دانشگاه تهران
پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

عنوان

مباحثی در حمله‌های جبری به سامانه‌های رمزنگاری

نگارنده
حسین هادی‌پور

استاد راهنما
دکتر حسین سبزرو

استاد مشاور
دکتر حسین حاجی‌ابوالحسن

پایان‌نامه
جهت دریافت درجه کارشناسی‌ارشد
در رشته ریاضی محض

شهریور ۱۳۹۵

بسم الله الرحمن الرحیم

دانشگاه تهران

پردیس علوم

پایان‌نامه برای دریافت درجه‌ی کارشناسی ارشد در رشته ریاضی محض

عنوان: مباحثی در حمله‌های جبری به سامانه‌های رمزنگاری

نگارنده: حسین هادی‌پور

این پایان‌نامه در تاریخ ۱۳۹۵/۰۶/۳۱ در مقابل هیات داوران دفاع گردید و مورد تصویب قرار گرفت.

معاون آموزشی و تحصیلات تکمیلی پردیس علوم: دکتر معصومه ملک

رییس دانشکده‌ی ریاضی، آمار و علوم کامپیوتر: دکتر حمید پزشک

استاد راهنما: دکتر حسین سبزرو

استاد مشاور: دکتر حسین حاجی ابوالحسن

عضو هیات داوران: دکتر مرتضی محمدنوری

عضو هیات داوران: دکتر حسین حاجی‌ابوالحسن

تعهدنامه‌ی اصالت اثر

اینجانب حسین هادی‌پور تایید می‌کنم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب است و به دستاوردهای پژوهشی دیگران که در این نوشته از آن‌ها استفاده شده است، مطابق مقررات ارجاع گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارایه نشده است.

حسین هادیپور
۱۳۹۵/۰۶/۳۱

تقدیم به پدر و مادرم

پروردگارا، نه می‌توانم موهایشان را که در راه عزت من سفید شد، سیاه کنم و نه برای دست‌های پینه بسته‌شان که ثمره تلاش برای افتخار من است، مرهمی دارم. پس توفیقم ده که هر لحظه شکر گزارشان باشم و ثانیه‌های عمرم را در عصای دست بودنشان بگذرانم.

سپاس گزاری

سپاس و ستایش تنها شایسته‌ی خدایی است که هیچ رمزی برایش پوشیده نیست، اما به مصداق «من لم یشکر المخلوق لم یشکر الخالق» بر خود فرض می‌دانم، قدردان زحمات کسانی باشم که در تمام مراحل تهیه این پایان‌نامه، یار و راهنمای من بوده‌اند. در ابتدا از پدر و مادر عزیزم که در تمام مراحل زندگی همراه و غم‌خوار من بوده‌اند تشکر می‌کنم. سپس از استاد راهنمایم **جناب آقای دکتر حسین سبزرو**، که با راهنمایی‌ها و دقت‌نظرهای خود یاری‌رسان بنده در تهیه این پایان‌نامه بوده‌اند و به‌خاطر صبر و تحمل ایشان در این مدت، تشکر و قدردانی می‌کنم. همچنین از همراهی صمیمانه **جناب آقای دکتر حسین حاجی‌ابوالحسن** که استاد مشاور بنده در تهیه این پایان‌نامه بوده‌اند سپاس گزاری می‌کنم.

چکیده

در این پایان‌نامه، بر پایه‌ی [۴۳]، گردایه‌ای از روش‌های جبری در رمزشکنی را ارائه می‌دهیم. بعد از معرفی مقدمات رمزنگاری و مفاهیم اولیه‌ی حمله‌های جبری و بررسی چندین سناریوی حمله جبری روی سامانه‌های رمزنگاری متقارن و نامتقارن، به مطالعه‌ی تعدادی از روش‌های خاص می‌پردازیم. به‌ویژه حمله‌های XL، XSL و MutantXL را مورد مطالعه قرار می‌دهیم که بر پایه‌ی روش‌های خطی‌سازی دستگاه‌های معادلات چندجمله‌ای چندمتغیره هستند. در ادامه حمله‌های جبری مبتنی بر پایه گروبنر و پایه‌های مرزی و نوع دیگری از حمله‌های جبری که بر مبنای روش‌های برنامه‌ریزی با عدد صحیح و مسئله صدق‌پذیری هستند را مطالعه می‌کنیم.

کلمات کلیدی:
سامانه‌های رمزنگاری، حمله‌های جبری، حل دستگاه معادلات چندجمله‌ای، بازخطی سازی، پایه‌های گروبنر، پایه‌های مرزی، برنامه‌ریزی با عدد صحیح، مسئله صدق‌پذیری

پیشگفتار

الف لام میم است آغاز کار
که رمزیست از سوی پروردگار
«امید مجد»

محور اصلی مباحث این پایان‌نامه بر اساس مقاله
Kreuzer, Martin. *Algebraic attacks galore!*. Groups–Complexity–Cryptology 1, no. **2** (2009), 231-259.
است.

علم رمزشناسی دارای دو وجه است، یک وجه آن شامل طرّاحی سامانه‌ها و پروتکل‌های رمزنگاری و وجه دیگر آن علم یا هنر تحلیل رمز یا رمزشکنی است، که در آن به مطالعه روش‌هایی برای شکستن سامانه‌ها و پروتکل‌های رمزنگاری می‌پردازند. تمرکز ما در این پایان‌نامه بر روی وجه دوّم است. منظور از شکستن یک سامانه رمزنگاری در ادبیات رمزنگاری، به‌دست آوردن کلید یا متن اصلی با تلاشی کمتر از جست‌وجوی فراگیر فضای کلید است، و به هر تلاشی برای شکستن یک سامانه حمله اتلاق می‌شود.

از آنجایی‌که هر نگاشت رمزنگاری بین فضاهای برداری با بعد متناهی، روی میدان متناهی را می‌توان به صورت یک نگاشت چندجمله‌ای نوشت، بیان مسئله حمله به یک سامانه رمز به صورت مسئله حل دستگاه معادلات چندجمله‌ای امری دور از انتظار نیست و به چنین رویکردی برای حمله به یک سامانه رمزنگاری *حمله جبری* گفته می‌شود.

حمله‌های جبری جزء حمله‌های شناخته شده دنیای رمزنگاری است و تا کنون تحقیقات زیادی در این زمینه صورت گرفته است، به‌طوری که حتی پایه‌گذاران رمزنگاری مدرن نیز ایده اصلی حمله‌های جبری را می‌شناختند. برای مثال شانون در مقاله معروف خود [۶۰] که در سال ۱۹۴۹ منتشر شد جمله‌ای با این مضمون دارد، که شکستن یک سامانه رمزنگاری خوب، نیازمند حداقل همان میزان تلاش برای حل یک دستگاه معادلات پیچیده با تعداد مجهولات فراوان است. با وجود مؤفقیت‌های به‌دست آمده، این حوزه هنوز مسائل باز و حل‌نشده‌ی فراوانی دارد. ایده اصلی حمله‌های جبری را می‌توان در سه گام زیر خلاصه کرد

۱. به‌دست آوردن روابط چندجمله‌ای بین متغیرهای به‌کار رفته در یک سامانه رمزنگاری و

تبدیل نگاشت رمزنگاری و رمزگشایی به نگاشت‌های چندجمله‌ای

۲. جایگذاری مقادیر معلوم در روابط به‌دست آمده و تشکیل یک دستگاه معادلات چندجمله‌ای روی یک میدان متناهی

۳. حلّ دستگاه معادلات به‌دست آمده

در این پایان‌نامه سعی بر این بوده، تا گردایه‌ای از حمله‌های جبری را مورد مطالعه و ارزیابی قرار دهیم. در این راستا در فصل اول یک آشنایی مختصر با رمزنگاری خواهیم داشت، سپس در فصل ۲ پایه گروبنر و پایه مرزی را معرفی می‌کنیم که از ابزارهای ما در حمله به سامانه‌های رمزنگاری هستند.

در فصل ۳ بعد از این که ثابت کردیم هر نگاشت رمزنگاری را می‌توان به یک نگاشت چندجمله‌ای تبدیل کرد، نحوه‌ی انجام این‌کار را نیز نشان می‌دهیم. در این رابطه الگوریتم بوخبرگر مولر [۱] را که از آن می‌توان برای یافتن روابط چندجمله‌ای بین بیت‌های کلید و متن رمز شده، یا روابط چندجمله‌ای بین بیت‌های متن اصلی و متن رمزشده استفاده کرد، معرفی می‌کنیم. در ادامه این فصل سناریوهای عمومی حمله جبری به سامانه‌های رمزنگاری متقارن و نامتقارن را با ذکر مثال‌هایی از جبری‌سازی الگوریتم‌های رمزنگاری واقعی، بررسی خواهیم کرد.

پس از تبدیل سامانه رمزنگاری به دستگاه معادلات چندجمله‌ای نوبت به بررسی الگوریتم‌های حل این دستگاه‌ها می‌رسد که در فصل ۴ به آن پرداخته‌ایم. این فصل را با روش‌های مبنی بر خطی‌سازی آغاز می‌کنیم و حمله‌ی XL را معرفی خواهیم کرد. این الگوریتم که از روش خطی‌سازی برای تحویل دستگاه چندجمله‌ای به یک دستگاه خطی استفاده می‌کند، اولین بار توسط شمیر [۲] و کیپنیس [۳] در [۴۲] معرفی شد. بر خلاف برخی از امیدهای اولیه پیچیدگی محاسباتی این روش زیرنمایی نبود و حافظه زیادی مصرف می‌کرد. پس از روشن شدن نقایص XL، یکی از اولین پیشنهادها برای بهبود آن، الگوریتم XSL بود که توسط کورتوا [۴] و پیپشیک [۵] در [۲۲] ارائه شد و موضوع بحث بعدی ما در این فصل است. اگر چه این روش با بهره گیری از ویژگی کم‌پشتی و ساختار خاص دستگاه‌هایی که از برخی سامانه‌های رمزنگاری به‌دست می‌آیند، در بهبود روش XL موفق بوده است، ولی تنها برای دسته‌ی خاصی از سامانه‌ها موسوم به سامانه‌های XSL مناسب است و یک روش کلی برای حل تمام دستگاه‌ها تلقی نمی‌شود. علاوه بر این بحث‌های زیادی بر سر کارایی این روش وجود دارد و عده‌ای از رمزنگارها اعتقادی به کارایی این روش ندارند. یک پیشنهاد دیگر برای بهبود الگوریتم XL الگوریتم MutantXL از دینگ [۶] است که آن را نیز در این

[1]Buchberger-Möller
[2]A. Shamir
[3]A. Kipnis
[4]Courtois
[5]Pieprzyk
[6]J. Ding

فصل معرفی می‌کنیم.

