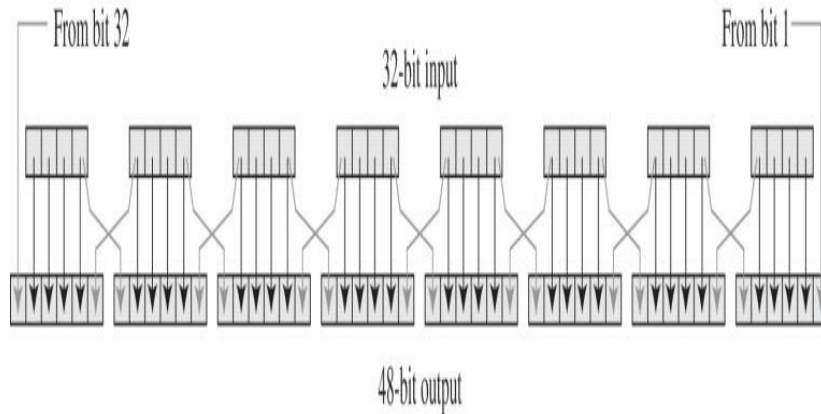
اگرچه رابطه‌ی بین ورودی و خروجی را می‌توان به صورت ریاضی تعریف کرد، اما DES برای تعریف جعبه‌ی P از جدول ۶-۲ استفاده می‌کند. توجه داشته باشید که تعداد درگاه‌های خروجی ۴۸ است اما محدوده‌ی ارزش آن فقط ۱ تا ۳۲ است. برخی از ورودی‌ها به بیش از یک خروجی منتقل می‌شوند. به عنوان مثال، مقدار بیت ۵ ورودی، مقدار خروجی بیت‌های ۶ و ۸ می‌شود.



شکل ۵-۶: تابع DES

=====

¹ Whitener



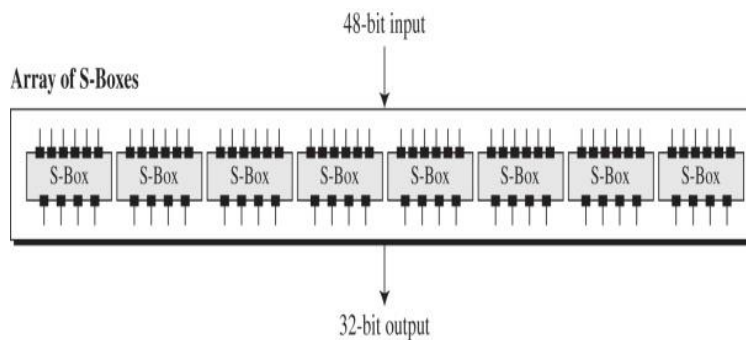
شکل ۶-۶: جایگشت گسترشی

جدول ۶-۲: جدول جعبه‌ی P گسترشی

32	01	02	03	04	05
04	05	06	07	08	09
08	09	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	31	31	32	01

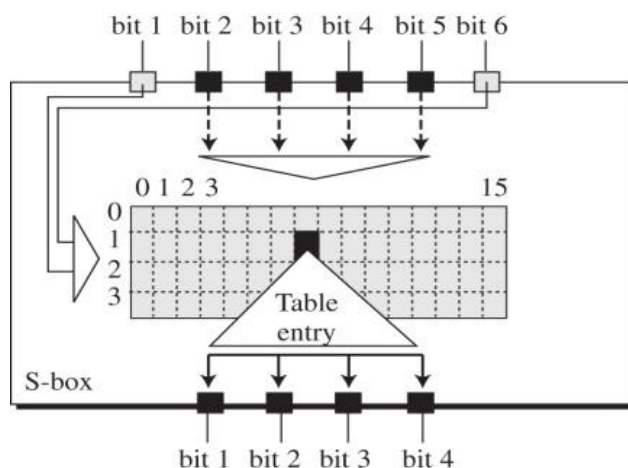
سپیدگر XOR: پس از جایگشت گسترشی، DES عمل XOR را برای بخش سمت راست گسترشی و کلید گردش اعمال می‌کند. توجه داشته باشید که طول قسمت سمت راست و کلید، ۴۸ بیت است. همچنین به خاطر داشته باشید که کلید گردش فقط در این بخش مورد استفاده قرار می‌گیرد.

جعبه‌های S: جعبه‌های S، ترکیب کردن واقعی (آشفته‌گی بیت‌ها) را انجام می‌دهند. DES از ۸ جعبه‌ی S، که هر کدام از یک ورودی ۶ بیتی و یک خروجی ۴ بیتی استفاده می‌کنند. به شکل ۶-۷ نگاه کنید.



شکل ۶-۷: جعبه‌های S

داده‌های ۴۸ بیتی حاصل از عمل دوم به هشت تکه‌ی ۶ بیتی تقسیم می‌شود و هر تکه به یک جعبه وارد می‌شود. نتیجه‌ی هر جعبه، یک تکه‌ی ۴ بیتی است. وقتی این تکه‌ها کنار هم قرار گیرد، یک متن ۳۲ بیتی به دست می‌آید. جایگزینی در هر جعبه، از قانون از پیش تعیین شده بر اساس جدول ۴ سطری و ۱۶ ستونی پیروی می‌کند. ترکیب بیت‌های ۱ و ۶ ورودی یکی از چهار سطر را مشخص می‌کند و همان‌گونه که در شکل ۶-۸ نشان داده شده است، ترکیب بیت‌های ۲ تا ۵ یکی از شانزده ستون را معین می‌کند. با مثال، این مسئله را روشن‌تر خواهیم کرد.



شکل ۶-۸: قانون جعبه‌ی S

از آنجا که هر جعبه‌ی S جدول خاص خود را دارد، برای تعیین خروجی‌های این جعبه‌ها به ۸ جدول نیاز داریم (به جدول‌های ۶-۳ تا ۶-۱۰ نگاه کنید). برای روشن‌تر شدن موضوع، مقادیر ورودی‌ها (تعداد سطر و تعداد ستون) و مقادیر خروجی‌ها به صورت اعداد ده‌دهی ارائه شده است. در عمل، این مقادیر باید به صورت دودویی درآیند.

جدول ۶-۳: جعبه‌ی s1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

جدول ۶-۴: جعبه‌ی s2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
1	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
2	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
3	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

جدول ۶-۵: جعبه‌ی s3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	08
1	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	01
2	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
3	01	10	13	00	06	09	08	07	04	15	14	03	11	05	02	12

جدول ۶-۶: جعبه‌ی s4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	07	13	14	03	00	6	09	10	1	02	08	05	11	12	04	15
1	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
2	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
3	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

جدول ۶-۷: جعبه‌ی s5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
1	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
2	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
3	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

جدول ۶-۸: جعبه‌ی s6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
1	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
2	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
3	04	03	02	12	09	05	15	10	11	14	01	07	10	00	08	13

جدول ۶-۹: جعبه‌ی S7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	00	08	13	03	12	09	07	05	10	06	01
1	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
2	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
3	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

جدول ۶-۱۰: جعبه‌ی S8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	10	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	10	15	03	05	08
3	02	01	14	07	04	10	8	13	15	12	09	09	03	05	06	11

مثال ۶-۳

ورودی به جعبه، یک مقدار ۱۰۰۰۱۱ است خروجی چیست؟

حل

اگر اولین و ششمین بیت را کنار هم بنویسیم، ۱۱ شکل دودویی به دست می‌آوریم که در مبنای ده دهی معادل ۳ می‌شود. سایر بیت‌های باقیمانده ۰۰۰۱ در مبنای دودویی معادل ۱ در مبنای ده هستند در جدول ۶-۳ (جعبه‌ی S1) به دنبال مقدار موجود در سطر ۳، ستون یک می‌گردیم. نتیجه در مبنای ده، ۱۲ و در مبنای دو، ۱۱۰۰ می‌شود؛ بنابراین، ورودی ۱۰۰۰۱۱ می‌شود و خروجی ۱۱۰۰.

مثال ۶-۴

ورودی به جعبه‌ی ۸ مقدار ۰۰۰۰۰۰ است خروجی چیست؟

حل

اگر اولین و ششمین بیت را کنار هم بنویسیم، ۰۰ به شکل دودویی به دست می‌آوریم که در مبنای ده می‌شود صفر. سایر بیت‌های باقیمانده، ۰۰۰۰ در مبنای دو است که در مبنای ده می‌شود صفر. در جدول ۶-۱۰ (جعبه‌ی S8) به دنبال مقدار موجود در سطر و ستون مربوطه می‌گردیم. نتیجه‌ی ۱۳ در مبنای ده است که در مبنای دو آن ۱۱۰۱ می‌شود؛ بنابراین، نتیجه‌ی ورودی ۰۰۰۰۰۰ و خروجی ۱۱۰۱ است.

جایگشت مستقیم: آخرین عمل در تابع DES جایگشت مستقیم با ورودی ۳۲ بیتی و خروجی ۳۲ بیتی است. رابطه‌ی بین ورودی/خروجی برای این عمل در جدول ۶-۱۱ نشان داده شده است و از همان قاعده‌ی کلی جدول جایگشت‌ها پیروی می‌کند. مثلاً هفتمین بیت ورودی، دومین بیت خروجی می‌شود.

جدول ۶-۱۱: جدول جایگشت مستقیم

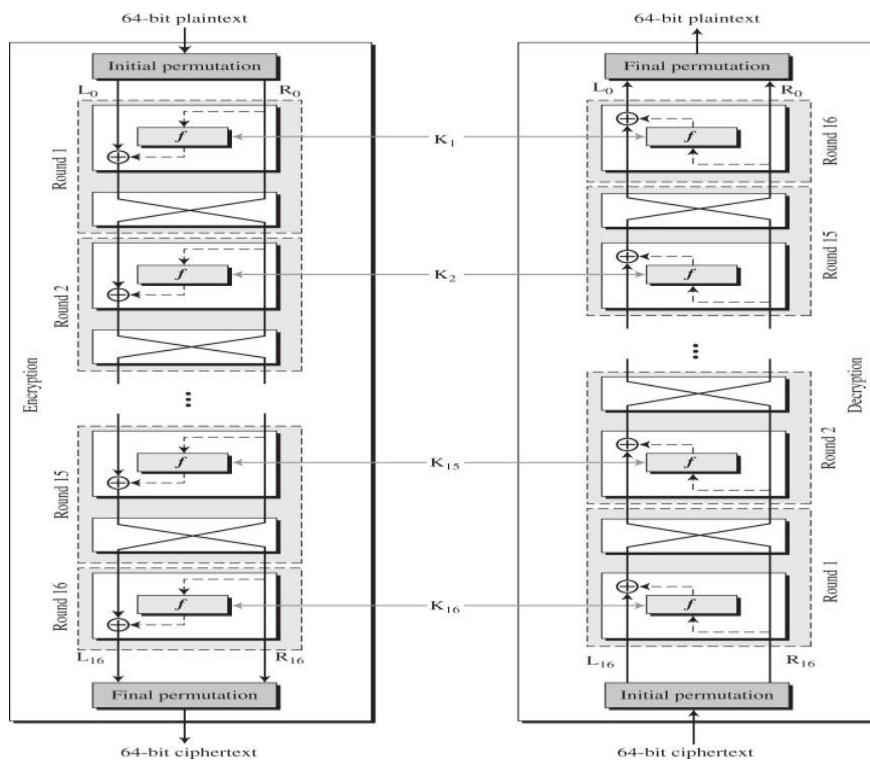
16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

رمز و رمز وارون

با استفاده از ترکیب‌کننده و مبادله‌گر می‌توانیم رمز و وارون رمز را که هر کدام ۱۶ گردش دارند ایجاد کنیم. رمز، در قسمت رمزنگاری و رمز وارون در قسمت رمزگشایی به کار می‌رود. هدف کلی، یکسان-سازی الگوریتم‌های رمز و رمز وارون است.

رویکرد نخست

برای دستیابی به این هدف، یک روش، متفاوت ساختن آخرین گردش از سایر گردش‌هاست. آخرین گردش فقط یک ترکیب‌کننده دارد و مبادله‌گر ندارد. در شکل ۶-۹ این روش را نشان می‌دهد.



شکل ۶-۹: رمز DES و رمز وارون برای رویکرد نخست

اگرچه گردش‌ها همسنگ نیستند، اما اجزاء آن (ترکیب‌کننده و مبادله‌گر) همسنگ‌اند. در فصل ۵ ثابت کردیم که ترکیب‌کننده، خود وارون است در نتیجه مبادله‌گر است. جایگشت‌های نهایی و نخستین نیز وارون یکدیگرند. قسمت سمت چپ متن عادی در مرحله‌ی رمزنگاری L و به عنوان L_{16} رمزنویسی می‌شود و در مرحله‌ی رمزگشایی، L_{16} به صورت L رمزگشایی می‌شود. وضعیت برای R و R_{16} نیز به همین منوال است.

نکته مهمی که باید در مورد رمزها به خاطر داشته باشیم این است که کلیدهای گردش (K_1 تا K_{16}) باید به ترتیب وارون به کار رود. در مرحله‌ی رمزنگاری، گردش اول از K_1 و گردش ۱۶ از K_{16} استفاده می‌کند. در مرحله‌ی رمزگشایی، گردش اول از K_{16} و گردش ۱۶ از K_1 استفاده می‌کند.

در رویکرد نخست، هیچ مبادله‌گر در گردش آخر وجود ندارد.

الگوریتم

نتیجه‌ی الگوریتم ۶-۱ شبه کدی برای رمزنگاری روال متناظر در رویکرد نخست را نشان می‌دهد. شبه کد سایر روال‌ها را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

```
Cipher (plainBlock[64], RoundKeys[16, 48], cipherBlock[64])
{
    permute (64, 64, plainBlock, inBlock, InitialPermutationTable)
    split (64, 32, inBlock, leftBlock, rightBlock)
    for (round = 1 to 16)
    {
        mixer (leftBlock, rightBlock, RoundKeys[round])
        if (round!=16) swapper (leftBlock, rightBlock)
    }
    combine (32, 64, leftBlock, rightBlock, outBlock)
    permute (64, 64, outBlock, cipherBlock, FinalPermutationTable)
}

mixer (leftBlock[48], rightBlock[48], RoundKey[48])
{
    copy (32, rightBlock, T1)
    function (T1, RoundKey, T2)
    exclusiveOr (32, leftBlock, T2, T3)
    copy (32, T3, rightBlock)
}

swapper (leftBlock[32], rightBlock[32])
{
    copy (32, leftBlock, T)
    copy (32, rightBlock, leftBlock)
    copy (32, T, rightBlock)
}
```

```

function (inBlock[32], RoundKey[48], outBlock[32])
{
    permute (32, 48, inBlock, T1, ExpansionPermutationTable)
    exclusiveOr (48, T1, RoundKey, T2)
    substitute (T2, T3, SubstitutionTables)
    permute (32, 32, T3, outBlock, StraightPermutationTable)
}

substitute (inBlock[32], outBlock[48], SubstitutionTables[8, 4, 16])
{
    for (i = 1 to 8)
    {
        row ← 2 × inBlock[i × 6 + 1] + inBlock[i × 6 + 6]
        col ← 8 × inBlock[i × 6 + 2] + 4 × inBlock[i × 6 + 3] +
            2 × inBlock[i × 6 + 4] + inBlock[i × 6 + 5]

        value = SubstitutionTables[i][row][col]

        outBlock[i × 4 + 1] ← value / 8;      value ← value mod 8
        outBlock[i × 4 + 2] ← value / 4;      value ← value mod 4
        outBlock[i × 4 + 3] ← value / 2;      value ← value mod 2
        outBlock[i × 4 + 4] ← value
    }
}

```

الگوریتم ۶-۱: شبه کد برای رمز

رویکرد جایگزین

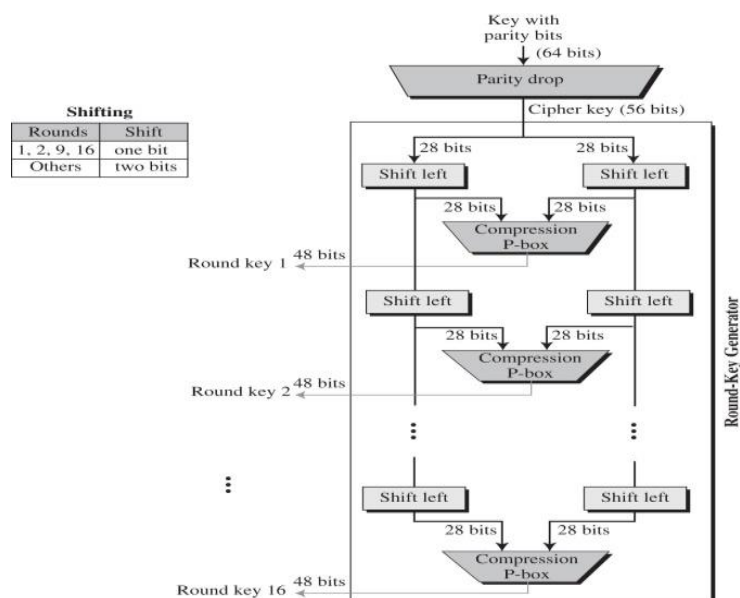
در رویکرد نخست، گردش شانزدهم با سایر گردش‌ها تفاوت داشت. در این گردش مبادله‌گر وجود ندارد. برای اینکه واپسین ترکیب‌کننده در مرحله‌ی رمز، نخستین ترکیب‌کننده در رمز وارون همسنگ باشد، باید مبادله‌گر حذف می‌شود. با گنجاندن یک مبادله‌گر در گردش شانزدهم و افزودن یک مبادله‌گر دیگر پس از آن (دو مبادله‌گر اثر یکدیگر را خنثی می‌کنند) می‌توان تمام ۱۶ گردش را همسان نمود. طراحی این روش را به عنوان تمرین به عهده شما می‌گذاریم.

مولد کلید

مولد کلید در گردش شانزدهم، کلید ۴۸ بیتی با استفاده از کلید رمز ۵۶ بیتی تولید می‌کند. با این وجود، معمولاً کلید رمز به صورت کلید ۶۴ بیتی در نظر گرفته می‌شود و در این کلید، ۸ بیت اضافی، بیت توازن هستند و همان‌گونه که در شکل ۶-۱۰ نشان داده شده است، پیش از فرایند رمزنگاری واقعی، می‌بایست کلید تولید شود.

ریزش بیت توازن^۱

فرآیند پیش از گسترش کلید، یک جایگشت تراکمی انجام می‌دهد که آن را «ریزش بیت توازن» می‌نامیم. این عمل بیت‌های توازن (بیت‌های ۸، ۱۶، ۲۴، ۳۲، ...، ۶۴) را از کلید ۶۴ بیتی حذف و بقیه بیت‌ها را بر اساس جدول ۶-۱۲ جابجا می‌کند. مقدار ۵۶ بیتی باقیمانده، کلید رمز واقعی است که برای تولید کلیدها در هر گردش به کار می‌رود. در جدول ۶-۱۲ جایگشت حاصل از ریزش بیت توازن برای جعبه P متراکم نشان داده شده است.



شکل ۶-۱۰: تولید کلید

جدول ۶-۱۲: جدول نادیده گرفتن بیت توازن

57	49	41	33	25	17	09	01
58	50	42	34	26	18	10	02
59	51	43	35	27	19	11	03
60	52	44	36	63	55	47	39
31	23	15	07	62	54	46	38
30	22	14	06	61	53	45	37
29	21	13	05	28	20	12	04

چرخش به چپ^۲

پس از جایگشت مستقیم، کلید به دو قسمت ۲۸ بیتی تقسیم می‌شود. هر قسمت، یک یا دو بیت به سمت چپ می‌چرخد (چرخش دایره‌ای). در گردش ۱، ۲، ۹ و ۱۲ چرخش یک بیت و در سایر گردش‌ها

^۱ Parity Bit Drop

^۲ Shift Left

دو بیت است؛ سپس هر دو قسمت به منظور تشکیل یک قسمت ۵۶ بیتی با هم ترکیب می‌شوند. جدول ۶-۱۳ تعداد چرخش برای هر گردش را نشان می‌دهد.

جدول ۶-۱۳: تعداد چرخش‌های در مقیاس بیت

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit shifts	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

جایگشت متراکم^۱

جایگشت متراکم (جعبه P) ۵۸ بیت را به ۴۸ بیت تبدیل می‌کند. این ۴۸ بیت به عنوان کلید برای یک گردش مورد استفاده قرار می‌گیرد. در جدول ۶-۱۴، جایگشت متراکم را مشاهده می‌نمایید

جدول ۶-۱۴: جدول کلید متراکم.

14	17	11	24	01	05	03	28
15	06	21	10	23	19	12	04
26	08	16	07	27	20	13	02
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

الگوریتم

اجازه دهید برای ایجاد کلیدهای هر گردش به همراه بیت‌های توازن، الگوریتمی ارائه دهیم. الگوریتم ۶-۲ از چندین روال موجود در الگوریتم ۶-۱ استفاده می‌کند. روال جدید، روال چرخش به چپ است که بر روی کد داده شده اعمال می‌گردد. توجه داشته باشید که T بلوک موقتی است.

```

Key_Generator (keyWithParities[64], RoundKeys[16, 48], ShiftTable[16])
{
    permute (64, 56, keyWithParities, cipherKey, ParityDropTable)
    split (56, 28, cipherKey, leftKey, rightKey)
    for (round = 1 to 16)
    {
        shiftLeft (leftKey, ShiftTable[round])
        shiftLeft (rightKey, ShiftTable[round])
        combine (28, 56, leftKey, rightKey, preRoundKey)
        permute (56, 48, preRoundKey, RoundKeys[round], KeyCompressionTable)
    }
}

```

=====

¹ Compression Permutation

```

shiftLeft (block[28], numOfShifts)
{
    for (i = 1 to numOfShifts)
    {
        T ← block[1]
        for (j = 2 to 28)
        {
            block [j-1] ← block [j]
        }
        block[28] ← T
    }
}

```

۶-۲: الگوریتم برای ایجاد کلیدها در هر گردش

مثال‌ها

پیش از تجزیه و تحلیل DES اجازه دهید برای مشاهده‌ی چگونگی رمزنگاری و رمزگشایی، تعداد بیت‌ها در هر گردش را تغییر دهیم سپس به بررسی چند مثال در این خصوص بپردازیم.

مثال ۶-۵

یک بلوک متن عادی تصادفی و یک کلید تصادفی انتخاب می‌کنیم و تعیین می‌کنیم که بلوک متن رمز چه خواهد بود (اعداد هم در مبنای شانزده هستند).

متن عادی: ۱۲۳۴۵۶ABCD۱۲۳۴۵۶

متن رمز: C۰B۷A۸D۰۵F۳A۸۲۹C

کلید رمز: AABBB۰۹۱۸۲۷۳۶CCDD

اجازه دهید که نتیجه‌ی هر گردش و متن ایجاد شده پیش و پس از هر گردش را نشان دهیم.

جدول ۶-۱۵ نخست نتیجه‌ی مراحل پیش از آغاز گردش نخستین را نشان می‌دهد.

جدول ۶-۱۵: پیگیری مراحل تغییر داده‌ها در مثال ۶-۵

Plaintext: 123456ABCD132536			
After initial permutation: 14A7D67818CA18AD			
After splitting: L ₀ =14A7D678 R ₀ =18CA18AD			
Round	Left	Right	Round Key
Round 1	18CA18AD	5A78E394	194CD072DE8C
Round 2	5A78E394	4A1210F6	4568581ABCCE
Round 3	4A1210F6	B8089591	06EDA4ACF5B5
Round 4	B8089591	236779C2	DA2D032B6EE3

Round 5	236779C2	A15A4B87	69A629FEC913
Round 6	A15A4B87	2E8F9C65	C1948E87475E
Round 7	2E8F9C65	A9FC20A3	708AD2DDB3C0
Round 8	A9FC20A3	308BEE97	34F822F0C66D
Round 9	308BEE97	10AF9D37	84BB4473DCCC
Round 10	10AF9D37	6CA6CB20	02765708B5BF
Round 11	6CA6CB20	FF3C485F	6D5560AF7CA5
Round 12	FF3C485F	22A5963B	C2C1E96A4BF3
Round 13	22A5963B	387CCDAA	99C31397C91F
Round 14	387CCDAA	BD2DD2AB	251B8BC717D0
Round 15	BD2DD2AB	CF26B472	3330C5D9A36D
Round 16	19BA9212	CF26B472	181C5D75C66D
After combination: 19BA9212CF26B472			
Ciphertext: C0B7A8D05F3A829C		(after final permutation)	

برای ایجاد ۶۴ بیت کاملاً متفاوت (۱۶ رقم مبنای شانزده)، متن عادی از جایگشت نخستین عبور می‌کند. پس از این مرحله، متن به دو نیم تقسیم می‌شود که آن را L و R می‌نامیم. این جدول، نتیجه‌ی ۱۶ گردش که شامل ترکیب و مبادله‌گر (به جز در گردش آخر) است را نشان می‌دهد. نتایج گردش‌های آخر (R_{16}, L_{16}) با هم ترکیب می‌شود. سرانجام جهت ایجاد متن رمز، حاصل وارد جایگشت پایانی می‌شود.

در اینجا بیان چند نکته شایان توجه است. نخست: قسمت راست از هر گردش با قسمت چپ گردش بعدی یکسان است. علت آن است که قسمت راست بدون تغییر از ترکیب کننده عبور می‌کند اما مبادله‌گر آن را به قسمت چپ منتقل می‌کند. به عنوان مثال R_1 از بین ترکیب کننده‌ی گردش دوم بدون تغییر می‌گذرد، اما سپس به خاطر وجود مبادله‌گر، به L_2 تبدیل می‌شود. نکته جالب این است که در گردش آخر مبادله‌گر نداریم. به همین علت است که R_{16} بجای اینکه به L_{16} تبدیل شود به R_{16} تبدیل می‌شود.

مثال ۶-۶

اگر موافق باشید بررسی کنیم که چگونه باب در مقصد می‌تواند با استفاده از همان کلید، متن رمز دریافتی از آلیس را رمزگشایی کند. جهت اختصار، به نشان دادن تنها چند گردش بسنده می‌کنیم. جدول ۶-۱۶ نکات جالبی را نشان می‌دهد. نخست، کلید گردش باید به ترتیب وارون مورد استفاده قرار گیرد. جدول ۶-۱۵ و جدول ۶-۱۶ را با هم مقایسه کنید. کلید گردش ۱ با کلید گردش ۱۶ یکی است. مقادیر L و R در خلال رمزگشایی با مقادیر L_{16} و R_{16} در خلال رمزنگاری برابر است. این مسئله در مورد سایر گردش‌ها نیز درست است. این نشان می‌دهد که نه تنها رمز و رمز وارون، وارون یکدیگرند، بلکه هر گردش در رمز، یک گردش وارون متناظر در رمز وارون دارد؛ در نتیجه ثابت می‌شود که مراحل جایگشت نخستین و پایانی نیز وارون یکدیگرند.

<i>Ciphertext:</i> C0B7A8D05F3A829C			
<i>After initial permutation:</i> 19BA9212CF26B472			
<i>After splitting:</i> L ₀ =19BA9212 R ₀ =CF26B472			
<i>Round</i>	<i>Left</i>	<i>Right</i>	<i>Round Key</i>
<i>Round 1</i>	CF26B472	BD2DD2AB	181C5D75C66D
<i>Round 2</i>	BD2DD2AB	387CCDAA	3330C5D9A36D
...
<i>Round 15</i>	5A78E394	18CA18AD	4568581ABCCE
<i>Round 16</i>	1A7D678	18CA18AD	194CD072DE8C
<i>After combination:</i> 14A7D67818CA18AD			
<i>Plaintext:</i> 123456ABCD132536		<i>(after final permutation)</i>	

منتقدان با دقت بسیار زیادی DES را تجزیه و تحلیل نموده‌اند. جهت اندازه‌گیری توانایی برخی از ویژگی‌های مطلوب در رمز بلوکی، آزمایش‌های زیادی انجام شده است. برای مشاهده تطابق DES با معیارهای موردنظر، اجزای آن را مورد بررسی موشکافانه قرار دادیم.

ویژگی‌های مطلوب یک رمز بلوکی، تأثیر بهمن‌گونه و تمامیت است.

تأثیر بهمن‌گونه به این معنی است که تغییر کوچک در متن عادی (یا کلید) باید تغییرهای چشمگیری در متن رمز ایجاد کند. DES ثابت کرده است که در این ویژگی بسیار توانا است.

برای بررسی کنترل تأثیر بهمن‌گونه در DES اجازه دهید دو بلوک متن عادی (با یک کلید یگانه) را که فقط در یک بیت تفاوت دارند را رمزنگاری کنیم و تفاوت بین تعداد بیت‌ها در هر گردش را مشاهده نماییم.

رمز: ٤٧٨٩FD٤٧٦E٨٢A٥F١

¹ Avalanche Effect

۱.....:متن عادی

اگرچه هر دو بلوک متن عادی فقط در بیت سمت راست تفاوت دارند، اما بلوک‌های متن رمز در ۲۹ بیت با هم متفاوت‌اند. این بدان معنی است که تغییر تقریباً ۱,۵ درصدی در متن عادی، تغییر تقریباً ۴۵ درصدی در متن رمز ایجاد می‌کند. جدول ۶-۱۷ تغییر در هر گردش را نشان می‌دهد. این جدول نشان می‌دهد که تغییرات چشمگیر در همان ابتدا در گردش سوم اتفاق می‌افتد.

<i>Rounds</i>	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bit differences	1	6	20	29	30	33	32	29	32	39	33	28	30	31	30	29

=====

¹ Completeness Effect

- (۳) اگر تنها یک بیت در ورودی را تغییر دهیم دو یا بیش از دو بیت در خروجی تغییر خواهد کرد.
- (۴) اگر در ورودی به یک جعبه‌ی S فقط از نظر دو بیت میانی (بیت‌های ۳ و ۴) با هم متفاوت باشند، خروجی باید دست‌کم در دو بیت متفاوت باشد؛ به عبارت دیگر $S(x)$ و $S(x \oplus 001100)$ باید دست‌کم در دو بیت متفاوت باشند، طوری که x ورودی و $S(x)$ خروجی است.
- (۵) اگر دو ورودی به یک جعبه S ، دو بیت اول آن‌ها (بیت‌های ۱ و ۲) با هم متفاوت و دو بیت پایانی آن‌ها (بیت‌های ۵ و ۶) یکسان باشد، دو خروجی باید متفاوت باشد؛ به عبارت دیگر باید رابطه‌ی $S(x) \neq S(x \oplus 11bc00)$ برقرار باشد که در این رابطه b و c بیت‌های دودویی هستند.
- (۶) فقط ۳۲ جفت کلمه‌ی ورودی ۶ بیتی وجود دارد، طوری که $x_i \oplus x_j \neq (000000)_2$ است. این ۳۲ جفت ورودی، ۳۲ جفت کلمه‌ی خروجی ۴ بیتی تولید می‌کند. اگر بین ۳۲ جفت خروجی تفاوت ایجاد کنیم، $d = y_i \oplus y_j$ بیش از ۸ عدد از آن‌ها نباید یکسان باشد.
- (۷) برای سه جعبه‌ی S ، معیاری شبیه به شماره‌ی ۶ اعمال می‌شود.
- (۸) در هر جعبه‌ی S ، اگر تنها یک بیت ورودی را ثابت نگه داریم (۰ یا ۱) و سایر بیت‌ها را به صورت تصادفی تغییر دهیم، تفاوت بین تعداد ۰ها و ۱ها به حداقل می‌رسد.