در ادامه‌ی فصل ۴، حمله‌های پایه گروبنر و پایه مرزی را مورد بررسی قرار می‌دهیم. بهترین الگوریتم‌ها برای یافتن پایه گروبنر تاکنون الگوریتم‌های F4 و F5 از فوجر [7] در [۳۰] و [۳۱] هستند که حمله‌های مبنی بر پایه گروبنر در رمزنگاری، از این الگوریتم‌ها برای یافتن پایه گروبنر و حل دستگاه استفاده می‌کنند. محبوبیت این الگوریتم‌ها پس از مؤفقیت آن‌ها در شکستن چالش HFE 80 افزایش یافته است. به‌ویژه پیشرفت‌های اخیر و بهینه‌سازی‌هایی که روی این الگوریتم‌ها صورت گرفته آن‌ها را قدرتمندتر ساخته و امنیت چندین سامانه‌ی امضای دیجیتال و رمز دنباله‌ای به واسطه‌ی آن‌ها تحت خطر جدی قرار گرفته است. در ادامه‌ی این فصل الگوریتم بهبودیافته پایه مرزی را، به عنوان روشی دیگر برای حل دستگاه‌ها و حمله به سامانه‌های رمزنگاری معرفی می‌کنیم. گرچه تا کنون هیچ حمله‌ی مؤفقی با استفاده از پایه‌های مرزی گزارش نشده ولی به نظر می‌رسد الگوریتم‌های محاسبه پایه مرزی اگر به همان اندازه الگوریتم‌های پایه گروبنر مورد توجه قرار گیرند، می‌توانند روشی چه‌بسا بهتر از روش پایه گروبنر، برای حمله به سامانه‌های رمزنگاری باشند.

در ادامه فصل ۴ به معرفی حمله‌های جبری مبنی بر برنامه‌ریزی خطی با عدد صحیح می‌پردازیم. در این بخش الگوریتمی برای تبدیل دستگاه معادلات چندجمله‌ای روی $\mathbb{GF}(۲)$ به مسئله برنامه‌ریزی خطی با متغیرهای صحیح معرفی می‌کنیم، تا بدین ترتیب بتوانیم از حل‌کننده‌های قدرتمند مسائل برنامه‌ریزی خطی نیز برای حمله به سامانه‌های رمزنگاری استفاده کنیم. در پایان به یکی از معروف‌ترین مسائل علوم کامپیوتر یعنی مسئله صدق‌پذیری پرداخته‌ایم و بعد از معرفی الگوریتمی برای تحویل مسئله حل دستگاه چندجمله‌ای به مسئله صدق‌پذیری، سعی کرده‌ایم تا قدرت حل‌کننده‌های مسئله صدق‌پذیری را در شکستن سامانه‌های رمزنگاری مورد آزمون قرار دهیم.

[7] Fauger

فهرست مطالب

فهرست الگوریتم‌ها

فهرست جداول

فهرست تصاویر

فصل ۱

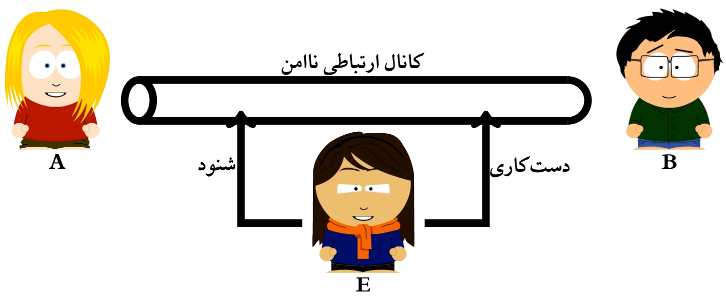
آشنایی با رمزنگاری

هدف از این فصل آشنایی با مفاهیم اولیه‌ی رمزنگاری نوین است. پس از مروری مختصر بر سیر تکاملی رمزنگاری، مفهوم امنیت را به‌صورت دقیق تعریف کرده و اهداف اصلی رمزنگاری را برمی‌شماریم. در ادامه ابزارهای اولیه‌ی رمزنگاری برای دست‌یابی به دو هدف اصلی، یعنی محرمانگی و احراز اصالت را معرفی می‌کنیم. برای آشنایی کامل با رمزنگاری نوین و مشاهده جزئیات بیشتر در رابطه با مفاهیم این فصل، می‌توانید به [۴۰]، رجوع کنید.

۱.۱ رمزنگاری درباره چیست؟

از نظر تاریخی رمزنگاری برای این بوجود آمد که طرفین یک ارتباط بتوانند محرمانگی ارتباط خود را حتی در حضور یک مهاجم که به کانال ارتباطی آن‌ها دسترسی دارد حفظ کنند. ولی اهداف رمزنگاری نوین، نه‌تنها *محرمانگی* بلکه *صحّت یا جامعیّت* و *احراز هویّت* و بسیاری از اهداف پیچیده و البته شگفت‌انگیز دیگر را در برمی‌گیرد.

مدل ساده شکل ۱.۱ را در نظر بگیرید که A و B قصد دارند از طریق یک کانال با هم ارتباط



شکل ۱.۱: مدل یک کانال ناامن[۳۷]

برقرار کنند. اگر کانال ارتباطی بین این دو، نفوذناپذیر باشد و هیچ شخص سومی نتواند به آنچه

در این کانال منتقل می‌شود دسترسی داشته باشد چنین کانالی یک کانال ایده‌ال است و در نتیجه پیام‌ها به صورت محرمانه و با صحّت کامل به مقصد می‌رسند و نیازی به رمز کردن آن‌ها نیست. ولی در دنیای واقعی هنوز کانال ایده‌ال وجود ندارد و ارتباط از طریق یک کانال مثل اینترنت یا امواج رادیویی که برای دیگران نیز در دسترس است برقرار می‌شود. برای مدل کردن خطرات متوجه یک کانال ارتباطی غیر ایده‌ال، شخص سوّمی به نام مهاجم را معرّفی می‌کنیم که در شکل با E نشان داده شده و می‌تواند ضمن شنود کانال، پیام ارسالی در کانال را تغییر دهد.

به طور کلّی مهاجم یا دشمن در رمزنگاری به عنوان منبع تمام خطرات احتمالی متوجه امنیت در نظر گرفته می‌شود و در عمل می‌تواند یک شخص متخاصم یا یک برنامه‌ی کامپیوتری مخرب، شرکت رقیب، گروهی از هکرها و یا یک سازمان جنایی باشد.

ابزار اصلی رمزنگاری برای غلبه بر مهاجم پروتکل‌های رمزنگاری هستند، پروتکل‌های رمزنگاری مجموعه‌ای از الگوریتم‌های رمزنگاری به انضمام قراردادهایی هستند که مشخص می‌کنند طرفین ارتباط چطور باید عمل کنند، ولی این که مهاجم چطور عمل می‌کند را مشخّص نمی‌کنند. پروتکل‌های رمزنگاری در اصل برنامه‌های رایانه‌ای توزیع شده هستند و به طور مختصر، رمزنگاری درباره طراحی و تحلیل پروتکل‌های رمزنگاری با هدف شکست دادن مهاجم است.

رمزنگاری قوانینی دارد، اولین قانون این است که ما برای غلبه بر مهاجم فقط مجازیم از پروتکل‌ها استفاده کنیم، ما اجازه نداریم با تهدید، دستگیری یا سم ریختن در چای مهاجم بر او چیره شویم، این روش‌ها شاید مؤثر واقع شود ولی از علم رمزنگاری به‌دور است!

قانون دوّم که اکثر رمزنگارها بر آن اصرار دارند، عمومی بودن پروتکل‌های رمزنگاری است. تنها چیزی که باید مخفی بماند کلید رمزنگاری است که البته کلید برخلاف الگوریتم‌ها و پروتکل‌ها یک نوع داده محسوب می‌شود و امنیت سامانه تنها باید متکی به مخفی بودن کلید باشد.

اهداف رمزنگاری

اهداف رمزنگاری را می‌توانیم به سه هدف زیر تقسیم کنیم:

۱. **محرمانگی** مهاجم با دیدن پیام رمزشده نتواند اطلاعات جدیدی راجع به متن اصلی به‌دست آورد.

۲. **صحّت یا جامعیّت** اگر پیام ارسالی تغییر کرد، گیرنده‌ی پیام متوجه شود.

۳. **احراز هویّت** گیرنده‌ی پیام بتواند از هویّت فرستنده اطمینان حاصل کند و فرستنده نتواند خود را جای کس دیگری معرفی کند.

ابزارهای لازم برای رسیدن به سه هدف فوق، ابزارهای پایه و اولیه رمزنگاری هستند، به‌طوری که می‌توان از آن‌ها برای ساخت پروتکل‌های پیچیده‌تر که اهداف امنیتی نظیر *گمنامی*، *تعهّد* و *عدم انکار* را برآورده می‌کنند نیز استفاده کرد.

۲.۱ سیرتکاملی رمزنگاری

تاریخچه

مطالعه‌های تاریخی نشان می‌دهد که اولین رمزنگاری‌ها متعلّق به ۱۹۰۰ سال قبل از میلاد مسیح است، یعنی زمانی که مصری‌های باستان برای مخفی نگاه‌داشتن مضمون نوشته‌های خود از رسم‌الخطّ هیروگلیف، به صورت نامتعارف استفاده می‌کردند. البته در میان تمدّن‌های ایرانی‌ها، هندی‌ها، بابلی‌ها و اسپارت‌ها نیز استفاده از رمزنگاری مرسوم بوده است. با این حال اولین مطالعات مربوط به تحلیل رمز یا رمزشکنی به هزار سال پس از میلاد مسیح مقارن با زمان بعثت پیامبر اسلام باز می‌گردد، به همین خاطر است که دیوید کان [1] مورّخ و روزنامه‌نگار آمریکایی در کتاب معروف خود راجع به تاریخ رمزنگاری [۳۹] می‌نویسد، رمزشناسی در میان اعراب مسلمان متولد شد. برای مثال ((أبو یوسف یعقوب بن إسحاق الصبّاح الکندي)) که از دانشمندان مسلمان بوده، اولین کسی است که روش تحلیل فرکانسی حروف متون رمزشده را، که مبنای بسیاری از روش‌های آماری برای شکستن رمزها تا جنگ جهانی دوّم بود در کتاب خود تحت عنوان ((رسالة في استخراج المعماة)) معرفی کرد.

در گذشته کاربرد رمزنگاری منحصر به کاربردهای نظامی و سیاسی بود امّا امروزه استفاده از رمزنگاری گسترش زیادی یافته است به‌طوری که بسیاری از ما بدون این‌که بدانیم روزانه از رمزنگاری استفاده می‌کنیم، برای مثال وقتی یک خرید اینترنتی انجام می‌دهیم از رمزنگاری برای انتقال محرمانه‌ی رمز کارت بانکی از رایانه‌ی شخصی ما به سرویس‌دهنده‌ی بانک استفاده می‌شود، یا در بانکداری الکترونیکی از رمزنگاری برای تشخیص جعلی و یا اصلی بودن چک‌های الکترونیکی استفاده می‌شود. به عنوان مثالی دیگر، امروزه برای حفظ محرمانگی مکالمات مشترکین تلفن همراه، از رمزنگاری حین انتقال در این شبکه استفاده می‌شود، به‌طوری که یک مهاجم با شنود سیگنال رادیویی بین دستگاه تلفن همراه و اولین گره شبکه، نتواند به داده منتقل شده دست یابد، علاوه بر این برای احراز اصالت مشترکین و به‌طور متقابل سرویس‌دهنده‌ها در این شبکه از پروتکل‌های احراز اصالت و اولیه‌های رمزنگاری استفاده می‌شود.

دوران تاریخی رمزنگاری را می‌توانیم به سه مرحله‌ی زیر تقسیم کنیم [۳۹]:

- **دوران کاغذ و مرکّب** در این دوران الگوریتم‌های رمزنگاری که بیشتر از روش‌های ساده‌ی جانشینی و جایگشت استفاده می‌کردند، با استفاده از کاغذ و مرکّب و گاهی ابزارهای مکانیکی بدوی پیاده‌سازی می‌شدند که برای نمونه می‌توان به الگوریتم سزار و روش رمزنگاری اسپارتان اشاره کرد.

- **دوران ماشین‌های رمزنگاری** در این دوران که از آغاز قرن بیست تا سال ۱۹۴۹ و واقعه‌ی

[1] David Kahn

جنگ جهانی دوّم را در برمی‌گیرد، برای پیاده‌سازی الگوریتم‌های رمزنگاری از ماشین‌های الکترومکانیکی استفاده می‌شد. برای نمونه می‌توان به ماشین انیگما که در جنگ جهانی دوّم توسط آلمان‌ها استفاده شد اشاره کرد.

• **دوران رمزنگاری نوین** شانون را پدر علم نظریه‌ی اطلاعات می‌دانند اما شاید بتوان آن را پدر رمزنگاری نوین هم قلمداد کرد، چرا که رمزنگاری نوین با انتشار مقاله سال ۱۹۴۹ وی [۶۰] درباره نظریه ارتباطات سیستم‌های محرمانه آغاز شد و تا کنون ادامه دارد.

رویکردهای مختلف در مطالعه‌ی رمزنگاری

رویکرد سنّتی

در این رویکرد الگوریتم‌های رمزنگاری با تمرکز روی غلبه بر حملات موجود طراحی می‌شدند و روند طراحی و تحلیل به صورت زیر بود.

۱. هدف رمزنگاری مشخص می‌شود.

۲. سامانه رمزنگاری متناسب با هدف، طراحی می‌شود.

۳. سامانه‌ی رمزنگاری مورد حمله قرار می‌گیرد.

۴. اگر حمله مؤفق بوده و منجر به شکست سامانه شود به گام ۲ بازمی‌گردیم و با تجدید نظر در طراحی، سامانه را در مقابل حمله جدید مقاوم می‌کنیم.

رویکرد سنتی دارای مشکلاتی است. یک مشکل آشکار چنین رویکردی این است که هیچ‌گاه نمی‌دانیم چه الگوریتمی می‌تواند الگوریتم صحیح و مقاومی باشد و فرآیند طراحی یک فرآیند بی‌پایان است، هیچ تضمینی بر امنیت سامانه وجود نداشته و هر لحظه ممکن است مشکلات جدیدی بروز کند.

اولین رویکرد اصولی در رمزنگاری که در آن به‌طور مؤثری از تعاریف و اثبات‌ها استفاده شد، متعلق به شانون است، وی برای اولین بار مفهومی تحت عنوان امنیت کامل را تعریف می‌کند [۶۰] . ایده شانون که در بخش‌های بعد آن‌را بیشتر بررسی می‌کنیم به طور مختصر این‌بود که ابهام مهاجم راجع به متن اصلی، با مشاهده‌ی متن رمزشده کمتر نشود و به عبارتی دیدن یا ندیدن متن رمز شده هیچ تأثیری در مؤفقیت مهاجم در شکستن رمز نگذارد. تعریف شانون از امنیت، مهاجمی با توانایی محاسباتی نامحدود را مدل می‌کرد که فقط توانایی شنود کانال را دارد. مهم‌تر از همه، وی ثابت کرد که برای رسیدن به امنیت کامل تعداد بیت‌های کلید باید بزرگتر یا مساوی با تعداد بیت‌های متن اصلی باشد و این یک محدودیت عملی اساسی بود. از طرفی فرض شانون مبنی بر نامحدود بودن توانایی محاسباتی مهاجم، یک فرض قوی بود چرا که مهاجم هر چقدر هم

قوی باشد، توانایی محاسباتی‌اش محدود است. به این ترتیب رمزنگارها رویکرد دیگری تحت عنوان امنیت محاسباتی را در پیش گرفتند، امنیتی که کامل نیست ولی کافی است.

در امنیت محاسباتی بر خلاف امنیت کامل شانون، فرض بر این است که توانایی محاسباتی مهاجم محدود است و چون به‌جای امنیت کامل، به‌دنبال امنیت کافی هستیم، می‌توانیم از کلیدی کوتاه‌تر از متن اصلی برای رمزکردن استفاده کنیم، در نتیجه حمله‌های مؤفق، امکان‌پذیر ولی دست‌یافتن به‌آن‌ها در عمل دشوار است. در چنین رویکردی روشن شدن درجه‌ی پیچیدگی محاسباتی الگوریتم‌های رمزنگاری از اهمیت زیادی برخوردار است، به این ترتیب رمزنگاری از قلمرو نظریه‌ی اطلاعات خارج شد و تحت سلطه‌ی علوم کامپیوتر درآمد.

رویکرد نوین یا امنیت اثبات‌پذیر

می‌توان گفت رمزنگاری نوین بر اصول زیر استوار است:

۱. **تعاریف دقیق** از ابزارها و مفاهیم پایه‌ی مورد استفاده در رمزنگاری. برای مثال باید به صورت دقیق و کمّی مشخّص کنیم که منظور ما از مفهوم امنیت چیست؟ و سپس تعریف دقیق هر عنصر پایه‌ای که برای رسیدن به امنیت استفاده می‌کنیم را شرح دهیم.

۲. **فرضیات دقیق و مقبول** هر فرضی که در نظر می‌گیریم را باید به طور دقیق بیان کنیم و مهم‌تر از آن، فرضیات نباید دور از انتظار و نامعقول باشند.

۳. **برهان‌های دقیق و مستحکم** بر اساس تعاریف و فرضیات در نظر گرفته شده، امنیت سامانه‌ها به صورت دقیق ثابت می‌شود.

در رویکرد امنیت اثبات‌پذیر، برای اثبات امنیت پروتکل‌های رمزنگاری از روش تحویل یا کاهش، که در نظریه‌ی پیچیدگی برای مقایسه‌ی پیچیدگی الگوریتم‌ها به‌کار می‌رود، استفاده می‌کنیم. در این روش با تحویل یک فرض مشخّص به مسئله‌ی شکستن سامانه‌ی رمزنگاری، نشان می‌دهیم که شکستن سامانه‌ی رمزنگاری راحت‌تر از نقض آن فرض مشخّص (فرض لگاریتم گسسته، فرض دیفی هلمن تصمیمی و...) نیست و چون آن فرض یک فرض مقبول است و جامعه‌ی رمزنگاری درستی آن را در زمان حیات سامانه قبول دارد، امنیت سامانه اثبات می‌شود.

عبارت امنیت اثبات‌پذیر کمی فریب‌دهنده است چرا که با وجود اثبات امنیت یک سامانه ممکن است آن سامانه باز هم شکسته شود! دلیل این امر می‌تواند در نظر گرفتن فرض‌های نامعقول و یا مدل‌های غیرواقعی و ناقص باشد. برای مثال اگر فرض اغراق آمیز «هر سامانه‌ای امن است!» را درنظر بگیریم، هر سامانه‌ای بدون‌نیاز به هیچ فرض اضافه و یا اثبات، امن خواهد بود! یا ممکن است سامانه به‌جای آن‌که تا حد امکان به مدل‌های واقعی قرابت داشته باشد، فقط طوری طراحی شود که امنیت آن تحت آن مدل قابل اثبات باشد.