جعبه‌های P

بین دو ردیف از جعبه‌های S (در دو گردش پیاپی)، یک جعبه‌ی P مستقیم (۳۲ تا ۳۲) و یک جعبه‌ی P گسترشی (۳۲ تا ۴۸) وجود دارد. این دو جعبه‌ی P با هم وظیفه‌ی پخش بیت‌ها را به عهده دارند. در فصل ۵، اصول کلی طراحی جعبه‌های P را مورد بحث قرار دادیم. در اینجا تنها به اصول اعمال شده بر جعبه‌های P استفاده شده در تابع DES می‌پردازیم. جهت نیل به این هدف، معیارهای زیر در طراحی جعبه‌های P به کار رفته است:

- (۱) هر ورودی جعبه‌ی S از خروجی یک جعبه‌ی S متفاوت (از گردش پیشین) می‌آید.
- (۲) هیچ ورودی به جعبه‌ی S معین از خروجی همان جعبه (از گردش پیشین) نمی‌آید.
- (۳) چهار خروجی از هر جعبه‌ی S به شش جعبه‌ی S متفاوت (در گردش بعدی) منتقل می‌شود.
- (۴) هیچ‌وقت دو بیت خروجی از یک جعبه‌ی S به همان جعبه‌ی S (در گردش پیشین) منتقل نمی‌شود.
- (۵) اگر هشت جعبه‌ی S را به صورت S_1, S_2, \dots, S_8 شماره‌گذاری می‌کنیم،
 - (a) یک خروجی S_{j-2} به یکی از اولین دو بیت S_j (در گردش بعد) منتقل می‌شود.

- (b) یک بیت خروجی از S_{j-1} به یکی از آخرین دو بیت S_j (در گردش بعد) منتقل می‌شود.
- (c) یک خروجی S_{j+1} به یکی از دو بیت میانی S_j (در گردش بعد) منتقل می‌شود.
- (۶) برای هر جعبه‌ی S ، دو بیت خروجی به دو بیت اول یا آخر یک جعبه‌ی S در گردش بعد منتقل می‌شود.
- (۷) اگر یک بیت خروجی از S_k به یکی از بیت‌های میانی در S_k منتقل شود (در گردش بعد) سپس یک بیت خروجی از S_k نمی‌تواند به بیت میانی S_j منتقل شود. اگر $j = k$ باشد بدین معنی است که هیچ یک از بیت‌های میانی یک جعبه‌ی S نمی‌تواند در گردش بعدی به یکی از بیت‌های میانی همان جعبه‌ی S منتقل شوند.

تعداد گردش‌ها

DES از شانزده گردش به‌سان رمزهای Feistel استفاده می‌کند. ثابت شده است که پس از هشت گردش تغییرات هر کلمه در متن رمز، تابعی از تمام بیت‌های متن عادی و تمام بیت‌های کلید است؛ متن رمز شده تابعی کاملاً تصادفی از متن عادی و متن رمزی است. بنابراین به نظر می‌رسد که هشت گردش کافی است. با این وجود، تجربه ثابت کرده است که استفاده از DES با تعداد گردش‌های کمتر از شانزده در برابر حملاتی به شیوه‌ی متن عادی شناخته شده بسیار آسیب‌پذیرتر از حمله از نوع آزمودن کلیه کلیدهاست. بنابراین طراحان DES استفاده از شانزده گردش را توصیه می‌کند.

نقاط ضعف DES

در طول سال‌های اخیر، منتقدان نقاط ضعفی در DES یافته‌اند.

ضعف در طراحی رمز

برخی از نقاط ضعفی که در طراحی رمز پیدا شده است را به اختصار بیان می‌کنیم.

جعبه‌های S: در تحقیقات دست‌کم به سه نقطه ضعف در زمینه‌ی جعبه‌های S اشاره شده است:

- (۱) در جعبه‌ی S_4 ، سه بیت آخر خروجی را می‌توان به همان روش بیت خروجی اول و به وسیله‌ی مکمل‌گیری برخی از بیت‌های ورودی به دست آورد.
- (۲) دو ورودی انتخاب شده ویژه برای آرایه‌ی جعبه‌ی S می‌توانند خروجی یکسان تولید کنند.
- (۳) این احتمال وجود دارد که تنها با تغییر بیت‌ها در سه جعبه‌ی S مجاور، یک خروجی یکتا در یک گردش به دست آورد.

جعبه‌های P: یک نکته ابهام و یک نقطه ضعف در طراحی جعبه‌های P یافت شد.

(۱) معلوم نیست که چرا طراحان DES از جایگشت‌های نخستین و پایانی استفاده کردند. این جایگشت‌ها هیچ کارکرد امنیتی ندارند.

(۲) در جایگشت گسترشی (در داخل تابع)، بیت‌های اول و چهارم هر سری چهار بیتی تکراری هستند.

نقاط ضعف کلید رمز

چندین نقطه ضعف به شرح زیر در کلید رمز پیدا شده است:

اندازه‌ی کلید: منتقدان بر این باورند که جدی‌ترین نقطه ضعف DES اندازه‌ی کلید آن (۵۶ بیت) است. برای انجام حمله از نوع آزمود، کلیه‌ی کلیدها بر روی یک بلوک متن رمز، رمزشکن باید فقط 2^{56} کلید را چک کند.

(a) با فن‌آوری موجود، آزمایش یک میلیون کلید در هر ثانیه امکان‌پذیر است؛ یعنی بیش از دو هزار سال لازم است تا با استفاده از یک کامپیوتر با یک پردازنده‌ی حمله از نوع آزمودن کلیه‌ی کلیدها را انجام دهیم.

(b) اگر بتوانیم کامپیوتری با یک میلیون پردازنده (با امکان پردازش موازی) بسازیم، می‌توانیم در مدت تقریباً ۲۰ ساعت اعضای کل دامنه‌ی کلیدها را آزمایش کنیم. وقتی DES ارائه شد، قیمت چنین کامپیوتری بیش از چند هزار دلار بود، اما قیمت‌ها به سرعت کاهش یافت. در سال ۱۹۹۸ میلادی کامپیوتری ساخته شد که در عرض ۱۱۲ ساعت کلید را پیدا می‌کرد.

(c) شبکه‌های کامپیوتری می‌توانند پردازش موازی را شبیه‌سازی کنند. در سال ۱۹۷۷ میلادی یک تیم تحقیقاتی توانست با استفاده از ۳۵۰۰ کامپیوتر متصل به اینترنت، کلید خواسته شده لائورائو RSA را در ۱۲۰ روز بیابد. دامنه‌ی کلید بین تمام این کامپیوترها تقسیم شده و هر کامپیوتر مسئول آزمودن کلید قسمت خود بودند.

(d) اگر ۳۵۰۰ کامپیوتر متصل به شبکه بتوانند کلید را ظرف ۱۲۰ روز پیدا کنند، یک انجمن سری با ۴۲۰۰۰ کامپیوتر می‌تواند کلید را ظرف ۱۰ روز پیدا کند.

(e) موارد بالا نشان می‌دهد که DES با کلید رمز ۵۶ بیتی چندان مطمئن نیست. در قسمت‌های بعدی این فصل خواهیم دید که یک راه‌حل استفاده از DES سه‌گانه (۳DES) با دو کلید (۱۱۲ بیت) یا DES سه‌گانه با سه کلید (۱۶۸ بیت) است.

کلیدهای ضعیف: چهار کلید از کل 2^{56} کلید را «کلیدهای ضعیف» می‌نامند. کلید ضعیف، کلیدی است که پس از حذف بیت‌های توازن (با استفاده از جدول ۶-۱۲)، تمام بیت‌ها ۰ و یا ۱ها یا نیمی از بیت‌ها ۰ و نیم دیگر از ۱ تشکیل شده باشد. در جدول ۶-۱۸ این کلیدها نشان داده شده است.

جدول ۶-۱۸: کلیدهای ضعیف

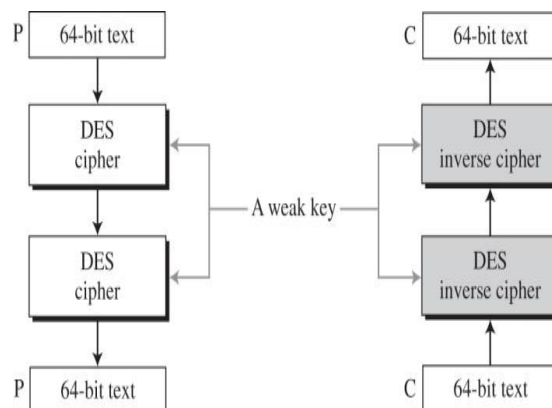
Keys before parities drop (64 bits)	Actual key (56 bits)
0101 0101 0101 0101	0000000 0000000
1F1F 1F1F 0E0E 0E0E	0000000 FFFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFFF

کلیدهای مورد استفاده در گردش‌ها از هر یک از این چهار کلید ضعیف، یکسان بوده و همان الگوی کلید رمز را دارند. به عنوان مثال، شانزده کلید گردش حاصل از کلید اول، همه صفر و کلید گردش‌های حاصل از کلید دوم، نصف صفر و نیم دیگر یک است.

علت آن است که الگوریتم مولد کلید ابتدا رمز را به دو نیمه تقسیم می‌کند. اگر تمام آن‌ها از صفر یا تمام آن‌ها از یک تشکیل شده باشند، تغییر یا جایگشت بلوک‌ها، تغییری در بلوک ایجاد نمی‌کند. عیب استفاده از کلید ضعیف چیست؟ اگر بلوکی را با یک کلید ضعیف رمزنگاری کنیم و متعاقباً نتیجه را با همان کلید ضعیف رمزنگاری کنیم، بلوک اصلی را به دست می‌آوریم. در این فرآیند، اگر بلوک را دو بار رمزگشایی کنیم همان بلوک اصلی را ایجاد می‌کنیم.

به عبارت دیگر، همان‌گونه که در شکل ۶-۱۱ نشان داده شده است هر کلید ضعیف وارون خود

$E_k(E_k(P))$ می‌باشد.



شکل ۶-۱۱: رمزنگاری و رمزگشایی دوگانه با یک کلید ضعیف

از آنجا که رقیب می‌تواند به سادگی کلیدهای ضعیف را روی متن رمز استراق سمع شده آزمایش کند، باید از انتخاب آن‌ها بپرهیزیم. اگر پس از دو بار رمزگشایی نتیجه یکسان باشد، رقیب کلید را یافته است.

مثال ۶-۸

اجازه دهید اولین کلید ضعیف در جدول ۶-۱۸ را برای رمزنگاری یک بلوک دو بار آزمایش کنیم. پس از دو بار رمزنگاری با همان کلید، بلوک متن عادی ایجاد می‌شود. توجه داشته باشید که الگوریتم رمزنگاری را دو بار استفاده کردیم نه اینکه نخست از الگوریتم رمزنگاری و سپس از الگوریتم رمزگشایی استفاده کنیم.

کلید: ۰X۰۱۰۱۰۱۰۱۰۱۰۱۰۱۰۱

متن رمز: ۰X۸۱۴FE۹۳۸۵۸۹۱۵۴F۷ متن عادی: ۰X۱۲۳۴۵۶۷۸۸۷۶۵۴۳۲۱

کلید: ۰X۰۱۰۱۰۱۰۱۰۱۰۱۰۱۰۱

متن رمز: ۰X۱۲۳۴۵۶۷۸۸۷۶۵۴۳۲۱ متن عادی: ۰X۸۱۴FE۹۳۸۵۸۹۱۵۴F۷

کلیدهای شبه ضعیف^۱: شش جفت کلید وجود دارد که به آن‌ها کلیدهای شبه ضعیف می‌گویند. این شش جفت را در جدول ۶-۱۹ نشان داده‌ایم (کلیدها در فرمت ۶۴ بیتی پیش از حذف بیت‌های توازن هستند).

یک کلید شبه ضعیف، تنها دو کلید متفاوت تولید می‌کند و هر یک از آن‌ها هشت بار تکرار می‌شود. علاوه بر این، کلیدهای ایجاد شده از هر جفت، یکسان ولی با ترتیب‌های گوناگون هستند.

جدول ۶-۱۹ کلیدهای نیمه ضعیف

First key in the pair	Second key in the pair
01FE 01FE 01FE 01FE	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	E01F E01F F10E F10E
01E0 01E1 01F1 01F1	E001 E001 F101 F101
1FFE 1FFE 0EFE 0EFE	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	FEE0 FEE0 FEF1 FEF1

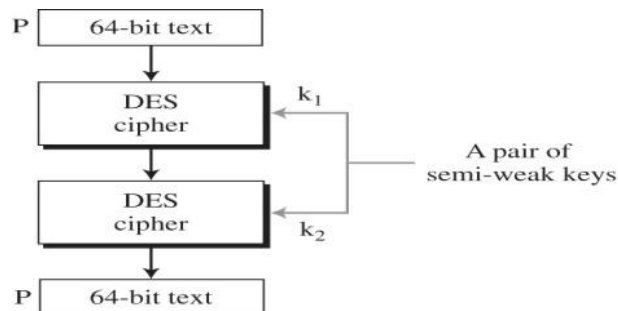
برای نشان دادن این موضوع، کلیدهای گردش را از اولین جفت به شکل زیر ایجاد کردیم.

=====

^۱ Semi-weak Keys

Round key 1	9153E54319BD	6EAC1ABCE642
Round key 2	6EAC1ABCE642	9153E54319BD
Round key 3	6EAC1ABCE642	9153E54319BD
Round key 4	6EAC1ABCE642	9153E54319BD
Round key 5	6EAC1ABCE642	9153E54319BD
Round key 6	6EAC1ABCE642	9153E54319BD
Round key 7	6EAC1ABCE642	9153E54319BD
Round key 8	6EAC1ABCE642	9153E54319BD
Round key 9	9153E54319BD	6EAC1ABCE642
Round key 10	9153E54319BD	6EAC1ABCE642
Round key 11	9153E54319BD	6EAC1ABCE642
Round key 12	9153E54319BD	6EAC1ABCE642
Round key 13	9153E54319BD	6EAC1ABCE642
Round key 14	9153E54319BD	6EAC1ABCE642
Round key 15	9153E54319BD	6EAC1ABCE642
Round key 16	6EAC1ABCE642	9153E54319BD

همان‌گونه که لیست بالا نشان می‌دهد، هشت کلید مساوی در هر کلید شبه ضعیف وجود دارد. به علاوه، کلید در گردش ۱ در مجموعه‌ی اول با کلید گردش ۱۶ در مجموعه‌ی دوم، کلید گردش ۲ در مجموعه‌ی اول با کلید گردش ۱۵ در مجموعه‌ی دوم یکسان است و الی آخر. این بدان معنی است که کلیدها وارون یکدیگرند $E_{k_2}(E_{k_1}(P))=P$ (شکل ۶-۱۲).



شکل ۶-۱۲: یک جفت کلید شبه ضعیف در رمزنگاری و رمزگشایی

کلیدهای ضعیف احتمالی^۱: ۴۸ کلید نیز وجود دارد که به آن‌ها کلیدهای ضعیف احتمالی می‌گویند. کلید ضعیف احتمالی کلیدی است که فقط چهار کلید مشخص تولید می‌کند؛ به عبارت دیگر، شانزده کلید به چهار گروه تقسیم می‌شود و هر گروه از چهار کلید مساوی تشکیل شده‌اند.

مثال ۶-۹

احتمال انتخاب تصادفی یک کلید ضعیف، شبه ضعیف، یا یک کلید ضعیف احتمالی چیست؟

حل

DES دارای دامنه‌ی کلید 2^{56} است. تعداد کلی کلیدها $(48+12+48) \times 64$ است و احتمال انتخاب یکی از این کلیدها، $10^{-16} \times 8,8$ است که تقریباً غیرممکن است.

=====

^۱ Possible Weak Keys

متمم کلید: در دامنه‌ی کلید 2^{56} ، مسلماً نیمی از کلیدها متمم نیمه دیگر هستند. «متمم کلید» را می‌توان با وارون کردن هر بیت در کلید (تغییر ۰ به ۱ یا ۱ به ۰) به دست آورد. آیا متمم کلید، کار رمزگشایی را ساده می‌کند؟ ممکن است این گونه باشد. رمزشکن می‌تواند تنها نیمی از کلیدهای احتمالی (2^{56}) برای انجام حمله‌ی آزمودن کلیدی کلیدها استفاده کند. این امر به این علت است که:

$$C = E(k, p) \rightarrow \bar{C} = E(\bar{k}, \bar{p})$$

به عبارت دیگر، اگر متمم متن عادی را با متمم کلید رمزنگاری کنیم، متمم متن رمز را به دست می‌آوریم. رمزشکن مجبور نیست تمام 2^{56} کلید احتمالی را آزمایش کند، او می‌تواند فقط نیمی از آن‌ها را آزموده و سپس نتیجه را مکمل‌گیری نماید.

مثال ۶-۱۰

اجازه دهید ادعا در مورد کلیدهای متمم را آزمایش کنیم. از کلید دلخواه و متن عادی برای یافتن متن رمز متناظر استفاده کردیم. اگر متمم کلید و متن عادی را داشته باشیم، می‌توانیم متمم متن رمز قبل (جدول ۶-۲۰) را به دست آوریم.

جدول ۶-۲۰: نتایج برای مثال ۶-۱۰

	Original	Complement
Key	1234123412341234	EDCBEDCBEDCBEDCB
Plaintext	12345678ABCDEF12	EDCBA987543210ED
Ciphertext	E112BE1DEFC7A367	1EED41E210385C98

خوشه‌بندی کلیدها: خوشه‌بندی کلید، نشان دهنده‌ی موقعیتی است که در آن دو یا بیش از دو کلید متفاوت می‌توانند متن رمز یکسانی از متن عادی یکسان تولید کنند. بدیهی است که هر جفت از کلیدهای شبه ضعیف، خوشه‌ای از کلیدها را تشکیل می‌دهد؛ با این وجود، خوشه‌های بیشتری برای DES پیدا نشده است. تحقیقات بعدی ممکن است خوشه‌های بیشتری را آشکار نماید.

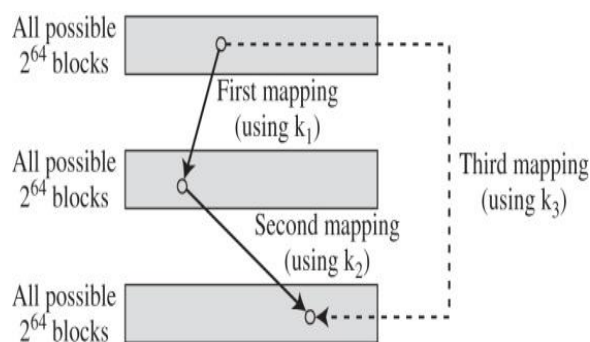
=====

¹ Key Clustering

۶-۴-DES چندگانه

همان‌گونه که دیدیم، انتقاد اصلی از DES مربوط به طول کلیدش است. با فناوری موجود و امکان انجام پردازش موازی، حمله از نوع آزمودن کلیدی کلیدها به DES امکان‌پذیر است. یک راه‌حل برای ارتقای امنیت DES، نادیده گرفتن آن و طراحی روش رمزنگاری نو است. در فصل ۷ و با پیدایش رمزنگاری AES، شاهد این راه حل خواهیم بود. راه‌حل دوم، استفاده از نمونه‌های چندگانه (آبشاری)^۱ از DES با کلیدهای چندگانه است. این راه‌حل که برای مدت کوتاهی مورد استفاده قرار گرفت، مستلزم سرمایه‌گذاری در سخت‌افزار و نرم‌افزار جدید نیست. در ادامه راه‌حل دوم را مورد مطالعه قرار می‌دهیم.

همان‌گونه که در فصل ۵ آموختیم هر جایگزینی که هر ورودی را به یک خروجی نگاشت می‌کند گروه نامیده می‌شود و این نگاشت‌هایی مانند اعضای مجموعه و ترکیب مانند عملگرش است. در این حالت، استفاده از دو نگاشت متوالی بی‌فایده است؛ زیرا همیشه می‌توانیم نگاشت سومی پیدا کنیم که با ترکیب دو نگاشت پی‌درپی یکسان باشد (ویژگی بسته بودن)؛ یعنی اگر DES گروه باشد، استفاده از DES دوگانه با دو کلید K_1 و K_2 بی‌فایده است زیرا یک DES تکی با کلید K_3 همان کار را انجام می‌دهد (شکل ۶-۱۳).



شکل ۶-۱۳: ترکیبی از نگاشت‌ها

خوشبختانه، DES بنا بر دو استدلال زیرگروه محسوب نمی‌شود:

(a) تعداد ورودی‌ها و خروجی‌های احتمالی در DES برابر $N=2^{64}$ است. این بدان معنی است که $N!=(2^{64})!=1.034 \times 10^{38}$ نگاشت گوناگون داریم. برای اینکه DES یک گروه باشد باید کاری کرد که با اندازه‌ی کلید $2^{70} \approx \log_2(N!) (2^{64})$ بیتی تمام این نگاشت‌ها را پشتیبانی کند.

=====

^۱ Multiple (cascaded)

اما می‌دانیم که طول کلید در DES فقط ۵۶ بیت است (فقط قسمت کوچکی از اندازه‌ی کلید هیولایی).

(b) راه دیگر فرض گروه بودن DES است که برای مجموعه‌ی نگاشت‌ها، زیرمجموعه‌ای از مجموعه‌ی استدلال اول وجود داشته باشد؛ اما ثابت شده است که هیچ یک از زیرگروه‌های ایجاد شده از گروه موجود در استدلال اول، اندازه‌ی کلید ۵۶ بیتی ندارند. اگر DES گروه نباشد، یافتن کلیدی مانند K_3 ، به شکل زیر، بسیار غیر محتمل است:

$$E_{K_2}(E_{K_1}(P)) = E_{K_3}(P)$$

به عبارت دیگر، می‌توانیم از DES دوگانه یا سه‌گانه برای افزایش اندازه‌ی کلید استفاده کنیم.

DES با کلید دوگانه

شیوه‌ی اول، استفاده از DES دوگانه^۱ می‌باشد. در این شیوه، از دو نمونه از رمزهای DES برای رمزنگاری و دو نمونه از رمزهای وارون برای رمزگشایی استفاده می‌کنیم. هر نمونه، از یک کلید مجزا استفاده می‌کند و این به معنی دو برابر شدن اندازه‌ی کلید است (۱۱۲ بیت)؛ با این وجود، همان‌گونه که در بخش بعدی خواهیم گفت، DES دوگانه در برابر حمله به شیوه‌ی متن عادی شناخته شده آسیب‌پذیر است.

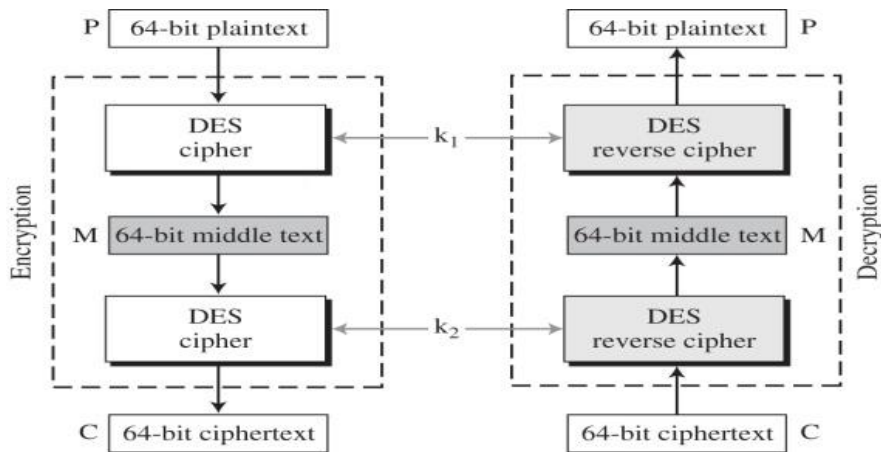
حمله از نوع ملاقات در میانه‌ی راه^۲

در نگاه اول چنین به نظر می‌رسد که DES دوگانه تعداد آزمایش‌ها برای جستجوی کلید را از 2^{56} (در DES تکی) به 2^{112} (در DES دوگانه) افزایش می‌دهد. با این وجود، استفاده از حمله به شیوه‌ی متن عادی شناخته شده موسوم به ملاقات در میانه راه ثابت می‌کند که DES دوگانه، این آسیب‌پذیری را کمی یعنی تا 2^{57} آزمایش ارتقا می‌دهد، نه تا 2^{112} . شکل ۶-۱۴ دیاگرام DES دوگانه را نشان می‌دهد. آلیس برای رمزنگاری متن عادی P به متن رمز C از دو کلید K_1 و K_2 استفاده می‌کند و باب نیز کلیدهای K_1 و K_2 را برای بازیابی DES به کار می‌برد.

=====

¹ Double DES (2DES)

² Meet-in-the-Middle Attack



شکل ۶-۱۴: حمله به شیوهی ملاقات در میانه‌ی راه برای DES دوگانه

نکته این است که متن میانی، متن ایجاد شده به وسیله‌ی نخستین رمزنگاری یا نخستین رمزگشایی -M- باید برای رمزنگاری و رمزگشایی یکسان باشد تا کار کند؛ به عبارت دیگر، دو رابطه‌ی زیر را برقرار است:

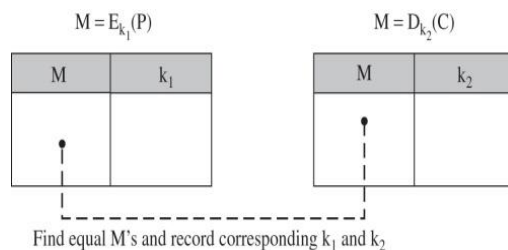
$$M = E_{K_1}(P) \quad M = D_{K_2}(C)$$

فرض کنید رمزشکن یک جفت P و C را پیشتر استراق سمع نموده است (حمله به شیوهی متن عادی شناخته شده). بر اساس اولین رابطه‌ی بالا، با استفاده از تمام مقادیر احتمالی K_1 (به میزان 2^{56})، P را رمزنگاری می‌کند و تمام مقادیر به دست آمده برای M را می‌نویسد. بر اساس دومین رابطه بالا او با استفاده از تمام مقادیر احتمالی K_2 (به میزان 2^{56})، C را رمزگشایی می‌کند و تمام مقادیر به دست آمده برای M را می‌نویسد. رمزشکن مقادیر به دست آمده برای M را در دو جدول نوشته سپس مقادیر M را با هم مقایسه می‌کند تا بتواند زوج‌های K_1 و K_2 را که برای آن‌ها مقدار M در هر دو جدول یکسان است را پیدا کند (شکل ۶-۱۵). توجه داشته باشید که در اینجا دست‌کم باید یک جفت کلید وجود داشته باشد، زیرا رمزشکن تحقیق جامعی روی ترکیب دو کلید انجام داده است:

(۱) اگر تنها یک جفت وجود داشته باشد، رمزشکن دو کلید (K_2, K_1) را یافته است. اگر بیش از یک جفت وجود داشته باشد، به مرحله بعدی می‌رود.

(۲) یک جفت متن عادی- متن رمز دیگری را انتخاب می‌کند و از هر جفت کلید به دست آمده استفاده می‌کند تا ببیند که آیا می‌تواند از متن رمز، متن عادی را به دست آورد.

(۳) اگر بیش از یک جفت کلید به دست آمد، مرحله ۲ را آن قدر تکرار می‌کند تا سرانجام یک جفت کلید منحصر به فرد به دست آورد.



شکل ۶-۱۵: جدول‌های حمله از نوع ملاقات در میانه راه

ثابت شده است که پس از اعمال گام دوم چند جفت متن عادی- متن رمز استراق سمع شده، کلیدها به دست می‌آید. این بدان معنی است که به جای استفاده از کلیدها 2^{112} ، رمز شکن آزمایش- های جستجوی کلید در 2^{56} را دو بار بکار می‌برد (اگر بیش از یک کلید کاندید به دست آمد، این آزمایش را چند بار دیگر انجام می‌دهد)؛ به عبارت دیگر، حرکت از DES تکی به DES دوگانه، استحکام DES را از 2^{56} به 2^{112} (نه تا 2^{112}) افزایش داده‌ایم.

DES سه‌گانه

برای ارتقای امنیت DES، DES سه‌گانه^۱ ارائه شد. این نوع DES از سه مرحله DES برای رمزنگاری و رمزگشایی استفاده می‌کند. امروزه از دو نسخه از DES سه‌گانه استفاده می‌کنند: DES سه‌گانه با دو کلید و DES سه‌گانه با سه کلید.

DES سه‌گانه با دو کلید

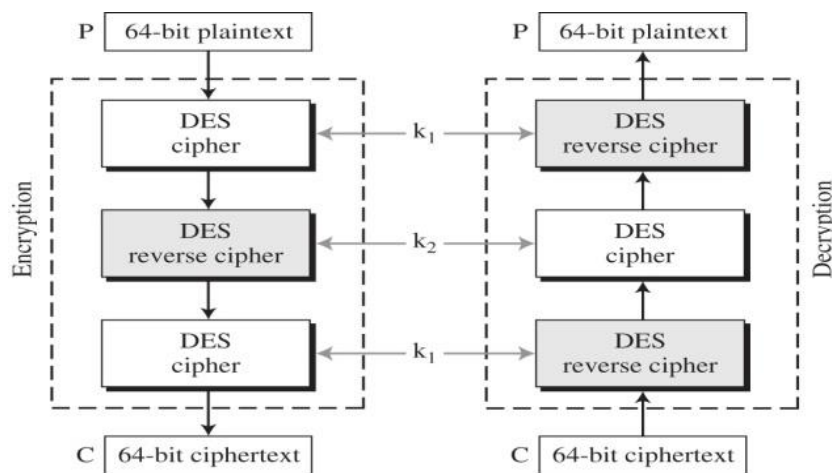
در DES سه‌گانه با دو کلید، فقط دو کلید K_1 و K_2 وجود دارد. در مرحله اول و سوم از K_1 و مرحله دوم از K_2 استفاده می‌کنند. برای سازگاری DES سه‌گانه با DES تکی، مرحله میانی از رمزگشایی (رمز وارون) در قسمت رمزنگاری و از رمزنگاری در قسمت رمزگشایی استفاده می‌کند. بدین ترتیب پیام رمزنگاری شده با DES تکی با کلید K را می‌توان با DES سه‌گانه رمزگشایی کرد به شرطی که $K_1 = K_2 = K$ باشد. اگرچه DES سه‌گانه با دو کلید نیز در برابر حمله‌ی متن عادی شناخته شده آسیب‌پذیر است ولی بسیار قوی‌تر از DES دوگانه می‌باشد. صنعت بانکداری، استفاده از این شیوه رمزنگاری DES را پذیرفته است. شکل ۶-۱۶ DES سه‌گانه با دو کلید را نشان می‌دهد.

=====

^۱ triple DES (3DES)

DES سه گانه با سه کلید

احتمال حملات متن عادی شناخته شده به DES سه گانه با دو کلید باعث شد که بسیاری از برنامه‌های کاربردی به استفاده از DES سه گانه با سه کلید روی آورند. اگرچه این الگوریتم می‌تواند برای سازگاری با DES تکی از سه مرحله رمز DES در قسمت رمزنگاری و سه مرحله رمز وارون در قسمت رمزگشایی استفاده کند، اما قسمت رمزنگاری از EDE و قسمت رمزگشایی از DED استفاده می‌کند (E نشان دهنده رمزنگاری و D نشان دهنده رمزگشایی است). با قرار دادن $K_1 = K_2$ و K_3 کلید دلخواه انتخاب شده به وسیله گیرنده، سازگاری با DES تکی محقق می‌گردد. بسیاری از برنامه‌های کاربردی مانند PGP از DES سه گانه با سه کلید استفاده می‌کنند.