دلایلی نظیر دلایل فوق سبب شده است، تا برخی از رمزنگاران معاصر نظیر مِنِزِس [۲] و کوبلیتز [۳] نقدهایی را بر رویکرد امنیت اثبات پذیر وارد کنند. نقد آن‌ها نه بر استفاده از فرضیات و اثبات‌ها در رمزنگاری، بلکه بر استفاده‌ی نادرست از آن‌ها و در نظر گرفتن فرضیات نامعقول و یا غیر واقعی و گاهاً اثبات‌های اشتباه است.

با این وجود تجربه نشان داده که در رویکرد امنیت اثبات پذیر، با تمرکز روی فرضیات و مدل‌ها می‌توان آن‌ها را به مدل‌های واقعی بسیار نزدیک کرد و یا فرضیات معقول را پذیرفت. با این وجود اگر سامانه شکسته شود کافی است تا مدل‌ها و فرضیات را اصلاح کنیم.

۳.۱ رمزنگاری متقارن

در سامانه‌های رمزنگاری متقارن یا کلید خصوصی از یک کلید برای رمزنگاری و رمزگشایی داده‌ها استفاده می‌شود، به این ترتیب لازم است تا قبل از شروع ارتباط، کلید از طریق یک کانال امن بین طرفین ارتباط به اشتراک گذاشته شود.

تعریف ۱.۱. سامانه‌ی رمزنگاری متقارن شامل یک مجموعه‌ی متناهی تحت عنوان فضای پیام اصلی $\mathcal{M}$ به همراه یک سه‌تایی مثل $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ از الگوریتم‌های کارا است که به صورت زیر عمل می‌کنند [۴۰]

- Gen الگوریتم تولید کلید نام دارد که یک الگوریتم تصادفی است و بر اساس یک توزیع احتمال مشخّص (معمولا توزیع یکنواخت) رشته تصادفی k به نام کلید را تولید می‌کند. مجموعه‌ی همه‌ی کلیدهای ممکن که Gen تولید می‌کند را فضای کلید نامیده و با $\mathcal{K}$ نشان می‌دهیم.

- Enc الگوریتم رمزنگاری است که متن اصلی $m \in \mathcal{M}$ و کلید $k \in \mathcal{K}$ را به عنوان ورودی دریافت کرده و متن رمز شده‌ی $c \in \mathcal{C}$ را تحویل می‌دهد. رمز شده‌ی متن اصلی m تحت کلید k را با $Enc_k(m)$ نشان می‌دهیم. الگوریتم رمزنگاری لزوما قطعی نیست و می‌تواند تصادفی باشد.

- Dec الگوریتم رمزگشایی است که با دریافت متن رمزشده‌ی c و کلید k متن اصلی m را بازمی‌گرداند. رمزگشایی متن رمزشده‌ی c تحت کلید k را با $Dec_k(c)$ نمایش می‌دهیم. الگوریتم رمزگشایی همواره قطعی است و در صورتی که حاصل رمزگشایی نامعتبر باشد یعنی $Dec_k(c) \notin \mathcal{M}$ خروجی الگوریتم $\perp$ خواهد بود.

[2] Alfred Menezes
[3] Neal Koblitz

- **شرط صحت:** $.\forall\, k \in \mathcal{K},\ m \in \mathcal{M}\ \ Dec_k(Enc_k(m)) = m$

در ادامه با چگونگی احراز اصالت و حفظ محرمانگی با استفاده از سامانه‌های رمزنگاری متقارن آشنا می‌شویم.

مدل‌های حمله

با توجه به توانایی و سطح دسترسی مهاجم، می‌توانیم مدل‌های حمله‌ی زیر را در نظر بگیریم:

۱. **حمله فقط متن رمزشده** در این مدل مهاجم فقط یک متن رمزشده را در اختیار دارد و هدفش به‌دست آوردن متن اصلی متناظر با آن است. این حمله زمانی مؤثر است که متن اصلی دارای افزونگی باشد.

۲. **حمله متن اصلی معلوم** در این حالت مهاجم با داشتن یک یا چند زوج متن اصلی و رمزشده متناظر با آن قصد دارد متن اصلی متناظر با یک متن رمزشده‌ی جدید را به‌دست آورد.

۳. **حمله متن اصلی انتخابی** در این حالت مهاجم قادر است رمزشده هر متنی را که بخواهد به‌دست آورد، به عبارت دیگر دستگاه رمزنگاری را در اختیار دارد و هدفش این است که متن اصلی متناظر با یک متن رمزشده‌ی جدید را بداند.

۴. **حمله متن رمزی انتخابی** در این حالت مهاجم قصد دارد یک متن رمزشده را رمزگشایی کند و علاوه براین که قادر است هر متنی را رمزکند (به دستگاه رمزنگاری دسترسی دارد) می‌تواند هر متن رمزشده‌ای جز متن رمزی مورد نظر را رمزگشایی کند.

در بین موارد فوق توانایی مهاجم در حمله‌ی متن رمزی انتخابی از همه‌ی موارد قبل بیشتر است و اگر سامانه‌ی رمزنگاری در مقابل این حمله امن باشد طبیعتاً در مقابل سایر حمله‌ها هم امن است، البته هدف مهاجم تنها به به‌دست آوردن متن اصلی متناظر با یک متن رمزشده محدود نمی‌شود بلکه، هر یک از موارد زیر نیز می تواند هدف حمله باشد:

- به‌دست آوردن کلید سامانه.

- اعمال تمایز، یعنی این که مهاجم می‌داند متن رمزی پیش‌روی او رمزشده یکی از دو متن اصلی m_1 یا m_2 است و فقط باید تمایز دهد که این متن رمزی متعلق به m_1 یا m_2.

- به‌دست آوردن تنها مقداری اطلّاعات راجع به متن اصلی.

امنیت کامل

تعریف ۲.۱ (امنیت کامل). فرض کنید فضاهای متن اصلی و متن رمزشده را به ترتیب با $\mathcal{M}$ و $\mathcal{C}$ نشان دهیم، در این صورت سامانه‌ی رمز متقارن $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ روی فضای $\mathcal{M}$ دارای امنیت کامل است هرگاه به ازای هر توزیع احتمال روی $\mathcal{M}$ داشته باشیم

$$\forall\, m \in \mathcal{M},\ c \in \mathcal{C} \quad \{Pr\{C = c\} > ۰ \Rightarrow Pr\{M = m|C = c\} = Pr\{M = m\}\}\,,$$

که M و C متغیرهای تصادفی متن اصلی و متن رمزشده هستند.

تعریف فوق حمله‌ای را مدل می‌کند که در آن مهاجم با توانایی محاسباتی نامحدود کانال را شنود کرده و فقط یک متن رمزشده را در اختیار دارد.

لم ۳.۱. سامانه رمزنگاری متقارن $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ روی فضای متن اصلی $\mathcal{M}$ دارای امنیت کامل است اگر و تنها اگر به ازای هر توزیع احتمال روی $\mathcal{M}$ داشته باشیم

$$\forall\, m \in \mathcal{M},\ c \in \mathcal{C} \quad \{Pr\{M = m\} > ۰ \Rightarrow Pr\{C = c|M = m\} = Pr\{C = c\}\}\,.$$

برهان. اگر $Pr\{M = m\} = ۰$ آن‌گاه شرط امنیت کامل به‌وضوح برقرار است و در غیر این‌صورت کافی است طرفین تساوی فوق را در $\frac{Pr\{M=m\}}{Pr\{C=c\}}$ ضرب کنیم و از قاعده‌ی احتمال بیز استفاده کنیم تا به تعریف امنیت کامل برسیم. اثبات در جهت عکس به طور مشابه به‌دست می‌آید. □

لم زیر تعریف معادلی از امنیت کامل ارائه می‌کند، که مستقل از توزیع احتمال روی $\mathcal{M}$ است.

لم ۴.۱. سامانه رمز متقارن $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ روی فضای متن اصلی $\mathcal{M}$ دارای امنیت کامل است اگر و تنها اگر

$$\forall\, m, m' \in \mathcal{M},\ \forall c \in \mathcal{C} \quad Pr\{C = c|M = m\} = Pr\{C = c|M = m'\}$$

یا به طور معادل

$$\forall\, m, m' \in \mathcal{M},\ \forall c \in \mathcal{C} \quad Pr\{Enc_K(m) = c\} = Pr\{Enc_K(m') = c\}$$

که K متغیر تصادفی متناظر با کلید است.

برهان. رجوع کنید به [۴۰].

□

تعریف ۵.۱ (آزمایش تمایزناپذیری در برابر مهاجم شنودگر). فرض کنید Π یک سامانه رمزنگاری با فضای پیام $\mathcal{M}$ باشد و مهاجم را که در مدل ما یک الگوریتم است با $\mathcal{A}$ نشان دهیم، آزمایش

تمایزناپذیری در برابر مهاجم شنودگر که آن‌را با نماد $\mathsf{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ نمایش می‌دهیم، آزمایشی(یا بازی) است که بین مهاجم و یک چالشگر فرضی و به صورت زیر انجام می‌شود

۱. چالشگر با اجرای الگوریتم Gen یک کلید k تولید می‌کند.

۲. مهاجم پیام‌های $m_۰, m_۱ \in \mathcal{M}$ را انتخاب کرده و به چالشگر می‌دهد.

۳. چالشگر پس از انتخاب تصادفی $b \in \{۰, ۱\}$ ، متن رمزشده‌ی $c \leftarrow Enc_k(m_b)$ را محاسبه کرده و به $\mathcal{A}$ می‌دهد. به c متن رمز چالش می‌گوییم.

۴. مهاجم $\mathcal{A}$ بیت $b' \in \{۰, ۱\}$ را تولید می‌کند.

۵. اگر $b = b'$ ، خروجی آزمایش ۱ است که با $\mathsf{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi} = ۱$ نمایش می‌دهیم و می‌گوییم مهاجم مؤفق شده و در غیر این صورت خروجی آزمایش صفر بوده و مهاجم شکست خورده است.

آزمایش فوق برای مدل کردن حمله‌ی فقط متن رمزشده به‌کار می‌رود.

نکته ۶.۱. اگر K متغیر تصادفی متناظر با کلید باشد، متغیرهای تصادفی $C = Enc_K(m)$ و $C' = Enc_K(m')$ به متغیر تصادفی K و مقادیر تصادفی که احتمالاً در الگوریتم رمزنگاری استفاده می‌شود بستگی دارند. به این ترتیب بیان دیگری از لم ۴.۱ این است که در سامانه امن کام، متغیرهای تصادفی C و C' دارای توزیع احتمال یکسانی هستند، یعنی اگر مهاجم بداند c رمزشده‌ی یکی از متن‌های اصلی m یا m' است نتواند با احتمال بهتر از $\frac{۱}{۲}$ متن درست را تمایز دهد.

لم ۷.۱. سامانه‌ی رمزنگاری متقارن Π دارای امنیت کامل است اگر و تنها اگر برای هر مهاجم $\mathcal{A}$ داشته باشیم:

$$Pr\{\mathsf{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}\} \leq \frac{۱}{۲}$$

برهان. به [۴۰] مراجعه کنید. □

سامانه رمزنگاری ورنام یا OTP[4]

سامانه‌ی رمزنگاری متقارنی که ما امروزه آن‌را با نام OTP می‌شناسیم یک سامانه‌ی امن کامل است که در حدود ۳۰ سال قبل از این‌که شانون مفهوم امنیت کامل را تعریف کند، در سال ۱۹۱۷ توسط ورنام به ثبت رسیده بود!

تعریف ۸.۱ (رمز ورنام). فرض کنید n یک عدد طبیعی باشد. فضای متن اصلی $\mathcal{M}$ ، فضای متن رمزشده $\mathcal{C}$ و فضای کلید $\mathcal{K}$ همگی برابر با مجموعه‌ی $\{۰, ۱\}^n$ هستند و الگوریتم‌های سامانه به صورت زیر عمل می‌کنند:

[4] one-time pad

- الگوریتم تولید کلید Gen یک رشته را بر اساس توزیع یکنواخت از فضای $K = \{۰,۱\}^n$ به عنوان کلید انتخاب می‌کند.
- الگوریتم Enc با دریافت $m \in \{۰,۱\}^n$ و $k \in \mathcal{K}$ متن رمز شده‌ی $c = m \oplus k$ را تولید می‌کند.
- الگوریتم Dec متن رمزشده c و کلید k را دریافت می‌کند و اگر $c \in \{۰,۱\}^n$ باشد متن اصلی $m = c \oplus k$ و در غیر این صورت $\perp$ را (به معنای نامعتبر بودن متن رمزی) تولید می‌کند.

قضیه ۹.۱. رمز ورنام دارای امنیت کامل است.

برهان. ابتدا $Pr\{C = c|M = m'\}$ را به ازای $c \in \mathcal{C}$ و $m' \in \mathcal{M}$ دلخواه محاسبه می‌کنیم.

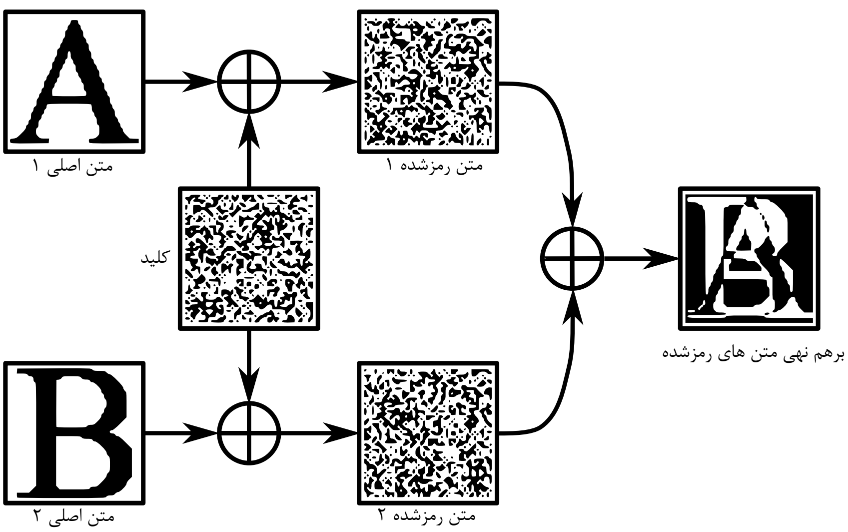
$$Pr\{C = c|M = m'\} = Pr\{Enc_K(m') = c\} = Pr\{K = m' \oplus c\} = ۲^{-n}$$

از طرف دیگر به‌ازای یک توزیع احتمال دلخواه روی $\mathcal{M}$ ، به ازای هر $c \in \mathcal{C}$ داریم:

$$\begin{aligned} Pr\{C = c\} &= \sum_{m' \in \mathcal{M}} Pr\{C = c|M = m'\} \cdot Pr\{M = m'\} \\ &= ۲^{-n} \cdot \sum_{m' \in \mathcal{M}} Pr\{M = m'\} = ۲^{-n}. \end{aligned}$$

در نتیجه طبق لم ۳.۱ حکم ثابت می‌شود. □

باید توجه کرد که سامانه رمز ورنام تنها زمانی امن کامل است که از هر کلید فقط یک‌بار استفاده کنیم، در غیر این‌صورت همان‌طور که در شکل ۲.۱ نمایش داده شده، اطلاعات نشت خواهد کرد.



شکل ۲.۱: نشت اطلاعات به‌هنگام رمز کردن دو پیام با یک کلید در رمز ورنام

قضیه‌ی شانون

قضیه ۱۰.۱ (قضیه‌ی شانون). اگر (Gen, Enc, Dec) یک سامانه‌ی رمز متقارن امن کامل با فضای پیام اصلی $\mathcal{M}$ و فضای کلید $\mathcal{K}$ باشد آن‌گاه، $|\mathcal{K}| \geq |\mathcal{M}|$.

برهان. نشان می‌دهیم اگر $|\mathcal{K}| < |\mathcal{M}|$ آن‌گاه سامانه فاقد امنیت کامل است. توزیع احتمال روی $\mathcal{M}$ را یکنواخت فرض می‌کنیم و $c \in \mathcal{C}$ را طوری انتخاب می‌کنیم که $Pr\{C = c\} > ۰$ ، مجموعه‌ی $\mathcal{M}(c)$ را به‌صورت زیر تعریف می‌کنیم:

$$\mathcal{M}(c) := \{m \in \mathcal{M} | \exists k \in \mathcal{K} : m = Dec_k(c)\}$$

از آن‌جایی که الگوریتم Dec یک الگوریتم قطعی است لذا $|\mathcal{M}(c)| \leq |\mathcal{K}|$، اگر $|\mathcal{K}| < |\mathcal{M}|$ آن‌گاه $m' \in \mathcal{M}$ وجود دارد به طوری که $m' \notin \mathcal{M}(c)$ که در این صورت:

$$Pr\{M = m' | C = c\} = ۰ \neq Pr\{M = m'\}$$

و امنیت کامل برقرار نخواهد بود. □

محدودیت‌های اساسی امنیت کامل:

- در تعریف امنیت کامل فرض کردیم توانایی محاسباتی مهاجم نامحدود است، حال آن‌که در عمل چنین نیست و قوی‌ترین مهاجم‌ها هم توانایی محاسباتی محدودی دارند.
- حتی مهاجم با توانایی محاسباتی نامحدود، نباید هیچ اطلاعاتی از متن رمزشده به‌دست آورد، که این یک شرط بیش از حد قوی است.
- نتیجه‌ی مستقیم قضیه‌ی ۱۰.۱ این است که اگر بخواهیم پیام‌هایی با طول یکسان را با کلیدهایی با طول یکسان رمز کنیم، طول کلید باید بزرگتر یا حداقل مساوی طول پیام باشد، که این یک محدودیت اساسی است.

امنیت کامل ضمن داشتن محدودیت‌های فوق، تنها مهاجمی را لحاظ می‌کند که توانایی شنود کانال و مشاهده فقط یک متن رمزشده را دارد و راه حلی برای مدل‌های دیگر حمله ارائه نمی‌کند. به این ترتیب به سراغ مفهوم دیگری از امنیت می‌رویم که محدودیت‌های مذکور را نداشته باشد.

امنیت محاسباتی

امنیتی که در آن شرط‌های زیر را در نظر می‌گیریم، امنیت محاسباتی نامیده می‌شود.

۱. مهاجم دارای توانایی محاسباتی محدود است. در واقع مهاجم در این مدل الگوریتم چندجمله‌ای تصادفی (غیریکنواخت) nuPPT است و امنیت فقط در مقابل چنین مهاجم‌هایی با زمان اجرای معقول و عملی، تضمین می‌شود.

۲. مهاجم به طور بالقوه می‌تواند با احتمال ناچیز مؤفّق شود که اگر این احتمال را به اندازه کافی کوچک بگیریم جای نگرانی نخواهد بود.

رویکردهای تحلیل سامانه‌های رمزنگاری

دو نوع رویکرد در تحلیل سامانه‌های رمزنگاری وجود دارد که عبارت‌اند از

- **رویکرد واقعی:** در این رویکرد امنیت یک سامانه با پارامترهای ثابت را با توجه به امکانات سخت‌افزاری موجود و در شرایط عملی بررسی می‌کنیم.

- **رویکرد مجانبی:** در این رویکرد امنیت خانواده‌ای از سامانه‌های رمزنگاری را به صورت مجانبی و بر حسب یک عدد طبیعی که اندیس خانواده‌ی مذکور است مورد بررسی قرار می‌دهیم.

رویکرد واقعی

تعریف ۱.۱۱ ((t, ε) - امن). [۴۰] سامانه‌ی رمزنگاری (Gen, Enc, Dec) را (t, ε) ـ امن گوییم هرگاه حداکثر احتمال مؤفقیت هر مهاجم با حداکثر زمان اجرای t برابر با ε باشد.