شکل ۶-۱۶: DES سه گانه با دو کلید

۶-۵- امنیت DES

DES به عنوان اولین رمز بلوکی مهم مورد بررسی بسیار دقیق قرار گرفته است. در میان انواع حمله‌ها، سه مورد بیشتر مورد توجه قرار گرفته است: آزمودن کلیدی کلیدها، رمزگشایی بر اساس تحلیل دیفرانسیلی رمز و رمزگشایی بر اساس تحلیل خطی.

حمله بر اساس آزمودن کلیدی کلیدها

در مورد نقاط ضعف کلید رمز کوتاه در DES صحبت کردیم. با ترکیب این نقاط ضعف با نقاط ضعف متمم کلید، پیداست که DES را می‌توان با استفاده از رمزنگاری به تعداد 2^{55} حالت شکست. با این وجود امروزه بسیاری از برنامه‌های کاربردی یا از ۳DES با دو کلید (اندازه کلید ۱۱۲) یا ۳DES با سه

کلید (اندازه کلید ۱۶۸) استفاده می‌کنند. این دو نسخه DES چندگانه، DES را در برابر این نوع حملات پایدار می‌سازد.

رمزگشایی با استفاده از تحلیل رمز دیفرانسیلی

در فصل ۵ تکنیک رمزگشایی دیفرانسیلی بر روی رمزهای بلوک پیشرفته صحبت کردیم. DES در مورد این نوع از حملات ایمن نیست؛ با این حال مشخص است که طراحان DES از این نوع حمله آگاه بوده و جعبه‌های S را طراحی کرده‌اند و ۱۶ را به عنوان تعداد گردش‌ها انتخاب نموده و بدین ترتیب DES را در برابر این نوع از حملات مقاوم کردند. امروزه مشخص شده است که اگر 2^{47} متن عادی انتخاب شده یا 2^{50} متن عادی شناخته شده داشته باشیم، می‌توانیم با استفاده از رمزگشایی دیفرانسیلی، رمز DES را بشکنیم. اگرچه این حمله کارآمدتر از حمله‌ی آزمودن کلیدی کلیدها به نظر می‌آید، اما یافتن 2^{47} متن عادی انتخاب شده یا 2^{50} متن عادی شناخته شده غیرعملی است؛ بنابراین می‌توانیم بگوییم که DES در برابر رمزگشایی دیفرانسیلی مقاوم است. مشخص شده است که افزایش تعداد گردش به عدد ۲۰ مستلزم بیش از 2^{64} متن عادی انتخاب شده برای این نوع حمله می‌باشد که عملاً غیر ممکن است؛ زیرا تعداد ممکن بلوک‌های متن عادی در DES فقط 2^{64} است.

رمزگشایی بر اساس تحلیل رمز خطی

در فصل ۵ در مورد تحلیل رمز بر روی رمزهای بلوکی پیشرفته صحبت کردیم. تحلیل رمز خطی، جدیدتر از تحلیل رمز دیفرانسیلی است. DES در برابر تحلیل رمز خطی آسیب‌پذیرتر از تحلیل رمز دیفرانسیلی است. شاید بدین علت که این نوع حمله برای طراحان DES ناشناخته بود. جعبه‌های S در برابر تحلیل رمز دیفرانسیلی چندان مقاوم نیستند. مشخص شده است که می‌توان با استفاده از 2^{43} جفت متن رمز شناخته شده، رمز DES را شکست؛ با این وجود، از دیدگاه عملی، یافتن این تعداد بعید است.

۶-۶- وب سایت‌های پیشنهادی:

وب سایت‌های زیر اطلاعات بیشتری در زمینه موضوعات مطرح شده در این فصل در اختیار شما قرار می‌دهند.

<http://www.itl.nist.gov/fipspubs/fip46-2.htm>
www.nist.gov/director/prog-ofc/report01-2.pdf
www.engr.mun.ca/~howard/PAPERS/ldc_tutorial.ps
islab.oregonstate.edu/koc/ece575/notes/dc1.pdf
homes.esat.kuleuven.be/~abiryuko/Cryptan/matsui_des
<http://nsfsecurity.pr.erau.edu/crypto/lincrypt.html>

۶-۷- چکیده

- * استاندارد رمزنگاری داده‌ها (DES)، رمز بلوکی کلید متقارن است که مؤسسه ملی استاندارد و فناوری تحت عنوان FIPS46 در فدرال ریجستر منتشر کرد.
- * در قسمت رمزنگاری، DES متن رمز ۶۴ بیتی را دریافت و بلوک ۶۴ بیتی از متن عادی را بازسازی می‌کند. کلید رمز ۵۶ بیتی یکتایی برای هردو قسمت بکار می‌رود.
- * پروسه رمزنگاری از دو جایگشت (جعبه‌های P)، که به آن‌ها جایگشت نخستین و پایانی می‌گوئیم و از شانزده گردش Feistel تشکیل شده است. در هر گردش DES رمز Feistel از دو جزء ترکیب کننده و مبادله گر تشکیل شده که هر یک از این اجزاء، وارون پذیرند.
- * قلب DES، تابع DES است. این تابع، کلید ۴۸ بیتی را بر ۳۲ بیت سمت راست اعمال می‌کند و بدین ترتیب خروجی ۳۲ بیتی تولید می‌کند. این تابع از ۴ عمل تشکیل شده است: یک جایگشت گسترشی، یک سپیدگر (که کلید اضافه می‌کند)، یک گروه از جعبه‌های S و یک جایگشت مستقیم.
- * مولد کلید در هر گردش، شانزده کلید ۴۸ بیتی از یک کلید رمز ۵۶ بیتی می‌سازد؛ با این وجود، کلید رمز معمولاً به صورت کلید ۶۴ بیت نشان داده می‌شود که ۸ بیت اضافه، بیت‌های توازن هستند. این بیت‌های توازن پیش از مرحله‌ی تولید کلید واقعی حذف می‌شوند.
- * DES با توجه به تأثیرات بهمن گونه و تمامیت، عملکرد خوبی از خود نشان داده است. نقطه ضعف در DES شامل طراحی رمز (جعبه‌های S و جعبه‌های P) و کلید رمز (طول کلید، کلیدهای ضعیف، کلیدهای شبه ضعیف، کلیدهای ضعیف احتمالی و متمم‌های کلید) می‌باشد.
- * از آنجا که DES گروه نیست، یک راه حل برای ارتقای امنیت DES، استفاده از DES چندگانه (DES دوگانه و سه گانه) می‌باشد. DES دوگانه در برابر حمله‌ی ملاقات در میانه‌ی راه آسیب پذیر است؛ بنابراین در برنامه‌های کاربردی بیشتر از DES سه گانه با دو کلید یا سه کلید استفاده می‌شود.
- * طراحی جعبه‌های S و تعداد گردش‌ها، DES را تقریباً در برابر تحلیل رمز دیفرانسیلی مقاوم می‌سازد. با این وجود، DES در برابر تحلیل رمز خطی آسیب پذیر است، البته به شرطی که رمز شکن بتواند متون عادی شناخته شده کافی جمع‌آوری کند.

۶-۸- مجموعه تمرین‌ها

پرسش‌های دوره‌ای

- (۱) اندازه‌ی بلوک در DES چقدر است؟ اندازه‌ی کلید رمز در DES چیست؟ اندازه‌ی کلید گردش در DES چند است؟
- (۲) تعداد گردش‌ها در DES چقدر است؟
- (۳) در شیوه‌ی اول رمزنگاری چه تعداد ترکیب‌کننده و مبادله‌گر مورد استفاده قرار گرفته است تا بتوان رمزنگاری و مبادله‌گر را وارون یکدیگر کرد؟ چه تعداد در شیوه‌ی دوم بکار رفته است؟
- (۴) در الگوریتم رمز DES چه تعداد جایگشت بکار رفته است؟ در مولد کلید گردش چند جایگشت استفاده شده است؟
- (۵) در رمز DES از چند عمل XOR استفاده شده است؟
- (۶) چرا تابع DES به یک جایگشت گسترشی نیاز دارد؟
- (۷) چرا مولد کلید گردش به یک جایگشت ریزشی نیاز دارد؟
- (۸) تفاوت بین کلید ضعیف، کلید نیمه ضعیف و کلید ضعیف احتمالی چیست؟
- (۹) DES دوگانه چیست؟ چه نوع حمله‌ای باعث بی‌استفاده شدن DES دوگانه می‌شود؟
- (۱۰) DES سه‌گانه چیست؟ DES سه‌گانه با دو کلید چیست؟ DES سه‌گانه با سه کلید چیست؟

تمرین‌ها

- (۱۱) به پرسش‌های زیر در مورد جعبه‌های S در DES پاسخ دهید:
 - (a) نتیجه گذراندن ۱۱۰۱۱۱ از میان جعبه‌ی S_۳ را نشان دهید.
 - (b) نتیجه گذراندن ۰۰۱۱۰۰ از میان جعبه‌ی S_۴ را نشان دهید.
 - (c) نتیجه گذراندن ۰۰۰۰۰۰ از میان جعبه‌ی S_۷ را نشان دهید.
 - (d) نتیجه گذراندن ۱۱۱۱۱۱ از میان جعبه‌ی S_۲ را نشان دهید.
- (۱۲) برای نشان دادن عبور ۰۰۰۰۰۰ از میان هر ۸ جعبه‌ی S جدولی رسم کنید. آیا الگویی در خروجی‌ها می‌بینید؟
- (۱۳) برای نشان دادن عبور ۱۱۱۱۱۱ از میان هر ۸ جعبه‌ی S جدولی رسم کنید. آیا الگویی در خروجی‌ها می‌بینید؟
- (۱۴) با استفاده از جفت‌های ورودی زیر، معیار سوم جعبه‌ی S_۳ را چک کنید.

(a) ۰۰۰۰۰۰ و ۰۰۰۰۰۱

(b) ۱۱۱۱۱۱ و ۱۱۱۰۱۱

(۱۵) با استفاده از جفت‌های خروجی زیر، چهارمین معیار طراحی جعبه‌ی S_2 را بررسی کنید.

(a) ۰۰۱۱۰۰ و ۱۱۰۰۰۰

(b) ۱۱۰۰۱۱ و ۰۰۱۱۱۱

(۱۶) با استفاده از جفت خروجی‌های زیر، پنجمین معیار طراحی جعبه‌ی S_4 را بررسی کنید.

(a) ۰۰۱۱۰۰ و ۱۱۰۰۰۰

(b) ۱۱۰۰۱۱ و ۰۰۱۱۱۱

(۱۷) برای کنترل ششمین معیار طراحی برای جعبه‌ی S_5 ، ۳۲ جفت ورودی ۶ بیتی ایجاد کنید.

(۱۸) نشان دهید که چگونه هشت معیار طراحی برای جعبه‌ی S_7 انجام می‌شود.

(۱۹) به وسیله کنترل ورودی به جعبه‌ی S_2 در گردش دوم، اولین معیار طراحی برای جعبه‌ی P را ثابت کنید.

(۲۰) به وسیله‌ی کنترل ورودی به جعبه‌ی S_3 در گردش چهارم، دومین معیار طراحی برای جعبه‌ی P را ثابت کنید.

(۲۱) به وسیله‌ی کنترل ورودی به جعبه‌ی S_4 در گردش سوم، سومین معیار طراحی برای جعبه‌ی P را ثابت کنید.

(۲۲) به وسیله‌ی کنترل ورودی به جعبه‌ی S_6 در گردش ۱۲، چهارمین معیار طراحی برای جعبه‌ی P را ثابت کنید.

(۲۳) به وسیله‌ی کنترل رابطه بین جعبه‌های S_3 ، ۴ و ۵ در گردش‌های ۱۰ و ۱۱، پنجمین معیار طراحی برای جعبه‌های P را ثابت کنید.

(۲۴) با کنترل مقصد یک جعبه‌ی S دلخواه، ششمین معیار طراحی برای جعبه‌های P را ثابت کنید.

(۲۵) با کنترل رابطه بین جعبه‌ی S_5 در گردش ۴ و جعبه‌ی S_7 در گردش ۵، هفتمین معیار طراحی برای جعبه‌های P را ثابت کنید.

(۲۶) با استفاده از جایگزینی، شکل ۶-۹ را دوباره رسم کنید.

(۲۷) ثابت کنید که رمز وارون در شکل ۶-۹ در واقع وارون رمز برای DES با سه گردش است. با یک متن عادی در آغاز رمز شروع کنید و ثابت نمایید که می‌توانید همان متن عادی را در پایان رمز وارون به دست آورید.

(۲۸) جایگشت متراکم کلید در جدول ۶-۱۴ را به دقت مطالعه کنید.

(a) چه درگاه‌های ورودی در خروجی ناپدید شده است؟

(b) آیا کل ۲۴ بیت خروجی چپ از ۲۸ بیت ورودی چپ می‌آید؟

(c) آیا کل ۲۴ بیت خروجی راست از کل ۲۸ بیت ورودی راست می‌آید؟

(۲۹) نتایج داده‌های مبنای شانزده زیر را پس از عبور از جعبه جایگشت نخستین نشان دهید.

۰۱۱۰ - ۱۰۲۳ - ۴۱۴۰ - ۱۰۲۳

(۳۰) نتایج داده‌های مبنای شانزده زیر را پس از عبور از جعبه جایگشت پایانی نشان دهید.

AAAA BBBB CCCC DDDD

(۳۱) اگر کلید دارای بیت توازن (۶۴ بیت) ۰۱۲۳ABCD۲۵۶۲۱۴۵۶ باشد، اولین گردش را پیدا کنید.

(۳۲) با فرض این که DES تنها از یک گردش تشکیل شده است، با استفاده از بلوک متن عادی که تمام آن ۰ است و کلید ۵۶ بیتی که تمام آن ۰ است؛ ضعف کلید متمم را ثابت کنید.

(۳۳) آیا می‌توانید حمله از نوع ملاقات در میانه برای یک DES سه‌گانه طراحی کنید؟

(۳۴) یک شبه کد برای جابجا کردن روال‌های استفاده شده در الگوریتم ۱-۶ بنویسید.

جابجا کن (m, n) ، داخل بلوک $[n]$ ، خارج بلوک $[m]$ ، جدول جایگشت $[m]$

(۳۵) یک شبه کد برای تقسیم کردن روال استفاده شده در الگوریتم ۱-۶ بنویسید.

تقسیم کن (m, n) ، داخل بلوک $[n]$ ، بلوک چپ $[m]$ ، بلوک راست $[m]$

(۳۶) یک شبه کد برای ترکیب کردن روال استفاده شده در الگوریتم ۱-۶ بنویسید.

ترکیب کن (m, n) ، بلوک چپ $[n]$ ، بلوک راست $[n]$ ، خارج بلوک $[m]$

(۳۷) یک شبه برنامه برای XOR روال استفاده شده در الگوریتم ۱-۶ بنویسید.

XOR (n) ، اولین داخل بلوک $[n]$ ، دومین داخل بلوک $[n]$ ، خارج بلوک $[n]$

(۳۸) برای نشان دادن روش جایگزین، الگوریتم ۱-۶ را تغییر دهید.

(۳۹) الگوریتم ۱-۶ را به گونه‌ای تکمیل کنید که بتوان از آن هم برای رمزنگاری و هم برای رمزگشایی استفاده کرد.

فصل هفتم

استاندارد رمزنگاری پیشرفته^۱

چشم‌انداز

در این فصل در مورد استاندارد رمزنگاری پیشرفته، رمز بلوکی کلید متقارن پیشرفته، که ممکن است جایگزین DES شود صحبت می‌کنیم. این فصل اهداف زیر را دنبال می‌کند:

✱ مروری گذرا بر تاریخچه AES.

✱ تعریف ساختار AES.

✱ تعیین تغییرات به کار رفته در AES.

✱ تشریح مراحل گسترش کلید.

✱ بحث در مورد چگونگی پیاده‌سازی‌های گوناگون.

✱ هدف نهایی این فصل، چگونگی تحقق اهداف امنیتی با استفاده از ساختارهای جبری مطرح

شده در فصل ۴ است.

۷-۱- مقدمه

استاندارد رمزنگاری پیشرفته، رمز بلوکی کلید متقارن است که در ماه دسامبر سال ۲۰۰۱ میلادی به وسیله‌ی مؤسسه‌ی ملی استاندارد و فناوری^۲ منتشر شد.

=====

^۱ Advanced Encryption Standard (AES)

^۲ National Institute of Standards and Technology (NIST)

تاریخچه

در سال ۱۹۹۷ میلادی NIST جستجوی جایگزین برای DES را که استاندارد رمزنگاری پیشرفته یا ADS نامیده شده بود، آغاز کرد. NIST خواستار مشخصات اندازه‌ی بلوکی ۱۲۸ بیتی و سه اندازه‌ی مختلف کلید ۱۲۸، ۱۹۲ و ۲۵۶ بیتی بود. همچنین این ویژگی‌ها تصریح می‌کرد که ADS باید الگوریتم منبع باز و در دسترس مردم در سراسر دنیا باشد. برای دریافت پاسخ از سراسر دنیا، درخواست به صورت بین‌المللی منتشر گردید.

پس از اولین کنفرانس کاندیداتوری ADS، NIST اعلام نمود که از ۲۱ الگوریتم دریافتی، ۱۵ الگوریتم با معیارها و ضوابط تطابق دارند و به عنوان اولین داوطلب انتخاب شده‌اند (آگوست ۱۹۹۸ میلادی). الگوریتم‌ها از چند کشور ارائه شده بود. تنوع طرح‌های پیشنهادی حاکی از گستردگی مشارکت جهانی بود. پس از دومین کنفرانس کاندیداتوری AES، که در رم برگزار شد، NIST اعلام نمود که از بین ۱۵ کاندیدا، ۵ کاندیدی -MARS, RC6, Rijndael, Serpent, twofish- به عنوان فینالیست انتخاب شدند (آگوست ۱۹۹۹ میلادی).

پس از سومین کنفرانس کاندیداتوری AES، NIST اعلام کرد که طرح پیشنهادی بلژیک موسوم به رین دول^۱ که توسط Vincent Rijment و Joan Daement طراحی شده است، به عنوان استاندارد رمزنگاری پیشرفته انتخاب شد (اکتبر ۲۰۰۰ میلادی).

در فوریه ۲۰۰۱ میلادی NIST اعلام کرد که طرح پیش‌نویس استاندارد پردازش اطلاعات فدرال^۲ جهت بررسی و اظهارنظر در دسترس عموم قرار می‌گیرد. سرانجام در دسامبر ۲۰۰۱ میلادی ADS تحت عنوان FIPS197 در فدرال ریجستر منتشر شد.

معیارها

معیارهای مشخص شده توسط NIST برای انتخاب ADS در سه حوزه بود: امنیت، هزینه و اجرا. در نهایت Rijndael به عنوان طرحی که ترکیب این معیارها را به بهترین نحو دارد، انتخاب شد.

=====

¹ Rain Doll

² Federal Information Processing Standard (FIPS)

امنیت

تأکید اصلی بر امنیت بود. از آن جا که NIST مشخصاً کلید ۱۲۸ بیتی را مد نظر داشت، این معیار بیشتر بر روی مقاومت در مقابل حملات مبتنی بر تحلیل رمز پافشاری می‌کند تا مقاومت در برابر حمله به شیوهی آزمودن کلیدی کلیدها.

هزینه

معیار دوم هزینه است که دربرگیرندهی بازده محاسباتی و نیاز ذخیره‌سازی در پیاده‌سازی مختلف در سخت افزار، نرم‌افزار یا کارت‌های هوشمند می‌باشد.

اجرا

این معیار دربرگیرندهی لزوم انعطاف‌پذیری الگوریتم (روی هر سکویی^۱ قابل اجرا باشد) و سادگی آن است.

تعداد گردش‌ها

AES رمز غیر Feistel است که بلوک داده‌ای ۱۲۸ بیتی را رمزنگاری و رمزگشایی می‌کند و از ۱۰، ۱۲ یا ۱۴ گردش تشکیل شده است. اندازه‌ی کلید که می‌تواند ۱۲۸، ۱۹۲ یا ۲۵۶ بیت باشد، به تعداد گردش‌ها بستگی دارد. شکل ۷-۱ طراحی کلی برای الگوریتم رمزنگاری موسوم به رمزنویسی^۲ را نشان می‌دهد. الگوریتم رمزگشایی^۳ (مرسوم به رمز وارون) شبیه به الگوریتم رمزنگاری است، با این تفاوت که کلیدها در هر گردش به صورت وارون اعمال می‌شود.

در شکل ۷-۱، N_r تعداد گردش‌ها را نشان می‌دهد؛ همچنین این شکل رابطه‌ی بین تعداد گردش‌ها و اندازه‌ی کلید را نشان می‌دهد و این به این معنی است که می‌توانیم سه نسخه مختلف AES داشته باشیم. این سه نسخه را AES-۱۲۸، AES-۱۹۲ و AES-۲۵۶ می‌نامیم. با این وجود، کلیدهای هر گردش که به وسیله-ی الگوریتم گسترش کلید ایجاد می‌شود، همیشه ۱۲۸ بیت می‌باشد، یعنی اندازه‌ی کلید و اندازه‌ی بلوک متن عادی با متن رمز یکسان است.

استاندارد AES سه نسخه با ۱۰، ۱۲ یا ۱۴ گردش تعریف کرده است.

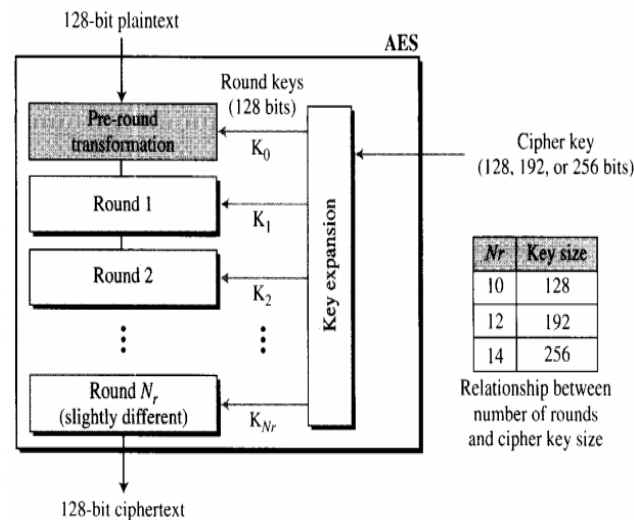
=====

¹ Platform

² Cipher

³ Decryption

هر نسخه از اندازه‌ی کلید رمز متفاوتی استفاده می‌کند (۱۲۸، ۱۹۲ یا ۲۵۶)، اما کلیدهای گردش‌ها معمولاً ۱۲۸ بیتی است.



شکل ۷-۱: طراحی کلی رمزنگاری AES

تعداد کلید گردش‌ی ایجاد شده به وسیله‌ی الگوریتم گسترش کلید، معمولاً یکی بیشتر از تعداد گردش‌هاست. به عبارت دیگر، داریم:

$$N_r + 1 = \text{تعداد کلید گردش‌ی}$$

کلیدهای گردش را با $K, K_1, K_2, \dots, K_{N_r}$ نشان می‌دهیم.

واحد داده‌ها

AES از پنج واحد برای اندازه‌گیری داده‌ها استفاده می‌کند: بیت، بایت، کلمه، بلوک و وضعیت^۱. بیت، کوچک‌ترین واحد و واحد یکپارچه (اتمیک) است. سایر واحدها را می‌توان در قالب واحد کوچک‌تری بیان کرد. شکل ۷-۲ واحدهای داده‌ای مرکب را نشان می‌دهد: بایت، کلمه، بلوک و وضعیت.

بیت

در AES هر بیت یک رقم دودویی با مقدار ۰ یا ۱ می‌باشد. برای نشان دادن یک بیت، از حروف کوچک استفاده می‌کنیم.

بایت

بایت، گروهی متشکل از هشت بیت است که یک نهاد یگانه^۱ به شکل ماتریس سطری (۸×۱) متشکل از هشت بیت تلقی می‌شود. وقتی ماتریس سطری در نظر گرفته می‌شود، بیت‌ها از چپ به راست در ماتریس

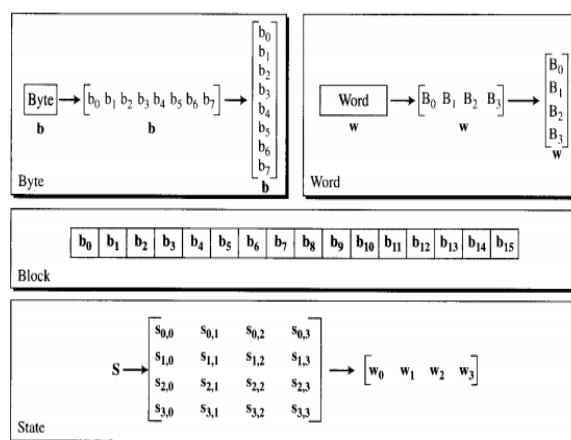
=====

¹ State

قرار داده می‌شوند. در ماتریس ستونی، بیت‌ها از بالا به پایین در ماتریس قرار می‌گیرند. برای نشان دادن یک بایت، از حروف کوچک توپر استفاده می‌کنیم.

کلمه

کلمه‌ی گروهی متشکل از ۳۲ بیت است که یک نهاد یگانه به شکل ماتریس سطری متشکل از چهار بایت یا ماتریس ستون متشکل از چهار بایت تلقی می‌شود. وقتی کلمه را ماتریس سطری در نظر بگیریم، بایت‌ها از چپ به راست در ماتریس قرار داده می‌شوند؛ وقتی آن را به شکل ماتریس ستونی در نظر بگیریم، بایت‌ها از بالا به پایین در ماتریس قرار می‌گیرند. از حرف کوچک توپر **w** برای نشان دادن کلمه استفاده می‌کنیم.



شکل ۷-۲: واحدهای داده‌ای به کاررفته در AES

بلوک

AES، داده‌ها را به صورت بلوکی رمزنگاری و رمزگشایی می‌کند. یک بلوک در AES، گروهی متشکل از ۱۲۸ بیت است. با این وجود، بلوک را می‌توان به شکل ماتریس سطری متشکل از ۱۶ بایت نشان داد.

وضعیت^۲

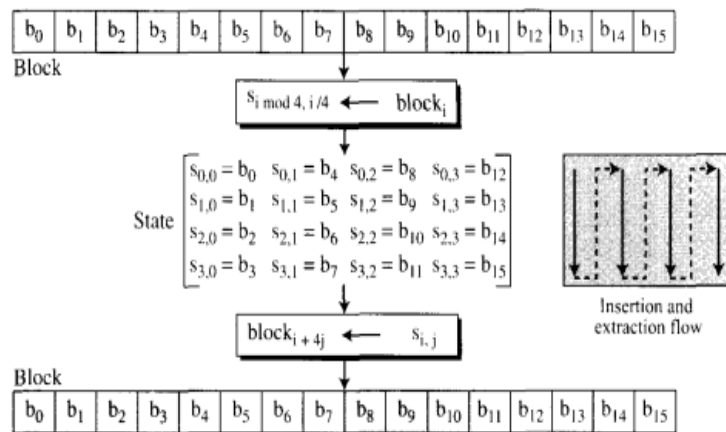
AES از چندین گردش استفاده می‌کند که هر یک از آن‌ها از چند مرحله تشکیل شده است. بلوک داده‌ای از هر مرحله به مرحله‌ی بعد منتقل می‌شود. AES در آغاز و پایان رمز، از عبارت بلوک داده‌ای^۳ استفاده می‌کند. پیش و پس از هر مرحله، به بلوک داده‌ای، وضعیت گفته می‌شود. از حرف بزرگ توپر برای نشان دادن وضعیت‌ها استفاده می‌کنیم. اگرچه معمولاً وضعیت در مراحل مختلف را **S** می‌نامند، اما در اینجا استثنائاً از حرف **T** برای نشان دادن وضعیت استفاده می‌کنیم. وضعیت‌ها مانند بلوک‌ها، از ۱۶ بایت تشکیل شده‌اند، اما

¹ Single Entity

² State

³ Data Block

معمولاً ماتریس‌های 4×4 در نظر گرفته می‌شوند. در این حالت، هر عضو از یک وضعیت را به صورت $S_{r,c}$ نشان می‌دهند، طوری که $(0-3)$ سطر و $(0-3)$ ستون را مشخص می‌کند. گاهی اوقات، یک وضعیت به عنوان ماتریس سطری (1×4) متشکل از کلمات در نظر گرفته می‌شود. این در صورتی مفهوم دارد که کلمه را به شکل ماتریس ستونی در نظر بگیریم. در آغاز رمزنویسی، بایت‌ها در بلوک داده‌ای در یک عبارت به شکل ستون به ستون و هر ستون از بالا به پایین درج می‌شود. در پایان رمزنویسی، بایت‌ها در هر عبارت به همین ترتیب - همان‌گونه که در شکل ۷-۳ نشان داده شده است - استخراج می‌شوند.



شکل ۷-۳: تبدیل بلوک به وضعیت و وضعیت به بلوک

مثال ۷-۱

اجازه دهید بررسی نماییم که چگونه بلوک ۱۶ کاراکتری را می‌توان به صورت ماتریس 4×4 نشان داد. فرض کنید بلوک متن عادی "AES uses a matrix" است. برای به دست آوردن "AESUSESAMATRIXZZ" دو حرف اضافی^۱ که جزء متن نیست به آخر جمله اضافه می‌کنیم. اکنون یک عدد صحیح بین ۰۰ و ۲۵ را جایگزین هر کاراکتر می‌کنیم؛ سپس هر بایت را به شکل یک عدد با دو رقم در مبنای شانزده (هگزادسیمال) نشان می‌دهیم. به عنوان مثال، کاراکتر "S" ابتدا به ۱۸ تغییر می‌یابد و سپس به شکل ۱۲_{۱۶} در مبنای شانزده نوشته می‌شود، سپس همان‌گونه که در شکل ۷-۴ نشان داده شده است ماتریس وضعیت، ستون به ستون پر می‌شود.

^۱ Bogus characters

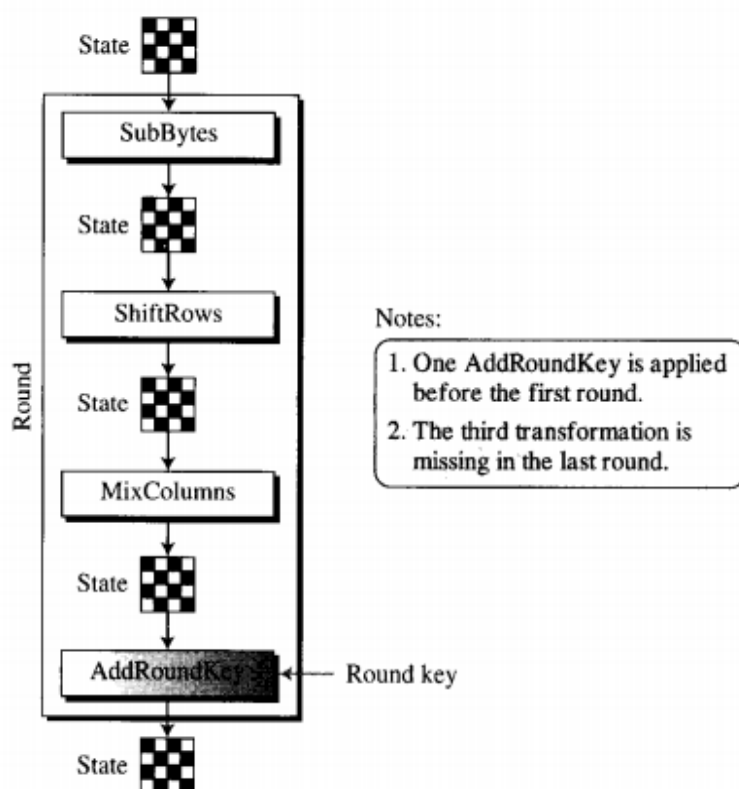
Text	A E S U S E S A M A T R I X Z Z															
Hexadecimal	00 04 12 14 12 04 12 00 0C 00 13 11 08 23 19 19															
	<div><div><div>00 12 0C 08</div><div>04 04 00 23</div><div>12 12 13 19</div><div>14 00 11 19</div></div><div>State</div></div>															

شکل ۷-۴: تغییر متن رمز به ماتریس وضعیت

ساختار هر گردش

شکل ۷-۵ ساختار گردش‌ها را در قسمت رمزنگاری نشان می‌دهد. هر گردش، به استثنای گردش آخر، از چهار تغییری استفاده می‌کند که وارون‌پذیر باشد. گردش آخر فقط سه تغییر دارد.