زمان اجرا در این رویکرد را معمولا با تعداد سیکل‌های پردازنده محاسبه می‌کنند. برای مثال در مورد یک سامانه‌ی رمزنگاری ادّعا می‌کنند که هیچ مهاجمی با حداکثر زمان اجرای ۲۰۰ سال که از سریع‌ترین ابرکامپیوترهای موجود استفاده کند نمی‌تواند با احتمال بیش از 2^{-60} مؤفق به شکستن سامانه شود یا می‌گویند هیچ مهاجمی با استفاده از حداکثر 2^{80} سیکل پردازنده نمی‌تواند با احتمال بیش از 2^{-60} مؤفق به شکستن سامانه شود. برای داشتن شهودی راجع به اعداد ذکر شده جالب است بدانید، اگر احتمال مؤفقیت یک مهاجم در بازیابی یک متن رمزشده در طول یک‌سال حداکثر 2^{-60} باشد، آن‌گاه مورد اصابت صاعقه قرار گرفتن فرستنده و گیرنده‌ی پیام در طی همین مدت، رویداد محتمل‌تری نسبت به مؤفقیت مهاجم است!

مثال ۱.۱۲. در سامانه‌های رمزنگاری متقارن نوین، وقتی از کلید n بیتی استفاده می‌شود، احتمال مؤفقیت مهاجم با زمان اجرای t (که معمولاً با شمارش سیکل‌های پرازنده اندازه گیری می‌شود) در شکستن سامانه، حداکثر برابر با $\frac{c \cdot t}{2^n}$ به ازای یک عدد ثابت c است. [۵] اگر فرض کنیم $c = 1$

[۵] این احتمال مربوط به حمله‌ی جست‌وجوی فراگیر فضای کلید بدون هیچ فرآیند پیش‌پردازشی است.

و طول کلید را ۶۰ بیت در نظر بگیریم آن‌گاه در مقابل رایانه‌های شخصی امنیت کافی خواهیم داشت، چرا که با یک پردازنده با فرکانس کاری ۴ گیگاهرتز می‌توان به ۴×۱۰^{۹} سیکل در هر ثانیه دست یافت، در این صورت اگر آزمودن هر کلید به یک سیکل نیاز داشته باشد برای آزمودن $۲^{۶۰}$ کلید موجود به $\frac{۲^{۶۰}}{(۴\times۱۰^{۹})}$ ثانیه یعنی حدوداً ۹ سال زمان نیاز است. با این حال ابرکامپیوترهای (چینی!) موجود در زمان نوشتن این رساله [۶] قادرند $۹٫۳ \times ۱۰^{۱۶}$ عملیات ممیز شناور را در یک ثانیه انجام دهند و این یعنی برای شکستن سامانه مذکور با استفاده از چنین کامپیوترهایی تنها به ۱۲٫۴ ثانیه نیاز است!

آن‌چه در نهایت مورد توجه کاربران پروتکل‌های رمزنگاری است، تضمین‌های واقعی و نه تئوری یا مجانبی از امنیت یک سامانه است و در این‌جا است که رویکرد واقعی در تحلیل سامانه اهمیت می‌یابد امّا تا قبل از آن به دلیل دشواری‌ها و و ملاحظات فنّی و تئوری متوجه رویکرد واقعی بهتر است از رویکرد مجانبی برای تحلیل سامانه‌ها استفاده کنیم، زیرا از این جهت که در آن نیازی به استفاده از پارامترهای t و ε و در نظر گرفتن ملاحظات فنّی در تعاریف و اثبات‌ها نیست، راحت‌تر است. همچنین هر تحلیل مجانبی، قابل تبدیل به تحلیل واقعی است. به همین دلیل در ادامه از رویکرد مجانبی برای تحلیل سامانه‌ها استفاده خواهیم کرد.

رویکرد مجانبی

تعریف ۱۳.۱ (امن مجانبی). [۴۰] سامانه رمزنگاری (Gen, Enc, Dec) را به صورت مجانبی امن گوییم هرگاه هر *مهاجم چندجمله‌ای تصادفی* با احتمال *ناچیزی* مؤفق به *شکستن* سامانه شود.

در رویکرد مجانبی امنیت خانواده‌ای از سامانه‌ها با مجموعه اندیس $\mathbb{N}$ مطرح است، که در این حالت اندیس یا شاخص هر سامانه را که به *پارامتر امنیتی* موسوم است با $n \in \mathbb{N}$ نمایش می‌دهیم، این رویکرد را از این جهت مجانبی می‌گوییم که امنیت سامانه‌ها به رفتار آن‌ها به ازای n های بزرگ بستگی دارد. *پارامتر امنیتی* توسط طرفین مجاز ارتباط رمزشده انتخاب می‌شود به طوری که مهاجم هم از آن آگاه است و نشان‌دهنده‌ی سطح امنیتی سامانه است. در این بحث می‌توانیم فرض کنیم پارامتر امنیتی همان طول کلید مورد استفاده است، و زمان اجرای (الگوریتم) مهاجم و طرفین مجاز و مزیت مهاجم نسبت به مهاجم تصادفی، همه را تابعی از این پارامتر در نظر می‌گیریم. تعریف ۱۳.۱ یک تعریف کلّی است و در ادامه رویکرد مجانبی را به صورت دقیق بررسی خواهیم کرد و مفاهیمی نظیر *احتمال ناچیز* و *شکستن سامانه* را به صورت دقیق تعریف خواهیم کرد.

تعریف ۱۴.۱ (مهاجم کارا). در رویکرد مجانبی زمان اجرای (الگوریتم) مهاجم تابعی از پارامتر امنیتی n است و برای لحاظ کردن توانایی (محدود) محاسباتی مهاجم فرض بر این است که این

[۶] سال ۱۳۹۵ هجری شمسی– ۲۰۱۶ میلادی

تابع چند جمله‌ای است. مهاجم می‌تواند به‌جای استفاده از الگوریتم‌های قطعی از الگوریتم‌های تصادفی استفاده کند، همچنین می‌تواند به‌جای این که برای همه‌ی اعضای خانواده‌ی سامانه‌های رمز، از یک الگوریتم ثابت استفاده کند، متناسب با هر عضو خانواده از الگوریتم متناسب با آن بهره ببرد و رفتاری غیر یکنواخت داشته باشد و برای حمله به عضو n ام خانواده از الگوریتم n ام خود استفاده کند. در نتیجه قوی‌ترین مدلی که برای مهاجم می‌توان تصور کرد، مهاجم چندجمله‌ای تصادفی غیریکنواخت است که به صورت $\mathcal{A}(1^n,.)$ نشان داده می‌شود. رشته‌ی ورودی 1^n نشان دهنده‌ی استفاده از پارامتر امنیتی n و نقطه نشان‌دهنده‌ی سایر ورودی‌های الگوریتم مهاجم است. در ادامه منظور ما از مهاجم کارا مهاجم چندجمله‌ای تصادفی است. مجموعه‌ی همه‌ی الگوریتم‌های چندجمله‌ای تصادفی را با PPT و مجموعه‌ی همه‌ی الگوریتم‌های چندجمله‌ای‌های تصادفی غیریکنواخت را با nuPPT نمایش می‌دهیم.

تعریف ۱.۱۵ (شکستن سامانه). مفهوم دقیق شکستن در رویکرد واقعی توسط یک آزمایش بیان می‌شود (مثل آزمایش تمایز ناپذیری)، ولی در رویکرد مجانبی خانواده‌ای از آزمایش‌ها را داریم که مجموعه‌ی اندیس آن، مجموعه‌ی مقادیر مجاز برای پارامتر امنیتی یعنی $\mathbb{N}$ است.

تعریف ۱.۱۶ (تابع ناچیز). تابع نامنفی $\varepsilon : \mathbb{N} \to \mathbb{R}_{\geq 0}$ را ناچیز گوییم هرگاه:

$$\forall\ c > 0\ \exists n_0 \in \mathbb{N}\ \forall n \in \mathbb{N}\ (n > n_0 \Rightarrow \varepsilon(n) < \frac{1}{n^c})$$

همچنین مجموعه‌ی همه‌ی توابع ناچیز را با NEG نمایش می‌دهیم.

تعریف ۱.۱۷ (خانواده‌ی سامانه‌های رمزمتقارن). یک خانواده از سامانه‌های رمزنگاری متقارن خانواده‌ای از سه‌تایی‌ها $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ از الگوریتم‌های چند جمله‌ای تصادفی (PPT) است به طوری که:

- Gen با دریافت رشته‌ی 1^n، کلید $k \in \mathcal{K}$ را تولید می‌کند. برای تأکید بر تصادفی بودن الگوریتم Gen در تولید کلید از نمایش $k \leftarrow \mathsf{Gen}(1^n)$ استفاده می‌کنیم. بدون کاستن از کلّیت فرض بر این است که $|k| \geq n$.
- از آن‌جایی که الگوریتم رمزنگاری Enc نیز ممکن است تصادفی باشد عملکرد آن‌را به صورت زیر نشان می‌دهیم: $\forall\ k \in \mathcal{K}, m \in \mathcal{M}\ \ c \leftarrow \mathsf{Enc}_k(m)$
- الگوریتم Dec قطعی است و عملکرد آن‌را به‌صورت زیر نمایش می‌دهیم:

$$\mathsf{Dec}_k(c) := \begin{cases} m & \text{اگر } c \text{ معتبر باشد} \\ \perp & \text{اگر } c \text{ نامعتبر باشد} \end{cases}$$

- شرط صحّت

$$\forall\, n \in \mathbb{N}, \forall\, k \leftarrow \mathsf{Gen}(1^n),\ \forall m \in \mathcal{M}\ \ \mathsf{Dec}_k(\mathsf{Enc}_k(m)) = m$$

نکته ۱۸.۱. در عمل، مخفی نگه داشتن طول پیام از مهاجم مقرون به صرفه نیست به این ترتیب فرض می‌کنیم در سامانه‌های رمزنگاری پیام‌هایی با طول یکسان به پیام‌های رمزی با طول یکسان رمز می‌شوند و در آزمایش تمایز ناپذیری و آزمایش‌هایی که بعد از این تعریف می‌کنیم فرض می‌کنیم پیام‌هایی که مهاجم تولید می‌کند طول یکسانی داشته باشند.