همان‌گونه که شکل ۷-۵ نشان می‌دهد، هر تغییر، ماتریس وضعیت را دریافت و ماتریس وضعیت دیگری را برای استفاده در تغییر بعدی یا گردش بعدی ایجاد می‌کند. قسمت پیش گردش^۱ فقط از یک تغییر^۲ و گردش آخر از سه تغییر استفاده می‌کند (تغییر Mixcolumns نادیده گرفته می‌شود).



شکل ۷-۵: ساختار گردش‌ها در قسمت رمزنگاری

=====

¹ Pre-round

² Add Round Key

نکات:

(۱) افزودن کلید در گردش‌ها، پیش از گردش اول اعمال می‌شود.

(۲) در گردش آخر، تغییر سوم وجود ندارد.

در قسمت رمزگشایی، تغییرات زیر به کار می‌رود:

InvSsubyte, InvShiftrows, InvMixColumns, AddRoundKey (این مورد، وارون خود

است).

۷-۲- تغییرات

جهت تأمین امنیت، AES از چهار نوع تغییر استفاده می‌کند: جایگزینی^۱، جایگشت^۲، ترکیب^۳ و افزودن کلید^۴. در مورد هر یک از این موارد به تفصیل صحبت خواهیم کرد.

جایگزینی

AES مانند DES از جایگزینی استفاده می‌کند اما با سازوکار متفاوت. نخست، جایگزینی برای هر بایت انجام می‌شود. دوم، برای تغییر هر بایت، تنها از یک جدول استفاده می‌شود؛ این بدان معنی است که اگر دو بایت مانند هم باشند، حاصل تغییر نیز یکسان است. سوم، تغییر یا از طریق مراحل مراجعه به جدول یا به وسیله محاسبه ریاضیاتی در میدان $GF(2^8)$ تعیین می‌گردد. AES از دو تغییر وارون‌پذیر استفاده می‌کند.

اولین تغییر subBytes در قسمت رمزنگاری به کار می‌رود. برای جایگزینی یک بایت، بایت را به شکل دو رقم در مبنای شانزده (هگزادسیمال) تفسیر می‌کنیم. رقم سمت چپ، سطر و رقم سمت راست، ستون جدول جایگزینی را مشخص می‌کند. دو رقم مبنای شانزده در محل برخورد سطر و ستون، مقدار جایگزین یا بایت جدید می‌باشد. شکل ۷-۶ این مطلب را نشان می‌دهد.

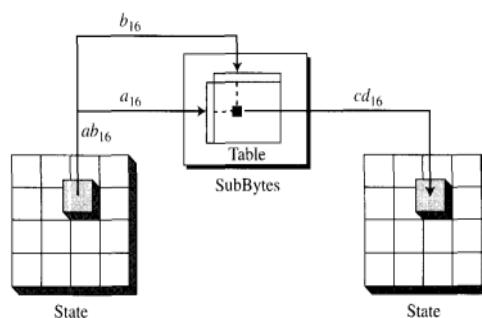
=====

¹ Substitution

² Permutation

³ Mixing

⁴ Key-Adding



شکل ۷-۶: تغییرات subBytes

در تغییر subBytes ماتریس 4×4 از بایت‌ها در نظر گرفته می‌شود و هر بار، تغییر بر روی یک بایت انجام می‌شود. محتوای بایت‌ها تغییر می‌کند اما ترتیب بایت‌ها در ماتریس بدون تغییر باقی می‌ماند. در این مرحله، هر بایت به صورت مستقل تغییر می‌یابد.

عملیات subBytes شامل شانزده تغییر بایت به بایت مستقل است.

جدول ۷-۱ جدول جایگزینی (جعبه‌ی S) برای تغییر subBytes را نشان می‌دهد. قطعاً این تغییر، ویژگی آشفتگی را برآورد می‌کند. مثلاً دو بایت $5A_{16}$ و $5B_{16}$ که تنها در یک بیت باهم تفاوت دارند (بیت سمت راستشان)، به شکل BE_{16} و 39_{16} که در چهار بیت باهم تفاوت دارند، تغییر می‌یابند.

جدول ۷-۱: جدول تغییر subBytes

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	CB	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

InvSubBytes

InvSubBytes وارون subBytes بوده و با استفاده از جدول ۷-۲ تغییرات انجام می‌شود. به سادگی می‌-

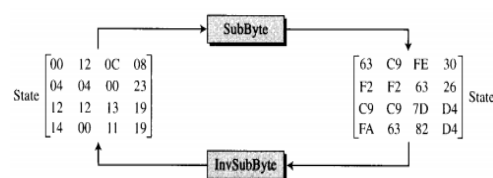
توانیم بررسی کنیم که دو تغییر وارون یکدیگرند.

جدول ۷-۲: جدول تغییر InvSubBytes

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	DB	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

مثال ۷-۲

شکل ۷-۷ نشان می‌دهد که چگونه یک وضعیت با استفاده از تغییر subBytes تغییر می‌یابد. همچنین این شکل نشان می‌دهد که تغییر InvSubBytes تغییرات بنیادین ایجاد می‌کند. توجه داشته باشید که اگر دو بایت، مقادیر یکسانی داشته باشند، تغییر آن‌ها نیز یکسان است. به عنوان مثال دو بایت ۰۴۱۶ و ۰۴۱۶ در عبارت سمت چپ به $F2_{16}$ و $F2_{16}$ در عبارت سمت راست و بالعکس تغییر می‌یابد. علت آن است که هر بایت از همان جدول استفاده می‌کند. در مقام مقایسه می‌بینیم که DES (فصل ۵) از هشت جعبه‌ی S مختلف استفاده می‌کرد.



شکل ۷-۷: تغییر subBytes برای مثال ۷-۲

تغییر با استفاده از میدان $GF(2^8)$

اگرچه می‌توانیم با استفاده از جدول ۷-۱ یا جدول ۷-۲ جایگزینی برای هر بایت پیدا کنیم، اما همان‌گونه که در شکل ۷-۸ نشان داده شده است، AES تغییر را با استفاده از میدان $GF(2^8)$ با چند جمله‌ای‌های کاهش‌ناپذیر $(X^8 + X^4 + X^3 + X + 1)$ به صورت جبری تعیین می‌کند.

تغییر **subBytes** روتین را که زیربایت^۱ نامیده می‌شود، شانزده بار تکرار می‌کند. **InvSubBytes** یک روتین مرسوم به وارون زیربایت را تکرار می‌کند. هر تکرار، یک بایت را تغییر می‌دهد. در روتین زیربایت، وارون ضربی بایت (به شکل یک رشته دودویی ۸ بیتی) با چندجمله‌ای کاهش-ناپذیر $(x^8 + x^4 + x^3 + x + 1)$ به عنوان پیمانه در $GF(2^8)$ یافت می‌شود. توجه داشته باشید که اگر بایت برابر 0016 باشد، وارون آن خودش است. سپس بیت وارون به شکل ماتریس ستونی با کم ارزش‌ترین بیت در بالا و پرارزش‌ترین بیت در پایین تفسیر می‌شود. این ماتریس ستونی در ماتریس مربع ثابت X ضرب می‌شود و حاصل که ماتریس ستونی است، با یک ماتریس ستونی ثابت Y جمع می‌شود و بدین ترتیب بایت جدید به دست می‌آید. توجه کنید که ضرب و جمع بیت‌ها در $GF(2^8)$ انجام می‌شود. وارون زیربایت، همین کار را به ترتیب وارون انجام می‌دهد.

پس از پیدا کردن وارون ضربی بایت، مراحل شبیه به رمز **Affin** است (به فصل ۳ رجوع کنید). در رمزنگاری، ضرب اول و جمع دوم انجام می‌شود؛ در رمزگشایی تفریق (جمع از طریق وارون) اول و تقسیم (ضرب از طریق وارون) دوم انجام می‌شود. به سادگی می‌توانیم ثابت کنیم که دو تغییر وارون یکدیگرند زیرا جمع و تفریق در $GF(2^8)$ در واقع عمل **XOR** است.

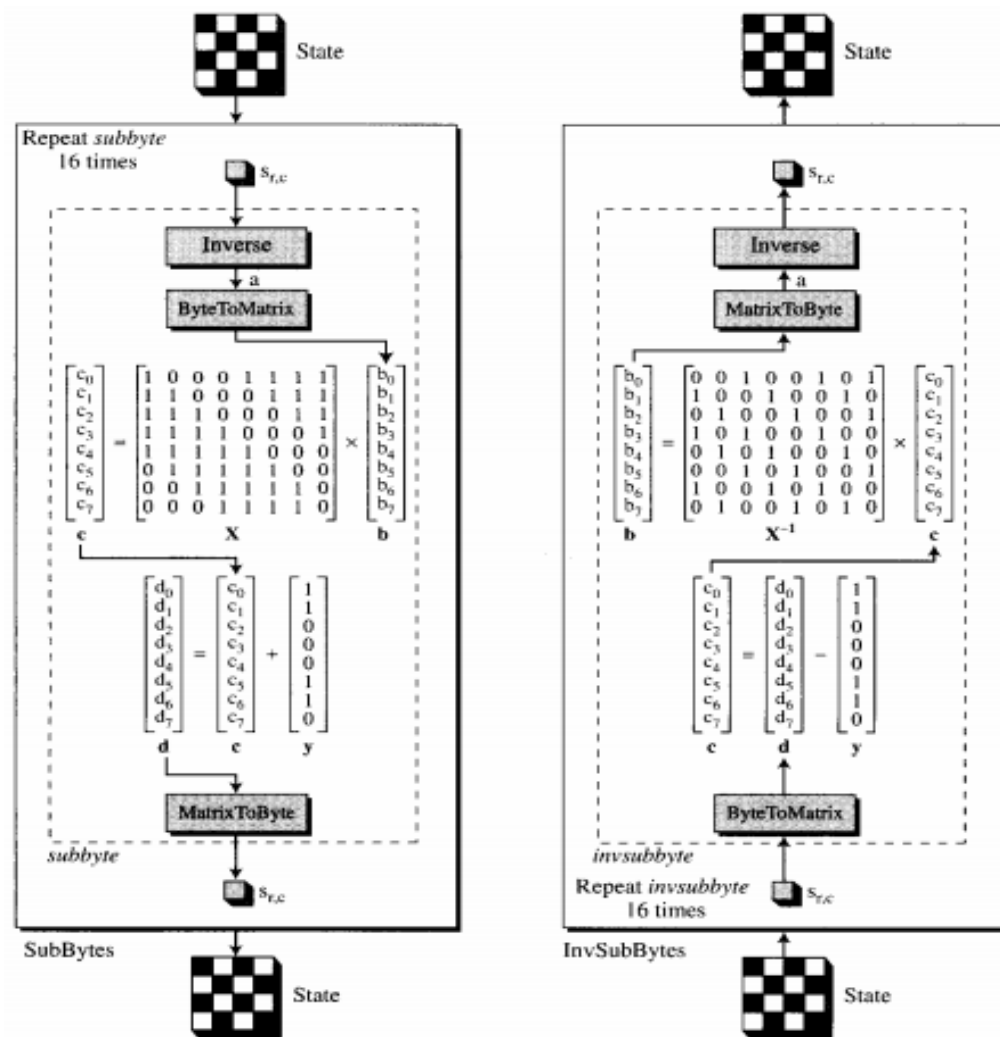
$$\text{Subbyte} \rightarrow d = X(s_{r,c})^{-1} \oplus y$$

$$\text{Invsubbyte} \rightarrow [X^{-1}(d \oplus y)]^{-1} = [X^{-1}(X(s_{r,c})^{-1} \oplus y \oplus y)]^{-1} = [(s_{r,c})^{-1}]^{-1} = s_{r,c}$$

تغییرات حاصل از **subBytes** و **InvSubBytes** وارون یکدیگرند.

=====

¹ subbyte



شکل ۷-۸: پردازش‌های subBytes و InvSubBytes

مثال ۷-۳

اگر موافق باشید بررسی کنیم که چگونه OC با استفاده از روتین زیر بایت به FE تبدیل می‌شود و به وسیله روتین وارون زیربایت دوباره به OC تبدیل می‌شود.

(۱) زیربایت

(a) وارون ضربی OC در میدان $GF(2^8)$ ، BO است بدین معنی است که $b = (10110000)$ است.

(b) ضرب ماتریس X در این ماتریس می‌شود $C = (10011101)$.

(c) نتیجه عمل XOR می‌شود: $d = (11111110)$ که در نمایش هگزادسیمال می‌شود FE.

(۲) وارون زیربایت

(a) نتیجه عمل XOR می‌شود: $C = (10011101)$.

(b) نتیجه ضرب در ماتریس X^{-1} می‌شود (11010000) یا BO.

(c) وارون ضربی BO می‌شود OC.

الگوریتم

اگرچه برای تأکید بر ماهیت جایگزینی (تغییر تکراری affine) ماتریس‌ها را نشان دادیم، اما الگوریتم لزوماً از ضرب و جمع ماتریس‌ها استفاده نمی‌کند؛ زیرا اکثر اعضای موجود در ماتریس، مربع ثابت و فقط ۰، ۱ هستند. مقدار ماتریس ستون ثابت و برابر ۶۳_{۱۶} است. می‌توانیم برای انجام subBytes، الگوریتم ساده‌تری بنویسیم. الگوریتم ۷-۱ روتین زیر بایت را ۱۶ بار، یعنی برای هر بایت در وضعیت یک‌بار فرامی‌خواند.

```

SubBytes (S)
{
  for (r = 0 to 3)
    for (c = 0 to 3)
      Sr,c = subbyte (Sr,c)
}

subbyte (byte)
{
  a ← byte-1 // Multiplicative inverse in GF(28) with inverse of 00 to be 00
  ByteToMatrix (a, b)
  for (i = 0 to 7)
  {
    ci ← bi ⊕ b(i+4) mod 8 ⊕ b(i+5) mod 8 ⊕ b(i+6) mod 8 ⊕ b(i+7) mod 8
    di ← ci ⊕ ByteToMatrix (0x63)
  }
  MatrixToByte (d, d)
  byte ← d
}

```

الگوریتم ۷-۱: شبه کد برای تغییر subBytes

روتین ByteToMatrix بایت را به شکل ماتریس ستونی ۸×۱ تبدیل می‌کند. روتین Matrix To

Byte ماتریس ستونی ۸×۱ را به یک بایت تبدیل می‌کند. بهبود این روتین‌ها و الگوریتم InvSubBytes

را به عنوان تمرین انجام دهید.

غیرخطی بودن

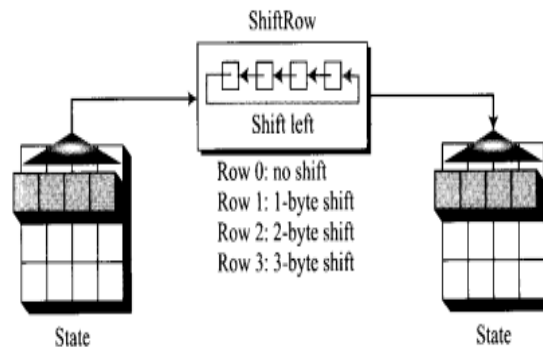
اگر چه ضرب و جمع ماتریس‌ها در روتین زیر بایت یک تغییر از نوع تکراری و خطی محسوب می‌شود، اما جایگزینی بایت به وسیله‌ی وارون ضربی آن در $GF(2^8)$ غیرخطی است. این مرحله باعث می‌شود که کل تغییر، غیرخطی شود.

جایگشت

تغییر دیگری که در هر گردش یافت می‌شد، تغییر مکان^۱ است که بایت‌ها را جابجا می‌کند. بر خلاف DES که جایگشت را در سطح بیت انجام می‌داد، تغییر مکان در AES در سطح بایت انجام می‌شود؛ ترتیب بیت‌ها در بایت تغییر نمی‌کند.

تغییر مکان سطرها^۲

در رمزنگاری، تغییری که ShiftRows نامیده می‌شود تغییر مکان به سمت چپ است. تعداد تغییر مکان‌ها به تعداد سطرهای (۰، ۱، ۲ یا ۳) ماتریس وضعیت بستگی دارد. این بدان معنی است که ردیف ۰ به هیچ‌وجه تغییر مکان نمی‌دهد و ردیف آخر، سه بایت تغییر مکان می‌دهد. شکل ۷-۹ تغییر مکان‌ها را نشان می‌دهد.



شکل ۷-۹: تغییر ShiftRows

توجه داشته باشید که تغییر مکان سطرها در یک زمان، بر روی تک سطر اعمال می‌شود.

وارون تغییر مکان سطر^۳

در رمزگشایی، تغییر مکان‌ها همان تعداد ردیف (۰، ۱، ۲ یا ۳) ماتریس وضعیت است. تغییرات ShiftRows و InvShiftRows وارون یکدیگرند.

الگوریتم

الگوریتم ۷-۲ برای تغییر مکان سطرها بسیار ساده است. با این وجود، برای تأکید بر این مطلب که در هر بار، تغییر فقط روی یک سطر انجام می‌شود؛ از یک روتین به نام تغییر مکان سطر استفاده می‌کنیم. این

=====

¹ Shifting

² ShiftRows

³ InvShiftRows

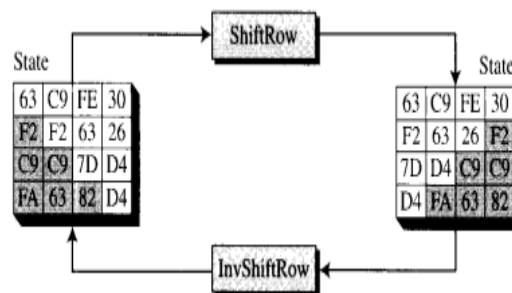
روتین، بایت موجود را فقط یک ردیف جابجا می‌کند. سه بار این روتین را فرا می‌خوانیم. روتین تغییر مکان سطر، نخست سطر را در ماتریس سطری موقت t کپی می‌کند و سپس سطر را جابجا می‌کند.

مثال ۷-۴

شکل ۷-۱۰ نشان می‌دهد که چگونه یک وضعیت با استفاده از تغییر مکان سطحی تغییر می‌کند. همچنین این شکل نشان می‌دهد که تغییر مکان سطری، عبارت اصلی را ایجاد می‌کند.

ShiftRows (S)	
{	
for ($r = 1$ to 3)	
shiftrow (s_r, r)	// s_r is the r th row
}	
shiftrow (row, n)	
// n is the number of bytes to be shifted	
{	
CopyRow (row, t)	// t is a temporary row
for ($c = 0$ to 3)	
row $_{(c-n) \bmod 4} \leftarrow t_c$	
}	

الگوریتم ۷-۲: شبه کد برای تغییر ShiftRows



شکل ۷-۱۰: تغییر ShiftRows در مثال ۷-۴

ترکیب

جایگزینی به وسیله **SubBytes** مقدار بایت را فقط بر اساس مقدار اصلی و ورودی از جدول تغییر می‌دهد. این روال شامل بیت‌های مجاور نمی‌شود. می‌توانیم ادعا کنیم که **SubBytes** تغییر درون بایتی است. جایگشت به وسیله تغییر **ShiftRows** بایت‌ها را بدون جابجا کردن بیت‌های داخل آن دگرگون می‌کند پس **ShiftRows** تغییر به شکل تبدیل بایت است. تغییر درون بایتی نیز لازم داریم تا بیت‌های داخل یک بایت را بر اساس بیت‌های موجود در بایت‌های مجاور عوض کند. برای تأمین آشفتگی در سطح بیت، باید بیت‌ها را باهم تلفیق کنیم.

تلفیق به وسیله‌ی دریافت چهار بایت و ترکیب آن‌ها با همدیگر و به دست آوردن چهار بایت جدید، محتوای بایت‌ها را عوض می‌کند. برای تضمین متفاوت بودن هر بایت جدید، (حتی اگر تمام چهار بایت یکسان باشند) روال و ترکیب مربوط به هر بایت را در یک مقدار ثابت متفاوت ضرب، سپس آن‌ها را باهم تلفیق می‌کند. با استفاده از ضرب ماتریس می‌توان عمل تلفیق را انجام داد. همان‌گونه که در فصل ۲ گفتیم، وقتی یک ماتریس مربع را در یک ماتریس ستون ضرب کنیم، نتیجه ماتریس ستونی جدید است. هر یک از اعضای ماتریس جدید، به هر چهار عضو ماتریس قبلی (قدیمی) که در مقادیر سطر در ماتریس ثابت ضرب شده‌اند بستگی دارد. شکل ۷-۱۱ این موضوع را نشان می‌دهد.

$$\begin{array}{l}
 ax + by + cz + dt \\
 ex + fy + gz + ht \\
 ix + jy + kz + lt \\
 mx + ny + oz + pt
 \end{array}
 \begin{array}{c}
 \left[\begin{array}{c} \boxed{} \\ \boxed{} \\ \boxed{} \\ \boxed{} \end{array} \right]
 \end{array}
 =
 \begin{bmatrix}
 a & b & c & d \\
 e & f & g & h \\
 i & j & k & l \\
 m & n & o & p
 \end{bmatrix}
 \times
 \begin{bmatrix}
 x \\
 y \\
 z \\
 t
 \end{bmatrix}$$

New matrix Constant matrix Old matrix

شکل ۷-۱۱: ترکیب بیت‌ها با استفاده از ضرب ماتریس

استاندارد AES برای دستیابی به این هدف، تغییری به نام Mix Columns را در نظر گرفت. همچنین تغییر وارون به نام InvMixColumns نیز وجود دارد. شکل ۷-۱۲ ماتریس‌های ثابت استفاده شده برای این تغییرها را نشان می‌دهد. وقتی که اعضای این ماتریس‌ها به شکل کلمات ۸ بیتی (یا چندجمله‌ای) با ضرایبی در $GF(2^8)$ تفسیر شوند، این دو ماتریس وارون یکدیگرند.

$$\begin{bmatrix}
 02 & 03 & 01 & 01 \\
 01 & 02 & 03 & 01 \\
 01 & 01 & 02 & 03 \\
 03 & 01 & 01 & 02
 \end{bmatrix}
 \xleftrightarrow{\text{Inverse}}
 \begin{bmatrix}
 0E & 0B & 0D & 09 \\
 09 & 0E & 0B & 0D \\
 0D & 09 & 0E & 0B \\
 0B & 0D & 09 & 0E
 \end{bmatrix}$$

C C⁻¹

شکل ۷-۱۲: ماتریس‌های ثابت به کاررفته در MixColumns و InvMixColumns

MixColumns

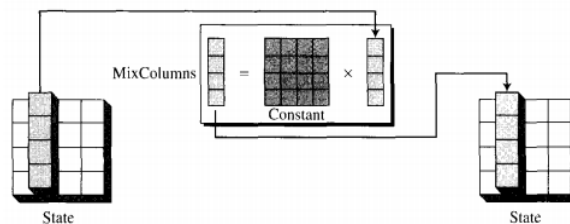
تغییر MixColumns در سطح ستون عمل می‌کند و هر ستون وضعیت را به یک ستون جدید تغییر می‌دهد؛ در واقع این تغییر، ضرب ماتریسی یک ستون ماتریس وضعیت در یک ماتریس مربع ثابت است. بایت‌ها در ستون وضعیت و ماتریس ثابت، به شکل کلمات ۸ بیتی (یا چندجمله‌ای) با ضرایبی در $GF(2^8)$ تفسیر می‌شود. ضرب بیت‌ها در میدان $GF(2^8)$ به پیمانه (10001101) یا $(x^8 + x^4 + x^3 + x + 1)$ انجام می‌شود.

جمع، مانند انجام عمل XOR کلمات ۸ بیتی است. شکل ۷-۱۳ تغییرات حاصل از MixColumns را نشان می‌دهد.

InvMixColumns

تغییر حاصل از InvMixColumns اساساً مانند تغییر MixColumns است. اگر دو ماتریس ثابت وارون یکدیگر باشد؛ به سادگی می‌توان ثابت کرد که دو تغییر، وارون یکدیگرند.

تغییرات MixColumns و InvMixColumns وارون یکدیگرند.



شکل ۷-۱۳: تغییر در MixColumns

الگوریتم

الگوریتم ۷-۳ تغییر MixColumns را نشان می‌دهد.

```

MixColumns (S)
{
    for (c = 0 to 3)
        mixcolumn (sc)
    }
mixcolumn (col)
{
    CopyColumn (col, t)           // t is a temporary column
    col0 ← (0x02) • t0 ⊕ (0x03) • t1 ⊕ t2 ⊕ t3
    col1 ← t0 ⊕ (0x02) • t1 ⊕ (0x03) • t2 ⊕ t3
    col2 ← t0 ⊕ t1 ⊕ (0x02) • t2 ⊕ (0x03) • t3
    col3 ← (0x03) • t0 ⊕ t1 ⊕ t2 ⊕ (0x02) • t3
}

```

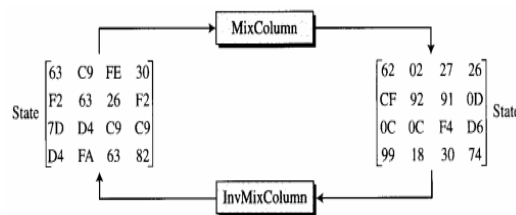
الگوریتم ۷-۳: شبه کد تغییر MixColumns

الگوریتم‌ها برای MixColumns و InvMixColumns شامل ضرب و جمع در میدان $GF(2^8)$ می‌باشد. همان‌گونه که در فصل ۴ دیدیم در این میدان، الگوریتم ساده و کارآمدی برای ضرب و جمع وجود دارد؛ با این وجود، برای نشان دادن ماهیت الگوریتم (تغییر یک ستون در یک زمان) از روالی به نام «MixColumns» استفاده می‌کنیم و الگوریتم آن را چهار بار فرامی‌خواند. این روال سطرهای ماتریس ثابت را در یک ستون از ماتریس وضعیت ضرب می‌کند. در الگوریتم فوق علامت (●) به کار رفته در روال ترکیب ستون، ضرب در میدان $GF(2^8)$ می‌باشد. همان‌گونه که در فصل ۴ گفته شد، می‌توان روال ساده‌تری

را جایگزین این روال کرد. پیاده‌سازی روال **InvMixColumns** را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

مثال ۷-۵

شکل ۷-۱۴ نشان می‌دهد که چگونه ماتریس وضعیت با استفاده از تغییر **MixColumns** تغییر می‌کند. هم چنین این شکل نشان می‌دهد که تغییر **InvMixColumns** ماتریس اصلی را بازسازی می‌کند.



شکل ۷-۱۴ تغییر **MixColumns** در مثال ۷-۵

توجه کنید که بایت‌های مساوی در عبارت قبلی (قدیمی) به هیچ‌وجه در عبارت جدید مساوی نیستند. به عنوان مثال، دو بایت F_2 در سطر دوم به CF و OD تغییر می‌یابد.

افزودن کلید

شاید مهم‌ترین تغییر، تغییری است که کلید رمز را در بر می‌گیرد. تمام تغییرات پیشین از الگوریتم‌های شناخته شده‌ای که وارون‌پذیرند استفاده می‌کنند. اگر در پایان هر گردش، کلید رمز به عبارت اضافه نشود، برای رقیب بسیار ساده است که با توجه به متن رمز، متن عادی را به دست آورد. در این حالت تنها سر بین آلیس و بابت کلید رمز است.

AES از روالی به نام گسترش کلید (که بعداً در این فصل توضیح داده خواهد شد) استفاده می‌کند. گسترش کلید، کلیدهای مرحله N_r+1 را از کلید رمز تولید می‌کند. طول کلید در هر گردش ۱۲۸ بیت است و به عنوان چهار کلمه‌ی ۳۲ بیتی در نظر گرفته می‌شود. به منظور افزودن کلید به ماتریس وضعیت، هر کلمه یک ماتریس ستونی در نظر گرفته می‌شود.

AddRoundkey

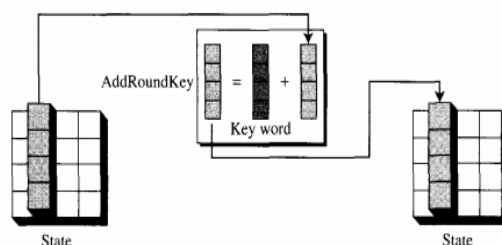
AddRoundkeys نیز در یک زمان یک ستون به سمت جلو می‌رود و از این جهت شبیه به **MixColumns** است. **MixColumns** یک ماتریس مربع ثابت را در هر ستون وضعیت ضرب می‌کند؛ **AddRoundkey** کلمه کلید گردش را با هر ستون ماتریس وضعیت جمع می‌کند. عمل در **MixColumns**، ضرب ماتریس و در **AddRoundkeys** جمع ماتریس‌هاست. از آنجا که در این میدان

جمع و تفریق یکسان هستند، تغییرات **AddRoundkeys** خود وارون هستند. شکل ۷-۱۵ تغییر **AddRoundkeys** را نشان می‌دهد.

تغییر **AddRoundkeys** وارون خودش است.

الگوریتم

تغییر **AddRoundkeys** را می‌توان با XOR هر ستون از ماتریس وضعیت با کلمه کلید مربوطه در نظر گرفت. توضیح خواهیم داد که چگونه کلید رمز، به مجموعه‌ای از کلمات کلیدی گسترش داده می‌شود. در حال حاضر می‌توانیم این تغییر را در الگوریتم ۷-۴ مشاهده نمایید. توجه داشته باشید که S_c و $w_{round+4c}$ ماتریس‌های ستونی 4×1 هستند.



شکل ۷-۱۵: تغییر **AddRoundkeys**

```

AddRoundKey(S)
{
  for (c = 0 to 3)
     $s_c \leftarrow s_c \oplus w_{round+4c}$ 
}

```

الگوریتم ۷-۴: شبه کد برای تغییر **AddRoundkey**

با این وجود شایان توجه است که علامت \oplus در اینجا به معنی XOR دو ماتریس ستونی که هر کدام ۴ بایت دارند می‌باشد.

۷-۳ گسترش کلید

برای تولید کلید مورد استفاده در هر گردش، AES از یک روال برای گسترش کلید استفاده می‌کند. اگر تعداد گردش N_r باشد، روتین گسترش کلید فقط از کلید رمز ۱۲۸ بیتی، N_r+1 کلید گردش ۱۲۸ بیتی ایجاد می‌کند. اولین کلید گردش برای تغییر پیش‌گردش **AddRoundkey** و سایر کلیدهای گردش برای آخرین تغییر (**AddRoundkey**) در پایان هر گردش به کار می‌روند.

روال گسترش کلید، کلیدهای هر گردش را کلمه به کلمه تولید می‌کند، طوری که a یک کلمه چهار بایتی است. این روال $\mathcal{E} \times (N_r + 1)$ کلمه تولید می‌کند که به صورت زیر نشان داده می‌شوند:

$$W, W_1, W_2, \dots, W_{\mathcal{E}(N_r+1)-1}$$

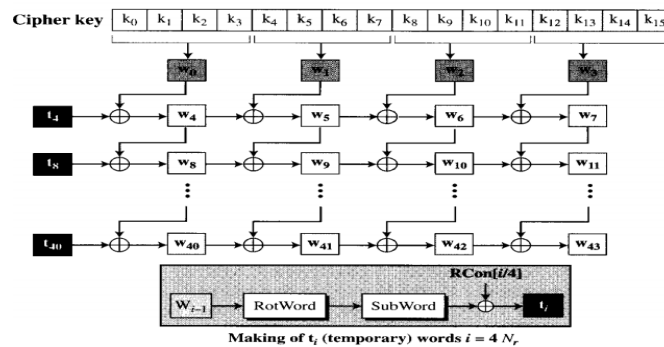
به عبارت دیگر، در نسخه AES-۱۲۸ (ده گردش) ۴۴ کلمه در نسخه AES-۱۹۲ (۱۲ گردش) ۵۲ کلمه و در نسخه AES-۲۵۶ (با ۱۴ گردش) ۶۰ کلمه وجود دارد. هر کلید گردش از چهار کلمه تشکیل شده است. جدول ۳-۷ رابطه‌ی بین گردش‌ها و کلمات را نشان می‌دهند.

جدول ۳-۷: کلمات تولیدی برای هر گردش

Round	Words			
Pre-round	w_0	w_1	w_2	w_3
1	w_4	w_5	w_6	w_7
2	w_8	w_9	w_{10}	w_{11}
...	...			
N_r	w_{4N_r}	w_{4N_r+1}	w_{4N_r+2}	w_{4N_r+3}

گسترش کلید در AES-۱۲۸

اجازه دهید ایجاد کلمات برای نسخه‌ی AES-۱۲۸ را نشان دهیم. این روال برای دو نسخه دیگر هم با کمی تغییر شبیه همین است. شکل ۱۶-۷ نشان می‌دهد که چطور ۴۴ کلمه از کلید اصلی ساخته می‌شود.



شکل ۱۶-۷: گسترش کلید در AES

این روال به شرح زیر می‌باشد:

- (۱) چهار کلمه اول (w, w_1, w_2, w_3) از کلمه‌ی رمز ساخته می‌شود. کلید رمز به عنوان آرایه‌ی ۱۶ بایتی (K_0 تا K_{15}) در نظر گرفته می‌شود. چهار بایت اول (K_0 تا K_3) می‌شود w ، چهار بایت بعدی (K_4 تا K_7) می‌شود w_1 و الی آخر؛ به عبارت دیگر، توالی کلمات در گروه، کلید رمز را عیناً بازسازی می‌کند (کپی می‌کند).

- (۲) بقیه‌ی کلمات (w_i for $i=4$ تا 43) به شرح زیر ساخته می‌شود:

(a) اگر $w_i = w_{i-1} \oplus w_{i-4}$ ، $(i \bmod 4) \neq 0$ باشد و به استناد شکل ۷-۱۶ این بدان معناست که هر کلمه از یک بیت در سمت چپ و یک بیت از بالا تشکیل شده است.

(b) اگر $(i \bmod 4) = 0$ ، $w_i = t \oplus w_{i-4}$ باشد؛ در کلمه‌ی موقت t ، نتیجه XOR دو رول Sub

Word و Rot Word بر روی w_{i-1} است. نتیجه با ثابت‌های گردش برابر Rcon می‌شود. به عبارت دیگر داریم:

$$T = \text{Sub Word}(\text{Rot Word}(w_{i-1})) \oplus \text{Rcon}_{i/4}$$

Rot Word

روتین Rot Word (کلمه چرخش) شبیه به تغییر shiftRows است، اما فقط بر روی یک سطر اعمال می‌شود. این روال، یک کلمه را به عنوان آرایه‌ی چهار بایتی گرفته سپس هر بایت را به سمت چپ حرکت می‌دهد.

SubWord

روال 'SubWord' شبیه به تغییر SubBytes است، اما فقط بر چهار بایت اعمال می‌شود. این روال، هر بایت در کلمه را گرفته و بایت دیگری را جایگزین آن می‌کند.

اعداد ثابت در گردش‌ها

هر عدد ثابت یک مقدار چهار بایتی است که سه بایت سمت راستش همیشه صفر است. جدول ۷-۴ مقادیر لازم برای نسخه AES-۱۲۸ (با ده گردش) را نشان می‌دهد.

Round	Constant (RCon)	Round	Constant (RCon)
1	(01 00 00 00) ₁₆	6	(20 00 00 00) ₁₆
2	(02 00 00 00) ₁₆	7	(40 00 00 00) ₁₆
3	(04 00 00 00) ₁₆	8	(80 00 00 00) ₁₆
4	(08 00 00 00) ₁₆	9	(1B 00 00 00) ₁₆
5	(10 00 00 00) ₁₆	10	(36 00 00 00) ₁₆

جدول ۷-۴: ثابت‌های RCon

روتین گسترش کلید می‌تواند هنگام محاسبه کلمه یا از جدول بالا استفاده کند یا همان‌گونه که در زیر نشان داده شده است برای محاسبه‌ی پویای بایت سمت چپ، از میدان $GF(2^8)$ استفاده کند (پریم، چندجمله‌ای کاهش‌ناپذیر است).

RC ₁	→ x^{1-1}	= x^0	mod prime	= 1	→ 00000001	→ 01 ₁₆
RC ₂	→ x^{2-1}	= x^1	mod prime	= x	→ 00000010	→ 02 ₁₆
RC ₃	→ x^{3-1}	= x^2	mod prime	= x^2	→ 00000100	→ 04 ₁₆
RC ₄	→ x^{4-1}	= x^3	mod prime	= x^3	→ 00001000	→ 08 ₁₆
RC ₅	→ x^{5-1}	= x^4	mod prime	= x^4	→ 00010000	→ 10 ₁₆
RC ₆	→ x^{6-1}	= x^5	mod prime	= x^5	→ 00100000	→ 20 ₁₆
RC ₇	→ x^{7-1}	= x^6	mod prime	= x^6	→ 01000000	→ 40 ₁₆
RC ₈	→ x^{8-1}	= x^7	mod prime	= x^7	→ 10000000	→ 80 ₁₆
RC ₉	→ x^{9-1}	= x^8	mod prime	= $x^4 + x^3 + x + 1$	→ 00011011	→ 1B ₁₆
RC ₁₀	→ x^{10-1}	= x^9	mod prime	= $x^5 + x^4 + x^2 + x$	→ 00110110	→ 36 ₁₆

=====

^۱ کلمه جایگزین

بایت سمت چپ که RC_i نامیده می‌شود، در واقع x^{i-1} است، به قسمی که i ، تعداد گردش می‌باشد. AES از چند جمله‌ای کاهش ناپذیر $(x^8 + x^4 + x^3 + x + 1)$ استفاده می‌کند.

الگوریتم

الگوریتم ۵-۷ الگوریتم ساده‌ای برای روال گسترش کلید می‌باشد (نسخه ۱۲۸-AES).

```

KeyExpansion ([key0 to key15], [w0 to w43])
{
  for (i = 0 to 3)
    wi ← key4i + key4i+1 + key4i+2 + key4i+3

  for (i = 4 to 43)
  {
    if (i mod 4 ≠ 0) wi ← wi-1 + wi-4
    else
    {
      t ← SubWord (RotWord (wi-1)) ⊕ RConi/4 // t is a temporary word
      wi ← t + wi-4
    }
  }
}

```

الگوریتم ۵-۷: شبه کد برای گسترش کلید در AES-128

مثال ۶-۷

جدول ۵-۷ نشان می‌دهد که با فرض این که کلید رمز ۱۲۸ بیتی توافق شده بین آلیس و باب ۸۷)۱۶

۵۴ AA ۱۳ ۰۰ ۱۲ E۲ ۳۱ ۵۶ ۷۵ ۳۴ B۳ A۲ ۷۵ (۲۴ باشد، چگونه کلید هر گردش محاسبه می‌شود.

جدول ۷-۵: مثال گسترش کلید

Round	Values of t 's	First word in the round	Second word in the round	Third word in the round	Fourth word in the round
—		$w_{00} = 2475A2B3$	$w_{01} = 34755688$	$w_{02} = 31E21200$	$w_{03} = 13AA5487$
1	AD20177D	$w_{04} = 8955B5CE$	$w_{05} = BD20E346$	$w_{06} = 8CC2F146$	$w_{07} = 9F68A5C1$
2	470678DB	$w_{08} = CE53CD15$	$w_{09} = 73732E53$	$w_{10} = FFB1DF15$	$w_{11} = 60D97AD4$
3	31DA48D0	$w_{12} = FF8985C5$	$w_{13} = 8CFAAB96$	$w_{14} = 734B7483$	$w_{15} = 2475A2B3$
4	47AB5B7D	$w_{16} = B822deb8$	$w_{17} = 34D8752E$	$w_{18} = 479301AD$	$w_{19} = 54010FFA$
5	6C762D20	$w_{20} = D454F398$	$w_{21} = E08C86B6$	$w_{22} = A71F871B$	$w_{23} = F31E88E1$
6	52C4F80D	$w_{24} = 86900B95$	$w_{25} = 661C8D23$	$w_{26} = C1030A38$	$w_{27} = 321D82D9$
7	E4133523	$w_{28} = 62833EB6$	$w_{29} = 049FB395$	$w_{30} = C59CB9AD$	$w_{31} = F7813B74$
8	8CE29268	$w_{32} = EE61ACDE$	$w_{33} = EAFE1F4B$	$w_{34} = 2F62A6E6$	$w_{35} = D8E39D92$
9	0A5E4F61	$w_{36} = E43FE3BF$	$w_{37} = 0EC1FCF4$	$w_{38} = 21A35A12$	$w_{39} = F940C780$
10	3FC6CD99	$w_{40} = DBF92E26$	$w_{41} = D538D2D2$	$w_{42} = F49B88C0$	$w_{43} = 0DDB4F40$

در هر گردش، محاسبه‌ی سه کلمه‌ی آخر بسیار ساده است. برای محاسبه‌ی کلمه‌ی اول باید نخست مقدار کلمه موقت (t) را حساب کنیم. مثلاً t برای گردش اول به صورت زیر محاسبه می‌شود:

Rot word ($12AA0587$)= $AA058713 \rightarrow$ **Sub word** ($AA058713$)= $AC20177D$

$t=AC20177D \oplus RCon_1=AC20177D \oplus 01000000=AD20177D$

مثال ۷-۷

هر کلید گردش در AES به کلید گردش پیشین بستگی دارد؛ با این وجود، به علت تغییرات حاصل از SubWord این وابستگی غیرخطی است. همچنین افزودن ثابت‌هایی در هر گردش تضمین می‌کند که کلید هر گردش با کلید قبلی متفاوت است.

مثال ۷-۸

از دو کلید رمزی که فقط در یک بیت با هم تفاوت دارند می‌توان دو مجموعه کلید گردش تولید کرد.

Cipher key₁: $12054A2A1 \ 2331A4A3 \ B2CCAA34 \ C2BB7723$

Cipher key₂: $12054A2A1 \ 2331A4A3 \ B2CCAB34 \ C2BB7723$

همان‌گونه که جدول ۷-۶ نشان می‌دهد، تفاوت‌های چشمگیری بین دو کلید گردش متناظر وجود

دارد (R یعنی گردش و B.D یعنی تفاوت بیت).

جدول ۷-۶: مقایسه دو مجموعه کلید گردش‌ها

R.	Round keys for set 1	Round keys for set 2	B. D.
—	1245A2A1 2331A4A3 B2CCAA34 C2BB7723	1245A2A1 2331A4A3 B2CCAB34 C2BB7723	01
1	F9B08484 DA812027 684D8A13 AAF6FD30	F9B08484 DA812027 684D8B13 AAF6FC30	02
2	B9E48028 6365A00F 0B282A1C A1DED72C	B9008028 6381A00F 0BCC2B1C A13AD72C	17
3	A0EAF11A C38F5115 C8A77B09 6979AC25	3D0EF11A 5E8F5115 55437A09 F479AD25	30
4	1E7BCEE3 DDF49FF6 1553E4FF 7C2A48DA	839BCEA5 DD149FB0 8857E5B9 7C2E489C	31
5	EB2999F3 36DD0605 238EE2FA 5FA4AA20	A2C910B5 7FDD8F05 F78A6ABC 8BA42220	34
6	82852E3C B4582839 97D6CAC3 C87260E3	CB5AA788 B487288D 430D4231 C8A96011	56
7	82553FD4 360D17ED A1DBDD2E 69A9BDCD	588A2560 EC0D0DED AF004FDC 67A92FCD	50
8	D12F822D E72295C0 46F948EE 2F50F523	0B9F98E5 E7929508 4892DAD4 2F3BF519	44
9	99C9A438 7EEB31F8 38127916 17428C35	F2794CF0 15EBD9F8 5D79032C 7242F635	51
10	83AD32C8 FD460330 C5547A26 D216F613	E83BDAB0 FDD00348 A0A90064 D2EBF651	52

مثال ۷-۹

مفهوم کلیدهای ضعیف، همان‌گونه که برای DES در فصل ۶ توضیح دادیم، برای AES به کار نمی‌رود. فرض کنید تمام بیت‌ها در کلید رمز ۰ باشند. در شکل زیر کلمات تولیدی در چند گردش نشان داده شده‌اند:

Pre-round:	00000000	00000000	00000000	00000000
Round 01:	62636363	62636363	62636363	62636363
Round 02:	9B9898C9	F9FBFBAA	9B9898C9	F9FBFBAA
Round 03:	90973450	696CCFFA	F2F45733	0B0FAC99
...
Round 10:	B4EF5BCB	3E92E211	23E951CF	6F8F188E

تمام کلمات در پیش‌گردش و گردش اول یکسان‌اند. در گردش دوم، کلمه‌ی اول با کلمه‌ی سوم، کلمه‌ی دوم با کلمه‌ی چهارم تطابق دارد؛ با این وجود، پس از گردش دوم، این الگو ناپدید می‌شود و هر کلمه متفاوت است.

گسترش کلید در AES-۱۲۸ و AES-۲۵۶

الگوریتم‌های گسترش کلید در نسخه‌های AES-۱۹۲ و AES-۲۵۶ بسیار شبیه به الگوریتم گسترش کلید در AES-۱۲۸ است؛ اما با تفاوت‌های زیر:

(۱) در AES-۱۹۲، کلمات به جای گروه‌های چهارتایی در گروه‌های شش‌تایی تولید می‌شوند.

(a) کلید رمز، شش کلمه اول را تولید می‌کند (w_0 تا w_5).

(b) اگر $i \bmod 6 \neq 0$ است $w_i \leftarrow w_{i-1} + w_{i-6}$ در غیر این صورت $w_i \leftarrow t + w_{i-6}$.

(۲) در AES-۲۵۶ کلمات به جای گروه‌های چهارتایی، در گروه‌های هشت‌تایی ایجاد می‌شود.

(a) کلید رمز، هشت کلمه اول را ایجاد می‌کند (w_0 تا w_7).

(b) اگر $i \bmod 8 \neq 0$ باشد $w_i \leftarrow w_{i-1} + w_{i-8}$ است در غیر این صورت $w_i \leftarrow t + w_{i-8}$.

(c) اگر $i \bmod 4 \neq 0$ باشد اما $i \bmod 8 \neq 0$ نباشد، پس $w_i = \text{Sub Word}(w_{i-1}) + w_{i-8}$ است.

تجزیه و تحلیل گسترش کلید

سازوکار گسترش کلید در AES به منظور تأمین چند ویژگی برای خشی کردن تحلیل رمز طراحی شده است.

(۱) حتی اگر رمز شکن قسمتی از کلید رمز یا مقادیر کلمات در برخی از کلیدهای گردش را بداند، باید پیش از یافتن تمام کلیدهای گردش، قسمت‌های دیگر کلید رمز را پیدا کند. این امر به خاطر غیرخطی بودن ناشی از تغییر Sub Word در روال گسترش کلید است.

(۲) مهم نیست که دو کلید رمز متفاوت چقدر شبیه یکدیگر باشند؛ آن‌ها دو گسترش را ایجاد می‌کنند که دست کم در دو گردش با هم متفاوت‌اند.

(۳) هر بیت از کلید رمز در چندین گردش پخش می‌شود. مثلاً تغییر فقط یک بیت در کلید رمز، چندین بیت را در چند گردش تغییر خواهد داد.

(۴) استفاده از ثابت RCon، هرگونه تقارنی که ممکن است به وسیله‌ی سایر تغییرات ایجاد شده باشد را از بین می‌برد.

(۵) در AES، برخلاف DES، هیچ کلید ضعیف مهمی وجود ندارد.

(۶) روال گسترش کلید را می‌توان به سادگی روی تمام سیستم‌عامل‌ها اجرا کرد.

(۷) روال گسترش کلید را می‌توان بدون ذخیره کردن حتی یک جدول اجرا کرد. تمام محاسبات را می‌توان با استفاده از میدان $GF(2^8)$ و $GF(2)$ انجام داد.

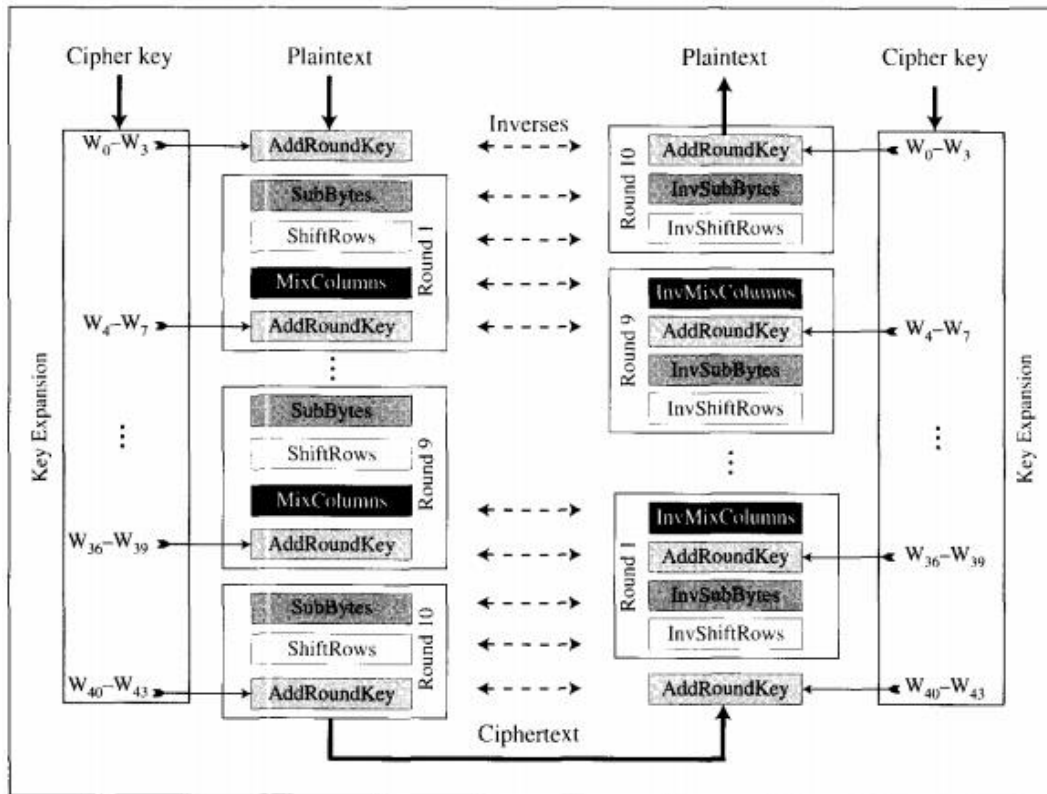
۷-۴ رمز نویسی

اجازه دهید ببینیم که ASE چطور از چهار نوع تغییر برای رمزنگاری و رمزگشایی استفاده می‌کند. استاندارد، الگوریتم رمزنگاری را رمز نویسی و الگوریتم رمزگشایی را رمز نویسی وارون نامیده است.

همان‌گونه که پیشتر گفتیم، AES رمز غیر Feistel است؛ بدین معنی که هر تغییر یا هر گروه از تغییرات باید وارون‌پذیر باشد. علاوه بر این، رمز نویسی و رمز نویسی وارون باید این اعمال را طوری به کار ببرند که اثر یکدیگر را از بین ببرند. کلیدهای گردش نیز باید به ترتیب وارون مورد استفاده قرار گیرند. دو طراحی مختلف برای استفاده در اجرای متفاوت مشخص شده‌اند. در مورد هر دو طراحی برای AES-۱۲۸ صحبت می‌کنیم. طراحی‌های نسخه‌های دیگر نیز به همین گونه است.

طرح اصلی

در طرح اصلی، ترتیب تغییرات هر گردش در رمزنویسی و رمزنویسی وارون یکسان نیستند. شکل ۷-۱۷ این طرح را نشان می‌دهد.



شکل ۷-۱۷: رمز و رمز وارون در طراحی اصلی

نخست، ترتیب SubBytes و ShiftRows در رمز وارون تغییر می‌کند. دوم، ترتیب MixColumns و AddRoundKey در رمز وارون عوض می‌شود. برای این که هر تغییر در رمز با وارون خود در رمز وارون در یک ردیف باشد، این تفاوت در ترتیب اجرا لازم است. در نتیجه الگوریتم رمزگشایی در قالب یک واحد کلی، وارون الگوریتم رمزنگاری است. ما فقط سه گردش را نشان داده‌ایم، اما سایر گردش‌ها نیز همین گونه است. توجه داشته باشید که کلید گردش به ترتیب وارون مورد استفاده قرار می‌گیرد. همچنین شایان یادآوری است که الگوریتم‌های رمزنگاری و رمزگشایی در طراحی اصلی شبیه هم نیستند.

الگوریتم

در الگوریتم ۶-۷ کد برای نسخه‌ی AES-۱۲۸ از این طراحی نشان داده شده است. رمز وارون را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

```

Cipher (InBlock [16], OutBlock[16], w[0 ... 43])
{
    BlockToState (InBlock, S)

    S ← AddRoundKey (S, w[0...3])
    for (round = 1 to 10)
    {
        S ← SubBytes (S)
        S ← ShiftRows (S)
        if (round ≠ 10) S ← MixColumns (S)
        S ← AddRoundKey (S, w[4 × round, 4 × round + 3])
    }

    StateToBlock (S, OutBlock);
}

```

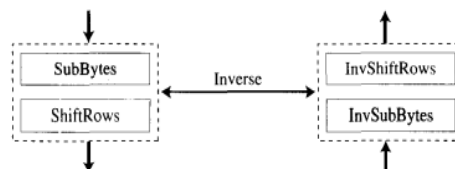
الگوریتم ۷-۶: شبه کد برای رمزنویسی در طراحی اصلی

طرح‌های جایگزین

برای آن دسته از برنامه‌های کاربردی که الگوریتم‌های مشابهی را برای رمزنگاری و رمزگشایی ترجیح می‌دهند، رمز وارون متفاوتی در نظر گرفته شده است. در این نسخه، تغییرات در رمز وارون دوباره مرتب شد تا ترتیب تغییرات در رمز و رمز وارون یکسان شود. در این طراحی به جای این که وارون‌پذیر بودن برای هر تغییر در نظر گرفته شود، برای یک جفت از تغییرات در نظر گرفته شده است.

زوج SubBytes / ShiftRows

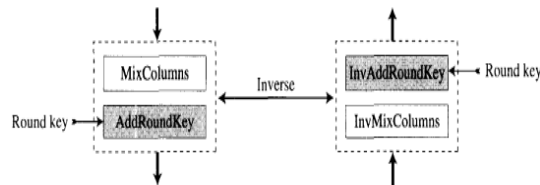
SubBytes محتوای هر بایت را بدون تغییر ترتیب بایت‌ها در وضعیت عوض می‌کند، ShiftRows ترتیب بایت‌ها را در وضعیت بدون تغییر محتوای آن‌ها عوض می‌کند. این بدان معنی است که می‌توانیم ترتیب این دو تغییر را در رمز وارون عوض کنیم، بدون این که بر وارون‌پذیری کل الگوریتم تأثیری داشته باشد. شکل ۷-۱۸ این ایده را نشان می‌دهد. توجه داشته باشید که ترکیب این دو تغییر در رمزنویسی و رمز وارون، وارون یکدیگرند.



شکل ۷-۱۸: وارون‌پذیری ترکیبات SubBytes و ShiftRows

زوج MixColumns / AddRoundKey

این دو تغییر گنجانده شده، ماهیت متفاوتی دارند؛ با این وجود، اگر ماتریس کلید را در وارون ماتریس ثابت به کار رفته در تغییر MixColumns ضرب کنیم، این جفت می‌توانند وارون یکدیگر باشند. حاصل را InvAddRoundKey می‌نامیم. شکل ۷-۱۹ پیکربندی جدید را نشان می‌دهد.



شکل ۷-۱۹: وارون‌پذیری ترکیب MixColumns و AddRoundKey

می‌توانیم ثابت کنیم که دو ترکیب وارون یکدیگرند. در رمز، عبارت ورودی را S و عبارت خروجی را T می‌نامیم. در رمز وارون، حاصل ورودی ترکیب ناشی از T است. در زیر نشان می‌دهیم که عبارت خروجی نیز S است. توجه داشته باشید که تغییر MixColumns در واقع ضرب ماتریس C (ماتریس ثابت در عبارت) است.

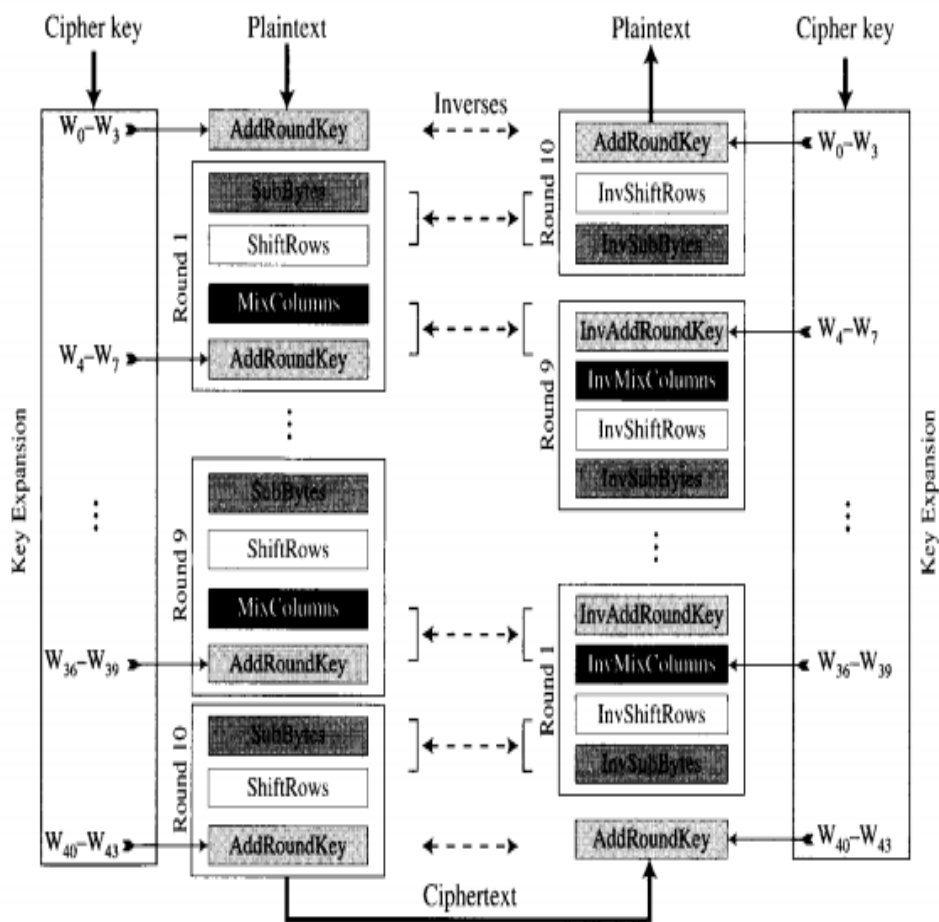
$$\text{رمز: } T = CS \oplus K$$

$$\text{رمز وارون: } C^{-1}T \oplus C^{-1}K = C^{-1}(CS \oplus K) \oplus C^{-1}K = C^{-1}CS \oplus C^{-1}K \oplus C^{-1}K = S$$

اکنون می‌توانیم رمز و رمز وارون را برای طراحی‌های جایگزین نشان دهیم. توجه داشته باشید که باید از دو تغییر AddRoundKey در رمزگشایی استفاده کنیم؛ به عبارت دیگر، همان‌گونه که در شکل ۷-۲۰ نشان داده شده است ۹ تغییر InvAddRoundKey و دو تغییر AddRoundKey داریم.

تغییر الگوریتم گسترش کلید

به جای استفاده از تغییر InvAddRoundKey در رمز وارون، می‌توان برای ایجاد مجموعه‌ی متفاوتی از کلید گردش‌ها در رمز وارون، الگوریتم گسترش کلید را تغییر داد.



شکل ۷-۲۰: رمز و رمز وارون در طراحی جایگزین

با این وجود، توجه کنید که کلید گردش برای عمل پیش‌گردش و گردش آخر نباید تغییر کند. کلیدهای گردش‌ها در گردش‌های ۱ تا ۹ باید در ماتریس ثابت ضرب شود. پیاده‌سازی این الگوریتم را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

۷-۵ مثال‌ها

در این بخش، برای تأکید برخی از نکات مطرح شده در بخش‌های پیش، مثال‌هایی از رمزنگاری و رمزگشایی و تولید کلید ارائه شده است.

مثال ۷-۱۰

مورد زیر، بلوک متن رمز تولید شده از بلوک متن عادی و با استفاده از کلید رمز تصادفی را نشان می‌دهد.

۱۹ ۱۹ ۲۳ ۰۸ ۱۱ ۱۳ ۰۰ C ۰۰ ۱۲ ۰۴ ۱۲ ۱۴ ۱۲ ۰۴ ۰۰ :متن اولیه

۸۷ ۵۴ AA ۱۳ ۰۰ ۱۲ E۲ ۳۱ ۸۸ ۵۶ ۷۵ ۳۴ B۳ ۲۳ ۷۵ A۲ :کلید رمز

متن رمز: BC ۰۲ ۸B D۳ E۰ E۳ B۱ ۹۵ ۸۸ ۰D ۶D FB E۶ F۱ ۸۲ ۴۱

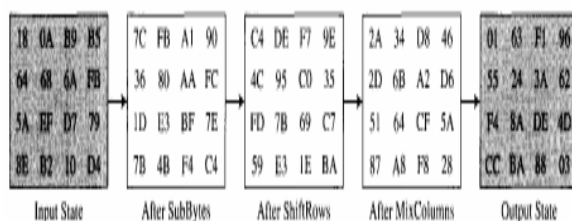
جدول ۷-۷، مقادیر ماتریس‌های وضعیت و کلیدهای گردش را برای این مثال نشان می‌دهد.