امنیت تک‌پیامی

آزمایش $\mathsf{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}$ را تعریف کردیم، ولی برای تأکید بر وابستگی این آزمایش به پارامتر امنیتی n در رویکرد مجانبی، آن را به صورت $\mathsf{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$ نمایش می‌دهیم و مراحل آن را به صورت زیر اصلاح می‌کنیم:

۱. چالشگر کلید k را تولید می‌کند: $k \leftarrow \mathsf{Gen}(1^n)$

۲. مهاجم پیام‌های با طول یکسان را تولید می‌کند، $m_0, m_1 \leftarrow \mathcal{A}(1^n);\ |m_0| = |m_1|$.

۳. چالشگر با توزیع یکناخت یکی از دو متن را انتخاب کرده، سپس رمز می‌کند و به مهاجم می‌دهد،
$b \leftarrow \{0, 1\},\ c \leftarrow \mathsf{Enc}_k(m_b)$.

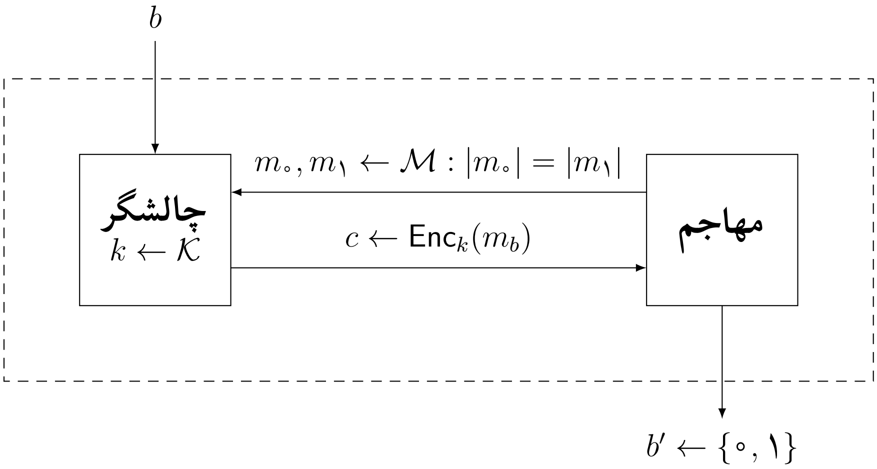
۴. مهاجم حدس می‌زند که c رمز شده‌ی کدام پیام است. برای این‌کار بیت b' را با توزیع یکنواخت انتخاب می‌کند. $b' \leftarrow \{0, 1\}$.

۵. نتیجه‌ی آزمایش عبارت است از

$$\mathsf{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n) = \begin{cases} 1 & b = b' \\ 0 & ow \end{cases}$$

تعریف ۱۹.۱ (امنیت تک‌پیامی). سامانه‌ی رمز متقارن Π دارای امنیت تک‌پیامی است هرگاه برای هر مهاجم چندجمله‌ای تصادفی $\mathcal{A}$، تابع ناچیز $\varepsilon(n)$ وجود داشته باشد به‌طوری که

$$Pr\{\mathsf{PrivK}^{\mathrm{eav}}_{\mathcal{A},\Pi}\}(n) \leq \frac{1}{2} + \varepsilon(n).$$



شکل ۱.۳: تعامل مهاجم و چالشگر در آزمایش امنیت تک‌پیامی

تمایزناپذیری محاسباتی و مولدهای شبه تصادفی

به بیان ساده، دو توزیع احتمال تمایزناپذیر محاسباتی هستند هرگاه هیچ مهاجم کارایی نتواند آن دو را از هم تمیز دهد. دو توزیع احتمال X و Y روی مجموعه رشته‌های با طول l را در نظر بگیرید، وقتی می‌گوییم الگوریتم تمایزگر $\mathcal{D}$ نمی‌تواند این دو را از هم تمایز دهد، یعنی اگر بر اساس یکی از توزیع‌های X یا Y یک نمونه از $\{۰,۱\}^l$ انتخاب کنیم و به $\mathcal{D}$ بدهیم، $\mathcal{D}$ نتواند با احتمال قابل توجهی، به‌درستی تشخیص دهد که نمونه بر اساس کدام‌یک از توزیع‌های X یا Y، انتخاب شده است. به عبارت دیگر اگر فرض کنیم تمایزگر $\mathcal{D}$ وقتی معتقد است نمونه بر اساس توزیع X انتخاب شده خروجی ۱ و در غیر این‌صورت خروجی ۰ را تولید می‌کند، آن‌گاه چه نمونه را براساس توزیع X و چه بر اساس توزیع Y به او بدهیم او باید با احتمال تقریباً یکسانی ۱ را تولید کند به عبارت دیگر ما می‌خواهیم عبارت زیر کوچک باشد

$$|Pr\{s \leftarrow X, D(s) = ۱\} - Pr\{s \leftarrow Y, D(s) = ۱\}|$$

تعریف ۲۰.۱ (تمایزناپذیری محاسباتی). دو خانواده از توزیع‌های احتمال $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ و $\mathcal{Y} = \{Y_n\}_{n\in\mathbb{N}}$ را تمایزناپذیر محاسباتی گوییم و به صورت $\mathcal{X} \overset{c}{\equiv} \mathcal{Y}$ نشان می‌دهیم، هرگاه به ازای هر تمایزگر چندجمله‌ای تصادفی $\mathcal{D}$، یک تابع ناچیز ε وجود داشته باشد که:

$$|Pr\{x \leftarrow X_n, \mathcal{D}(۱^n, x) = ۱\} - Pr\{y \leftarrow Y_n, \mathcal{D}(۱^n, y) = ۱\}| \leq \varepsilon(n).$$

در تعریف فوق رشته‌ی ۱^n را برای تأکید بر چندجمله‌ای بودن زمان اجرای $\mathcal{D}$ بر حسب n در ورودی $\mathcal{D}$ نمایش داده‌ایم.

شبه‌تصادفی بودن حالت خاصی از مفهوم تمایزناپذیری است و با فرض این‌که U_n به ازای هر

عدد طبیعی n نشان‌دهنده‌ی توزیع یکنواخت روی مجموعه‌ی $\{۰,۱\}^n$ باشد، به صورت زیر تعریف می‌شود.

تعریف ۲۱.۱ (خانواده توزیع‌های شبه‌تصادفی). فرض کنید $l(n)$ یک چندجمله‌ای و X_n یک توزیع احتمال روی مجموعه رشته‌های $\{۰,۱\}^{l(n)}$ باشد، در این صورت خانواده توزیع‌های $\mathcal{X} = \{X_n\}_{n\in\mathbb{N}}$ را شبه تصادفی گوییم هرگاه $\mathcal{X} \overset{c}{\equiv} \{U_{l(n)}\}_n$.

تعریف ۲۲.۱ (مولد شبه‌تصادفی). [۴۰] فرض کنید $l(n)$ چندجمله‌ای و G یک الگوریتم قطعی چندجمله‌ای باشد که به ازای هر $n \in \mathbb{N}$ رشته‌ی $s \in \{۰,۱\}^n$ را به $G(s) \in \{۰,۱\}^{l(n)}$ تبدیل کند، در این صورت G یک مولد شبه تصادفی است هرگاه

- **(گسترش)** $\forall\, n \in \mathbb{N}\;\; l(n) > n$. $l(n)$ را ضریب گسترش مولد می‌گوییم.
- **(شبه‌تصادفی)** $\{G(U_n)\} \overset{c}{\equiv} \{U_{l(n)}\}$
یا به طور معادل به ازای هر تمایزگر چندجمله‌ای تصادفی مثل $\mathcal{D}$ تابع ناچیز $\varepsilon(n)$ وجود داشته باشد به‌طوری که

$$|Pr\{s \xleftarrow{unif} \{۰,۱\}^n, \mathcal{D}(G(s)) = ۱\} - Pr\{r \xleftarrow{unif} \{۰,۱\}^{l(n)},\ \mathcal{D}(r) = ۱\}| \leq \varepsilon(n)$$

به بیان دیگر مولد شبه‌تصادفی ضمن گسترش طول کلید یا بذر اوّلیه، خروجی‌اش باید از رشته‌های تصادفی تمایزناپذیر محاسباتی باشد. در مثال بعد به کمک مولد شبه تصادفی یک سامانه رمزنگاری دارای امنیت تک‌پیامی می‌سازیم.

مثال ۲۳.۱. به ازای عدد طبیعی n و چند جمله‌ای $l(n)$ فرض کنید G یک مولد شبه تصادفی با ضریب گسترش $l(n)$ باشد، سامانه‌ی رمزنگاری با فضای پیام $\{۰,۱\}^{l(n)}$ را به صورت زیر می‌سازیم:

- Gen: با دریافت رشته‌ی ۱^n ، $k \in \{۰,۱\}^n$ را بر اساس توزیع یکنواخت به عنوان کلید انتخاب می‌کند.
- Enc: با دریافت متن اصلی $m \in \{۰,۱\}^{l(n)}$ و کلید $k \in \{۰,۱\}^n$ متن رمزشده را به‌صورت زیر تولید می‌کند
$$c := G(k) \oplus m.$$
- Dec: با دریافت پیام رمزشده $c \in \{۰,۱\}^{l(n)}$ و کلید $k \in \{۰,۱\}^n$ پیام اصلی را به صورت زیر تولید می‌کند
$$m := G(k) \oplus c.$$

در [۴۰]، به روش تحویل، ثابت شده است که سامانه‌ی رمزنگاری فوق دارای امنیت تک‌پیامی است.