جدول ۷-۷: مثال رمزنگاری

Round	Input State	Output State	Round Key
Pre-round	00 12 0C 08 04 04 00 23 12 12 13 19 14 00 11 19	24 26 3D 1B 71 71 E2 89 B0 44 01 4D A7 88 11 9E	24 34 31 13 75 75 E2 AA A2 56 12 54 B3 88 00 87
1	24 26 3D 1B 71 71 E2 89 B0 44 01 4D A7 88 11 9E	6C 44 13 BD B1 9E 46 35 C5 B5 F3 02 5D 87 FC 8C	89 BD 8C 9F 55 20 C2 68 B5 E3 F1 A5 CE 46 46 C1
2	6C 44 13 BD 31 9E 46 35 C5 B5 F3 02 5D 87 FC 8C	1A 90 15 B2 66 09 1D FC 20 55 5A B2 2B CB 8C 3C	CE 73 FF 60 53 73 B1 D9 CD 2E DF 7A 15 53 15 D4
3	1A 90 15 B2 66 09 1D FC 20 55 5A B2 2B CB 8C 3C	F6 7D A2 B0 1B 61 B4 B8 67 09 C9 45 4A 5C 51 09	FF 8C 73 13 89 FA 4B 92 85 AB 74 0E C5 96 83 57
4	F6 7D A2 B0 1B 61 B4 B8 67 09 C9 45 4A 5C 51 09	CA E5 48 BB D8 42 AF 71 D1 BA 98 2D 4E 60 9E DF	B8 34 47 54 22 D8 93 01 DE 75 01 0F B8 2E AD FA
5	CA E5 48 BB D8 42 AF 71 D1 BA 98 2D 4E 60 9E DF	90 35 13 60 2C FB 82 3A 9E FC 61 ED 49 39 CB 47	D4 E0 A7 F3 54 8C 1F 1E F3 86 87 88 98 B6 1B E1
6	90 35 13 60 2C FB 82 3A 9E FC 61 ED 49 39 CB 47	18 0A B9 B5 64 68 6A FB 5A EF D7 79 8E B2 10 4D	86 66 C1 32 90 1C 03 1D 0B 8D 0A 82 95 23 38 D9
7	18 0A B9 B5 64 68 6A FB 5A EF D7 79 8E B2 10 4D	01 63 F1 96 55 24 3A 62 F4 8A DE 4D CC BA 88 03	62 04 C5 F7 83 9F 9C 81 3E B3 B9 3B B6 95 AD 74
8	01 63 F1 96 55 24 3A 62 F4 8A DE 4D CC BA 88 03	2A 34 D8 46 2D 6B A2 D6 51 64 CF 5A 87 A8 F8 28	EE EA 2F D8 61 FE 62 E3 AC 1F A6 9D DE 4B E6 92
9	2A 34 D8 46 2D 6B A2 D6 51 64 CF 5A 87 A8 F8 28	0A D9 F1 3C 95 63 9F 35 2A 80 29 00 16 76 09 77	E4 0E 21 F9 3F C1 A3 40 E3 FC 5A C7 BF F4 12 80
10	0A D9 F1 3C 95 63 9F 35 2A 80 29 00 16 76 09 77	BC E0 55 E6 02 E3 0D F1 8B B1 6D 82 D3 95 F8 41	DB D5 F4 0D F9 38 9B DB 2E D2 88 4F 26 D2 C0 40

مثال ۷-۱۱

شکل ۷-۲۱ ورودی‌های وضعیت گردش ۷ مثال ۷-۱۰ نشان می‌دهد.



شکل ۷-۲۱: وضعیت حاصل از یک روند

AES پس از DES طراحی شد. اکثر جملات شناخته شده به DES، بر روی AES نیز آزمایش شده‌اند. تاکنون هیچ یک از این حملات نتوانسته است امنیت AES را به چالش بکشند.

حمله‌ی آزمودن کلیدها

AES به خاطر کلید دارای اندازه‌ی بزرگ (۱۲۸، ۱۹۳ و ۲۵۶ بیتی) بسیار مطمئن‌تر از DES است. اجازه دهید DES با کلید رمز ۵۶ بیتی را با AES با کلید رمز ۱۲۸ بیتی مقایسه کنیم. در مورد DES (صرف‌نظر از مسئله‌ی متمم کلید) برای یافتن کلید ۲۵۶ آزمایش باید انجام شود. این بدین معنی است که اگر برای شکستن DES، به t ثانیه نیاز داریم، برای شکستن AES به $2^{128}t$ ثانیه نیاز است. می‌توان گفت که این امر تقریباً غیرممکن است. علاوه بر این AES، دو نسخه‌ی دیگر با کلیدهای رمز طولانی‌تر نیز وجود دارد. یک مزیت دیگر AES بر DES نبودن کلیدهای ضعیف است.

حملات آماری

پخش و آشفتگی قوی ایجاد شده به وسیله‌ی ترکیب تغییرات حاصل از ShiftRows، Sbytes، MixColumns هر نوع الگوی تکرار در متن اولیه را از بین می‌برد. آزمایش‌های گوناگونی در زمینه‌ی تجزیه و تحلیل آماری متن رمز ناموفق بوده‌اند.

حملات دیفرانسیلی و خطی

AES پس از DES طراحی شد. بدون شک حملات تحلیل رمز خطی و دیفرانسیلی نیز در نظر گرفته شده است. تاکنون هیچ حمله‌ی خطی و دیفرانسیلی روی AES انجام نشده است.

پیاده‌سازی

AES را می‌توان در نرم‌افزار، سخت‌افزار و سیستم‌عامل پیاده‌سازی نمود. روال پیاده‌سازی می‌تواند با مراجعه به جدول یا روتین‌هایی که از لحاظ ساختارهای جبری خوب تعریف شده است انجام گیرد. این تغییرات می‌تواند مبتنی بر بایت یا مبتنی بر کلمه باشد. در نسخه‌ی مبتنی بر بایت، کل الگوریتم می‌تواند از یک پردازشگر ۸ بیتی و در نسخه‌ی مبتنی بر کلمه، از یک پردازشگر ۳۲ بیتی استفاده کند. در هر یک از این حالات، طراحی ثابت‌ها، عمل پردازش را بسیار سریع می‌کند.

سهولت و هزینه

الگوریتم‌های به کار رفته در AES چنان ساده‌اند که می‌توان آن‌ها را به آسانی با استفاده از پردازشگرهای ارزان قیمت و میزان ناچیز از حافظه اجرا کرد.

۷-۹ چکیده

- * استاندارد رمزنگاری پیشرفته رمز بلوکی کلید متقارن است که توسط NIST به عنوان FIPS منتشر شد. AES مبتنی بر الگوریتم Rijndael است.
- * AES، رمز غیر Feistel است که بلوک داده‌ای ۱۲۸ بیتی را رمزنگاری و رمزگشایی می‌کند. این رمز از ۱۰، ۱۲ یا ۱۴ گردش استفاده می‌کند. اندازه‌ی کلیدها که می‌تواند ۱۲۸، ۱۹۲ یا ۲۵۶ بیت باشد، به تعداد گردش‌ها بستگی دارد.
- * AES مبتنی بر بایت است. متن اولیه یا متن رمز ۱۲۸ بیتی را به شکل شانزده بایت هشت بیتی در نظر می‌گیرند. AES برای این که بتواند برخی از تغییرات ریاضیاتی روی بایت‌ها انجام دهد، مفهوم وضعیت را تعریف کرده است. وضعیت، ماتریس 4×4 است که هر ورودی به آن یک بایت است.
- * به منظور تأمین امنیت، AES از چهار تغییر سود می‌برد؛ جایگزینی، جایگشت، تلفیق و افزودن کلید. هر گردش AES به جز گردش آخر، از هر چهار تغییر استفاده می‌کند. گردش آخر فقط از سه تغییر استفاده می‌کند.
- * جایگزینی یا از طریق مراجعه به جدول یا از طریق محاسبه ریاضیاتی در میدان $GF(2^8)$ تعیین می‌شود. AES از دو تغییر به شکل وارون، SubBytes و InvSubBytes که وارون یکدیگرند، استفاده می‌کند.
- * دومین تغییر در گردش، چرخش است که بایت‌ها را جابجا می‌کند. در رمزنگاری، این تغییر را ShiftRows و در رمزگشایی آن را InvShiftRows می‌نامند. تغییرات ShiftRows و InvShiftRows وارون یکدیگرند.
- * تغییرهای حاصل از تلفیق چهار بایت در یک زمان و ترکیب آن‌ها با یکدیگر و ایجاد چهار بایت جدید، محتوای هر بایت را عوض می‌کند. AES دو تغییر MixColumns و InvMixColumns را برای استفاده در رمزنگاری و رمزگشایی تعریف کرده است. MixColumns، ماتریس وضعیت را در یک ماتریس مربع اعداد ثابت ضرب می‌کند. InvMixColumns همین کار را با استفاده از ماتریس ثابت وارون انجام می‌دهد. تغییرات MixColumns و InvMixColumns وارون یکدیگرند.
- * تغییری که سپیدگر انجام می‌دهد، AddRoundKey است. برای ایجاد وضعیت جدید، وضعیت پیشین با کلید ماتریس گردش جمع می‌شود (جمع ماتریس). جمع عضوها در دو ماتریس، در میدان $GF(2^8)$ انجام می‌شود و این به معنی این است که کلمات ۸ بیتی XOR می‌شوند. تغییرات AddRoundKey خود وارون هستند.

* در نخستین پیکربندی (۱۰ گردش با کلیدهای ۱۲۸ بیتی) مولد کلید، یازده کلید گردش ۱۲۸ بیتی از کلید رمز ۱۲۸ بیتی می‌سازد. AES از مفهوم کلمه برای تولید کلید استفاده می‌کند. هر کلمه متشکل از چهار بایت است. کلیدها در هر گردش، کلمه به کلمه ایجاد می‌شود. AES کلمات را از w_0 تا w_{23} شماره‌گذاری می‌کند. این روال را «گسترش کلید» می‌نامند.

* رمز AES از دو الگوریتم برای رمزگشایی استفاده می‌کند. در طراحی اصلی، ترتیب تغییرات هر گردش در رمزنگاری و رمزگشایی یکسان نیست. در طراحی‌های جایگزین، برای یکسان‌سازی ترتیب در رمزنگاری و رمزگشایی، الگوریتم‌های رمزگشایی دوباره مرتب شده‌اند. در نسخه دوم، وارون‌پذیری برای هر زوج از تغییرات در نظر گرفته شده است.

۷-۱۰- مجموعه تمرین‌ها

پرسش‌های دوره‌ای

- (۱) معیارهایی که NIST برای AES تعیین کرده بود را نام ببرید.
- (۲) پارامترها (اندازه‌ی بلوک، اندازه‌ی کلید، و تعداد گردش‌ها) برای سه نسخه AES را نام ببرید.
- (۳) در هر نسخه AES چند تغییر وجود دارد؟ برای هر نسخه چند کلید گردش لازم است؟
- (۴) AES و DES را مقایسه کنید. کدام یک مبتنی بر بیت و کدام یک مبتنی بر بایت است؟
- (۵) وضعیت در AES را تعریف کنید. در هر نسخه AES چند وضعیت وجود دارد؟
- (۶) کدام یک از تغییرات تعیین شده برای AES، محتوای بایت‌ها را تغییر می‌دهد؟ کدام یک محتوای بایت‌ها را تغییر نمی‌دهد؟
- (۷) جایگزینی در رمزهای AES و DES را مقایسه کنید. چرا در AES فقط یک جدول جایگزینی (جعبه‌ی S) داریم در حالی که در DES چندین جعبه‌ی S وجود دارد؟
- (۸) جایگشت در AES و DES را مقایسه کنید. چرا در DES جایگشت‌ها گسترش و فشرده لازم است در حالی که در AES لازم نیست؟
- (۹) کلیدهای گردش در AES و DES را مقایسه کنید. در کدام رمز اندازه‌ی کلید گردش و اندازه‌ی بلوک یکسان است؟
- (۱۰) به نظر شما چرا تأثیر تلفیق MixColumns در DES لازم نیست در حالی که در AES لازم است؟

تمرین‌ها

- (۱۱) شیوه‌ی رمزنگاری جعبه‌های S می‌تواند ایستا یا پویا باشد. پارامترها در یک جعبه‌ی S ایستا، به کلید بستگی ندارد.
- (a) چند مزیت و چند نقص جعبه‌های S ایستا و پویا را بیان کنید.
- (b) جعبه‌های S (جدول‌های جایگزینی) در AES ایستا است یا پویا؟
- (۱۲) AES نسبت به DES دارای اندازه‌ی بلوک بزرگ‌تری است (۱۲۸ در مقابل ۶۴). این مزیت است یا نقص؟ توضیح دهید.
- (۱۳) AES سه شیوه اجرا با سه تعداد متفاوت گردش (۱۰، ۱۲، ۱۴) تعریف کرده است؛ DES فقط یک اجرا با ۱۶ گردش را تعریف نموده است. با توجه به این تفاوت، مزایا و معایب AES نسبت به DES چیست؟

(۱۴) AES سه اندازه مختلف کلید رمز (۱۲۸، ۱۹۲، ۲۵۶) تعریف کرده است؛ در حالی که DES فقط یک اندازه کلید (۵۶) تعریف نموده است. با توجه به این تفاوت، مزایا و معایب AES نسبت به DES چیست؟

(۱۵) در AES اندازه‌ی بلوک با اندازه‌ی کلید گردش یکی است (۱۲۸) در حالی که در DES اندازه‌ی بلوک ۶۴ بیت، اما اندازه‌ی کلید در گردش تنها ۴۸ بیت است. با توجه به این نکته مزایا و معایب AES نسبت به DES چیست؟

(۱۶) پس از انجام موارد زیر ثابت کنید که تغییرات حاصل از ShiftRows و InvShiftRows جایگشتی هستند:

(a) جدول جایگشت ShiftRows را تشکیل دهید. جدول باید ۱۲۸ ورودی داشته باشد اما از آنجا که محتوای یک بایت تغییر نمی‌کند، این جدول می‌تواند با فرض این که هر ورودی نشان دهنده‌ی یک بایت است، تنها ۱۶ ورودی داشته باشد.

(b) قسمت a را برای تغییرات حاصل InvShiftRows تکرار کنید.

(c) با استفاده از نتایج قسمت‌های a و b ثابت کنید که تغییرات ShiftRows و InvShiftRows وارون یکدیگرند.

(۱۷) با استفاده از یک کلید رمز واحد، هر یک از تغییرات زیر را بر متون عادی که تنها در بیت اول با هم تفاوت دارند، اعمال کنید. تعداد بیت‌های تغییر یافته پس از هر تغییر را پیدا کنید. هر تغییر را به طور مستقل اعمال نمایید.

- a) SubBytes
- b) ShiftRows
- c) MixColumns
- d) AddRoundKey

(۱۸) برای مشاهده‌ی غیرخطی بودن تغییرات SubBytes نشان دهید که اگر در دو بایت a, b داریم:

$$\text{SubBytes}(a \oplus b) \neq \text{SubBytes}(a) \oplus \text{SubBytes}(b)$$

از $a = 0x57$ و $b = 0xA2$ به عنوان نمونه استفاده کنید.

(۱۹) فرمول ثابتی برای محاسبه‌ی تعداد حاصل از تغییرات AddRoundKey, MixColumns, ShiftRowsKey, SubBytes و تعداد تغییرات کلی برای هر نسخه از AES ارائه دهید. این فرمول باید بر اساس تعداد گردش‌ها باشد.

(۲۰) شکل ۶-۷ را برای AES-۱۹۲ و AES-۲۵۶ دوباره رسم کنید.

(۲۱) دو جدول جدید ایجاد کنید که ثابت‌های Rcons را برای AES-۱۹۲ و AES-۲۵۶ نشان دهد. (به جدول ۴-۷ مراجعه کنید).

- (۲۲) در AES-۱۲۸ کلید گردش بکار رفته در عمل پیش گردش با کلید رمز یکی است. آیا برای ۱۹۲-AES هم همین طور است؟ آیا AES-۲۵۶ نیز همین گونه است.
- (۲۳) در شکل ۷-۸ ماتریس‌های X و X^{-1} را در هم ضرب کنید تا ثابت کنید که وارون یکدیگرند.
- (۲۴) با استفاده از شکل ۷-۱۲ و با استفاده از چند جمله‌هایی با ضرایبی در $GF(2)$ ، ماتریس‌های C و C^{-1} را دوباره بنویسید. دو ماتریس را در هم ضرب کنید و ثابت کنید که وارون یکدیگرند.
- (۲۵) ثابت کنید که کد موجود در الگوریتم ۷-۱ (تغییر شکل در SubBytes) با مراحل نشان داده‌شده در شکل ۷-۸ تطابق دارد.
- (۲۶) با استفاده از الگوریتم ۷-۱ (تغییر SubBytes)، موارد زیر را انجام دهید:
- کدی برای یک روتین بنویسید که وارون یک بایت را در $GF(2^8)$ محاسبه کند.
 - کدی برای Byte to Matrix بنویسید.
 - کدی برای Matrix to Byte بنویسید.
- (۲۷) الگوریتمی برای تغییر InvSubBytes بنویسید.
- (۲۸) ثابت کنید که کد موجود در الگوریتم ۷-۲ (تغییر ShiftRows) با مراحل نشان داده‌شده در شکل ۷-۹ تطابق دارد.
- (۲۹) با استفاده از الگوریتم ۷-۲ (تغییر ShiftRows) کدی برای روتین CopyRow بنویسید.
- (۳۰) الگوریتمی برای تغییر InvShiftRow بنویسید.
- (۳۱) ثابت کنید که کد موجود در الگوریتم ۷-۳ (تغییر MixColumns) با مراحل نشان داده‌شده در شکل ۷-۱۳ تطابق دارد.
- (۳۲) با استفاده از الگوریتم ۷-۳ (تغییرات MixColumns) کدی برای روتین CopyColumns بنویسید.
- (۳۳) برای محاسبه‌ی ضرب دو بایت در میدان $GF(2^8)$ به جای عملگر (\bullet) در الگوریتم ۷-۳ (تغییر MixColumns) روتین موسوم به MultField را بدون تغییر نگهدارید و الگوریتم را بازنویسی کنید.
- (۳۴) الگوریتمی برای تغییر InvMixColumns بنویسید.
- (۳۵) ثابت کنید که کد موجود در الگوریتم ۷-۴ (مربوط به تغییر AddRoundKey) با مراحل نشان داده‌شده در شکل ۷-۱۵ تطابق دارد.
- (۳۶) در الگوریتم ۷-۵ (گسترش کلید)
- کد روتین SubWord را بنویسید.
 - کد روتین Rotword را بنویسید.

۳۷) دو الگوریتم جدید برای گسترش کلید در AES-۱۹۲ و AES-۲۵۶ ارائه دهید. (به الگوریتم ۵-۷ نگاه کنید).

۳۸) الگوریتم گسترش کلید برای رمز وارون جایگزین بنویسید.

۳۹) الگوریتمی برای رمز وارون در طراحی اصلی بنویسید.

۴۰) الگوریتمی برای رمز وارون در طراحی جایگزین بنویسید.

فصل ۸

رمزنویسی با استفاده از رمزهای کلید متقارن مدرن

چشم‌انداز

این فصل اهداف زیر را دنبال می‌کند:

✱ نشان دادن این که چطور رمزهای استاندارد مدرن DES و AES را می‌توان برای رمزنویسی پیام-های طولانی به کار برد.

✱ بحث و بررسی پنج حالت عملکرد در نظر گرفته شده برای استفاده در رمزهای بلوکی مدرن.

✱ تعیین این که کدام حالت از عملکردها، رمزهای جریانی را از رمزهای بلوکی پایه ایجاد می‌کند.

✱ بحث و بررسی مسئله‌ی امنیت و انتشار خطا در حالت‌های مختلف عملکرد.

✱ بحث و بررسی دو رمز جریانی مورد استفاده در پردازش آنی داده‌ها.

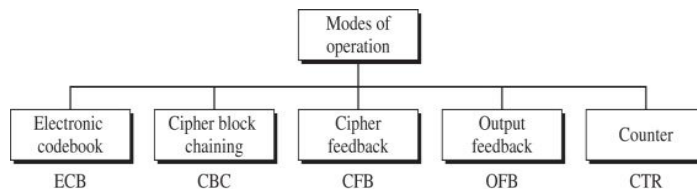
فصل ۸ نشان می‌دهد که چگونه مفاهیم مطرح شده در فصل ۵ و دو رمز بلوکی پیشرفته مشروحه در

فصول ۶ و ۷ را می‌توان برای رمزنویسی پیام‌های طولانی به کار برد. همچنین این فصل، دو شیوه‌ی رمز جریانی را معرفی می‌کند.

۸-۱- کاربرد رمزهای بلوکی پیشرفته

با استفاده از رمزهای بلوکی پیشرفته می‌توان رمزنویسی را بر پایه‌ی کلید متقارن انجام داد. دو رمز بلوکی پیشرفته مشروحه در فصول ۶ و ۷ یعنی DES و AES برای رمزنویسی و رمزگشایی بلوکی از متن با اندازه-ی ثابت طراحی شده‌اند. DES یک بلوک ۶۴ بیتی و AES یک بلوک ۱۲۸ بیتی را می‌تواند رمزنگاری و رمزگشایی نماید. در کاربردهای روزمره، متنی که باید رمزنویسی شود دارای اندازه متغیر می‌باشد و معمولاً

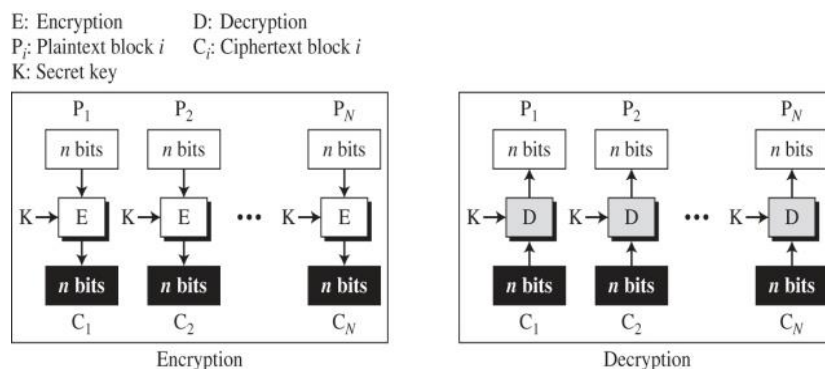
طولانی‌تر از ۶۸ یا ۱۲۸ بیت است. حالت‌های عملکرد گوناگونی برای رمزنویسی متن به هر اندازه‌ای با استفاده از DES یا AES ابداع شده است. شکل ۸-۱ پنج حالت عملکرد را که در فصل مورد بحث قرار خواهد گرفت نشان می‌دهد.



شکل ۸-۱: حالت عمل

حالت کتاب کد الکترونیکی^۱

ساده‌ترین حالت عمل را حالت کتاب کد الکترونیکی می‌نامند. متن اولیه به N بلوک تقسیم می‌شود. اندازه‌ی هر بلوک n بیت است. اگر اندازه‌ی متن اولیه چند برابر اندازه‌ی بلوک باشد، به متن نویسه‌هایی اضافه می‌گردد تا بدین ترتیب، اندازه‌ی بلوک آخر همانند سایر بلوک‌ها شود. برای رمزنگاری و رمزگشایی هر بلوک، از کلید یکتایی استفاده می‌شود. شکل ۸-۲ رمزنگاری و رمزگشایی را در این حالت نشان می‌دهد.



شکل ۸-۲: حالت کتاب کد الکترونیکی (ECB)

رابطه‌ی بین متن عادی و متن رمز به صورت زیر تعریف می‌شود.

$$C_i = E_k(P_i) \quad \text{رمزگذاری} \qquad P_i = D_k(C_i) \quad \text{رمزگشایی}$$

مثال ۸-۱

می‌توان ثابت کرد که هر بلوک از متن عادی نزد آلیس، دقیقاً در جای مناسب خود به وسیله‌ی باب بازیابی می‌شود. از آن جا که رمزنگاری و رمزگشایی وارون یکدیگرند، پس

$$P_i = D_k(C_i) = D_k(E_k(P_i)) = P_i$$

=====

¹ Electronic Codebook (ECB)

مثال ۸-۲

این حالت را کتاب کد الکترونیکی می‌نامند زیرا می‌توان کتاب‌های کد 2^K را از پیش همسان نمود (یک کتاب کد برای هر کلید)، که در آن هر کتاب کد 2^n ورودی در دو ستون داشته باشد. هر ورودی می‌تواند متن عادی و بلوک‌های متن رمز متناظر آن را لیست کند. با این وجود اگر K و n اعداد بزرگی باشند، کتاب کد نیز بسیار بزرگ‌تر از آن خواهد بود که بتوان آن را از پیش همسان و حفظ کرد.

چالش‌های امنیتی

چالش‌های امنیتی در حالت CBC موارد زیر می‌باشد:

۱. الگوها در سطح بلوکی حفظ می‌شود. مثلاً هر بلوک در متن عادی به بلوک متناظر در متن رمز تبدیل می‌شود. اگر رمزشکن بفهمد که بلوک‌های ۱، ۵ و ۱۰ متن رمز یکسان‌اند، نتیجه می‌گیرد که بلوک‌های ۱، ۵ و ۱۰ متن عادی نیز یکسان‌اند. این مسئله نقطه ضعف امنیتی محسوب می‌شود. به عنوان مثال، رمزشکن می‌تواند برای رمزگشایی تنها با یکی از این بلوک‌ها، بررسی جامعی انجام دهد و بدین ترتیب محتوای تمام آن‌ها را پیدا کند.
۲. عدم وابستگی بلوکی این امکان را برای رمزشکن فراهم می‌کند تا بدون دانستن کلید، برخی از بلوک‌های متن رمز را عوض کند. مثلاً اگر رمزشکن بداند بلوک ۸ همیشه حاوی اطلاعات خاصی است، می‌تواند این بلوک را با بلوک متناظر آن در پیام دریافت شده پیشین جابجا کند.

مثال ۸-۳

فرض کنید رمزشکن چند ساعت در ماه در شرکتی کار می‌کند (وی حقوق بسیار کمی دریافت می‌کند). او می‌داند که این شرکت از چندین بلوک اطلاعاتی برای هر کارمند استفاده می‌کند و هفتمین بلوک، مقدار پولی است که باید به حساب کارمندان واریز شود. رمزشکن می‌تواند ارسال متن رمز را مختل و در پایان ماه، در حالی که کپی بلوک اطلاعات حقوقی همکار تمام وقتش را جایگزین اطلاعات حقوقی خود نموده است، به بانک ارسال نماید. او هر ماه می‌تواند بیش از آن چه که مستحقش است حقوق بگیرد.

انتشار خطا

فقط یک خطای بیتی در زمان انتقال می‌تواند موجب بروز چندین خطا در بلوک مربوطه (معمولاً نصف بیت‌ها یا تمام بیت‌ها در بلوک) می‌شود. با این وجود، این خطا تأثیری بر بلوک‌های دیگر ندارد.

الگوریتم

برای رمزنگاری و رمزگشایی می‌توان الگوریتم‌های ساده‌ای نوشت. الگوریتم ۸-۱ روتین شبه کد لازم برای رمزنگاری ارائه می‌دهد. روتین شبه کد را برای رمزگشایی به عنوان تمرین به عهده‌ی شما می‌گذاریم. E_k فقط یک بلوک را رمزنگاری می‌کند و می‌تواند یکی از شیوه‌های رمزنگاری مطرح شده در فصول ۷-۶ (DES یا AES) را به کار گیرد.

ECB_Encryption (K, Plaintext blocks)

```
{
  for (i = 1 to N)
  {
     $C_i \leftarrow E_K(P_i)$ 
  }
  return Ciphertext blocks
}
```

الگوریتم ۸-۱: رمزنگاری بر دو حالت ECB

سرقت متن رمز

در حالت ECB، اگر طول بیت‌ها n نباشد، باید به بلوک آخر داده‌های ساختگی اضافه شود. ولی این کار همیشه امکان‌پذیر نیست. به عنوان مثال، اگر لازم باشد متن رمز در بافری ذخیره شود که پیشتر متن عادی در آن ذخیره شده است، متن عادی و متن رمز باید یکسان باشند. فنی به نام سرقت متن رمز^۱ استفاده از حالت ECB بدون انجام افزودن داده‌های ساختگی را امکان‌پذیر می‌کند. همان‌گونه که در زیر نشان داده شده است، در این فن دو بلوک آخر متن عادی P_N و P_{N-1} به صورت مجزا و خارج از ترتیب رمزنگاری می‌شود، با فرض این که P_{N-1} ، n بیت و P_N ، m بیت دارد. طوری که $m \leq n$ است.

$$X = E_k(P_{N-1}) \rightarrow C_N = \text{head}_m(X)$$

$$Y = P_N | \text{tail}_{n-m}(X) \rightarrow C_{N-1} = E_k(Y)$$

تابع head_m بیت‌های سمت چپ، و تابع tail_{n-m} بیت‌های سمت راست را انتخاب می‌کند. دیاگرام دقیق و شبه کد رمزنگاری و رمزگشایی را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

=====

¹ Ciphertext Stealing (CTS)

کاربردها

در حالت ECB عمل برای رمزنگاری پیام‌های بیش از یک بلوک که باید از طریق کانال ناامن منتقل شود، توصیه نمی‌شود. اگر پیام به قدری کوتاه باشد که بتوان آن را در یک بلوک جا داد، چالش‌های امنیتی و انتشار خطا قابل چشم‌پوشی است.

حوزه‌ای که در آن عدم وابستگی بین بلوک‌های متن رمز می‌تواند مفید باشد، جایی است که باید پیش از ذخیره شدن در پایگاه داده، رمزنگاری یا پیش از بازیابی، رمزگشایی شود. از آن جا که در این حالت ترتیب رمزنگاری و رمزگشایی بلوک‌ها اهمیت ندارد، اگر هر رکورد در یک یا چند بلوک ذخیره شده باشد، دستیابی به پایگاه داده می‌تواند تصادفی (غیر ترتیبی) باشد. رکوردها را می‌توان از وسط هم بازیابی کرد و پس از تغییر بدون تأثیر بر ترتیب سایر رکوردها، رمزنگاری و رمزگشایی نمود.

مزیت دیگر این حالت زمانی است که مثلاً ایجاد پایگاه داده‌ای بسیار بزرگ ضروری باشد و لزوم استفاده از پردازش موازی محسوس شود.

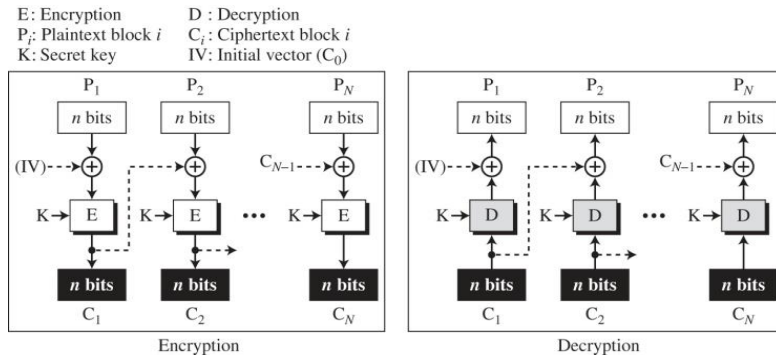
حالت زنجیره‌ای بلوک رمز^۲

تحول بعدی در حالت‌های گوناگون عملکرد، حالت زنجیره‌ای بلوک رمز است. در حالت CBC، هر بلوک متن عادی پیش از رمزنگاری، با بلوک پیشین متن رمز XOR می‌شود. وقتی بلوک رمز نویسی شد ارسال می‌گردد اما جهت استفاده در رمزنگاری بلوک بعدی، یک کپی از آن در حافظه ذخیره می‌شود. در مورد بلوک اول ممکن است در ذهن شما این پرسش ایجاد شود که پیش از بلوک اول، بلوک متن رمز وجود ندارد. در این حالت، بلوک ساختگی به نام بردار مقداردهی اولیه (IV) به کار می‌رود. فرستنده و گیرنده بر سر استفاده از یک IV پیش تعیین شده توافق می‌کنند؛ به عبارت دیگر، IV به جای C، که موجود نیست به کار می‌رود. شکل ۸-۳ حالت عملکرد CBC را نشان می‌دهد. در قسمت فرستنده، عمل XOR پیش از رمزنگاری انجام می‌شود و در قسمت گیرنده رمزگشایی پس از XOR انجام می‌شود.

=====

¹ Record

² Cipher Block chaining (CBC)



شکل ۸-۳: حالت زنجیره‌ای بلوک رمز

رابطه‌ی بین بلوک‌های متن عادی و متن رمز به شرح زیر می‌باشد.

رمزنگاری

رمزگشایی

$$C_i = IV$$

$$C_i = IV$$

$$C_i = E_k(P_i \oplus C_{i-1})$$

$$P_i = D_k(C_i) \oplus C_{i-1}$$

مثال ۸-۴

می‌توان ثابت کرد که هر بلوک متن عادی آلیس دقیقاً در محل خود توسط باب بازیابی می‌شود. از آن

جا که رمزنگاری و رمزگشایی وارون یکدیگرند،

$$P_i = D_k(C_i) \oplus C_{i-1} = D_k(E_k(P_i \oplus C_{i-1})) \oplus C_{i-1} = P_i \oplus C_{i-1} \oplus C_{i-1} = P_i$$

بردار مقداردهی اولیه^۱

فرستنده و گیرنده باید بردار مقداردهی اولیه را بدانند. اگر چه پنهان نگه‌داشتن بردار لازم نیست اما انسجام

بردار نقش مهمی در امنیت حالت CBC ایفا می‌کند. IV باید از تغییر دور نگه داشته شود. اگر رمزشکن

بتواند مقادیر بیت IV را تغییر دهد، می‌تواند مقادیر بیت بلوک اول را تغییر دهد.

برای استفاده از IV چند روش توصیه می‌شود. فرستنده می‌تواند یک عدد شبه تصادفی را انتخاب کند

و از طریق کانال مطمئن (مثلاً با استفاده از حالت ECB)، آن را انتقال دهد. وقتی کلید سری ایجاد شد، باب

و آلیس می‌توانند بر سر یک مقدار ثابت توافق کنند. این مقدار ثابت می‌تواند قسمتی از کلید سری هم باشد،

و الی آخر.

¹ Initialization Vector (IV)

چالش‌های امنیتی

در حالت CBC دو چالش امنیتی وجود دارد:

- (۱) در حالت CBC، بلوک‌های برابر متن عادی از یک پیام به صورت بلوک‌های متفاوت متن رمز، رمزنویسی می‌شود؛ به عبارت دیگر، الگوها در سطح بلوک حفظ نمی‌شود. با این وجود اگر دو پیام برابر باشند، رمزنویسی آن‌ها نیز یکی است، البته به شرطی که از یک IV استفاده کنند؛ در واقع اگر بلوک اول دو پیام مختلف یکسان باشند، در صورت عدم استفاده از دو IV متفاوت، به صورت بلوک‌های یکسان رمزنویسی می‌شوند؛ به همین دلیل برخی، استفاده از برچسب زمان را به عنوان IV پیشنهاد می‌کنند.
- (۲) رمزشکن می‌تواند چند بلوک متن رمز را به انتهای جریان متن رمز اضافه کند.

انتشار خطا

در حالت CBC، تنها یک خطای بیتی در بلوک متن رمز C_i در حین انتقال می‌تواند در زمان رمزگشایی در اکثر بیت‌های بلوک متن عادی P_j خطا ایجاد کند؛ با این وجود، این خطا فقط یک بیت در بلوک متن عادی P_{j+1} (بیت در همان موقعیت) را متأثر می‌کند. اثبات این موضوع را به عنوان تمرین به عهده‌ی شما می‌گذاریم. این خطای بیتی، بلوک‌های P_{j+2} تا P_N را تحت تأثیر قرار نمی‌دهد. تنها یک خطای بیتی در متن رمز خودبه‌خود بازیابی می‌شود.

الگوریتم

الگوریتم ۸-۲، شبه کد برای رمزنگاری ارائه می‌دهد. این الگوریتم، روتین رمزنگاری را که تنها یک بلوک را رمزنگاری می‌کند (مثلاً DES یا AES) فرامی‌خواند. الگوریتم رمزگشایی را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

CBC_Encryption (IV, K, Plaintext blocks)

```
{
   $C_0 \leftarrow IV$ 
  for ( $i = 1$  to  $N$ )
  {
     $Temp \leftarrow P_i \oplus C_{i-1}$ 
     $C_i \leftarrow E_K(Temp)$ 
  }
  return Ciphertext blocks
}
```

الگوریتم ۸-۲: الگوریتم رمزنگاری برای حالت CBC

سرقت متن رمز

همان‌گونه که مشاهده می‌کنید، شگرد سرقت متن رمز، که برای حالت ECB توصیف شد را می‌توان برای CBC نیز به کار برد:

$$U = P_{N-1} \oplus C_{N-2} \rightarrow X = E_k(U) \rightarrow C_N = \text{head}_m(X)$$

$$V = P_N \mid \text{pad}_{n-m}(\cdot) \rightarrow Y = X \oplus V \rightarrow C_{N+1} = E_k(Y)$$

تابع head با آنچه که برای حالت ECB توصیف شد، یکسان است؛ تابع Pad مقدار 0 را به متن اضافه می‌کند.

کاربردها

از حالت CBC می‌توان برای رمزنویسی پیام‌ها استفاده کرد؛ با این وجود، به علت وجود سازوکار زنجیره‌ای، انجام پردازش موازی غیرممکن است. از حالت CBC نمی‌توان برای رمزنگاری و رمزگشایی رکورد در فایل‌ها با دسترسی تصادفی استفاده کرد، زیرا رمزنگاری و رمزگشایی مستلزم دسترسی به رکوردهای پیشین است. حالت CBC را می‌توان برای احراز هویت به کار برد.

حالت بازخورد رمز^۱

ECB و CBC بلوک‌های پیام را رمزنگاری و رمزگشایی می‌کنند. اندازه‌ی بلوک، n ، با استفاده از رمز پایه و از پیش تعیین می‌شود. به عنوان مثال $n=64$ برای DES و $n=128$ برای AES. برخی مواقع لازم است از DES یا AES به عنوان رمزهای مطمئن استفاده کنیم، اما اندازه‌ی بلوکی متن عادی و متن رمز باید کوچک‌تر باشند. مثلاً برای رمزنگاری و رمزگشایی کاراکترهای ۸ بیتی ASCII نباید از رمزهای سنتی مطرح شده در فصل ۳ استفاده کنید چون این رمزها نامطمئن هستند. راه حل، استفاده از DES یا AES در حالت بازخورد رمز (CFB) می‌باشد. در این حالت، اندازه‌ی بلوک به کار رفته در DES یا AES، n است، اما اندازه‌ی بلوک متن عادی یا متن رمز r می‌باشد، طوری که $r \leq n$ است.

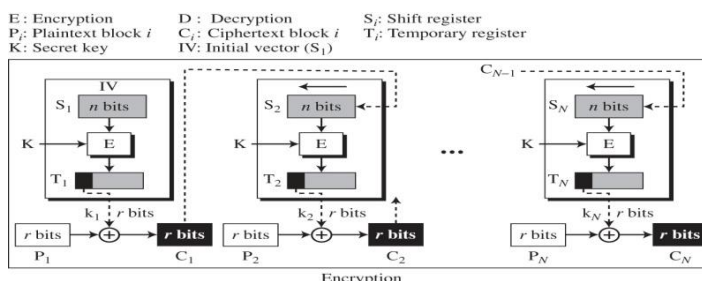
هدف ما از استفاده از DES یا AES؛ رمزنگاری یا رمزگشایی محتوای یک شیفت ریجستر $S -$ به اندازه n می‌باشد؛ نه رمزنگاری متن عادی یا رمزگشایی متن رمز شده. رمزنگاری به وسیله XOR یک بلوک r بیتی متن عادی با r بیت از شیفت ریجستر انجام می‌شود. رمزگشایی به وسیله XOR یک بلوک r بیتی

=====

¹ Cipher Feedback (CFB)

متن رمز با r بیت از شیفت ریجستر انجام می‌شود. با جابجا کردن شیفت ریجستر S_{i-1} (شیفت ریجستر پیشین) r بیت به سمت چپ و پرکردن بیت‌های r سمت راست با C_{i-1} ، شیفت ریجستر S_i برای هر بلوک ایجاد می‌شود. سپس S_i برای هر بلوک ایجاد می‌شود. سپس S_i به صورت T_i رمزنگاری می‌شود. فقط r بیت سمت راست T_i با بلوک متن عادی P_i XOR می‌شود؛ بدین ترتیب C_i به دست می‌آید. توجه داشته باشید که S_1 که جابجا نمی‌شود، برای استفاده در بلوک اول به IV فرستاده می‌شود.

شکل ۸-۴ حالت CFB را برای رمزنویسی نشان می‌دهد. رمزگشایی نیز به همین شکل است با این تفاوت که نقش بلوک‌های متن عادی (P_i, S) و بلوک‌های متن رمز (C_i, S) عوض می‌شود. توجه کنید که هم رمزنویسی و هم رمزگشایی از تابع رمزنگاری رمز بلوکی پایه (مثلاً DES یا AES) استفاده می‌کنند.



شکل ۸-۴: رمزنگاری در حالت بازخورد رمز (CFB)

در حالت CFB، رمزنویسی و رمزگشایی از تابع رمزنگاری رمز بلوکی پایه استفاده می‌کنند.

روابط بین بلوک‌های متن عادی و متن رمز به شکل زیر می‌باشد:

$$\text{Encryption: } C_i = P_i \oplus \text{SelectLeft}_r \{ E_k [\text{shiftLeft}_r (S_{i-1}) | (C_{i-1})] \}$$

$$\text{Decryption: } C_i = P_i \oplus \text{SelectLeft}_r \{ E_k [\text{shiftLeft}_r (S_{i-1}) | (C_{i-1})] \}$$

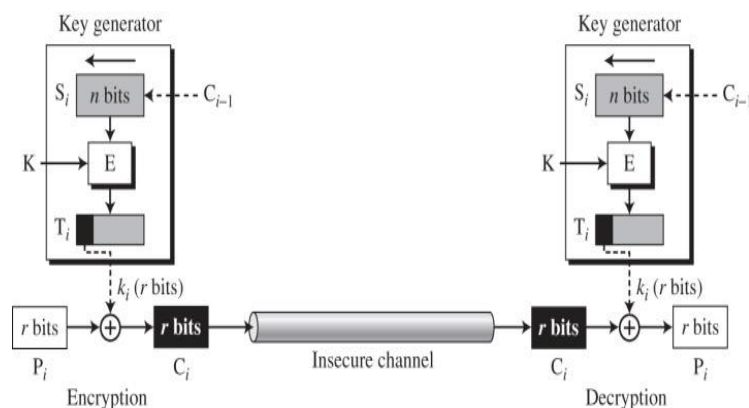
روتین ShiftLeft_r r بیت مربوطه به خود را به سمت چپ جابجا می‌کند. (سمت چپ‌ترین r بیت-ها حذف می‌شوند). علامت | پیوستگی را نشان می‌دهد. روتین SelectLeft_r فقط سمت چپ‌ترین بیت-های r مورد بحث را انتخاب می‌کند. می‌توان اثبات کرد که هر بلوک متن عادی ارسالی توسط آلیس دقیقاً در همان قسمت به وسیله‌ی باب بازیابی می‌شود. اثبات را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

نکته شایان توجه در مورد این حالت، عدم نیاز به افزودن داده‌های ساختگی به متن عادی است، زیرا اندازه‌ی بلوک، r ، معمولاً متناسب با واحد داده که باید رمزنگاری شود (مثلاً یک نویسه)، انتخاب می‌گردد. نکته جالب دیگر این است که لازم نیست سیستم پیش از آغاز رمزنگاری منتظر دریافت بلوک بزرگی از

داده‌ها (۶۴ بیت یا ۱۲۸ بیت) باشد. قاعده‌ی رمزنگاری برای بلوک کوچکی از داده‌ها (نظیر یک نویسه) انجام می‌شود. این دو مزیت با یک عیب همراه است. CFB به اندازه‌ی CBC یا ECB کارآمد نیست زیرا تابع رمزنگاری رمز بلوکی پایه باید بر روی بلوک‌های کوچک به اندازه‌ی ۲ اعمال شود.

کاربرد CFB در رمز جریانی

اگر چه شیوه‌ی کارکرد CFB استفاده از رمزهای بلوکی نظیر DES یا AES است، اما نتیجه رمز جریانی است. در واقع، CFB رمز جریانی نامتقارن است که جریان کلید به متن رمز بستگی دارد. شکل ۵-۸ روش رمزنگاری و رمزگشایی را با فرض مشخص بودن مولد کلید نشان می‌دهد.



شکل ۵-۸: حالت بازخورد رمز (CFB) به عنوان یک رمز جریانی

شکل ۵-۸ نشان می‌دهد که رمز پایه (DES یا AES)، کلید رمز (K) و بلوک رمز قبلی (C_{i-1}) فقط برای ایجاد جریان‌های کلید (k_1, k_2, \dots, k_n) استفاده می‌شود.

الگوریتم

الگوریتم ۳-۸ روتین برای رمزنگاری را ارائه می‌دهد. این الگوریتم چندین روتین دیگر را فرامی‌خواند که جزئیات اجرای آن‌ها را به عنوان تمرین به عهده‌ی شما می‌گذاریم. توجه داشته باشید که الگوریتم را طوری نوشته‌ایم که ماهیت جریانی (وضعیت بلادرنگ) را نشان دهد. تا وقتی که بلوک متن عادی برای رمزنگاری وجود داشته باشد، الگوریتم ادامه می‌یابد.

```

CFB_Encryption (IV, K, r)
{
  i ← 1
  while (more blocks to encrypt)
  {
    input (Pi)
    if (i = 1)
      S ← IV
    else
    {
      Temp ← shiftLeftr(S)
      S ← concatenate (Temp, Ci-1)
    }
    T ← EK(S)
    ki ← selectLeftr(T)
    Ci ← Pi ⊕ ki
    output (Ci)
    i ← i + 1
  }
}

```

الگوریتم ۸-۳: الگوریتم رمزنگاری برای CFB

چالش‌های امنیتی

در حالت CFB سه چالش امنیتی اساسی وجود دارد:

- (۱) دقیقاً مانند CBC، الگو در سطح بلوکی حفظ نمی‌شود.
- (۲) بیش از یک پیام را می‌توان با یک کلید رمزنگاری کرد، به شرطی که مقدار IV را برای هر پیام تغییر داد؛ یعنی آلیس باید برای هر پیامی که می‌خواهد ارسال کند از IV جداگانه‌ای استفاده نماید.
- (۳) رمزشکن می‌تواند چند بلوک متن رمز به انتهای جریان متن رمز اضافه کند.

انتشار خطا

در CFB، یک خطای بیتی در بلوک متن رمز C در خلال انتقال، فقط یک خطای بیتی (در همان موقعیت) در بلوک متن عادی P_j ایجاد می‌کند؛ با این وجود، تا وقتی که برخی از بیت‌های C_j هنوز در شیفت ریجستر باشند، اکثر بیت‌ها در بلوک‌های متن عادی نیز خطا دار هستند (احتمال ۵۰ درصد). محاسبه‌ی تعداد بلوک‌های متأثر را به عنوان تمرین به عهده شما می‌گذاریم. پس از این که شیفت ریجستر به طور کلی از نو شروع به کار کرد سیستم از حالت خطا خارج می‌شود.

کاربرد:

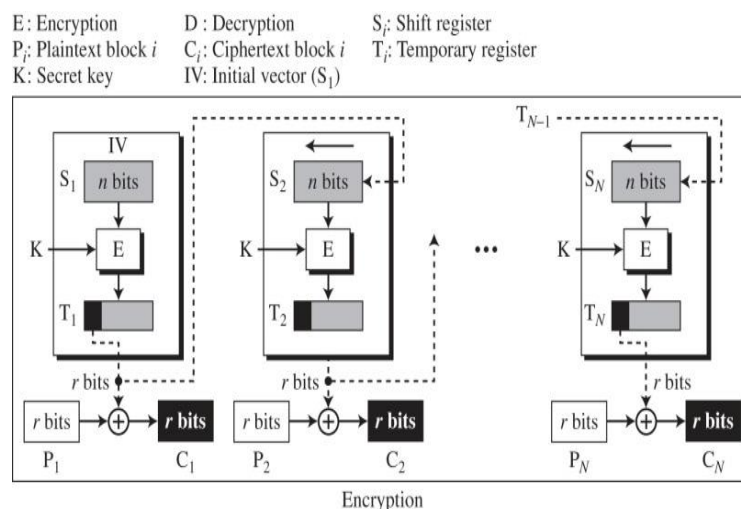
از حالت CFB می‌توان برای رمزنویسی بلوک‌هایی با اندازه‌ی کوچک یک نویسه‌ای یا تک بیت استفاده کرد. افزودن داده‌های ساختگی لازم نیست زیرا معمولاً اندازه‌ی بلوک متن عادی ثابت است (عدد ۸ برای یک نویسه یا عدد ۱ برای یک بیت).

حالت خاص

اگر بلوک‌ها در متن عادی و رمز پایه دارای اندازه‌ی یکسانی باشند ($n=R$) رمزنگاری/رمزگشایی ساده‌تر می‌شود. ارائه‌ی بلوک دیاگرام و الگوریتم را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

حالت بازخورد خروجی^۱

حالت بازخورد خروجی بسیار شبیه حالت CFB است، با این تفاوت که هر بیت در متن رمز، مستقل از بیت یا بیت‌های پیش از خود است. این مسئله باعث پیشگیری از انتشار خطا می‌شود. اگر خطایی در زمان انتقال رخ دهد، روی بیت‌های بعدی تأثیر نمی‌گذارد. توجه داشته باشید که مانند CFB فرستنده و گیرنده از الگوریتم رمزنگاری استفاده می‌کنند. شکل ۶-۸ حالت OFB را نشان می‌دهد.

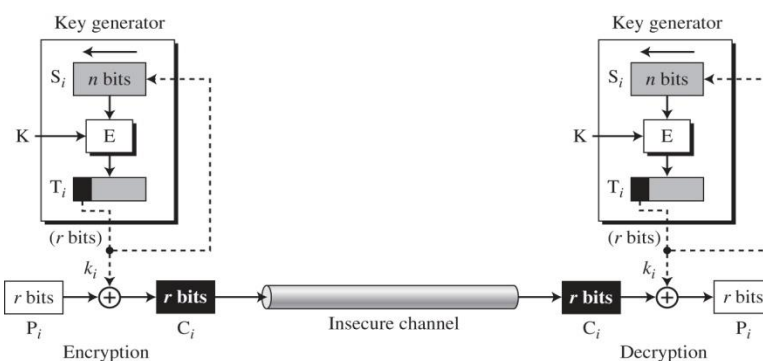


شکل ۶-۸: رمزنگاری در حالت بازخورد خروجی (OFB)

¹ Output Feedback (OFB)

به کارگیری OFB در رمز جریانی

OFB، مانند CFB، بر اساس رمز بلوکی پایه، رمز جریانی می‌سازد. با این وجود، جریان کلید به متن عادی یا متن رمز بستگی ندارد و همان‌گونه که در فصل ۵ گفتیم، این بدان معناست که رمز جریانی متقارن است. شکل ۷-۸ رمزنگاری و رمزگشایی را نشان می‌دهد که مولد کلید از پیش تعیین شده است.



شکل ۷-۸: حالت بازخور خروجی (OFB) به عنوان یک رمز جریانی

الگوریتم

الگوریتم ۸-۴ روتین برای رمزنگاری را ارائه می‌دهد. این الگوریتم، چندین روتین دیگر را فرا می‌خواند که جزئیات اجرای آن را به عنوان تمرین به عهده‌ی شما می‌گذاریم. توجه داشته باشید که الگوریتم را طوری نوشته‌ایم که ماهیت جریانی (موقعیت بلادرنگ) را نشان دهد. تا وقتی که بلوک متن عادی برای رمزنگاری وجود داشته باشد، الگوریتم ادامه می‌یابد.

```

OFB_Encryption (IV, K, r)
{
    i ← 1
    while (more blocks to encrypt)
    {
        input (Pi)
        if (i = 1) S ← IV
        else
        {
            Temp ← shiftLeftr (S)
            S ← concatenate (Temp, ki-1)
        }
        T ← EK (S)
        ki ← selectLeftr (T)
        Ci ← Pi ⊕ ki
        output (Ci)
        i ← i + 1
    }
}

```

الگوریتم ۸-۴: الگوریتم رمزگذاری برای OFB

دو مورد از مشکلات امنیت در حالت OFB به شرح زیر می باشد:

(۱) دقیقاً مانند حالت CFB، الگوها در سطح بلوکی حفظ نشده اند.

(۲) هر تغییر در متن رمز بر روی متن عادی رمزنگاری شده توسط دریافت کننده تأثیر می گذارد.

انتشار خطا

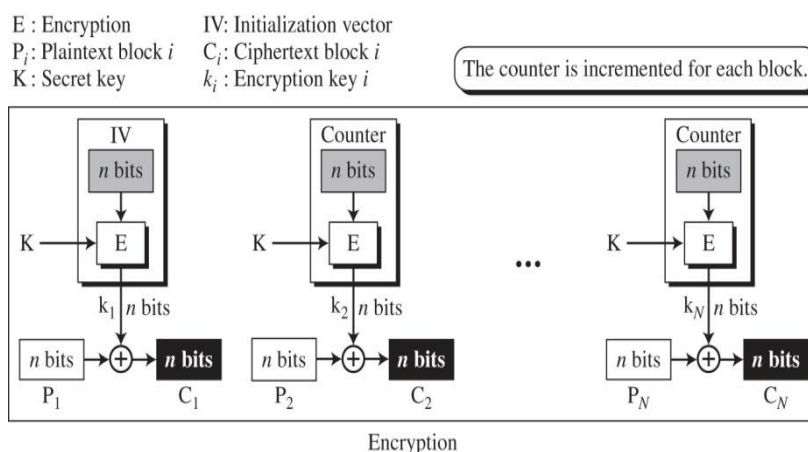
یک خطا در متن رمز فقط بر بیت متناظر آن در متن عادی اثر می گذارد.

حالت خاص

اگر بلوکها در متن عادی و رمز پایه اندازه ی یکسانی داشته باشند ($n=R$)، رمزنگاری و رمزگشایی ساده تر می شود. ارائه ی بلوک دیاگرام و الگوریتم مربوطه را به عنوان تمرین به عهده ی شما می گذارم.

حالت شمارنده^۱

در حالت شمارنده، هیچ بازخوردی وجود ندارد. با استفاده از شمارنده می توان به شبه اتفاقی بودن در جریان کلید دست یافت. مقدار عادی شمارنده n بیتی بر اساس مقدار از پیش تعیین شده (IV) مشخص می شود و بر اساس قاعده ی از پیش تعیین شده ($\text{mod } 2^n$) افزایش می یابد. برای دستیابی به تصادفی بودن بیشتر می توان مقدار افزایش را بر پایه ی عدد بلوکی که باید افزایش یابد، قرار داد. اندازه ی بلوک متن عادی و متن رمز با رمز پایه (مثلاً DES یا AES) یکسان است. بلوک های متن عادی با اندازه n برای تولید بلوک های متن رمز با اندازه n رمزنگاری می شوند. شکل ۸-۸ حالت شمارنده را نشان می دهد.



شکل ۸-۸: رمزگذاری در حالت شمارنده (CTR)

مقدار شمارنده برای هر بلوک افزایش می یابد.

^۱ Counter

رابطه‌ی بین بلوک‌های متن عادی و متن رمز به شکل زیر است.

$$\text{Encryption: } C_i = P_i \oplus E_{k_i}(\text{counter}) \quad \text{Decryption: } P_i = C_i \oplus E_{k_i}(\text{counter})$$

CTR هم برای رمزنگاری و هم برای رمزگشایی از تابع رمزنگاری رمز بلوکی پایه (E_k) استفاده می‌کند.

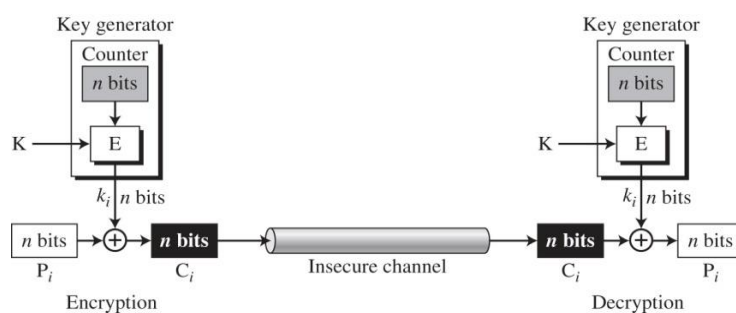
به سادگی می‌توان اثبات کرد که بلوک P_i متن عادی را می‌توان از C_i متن رمز بازیابی نمود. اثبات این مسئله را به عنوان تمرین به عهده‌ی شما می‌گذاریم.

می‌توانیم حالت CTR را با حالت‌های OFB و ECB مقایسه کنیم. CTR مانند OFB جریان کلیدی می‌سازد که به بلوک قبلی متن رمز بستگی ندارد، اما CTR از بازخورد استفاده نمی‌کند. ECB مانند CTR، n بیت بلوک متن رمز ایجاد می‌کند که مستقل از یکدیگرند و فقط به مقدار شمارنده بستگی دارند. با دید منفی، این بدان معنی است که از حالت CTR همانند حالت ECB، نمی‌توان برای پردازش بلادرنگ استفاده نمود. الگوریتم رمزنگاری باید پیش از آغاز رمزنگاری کل بلوک n بیتی داده‌ها را دریافت کند. با نگاه مثبت، حالت CTR همانند حالت ECB تا وقتی که بتوان مقدار شمارنده را به تعداد رکورد در فایل مرتبط ساخت، می‌توان از آن برای رمزنگاری و رمزگشایی فایل‌ها با دسترسی تصادفی استفاده نمود.

CTR در قالب رمز جریانی

CTR مانند CFB و OFB در واقع یک رمز جریان است (بلوک‌های متفاوت با کلیدهای متفاوت XOR می‌شوند).

شکل ۸-۹ رمزنگاری و رمزگشایی i امین بلوک داده را نشان می‌دهد.



شکل ۸-۹: حالت شمارنده CTR در قالب رمز جریانی

الگوریتم

الگوریتم ۵-۸ روتین رمزنگاری را به صورت شبه کد نشان می‌دهد؛ الگوریتم رمزگشایی را به عنوان تمرین به عهده‌ی شما می‌گذاریم. در این الگوریتم، مقدار افزایش به تعداد بلوک بستگی دارد؛ به عبارت دیگر،

مقدار شمارنده $IV, IV+1, IV+3, IV+6$ و الی آخر می‌باشد. همچنین فرض بر این است که تمام N بلوک متن عادی پیش از آغاز رمزنگاری آماده‌اند، اما می‌توان برای گریز از این فرض، الگوریتم را بازنویسی کرد.

چالش‌های امنیتی

چالش‌های امنیتی حالت CTR با حالت OFB یکسان است.

انتشار خطا

یک خطا در متن رمز فقط بر بیت متناظر آن بیت در متن عادی اثر می‌گذارد.

```

CTR_Encryption (IV, K, Plaintext blocks)
{
    Counter ← IV
    for (i = 1 to N)
    {
        Counter ← (Counter + i - 1) mod  $2^N$ 
         $k_i \leftarrow E_K(\text{Counter})$ 
         $C_i \leftarrow P_i \oplus k_i$ 
    }
    return Ciphertext blocks
}

```

الگوریتم ۸-۵: الگوریتم رمزنگاری برای CTR

مقایسه حالت‌های مختلف

جدول ۸-۱ پنج حالت مختلف عمل را که در این فصل بیان شده را به شکل گذرا مقایسه می‌کند.

جدول ۸-۱: خلاصه حالت‌های عمل

حالت عمل	توصیف	نوع نتیجه	اندازه واحد داده
ECB	هر بلوک n بیتی به طور مستقل با همان کلید رمزنگاری می‌شود	رمز بلوکی	n
CBC	مانند ECB است با این تفاوت که هر بلوک ابتدا با متن رمز پیشین XOR می‌شود.	رمز بلوکی	N
CFB	هر بلوک r بیتی با یک کلید r بیتی XOR می‌شود. این کلید r بیتی قسمتی از متن رمز پیشین است.	رمز جریانی	
OFB	مانند CFB است با این تفاوت که شیفت رجیستر به وسیله کلید r بیتی پیشین به‌روز می‌شود.	رمز جریانی	
CTR	مانند OFB است با این تفاوت که به جای شیفت رجیستر، از شمارنده استفاده می‌شود	رمز جریانی	n

۸-۲- کاربردهای رمزهای جریانی

اگرچه پنج حالت عمل استفاده از رمزهای بلوکی برای رمزنویسی پیام‌ها یا فایل‌ها در اندازه‌های بزرگ با استفاده از ECB, CBC, CRT و اندازه‌های کوچک با استفاده از CFB, OFB را فراهم نموده است، اما گاهی اوقات برای رمزنویسی واحدهای کوچک داده‌ها نظیر نویسه‌ها و بیت‌ها نیاز به جریان خالص^۱ می‌باشد. رمزهای جریانی برای پردازش بلادرنگ کارآمدترند. در خلال چند دهه اخیر، رمزهای جریانی متعددی در پروتکل‌های مختلف به کار رفته است. در این جا فقط به ذکر دو مورد اکتفا می‌کنیم: RC۴ و A۵/۱.

RC ۴

RC۴ رمز جریانی است که در سال ۱۹۸۴ توسط فردی به نام رونالد یوست برای تامین امنیت داده‌ای RSA طراحی شد. RC4 در بسیاری از پروتکل‌های شبکه و ارتباطات داده‌ای، از جمله در SSL/TLS و ۱۱. IEEE۸۰۲ استاندارد LAN بی‌سیم به کار رفته است.

RC۴ رمز جریانی بر پایه بایت است که در آن جهت ایجاد یک بایت از متن رمز، یک بایت (۸ بیت) از متن عادی با یک بایت از کلید XOR می‌شود. کلید سری، که کلیدهای یک بایتی در جریان کلید از آن مشتق می‌شود، می‌تواند از ۱ تا ۲۵۶ بایت را دربر گیرد.

مفهوم وضعیت

RC۴ مبتنی بر مفهوم وضعیت است. در هر لحظه، فقط یک وضعیت ۲۵۶ بایتی فعال است که یکی از بایت‌های آن به صورت تصادفی انتخاب می‌شود تا به عنوان کلید برای رمزنگاری مورد استفاده قرار گیرد. این مسئله را می‌توان به صورت آرایه‌ای از بایت‌ها نشان داد:

$$S[0] \quad S[1] \quad S[2] \quad \dots \quad S[255]$$

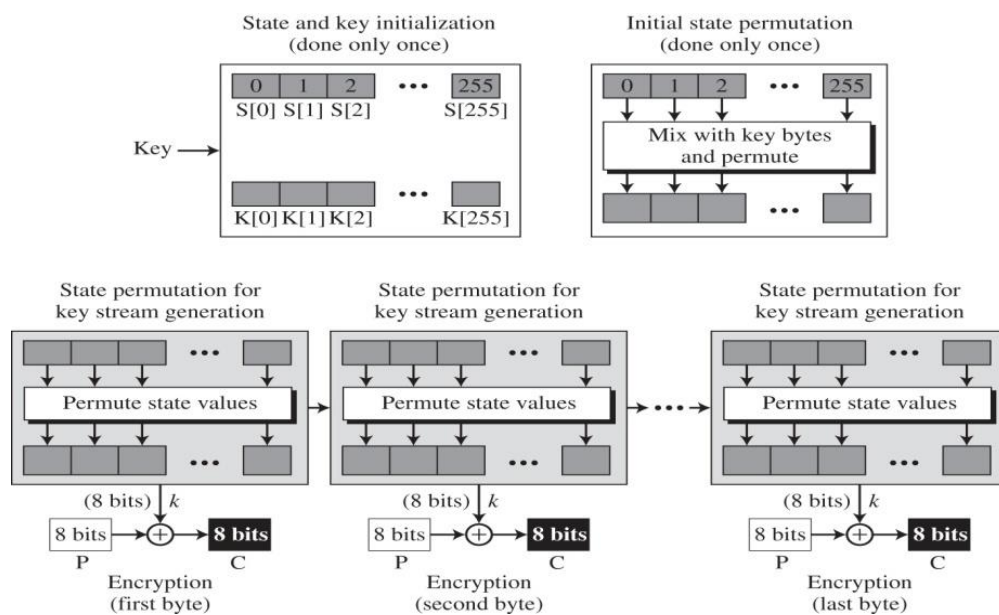
توجه داشته باشید که شاخص عناصر بین صفر تا ۲۵۵ متغیر است. محتوای هر بایت نیز ۸ بیتی است که می‌توان آن را به صورت عدد صحیحی بین صفر تا ۲۵۵ تفسیر کرد.

=====

^۱ Pure stream

هدف

شکل ۸-۱۰ ایده کلی RC۴ را نشان می‌دهد. دو جعبه اول فقط یک بار مقداردهی اولیه را انجام می‌دهد. برای تولید جریان کلید تا وقتی که بایت متن عادی برای رمزنگاری وجود داشته باشد، جایگشت تکرار می‌شود.



شکل ۸-۱۰: ایده‌ی رمز جریانی RC۴

مقداردهی اولیه: مقداردهی اولیه در دو مرحله انجام می‌گیرد.

(۱) در مرحله نخست، عبارت با مقادیر ۰، ۱، ...، ۲۵۵، آرایه‌ای از کلیدها به صورت $K[0], K[1], \dots, K[255]$ ایجاد می‌شود. اگر کلید سری دقیقاً ۲۵۶ بایت داشته باشد، بایت‌ها در آرایه‌ی K کپی می‌شوند؛ در غیر این صورت، بایت‌ها آن قدر تکرار می‌شود تا آرایه‌ی K پر شود.

for ($i = 0$ to 255)

```
{
S[i] ← i
K[i] ← key [i mod key length] }
```

(۲) در مرحله دوم، عبارت مقداردهی شده بر اساس مقدار بایت‌ها در $K[i]$ وارد یک جایگشت می‌شود (جایگزینی عناصر). فقط در این مرحله جهت تعیین این که کدام عضو باید جایگزین شود، بایت کلید مورد استفاده قرار می‌گیرد. پس از این مرحله، بایت‌های عبارت به طور کامل جابجا می‌شوند.

$z \leftarrow 0$

for { $(i=0 \text{ to } 255)$

$z \leftarrow (z + S[i] + K[i]) \bmod 256$

swap ($S[i], S[z]$)

}

تولید جریان کلید: کلیدها در جریان کلید به شکل $K[i]$ ، تک تک تولید می‌شود. نخست، بر اساس مقادیر عناصر وضعیت و مقادیر دو متغیر منفرد i و z عبارت جابجا می‌شود. دوم، مقادیر دو عضو عبارت در موقعیت i و z برای تعیین اندیس عضو وضعیتی که به عنوان K انتخاب می‌شود، به کار می‌رود. کد زیر برای هر بایت از متن عادی تکرار می‌شود و بدین ترتیب عضو کلید جدید در جریان کلید تولید می‌شود. مقدار اولیه‌ی متغیرهای i و z پیش از تکرار نخست، صفر در نظر گرفته می‌شود، اما مقادیر از تکرار پیشین به تکرار بعد کپی می‌شود.

$i \leftarrow (i+1) \bmod 256$

$z \leftarrow (z + S[i]) \bmod 256$

swap ($S[i], S[z]$)

$k \leftarrow s [(S[i] + S[z]) \bmod 256]$

رمزنگاری یا رمزگشایی: پیش از ایجاد K برای تولید بایت‌های متن رمز، بایت متن عادی رمزنگاری می‌شود. رمزگشایی در روال وارون مورد استفاده قرار می‌گیرد.

الگوریتم

الگوریتم ۸-۶ روتین پیش کد برای RC۴ را نشان می‌دهد.

```

RC4_Encryption (K)
|
| // Creation of initial state and key bytes
| for (i = 0 to 255)
| | S[i] ← i
| | K[i] ← Key [i mod KeyLength]
| |
| // Permuting state bytes based on values of key bytes
| j ← 0
| for (i = 0 to 255)
| | j ← (j + S[i] + K[i]) mod 256
| | swap (S[i] , S[j])
| |
| // Continuously permuting state bytes, generating keys, and encrypting
| i ← 0
| j ← 0
| while (more byte to encrypt)
| | i ← (i + 1) mod 256
| | j ← (j + S[i]) mod 256
| | swap (S [i] , S[j])
| | k ← S [(S[i] + S[j]) mod 256]
| | // Key is ready, encrypt
| | input P
| | C ← P ⊕ k
| | output C
| |
|
|

```

الگوریتم ۸-۶: الگوریتم رمزنگاری برای RC۴

مثال ۸-۵

برای نشان دادن تصادفی بودن کلید جریان، از کلیدی با بایت‌های مساوی * استفاده کردیم. جریان کلید برای ۲۰ عدد K به شکل زیر است:

(۲۲۲، ۲۴، ۱۳۷، ۶۵، ۱۶۳، ۵۵، ۹۳، ۵۸، ۱۳۸، ۶، ۳۰، ۱۰۳، ۸۷، ۱۱۰، ۱۴۶، ۱۰۹، ۱۹۹، ۲۶، ۱۲۷، ۱۶۳)

مثال ۸-۶

مثال ۸-۵ را تکرار کنید؛ فرض کنید کلید سری پیش‌بایت (۸، ۶، ۲۲، ۲۰۲، ۱۵) باشد. جریان کلید حاصل به شکل زیر است. باز هم تصادفی بودن در جریان کلید مشهود است.

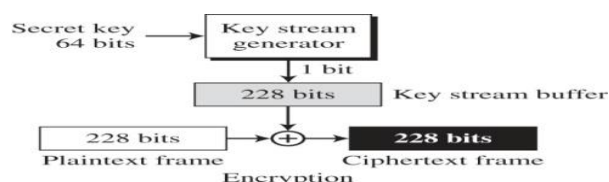
(۹۴، ۱۴۴، ۳۴، ۳۰، ۲۴۲، ۳۹، ۲۱۴، ۱۰۳، ۱۰۶، ۱۰۸، ۲۳۸، ۵۱، ۱۳۳، ۱۸۶، ۲۳۷، ۲۱۲، ۵۴، ۱۰۲، ۱۸۴، ۲۴۸)

چالش‌های امنیتی

چنین گفته می‌شود که اگر اندازه‌ی کلید حداقل ۱۲۸ بیت (۱۶ بایت) باشد، رمز مطمئن است. گزارش‌هایی در خصوص حملات انجام شده در اندازه‌های کوچک‌تر کلید (کمتر از ۵ بایت) وجود دارد، اما امروزه پروتکل‌هایی که از RC۴ استفاده می‌کنند، همه از اندازه‌های کلیدی استفاده می‌کنند که RC۴ را قابل اعتماد می‌سازد؛ با این وجود، مانند بسیاری از رمزهای دیگر، توصیه می‌شود که برای ارتباطات مختلف از کلیدهای مختلف استفاده شود. این کار باعث می‌شود رمزشکن نتواند تحلیل رمز دیفرانسیلی را بر روی رمز انجام دهد.

A۵/۱

در این قسمت، رمز جریانی‌ای به نام A۵/۱ را معرفی می‌کنیم که برای تولید جریان بیت‌ها، از شیفت رجستر پس‌خور خطی استفاده می‌کند (به فصل ۵ رجوع کنید). A۵/۱ (یکی از اعضای خانواده رمزهای A۵) در سیستم جهانی ارتباطات موبایل^۱ است؛ شبکه‌ای که برای ارتباطات تلفن همراه مورد استفاده قرار می‌گیرد. ارتباطات تلفنی در GSM به شکل سلسله‌ای از چارچوب‌های ۲۲۸ بیتی انجام می‌شود که هر چارچوب ۴/۶ هزارم ثانیه طول می‌کشد. A۵/۱ از کلید ۶۴ بیتی، یک جریان بیتی ساخته، سپس همان‌گونه که در شکل ۸-۱۱ نشان داده شده است، جریان‌های بیتی در بافر ۲۲۸ بیتی قرار می‌دهد تا با یک چارچوب ۲۲۸ بیتی XOR شوند.



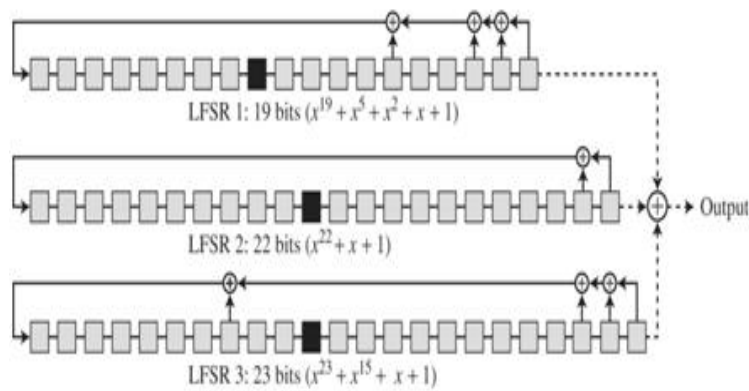
شکل ۸-۱۱: رئوس کلی A۵/۱

مولد کلید

A۵/۱ از سه شیفت رجستر پس‌خور خطی با ۱۹، ۲۲ و ۲۳ بیت استفاده می‌کند. در شکل ۸-۱۲ شیفت رجستر پس‌خور خطی، چندجمله‌ای‌های ویژه و بیت‌های clocking نشان داده شده است.

=====

¹ Global System for Mobile Communication (GSM)



شکل ۸-۱۲: شیفت رجستر پس‌خور خطی مورد استفاده در A5/۱

توجه: سه خانه سیاه در تابع اکثریت به کار رفته است.

خروجی یک بیتی به بافر ۲۲۸ بیتی داده می‌شود تا برای انجام رمزنگاری (یا رمزگشایی) مورد استفاده قرار گیرد.

مقداردهی اولیه: برای هر چارچوب در رمزنگاری (یا رمزگشایی)، مقداردهی اولیه انجام می‌شود. مقداردهی اولیه از کلید رمز ۶۴ بیتی و ۲۲ بیت از تعداد چارچوب مربوطه استفاده می‌کند. مراحل کار به شکل زیر است.

(۱) تمام بیت‌های موجود در LFSR ها را مساوی صفر قرار می‌دهیم.

(۲) بر اساس کد زیر، کلید ۶۴ بیتی را با مقدار رجستر مخلوط می‌کنیم؛ بدین معنی است که هر شیفت رجستر پس‌خور خطی از یک رجستر چرخش عبور داده می‌شود.

for ($i = 0$ to ۶۳)

{

Exclusive- or $K[i]$ with the left most bit in all three registers.

Clock all three LFSRs

}

(۳) سپس روال پیشین را تکرار می‌کنیم اما از تعداد چارچوب ۲۲ بیتی استفاده می‌کنیم.

for ($i = 0$ to ۲۱)

{

Exclusive- or FrameNumber [i] with the left most bit in all three registers.

Clock all three LFSRs

}

۴) برای ۱۰۰ چرخه، کل مولدها را clock می‌کنیم. اما برای این که ببینیم کدام شیفت رجستر پس-خور خطی باید clock شود، از تابع اکثریت استفاده می‌کنیم (به قسمت بعد نگاه کنید). توجه داشته باشید که clock در این جا یعنی این که گاهی اوقات دو و گاهی اوقات هر سه شیفت رجستر پس‌خور خطی از رجستر چرخش عبور می‌کنند.

for (i= ۰ to ۹۹)

{

Clock the whole generator based on the majority function

تمام مولدها را بر مبنای تابع اکثریت clock کن

}

تابع اکثریت: تابعی است که در (b_1, b_2, b_3) ، اگر اکثر بیت‌ها ۱ باشد، حاصل ۱ و اگر اکثر بیت‌ها صفر باشد، حاصل صفر است. به عنوان مثال $\text{Majority}(1, 0, 1) = 1$ اما $\text{Majority}(0, 0, 1) = 0$ است. پیش از هر تریگر زمانی، تابع اکثریت مقداری دارد؛ سه بیت ورودی را بیت‌های clocking می‌نامند. بیت‌های [۱۱] ۳ LFSR، [۱۱] LFSR_۲، [۱۰] LFSR_۱ به شرطی که بیت سمت راست، بیت صفر باشد. توجه داشته باشید که کتاب‌ها و نوشته‌ها با شمارش از سمت چپ، این بیت‌ها را ۱۰، ۱۰، ۸ می‌نامند، اما ما با شمارش از سمت راست، آن‌ها را ۱۱، ۱۱، ۱۰ می‌نامیم. به منظور هماهنگی و تطابق با چندجمله‌ای خاص، از این قرارداد استفاده می‌کنیم.

بیت‌های جریان کلید: مولد کلید در هر تریگر زمان یک بیت جریان کلید را تولید می‌کند. پیش از تولید کلید، تابع اکثریت اجرا می‌شود؛ سپس اگر بیت clock یک شیفت رجستر پس‌خور خطی با نتیجه‌ی تابع اکثریت تطابق داشته باشد، با آن شیفت رجستر پس‌خور خطی clock می‌شود.

مثال ۸-۷

در یک برهه‌ی زمانی، بیت‌های clocking ۱، ۰، ۱ هستند. کدام شیفت رجستر پس‌خور خطی clock

می‌شود (جابجا می‌شود)؟

حل

نتیجه $(1,0,1)=1$ Majority است پس شیفت رجستر پس‌خور خطی $LFSR^1$ و $LFSR^3$ جابجا می‌شود اما شیفت رجستر پس‌خور خطی $LFSR^2$ جابجا نمی‌شود.

رمزنگاری / رمزگشایی

جریان‌های بیت‌ی تولید شده از مولد کلید برای تشکیل کلید ۲۲۸ بیت‌ی بافر می‌شود. این کلید ۲۲۸ بیت‌ی به منظور تولید چارچوب متن رمز، با چارچوب متن عادی XOR می‌شود. هر بار رمزنگاری / رمزگشایی با یک چارچوب انجام می‌شود.

چالش‌های امنیتی

اگر چه GSM به استفاده از A5/1 ادامه می‌دهد، چندین حمله به GSM گزارش شده است. به ذکر دو مورد از آن‌ها اکتفا می‌کنیم. در سال ۲۰۰۰ میلادی آلکس بیروکوف، ادی شمیر و دیوید واگنر با یک حمله بلادرنگ^۱ نشان دادند که کلید را ظرف چند دقیقه می‌توان از متون عادی شناخته شده کوچکی به دست آورد؛ اما این حمله مستلزم اجرای مرحله‌ی پیش‌پردازش با 2^{48} مرحله می‌باشد. در سال ۲۰۰۳ میلادی اکدال و جانسون حمله‌ای را منتشر کردند که در زمان ۲ تا ۵ دقیقه‌ای توانست از متن عادی، رمز A5/1 را بشکند. با در نظر گرفتن حملات جدید، شاید لازم باشد GMS در آینده A5/1 را جایگزین یا تقویت نماید.

۸-۳- سایر چالش‌ها

رمزنویسی با استفاده از بلوک کلید متقارن یا رمزهای جریان‌ی مستلزم داشتن آگاهی در مورد سایر چالش‌ها می‌باشد.

مدیریت کلید

آلیس و باب برای این که بتوانند با استفاده از رمز کلید متقارن ارتباط مطمئن داشته باشند، می‌بایست کلید سری مشترکی را برای خودشان انتخاب کنند. اگر در یک گروه، n عضو وجود داشته باشد، باید هر یک از این اعضا با $n-1$ عضو دیگر ارتباط داشته باشد. بنابراین $n(n-1)$ کلید سری لازم است. با این وجود، در رمزنویسی کلید متقارن می‌توان از یک کلید برای هر دو طرف استفاده کرد (از آلیس به باب و از باب به

=====

¹ Real-Time Attack

آلیس) یعنی $n(n-1)/2$ کلید کافی است. اگر n حدود یک میلیون باشد، در این صورت تقریباً نیم میلیون کلید باید مبادله شود. از آن جا که این امر ممکن نیست، راه‌های دیگری پیش‌روست. نخست، هر بار که آلیس و باب می‌خواهند باهم ارتباط برقرار کنند می‌توانند کلید خاص همان نشست را موقتاً تولید کنند. دوم، می‌توان یک یا چند مرکز توزیع کلید در گروه، برای توزیع کلیدهای جلسه به اعضا دایر نمود. تمام این موارد قسمتی از مدیریت کلید محسوب می‌شود که پس از بررسی ابزارهای مورد نیازش، به آن خواهیم پرداخت.

تولید کلید

مسئله‌ی دیگر در رمزنویسی کلید متقارن، تولید کلید مطمئن است. رمزهای مختلف کلید رمز متقارن نیازمند کلیدهایی با اندازه‌های مختلف می‌باشند. برای پیشگیری از نشت مسائل امنیتی، کلید باید بر اساس رویکرد سامانمند انتخاب شود. اگر آلیس و باب کلید جلسه بین خودشان را ایجاد کنند، باید چنان به صورت تصادفی این کلید را انتخاب کنند که رمزشکن نتواند کلید مربوط به آلیس و باب را از کلید مربوط به جان و رمزشکن پیدا کند. این امر به معنی نیاز به مولد عدد تصادفی (یا شبه تصادفی) است. از آن جا که مبحث مولد اعداد تصادفی شامل موضوعاتی است که تاکنون درباره‌ی آن‌ها بحث نکرده‌ایم، پس مطالعه‌ی مولدهای عدد تصادفی را به عهده شما می‌گذاریم.

۸-۴- وب سایت‌های پیشنهادی

وب سایت‌های زیر، اطلاعات بیشتری در مورد موضوعات این فصل ارائه می‌دهد.

http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation

<http://www.itl.nist.gov/fipspubs/fip81.htm>

en.wikipedia.org/wiki/A5/1

en.wikipedia.org/wiki/RC4

۸-۵- چکیده

* در کاربردهای واقعی، متنی که باید رمزنویسی شود معمولاً اندازه‌ی متغیر دارد و اندازه‌ی آن بسیار بزرگ‌تر از اندازه‌ی بلوکی تعیین شده برای رمزهای بلوکی پیشرفته می‌باشد. حالت‌های عمل، برای رمزنویسی متن با هر اندازه‌ای با استفاده از رمزهای بلوکی پیشرفته اختراع شد. در این فصل، پنج حالت را بررسی کردیم.

* ساده‌ترین حالت عمل را حالت کتاب کد الکترونیک می‌نامند. متن عادی به n بلوک تقسیم می‌شود. اندازه‌ی هر بلوک، n بیت است. برای رمزنگاری و رمزگشایی هر بلوک، کلید یکتا به کار می‌رود.

* در حالت زنجیره‌ای بلوک رمز، هر بلوک متن عادی پیش از این که رمزنگاری شود با بلوک متن رمز پیشین XOR می‌شود. وقتی یک بلوک رمزنویسی شد، بلوک ارسال می‌شود؛ اما یک کپی از آن برای استفاده در رمزنگاری بلوک بعدی در حافظه می‌ماند. فرستنده و گیرنده بر سر یک بردار مقداردی اولیه (IV) از پیش تعیین شده‌ای توافق می‌کنند. این بردار با بلوک اول متن رمز XOR می‌شود.

* برای رمزنویسی واحدهای داده‌ای کوچک در پردازش بلادرنگ، حالت بازخورد رمز ارائه شد. CFB از یکی از رمزهای بلوکی استاندارد همچون DES یا AES برای رمزنگاری شیفت ریجستر استفاده می‌کند. اما برای رمزنگاری و رمزگشایی واحدهای داده‌ای واقعی از XOR استفاده می‌کند. حالت CFB از رمزهای بلوکی استفاده می‌کند، اما حاصل آن رمز جریانی است زیرا هر واحد داده‌ای با کلید متفاوت رمزنویسی می‌شود.

* حالت بازخورد خروجی بسیار شبیه به حالت CFB است با این تفاوت که هر بیت در متن رمز به بیت یا بیت‌های قبل بستگی ندارد. این امر باعث جلوگیری از انتشار خطا می‌شود. به جای استفاده از بلوک متن رمز قبلی، OFB از کلید قبلی به عنوان بازخورد استفاده می‌کند.

* در حالت شمارنده، بازخوردی وجود ندارد. با استفاده از شمارنده، شبه تصادفی بودن در کلید تحقق می‌یابد. مقدار اولیه شمارنده‌ی n بیتی با مقدار از پیش تعیین شده‌ی IV مشخص می‌شود و بر اساس قاعده‌ی از پیش تعریف شده‌ای افزایش می‌یابد.

* برای رمزنویسی واحدهای کوچک داده، مانند نویسه‌ها و بیت‌ها، از دیرباز رمزهای جریانی گوناگونی طراحی شده است. این رمزهای جریانی برای پردازش بلادرنگ بسیار کارآمدترند. در این فصل، فقط در مورد دو رمز جریانی خالص بحث شد: A5/1 و RC4.

* RC۴ رمز جریانی مبتنی بر بایت است که در یک بایت (۸ بیت) از متن عادی با یک بایت از کلیدی XOR می‌شود و بدین ترتیب یک بایت از متن رمز ایجاد می‌گردد. کلید سری، که از آن کلیدهای یک بایتی در جریان کلید تولید می‌شود، در هر جا می‌تواند شامل ۱ تا ۲۵۶ بایت باشد. مولد جریان کلید مبتنی بر جایگشت، وضعیت ۲۵۶ بایتی است.

* A۵/۱ رمز جریان است که در ارتباطات تلفن همراه به کار می‌رود. A۵/۱ با استفاده از سه شیفت رجستر پس‌خور خطی، جریان بیتی از کلید ۶۴ بیتی تولید می‌کند.

۸-۶- مجموعه تمرین‌ها

پرسش‌های دوره‌ای

- ۱) توضیح دهید که چرا در صورت استفاده از رمزهای بلوکی پیشرفته در رمزنویسی، حالت‌های عمل لازم است؟
- ۲) پنج حالت عمل مشروح در این فصل را نام ببرید.
- ۳) ECB را تعریف کرده و مزایا و معایب آن را نام ببرید.
- ۴) CBC را تعریف کرده و مزایا و معایب آن را نام ببرید.
- ۵) CFB را تعریف کرده و مزایا و معایب آن را نام ببرید.
- ۶) OFB را تعریف کرده و مزایا و معایب آن را نام ببرید.
- ۷) CTR را تعریف کرده و مزایا و معایب آن را نام ببرید.
- ۸) پنج حالت عمل را به دو گروه تقسیم کنید: آن‌هایی که از توابع رمزنگاری و رمزگشایی رمز پایه (مثلاً DES و AES) استفاده می‌کنند و آن‌هایی که فقط از تابع رمزنگاری استفاده می‌کنند.
- ۹) پنج حالت عمل را به دو گروه تقسیم کنید: آن‌هایی که افزودن داده‌های ساختگی لازم دارند و آن‌هایی که نیاز ندارند.
- ۱۰) پنج حالت عمل را به دو گروه تقسیم کنید: آن‌هایی که از کلید برای رمزنویسی تمام بلوک‌ها استفاده می‌کنند و آن‌هایی که از جریان کلیدها برای رمزنویسی بلوک‌ها استفاده می‌کنند.
- ۱۱) تفاوت اصلی بین A5/1 و RC4 را توضیح دهید. کدام یک از شیفت رجستر پس‌خور خطی استفاده می‌کند؟
- ۱۲) اندازه‌ی واحد داده‌ها در RC4 چقدر است؟ اندازه‌ی واحد داده‌ها در A5/1 چیست؟
- ۱۳) حالت‌های عملی را نام ببرید که با پردازش موازی تسریع می‌شوند.
- ۱۴) حالت‌های عملی را نام ببرید که می‌توان از آن‌ها برای رمزنویسی فایل‌ها با دسترسی تصادفی استفاده کرد.
- ۱۵) نشان دهید که چرا حالت CFB، رمز جریانی غیر همزمان ایجاد می‌کند اما OFB رمز جریانی همزمان؟
- ۱۶) در حالت CFB، چرا بسیاری از بلوک‌ها تحت تأثیر یک خطا در انتقال قرار می‌گیرند؟
- ۱۷) در حالت ECB، بیت ۱۷ در بلوک ۸ متن رمز در طول انتقال خراب شده است. بیت‌های خراب شده‌ی احتمالی در متن عادی را پیدا کنید.

- (۱۸) در حالت CBC، بیت‌های ۱۷ و ۱۸ در بلوک ۹ متن رمز در حین انتقال خراب شده است. بیت‌های خراب شده احتمالی در متن عادی را پیدا کنید.
- (۱۹) در حالت CFB، بیت‌های ۳ و ۶ در بلوک ۱۱ متن رمز خراب شده است ($r=8$). بیت‌های خراب شده احتمالی در متن عادی را پیدا کنید.
- (۲۰) در حالت CTR، بلوک‌های ۳ و ۴ به کلی خراب شدند. بیت‌های خراب شده احتمالی در متن عادی را پیدا کنید.
- (۲۱) در حالت OFB، کل بلوک ۱۱ متن رمز خراب شده است ($r=8$)، بیت‌های خراب شده احتمالی در متن عادی را پیدا کنید.
- (۲۲) ثابت کنید که باب می‌تواند در حالت CFB متن عادی آلیس را بازیابی کند.
- (۲۳) ثابت کنید که باب می‌تواند در حالت OFB متن عادی آلیس را بازیابی کند.
- (۲۴) ثابت کنید که باب می‌تواند در حالت CTR متن عادی آلیس را بازیابی کند.
- (۲۵) دیاگرام رمزنگاری و رمزگشایی در حالت CFB وقتی که $r=n$ است را ترسیم نمایید.
- (۲۶) دیاگرام رمزنگاری و رمزگشایی در حالت OFB وقتی که $r=n$ است را ترسیم نمایید.
- (۲۷) اگر متن رمز (CTS) سرقت شده باشد، روال استفاده شده برای الگوریتم رمزگشایی در حالت ECB را نشان دهید.
- (۲۸) دیاگرام رمزنگاری و رمزگشایی برای حالت ECB (فقط دو بلوک آخر) را وقتی که متن رمز (CTS) سرقت شده باشد را نشان دهید.
- (۲۹) اگر متن رمز (CTS) سرقت شده باشد، روال‌های استفاده شده برای الگوریتم رمزگشایی در حالت CBC را نشان دهید.
- (۳۰) دیاگرام رمزنگاری و رمزگشایی برای حالت CBC (فقط دو بلوک آخر) را وقتی که متن رمز (CTS) سرقت شده باشد را نشان دهید.
- (۳۱) توضیح دهید که چرا در حالت‌های CFB، OFB و CTR نیاز به سرقت متن رمز نیست.
- (۳۲) وقتی که ECB از شیوهی CTS استفاده می‌کند، تأثیر انتشار خطا را نشان دهید.
- (۳۳) وقتی که CBC از شیوهی CTS استفاده می‌کند، تأثیر انتشار خطا را نشان دهید.
- (۳۴) حالت زنجیره‌ای بلوک (BC) نوعی CBC است که در آن تمام بلوک‌های متن رمز قبلی پیش از رمزنگاری با بلوک متن عادی فعلی XOR می‌شود. دیاگرامی رسم کنید که رمزنگاری و رمزگشایی آن را نشان دهد.

(۳۵) حالت رنجیره‌ای بلوک رمز انتشاری (PCBC) نوعی CBC است که در آن هم بلوک متن عادی پیشین و هم بلوک متن رمز قبلی، پیش از رمزنگاری با بلوک متن عادی فعلی XOR می‌شوند. دیاگرامی رسم کنید که رمزنگاری و رمزگشایی آن را نشان دهد.

(۳۶) حالت زنجیره‌ای بلوک رمز دارای کنترل (CBCC) نوعی CBC است که در آن تمام بلوک‌های متن عادی قبلی، پیش از رمزنگاری با بلوک متن عادی فعال XOR می‌شود. دیاگرامی رسم کنید که رمزنگاری و رمزگشایی و همچنین گردش آن را نشان دهد.

(۳۷) در RC۶، اگر کلید سری فقط ۷ بایت با مقادیر ۱ و ۲ و ۳ و ۴ و ۵ و ۶ و ۷ باشد، ۲۰ عضو اول جریان کلید را نشان دهید. شاید برای انجام این کار لازم باشد برنامه کوچکی بنویسید.

(۳۸) در RC۴، مقداری برای کلید سری پیدا کنید که پس از مراحل مقداردهی اولیه اول و دوم، عبارت را تغییر ندهد.

(۳۹) آلیس و باب برای پنهان کاری بیشتر در ارتباطاتشان از رمز RC۴ با کلید سری ۱۶ بایتی استفاده می‌کنند. این کلید سری هر بار با استفاده از تعریف بازگشتی $K_i = (K_{i-1} + K_{i-2}) \bmod 2^{128}$ تغییر می‌کند. نشان دهید که آن‌ها می‌توانند پیش از این که این الگو دوباره تکرار شود، چند پیام رد و بدل کنند.

(۴۰) در A۵/۱، حداکثر دوره‌ی هر شیفت رجستر پس‌خور خطی را پیدا کنید.

(۴۱) در A۵/۱، مقدار توابع زیر را به دست آورید. در هر مورد نشان دهید که چند شیفت رجستر پس‌خور خطی clock می‌شود.

- a) Majority (۱, ۰, ۰)
- b) Majority (۰, ۱, ۱)
- c) Majority (۰, ۰, ۰)
- d) Majority (۱, ۱, ۱)

(۴۲) در A۵/۱، عبارتی برای تابع Majority بیابید.

(۴۳) الگوریتم رمزگشایی در شبه کد برای حالت ECB را بنویسید.

(۴۴) الگوریتم رمزگشایی در شبه کد برای حالت CBC را بنویسید.

(۴۵) شبه کد الگوریتم رمزگشایی برای حالت CFB را بنویسید.

(۴۶) الگوریتم رمزگشایی در شبه کد برای حالت OFB را بنویسید.

(۴۷) الگوریتم رمزگشایی در شبه کد برای حالت CTR را بنویسید.

- (۴۸) الگوریتمی برای روتین ShiftLeft به کار رفته در الگوریتم ۸-۴ بنویسید.
- (۴۹) الگوریتمی برای روتین SelectLeft به کار رفته در الگوریتم ۸-۴ بنویسید.
- (۵۰) الگوریتمی برای روتین پیوند^۱ به کار رفته در الگوریتم ۸-۴ بنویسید.

پایان

=====

^۱ Concatenate