رمزهای دنباله‌ای

روش رمزنگاری مثال ۲۳.۱ مانند رمز ورنام است با این تفاوت که به‌جای کلید کاملاً تصادفی از خروجی مولد شبه تصادفی به طور مستقیم استفاده کرده‌ایم، به همین دلیل محدودیت‌های رمز ورنام را نیز باید رعایت کنیم از جمله این‌که از هر کلید فقط یک‌بار و برای رمز کردن پیام‌هایی که طولشان حداکثر برابر طول خروجی مولد شبه‌تصادفی است استفاده کنیم. از طرفی استفاده از مولد شبه تصادفی به این صورت دو محدودیت دارد، اوّل این‌که طول خروجی مولد شبه تصادفی ثابت است، در حالی که ممکن است متن‌های اصلی طول‌های متفاوتی داشته باشند، در ثانی، مولد شبه تصادفی به یک‌بار ه یک رشته‌ی با طول ثابت را تولید می‌کند در حالی که ما به ابزاری نیاز داریم که همزمان با تولید بیت‌های متن اصلی، بیت‌های کلید اجرایی را تولید کند تا به این ترتیب فقط به اندازه‌ی نیاز، یعنی هم طول با متن اصلی، بیت‌های کلید را تولید کرده باشیم. برای رفع این مشکل‌ها *رمزدنباله‌ای* را معرفی می‌کنیم. رمزهای دنباله‌ای دسته‌ای از رمزهای متقارن محسوب می‌شوند که تعریف دقیق آن را در ادامه آورده‌ایم.

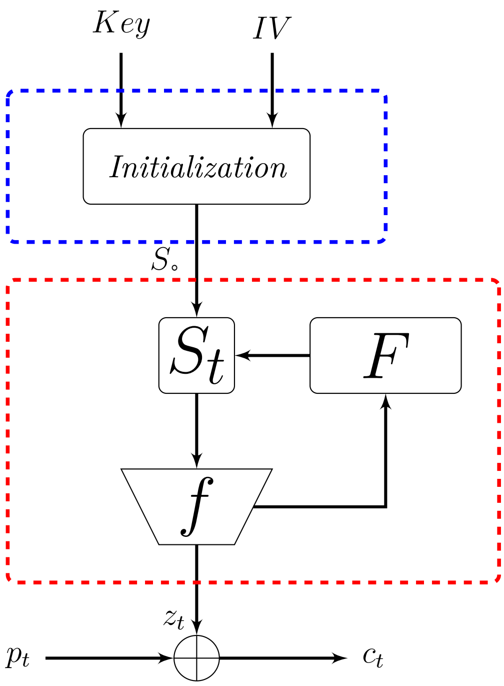
تعریف ۲۴.۱ (رمزهای دنباله‌ای)**.** رمز دنباله‌ای دوتایی (Init, GetBits) از الگوریتم‌های قطعی است به طوری که:

- Init : الگوریتم آغازسازی که کلید k و یک مقدار اوّلیه IV را دریافت کرده و st_0 را تولید می‌کند.
- GetBits الگوریتم تولید کلید اجرایی که با دریافت حالت فعلی st_i ضمن مشخص نمودن حالت بعدی یعنی st_{i+1}، یک بیت از کلید اجرایی z، را نیز مشخص می‌کند.

رمزنگاری در رمزهای دنباله‌ای، همان طور که در شکل ۴.۱ نمایش داده شده، به این‌صورت است که هر بیت از کلید اجرایی پس از تولید با بیت متناظرش در متن اصلی، در پیمانه‌ی ۲ جمع شده و بیت متناظر در متن رمز شده به‌دست می‌آید. مقدار اوّلیه یا IV معمولاً عمومی است و به همراه متن رمزی ارسال می‌شود. به این ترتیب با وجود ثابت بودن کلید، به دلیل تغییر مقدار اوّلیه در هر بار رمزنگاری، دنباله‌ی کلید اجرایی متفاوتی خواهیم داشت.

امنیت چندپیامی

امنیت تک‌پیامی امنیتی را مدل می‌کند که در آن مهاجم فقط توانایی شنود یک پیام رمزشده را دارد، اکنون فرض کنید فرستنده و گیرنده به‌جای یک‌پیام چند پیام را با یک کلید رمزکرده و از طریق کانالی که مهاجم آن را شنود می‌کند برای یکدیگر ارسال کنند برای مدل کردن چنین مهاجمی امنیت چند پیامی را تعریف می‌کنیم.



شکل ۴.۱: نمای کلی رمزنگاری در رمزهای دنباله‌ای

آزمایش شنود چندپیامی

- مهاجم با دریافت پارامتر امنیتی 1^n دو لیست از پیام‌های هم‌طول به صورت

$$M_\circ = (m_{\circ,1}, ..., m_{\circ,t}),\ M_1 = (m_{1,1}, ..., m_{1,t})$$

را تولید کرده و برای چالشگر فرضی می‌فرستد که t یک چندجمله‌ای بر حسب n است و به ازای هر $i = 1, ..., t$ داریم: $|m_{\circ,i}| = |m_{1,i}|$.

- چالشگر $\mathsf{Gen}(1^n)$ را اجرا کرده و بیت $b \in \{\circ, 1\}$ را با توزیع یکنواخت انتخاب می‌کند. سپس برای هر i متن رمزی $c_i \leftarrow \mathsf{Enc}_k(m_{b,i})$ را محاسبه کرده و لیست $C = (c_1, ..., c_t)$ را برای مهاجم می‌فرستد.

- مهاجم بیت b' را تولید می‌کند.

- اگر $b = b'$ خروجی آزمایش ۱ است که آن را با $\mathsf{PrivK}^{\mathrm{mult}}_{\mathcal{A},\Pi}(n) = 1$ نمایش می‌دهیم و می‌گوییم مهاجم مؤفق شده.

تعریف ۲۵.۱ (امنیت چند پیامی). سامانه‌ی رمز متقارن $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ دارای امنیت چندپیامی در حضور مهاجم شنودگر است هرگاه، به ازای هر مهاجم چندجمله‌ای تصادفی $\mathcal{A}$، تابع ناچیز $\varepsilon(n)$

موجود باشد به‌طوری که

$$Pr\{\mathsf{PrivK}^{\text{mult}}_{\mathcal{A},\Pi}(n)=1\}\leq \frac{1}{2}+\varepsilon(n).$$

مثال ۱.۲۶. سامانه‌ی رمزنگاری متقارنی وجود دارد که امنیت تک‌پیامی دارد ولی امنیت چندپیامی ندارد.

مثال ۱.۲۳ دارای چنین خصوصیتی است، کافی است تا مهاجم در آزمایش شنود تک‌پیامی دو لیست پیام زیر را در گام اوّل برای چالشگر ارسال کند:

$$(m_{0,1},m_{0,2})=(0^{l(n)},0^{l(n)})\quad (m_{1,1},m_{1,2})=(0^{l(n)},1^{l(n)})$$

حال اگر $c_1=c_2$ بود، مهاجم b' را برابر ۰ و در غیر این‌صورت برابر ۱ قرار می‌دهد. این مهاجم با احتمال ۱ مؤفق می‌شود و لذا سامانه امنیت چندپیامی ندارد.

مثال فوق تنها سامانه‌ای نیست که فاقد امنیت چندپیامی است، می‌توان ثابت کرد [۴۰] که به طور کلّی هر سامانه‌ی رمزنگاری که الگوریتم رمزنگاری آن قطعی و غیر تصادفی باشد، دارای امنیت چندپیامی نخواهد بود. به این ترتیب برای دست‌یابی به امنیت چندپیامی به سامانه رمزنگاری نیازمندیم که الگوریتم رمزنگاری آن تصادفی باشد، به طوری که حاصل رمزنگاری یک متن اصلی تحت یک کلید ثابت طی دفعات متوالی متفاوت باشد، این در حالی است که الگوریتم رمزگشایی قطعی باقی می‌ماند!

امنیت متن اصلی انتخابی

برای مدل کردن مهاجمی که علاوه بر شنود ارتباط رمز شده به دستگاه رمزنگاری نیز دسترسی دارد مدل دیگری از حمله تحت عنوان حمله‌ی متن اصلی انتخابی را معرفّی می‌کنیم. در این مدل فرض بر این است که مهاجم می‌تواند رمزشده‌ی هر پیامی (تحت یک کلید ثابت) را به‌دست آورد. این مدل حمله شامل حمله متن اصلی معلوم که در آن مهاجم به تعدای زوج متن اصلی و رمزشده‌ی نظیر آن‌ها دسترسی دارد، نیز می‌شود. جالب توجه است که حمله‌ی متن اصلی انتخابی یکی از حملات مؤفق در تاریخ جنگ‌های نظامی بوده است. برای مثال می‌توان به نبرد میدوِی [7] که یکی از تعیین کننده‌ترین نبردهای جنگ جهانی دوّم میان آمریکا و ژاپن در جنگ جهانی دوّم در منطقه‌ی اقیانوس آرام بود، اشاره کرد. در ماه مه سال ۱۹۴۹، رمزنگارهای نیروی دریایی آمریکا پیام‌هایی رمزشده از سوی ژاپنی‌ها دریافت کردند که قادر بودند بخشی از آن را رمزگشایی کنند. نتایج این رمزگشایی‌های ناقص حاکی از این بود که ژاپنی‌ها طرح یک حمله به AF را ریخته‌اند، امّا AF جزو

[7] Battle of Midway

رمزهایی بود که آمریکایی‌ها هنوز رمزگشایی نکرده بودند. به دلایلی آمریکایی‌ها می‌دانستند که جزیره‌ی میدوی باید هدف حمله‌ی ژاپنی‌ها باشد، امّا تلاش‌هابرای متقاعد کردن فرماندهان اصلی ناامیدکننده بود تا این‌که رمزنگارهای نیروی دریایی دست به اقدامی جالب زدند. آن‌ها به نیروهای مستقر در جزیره‌ی میدوی دستور دادند تا پیام‌ها فریب دهنده‌ای مبنی بر کمبود آب آشامیدنی در جزیره میدوی را مخابره کنند. ژاپنی‌ها با شنود این پیام‌ها بلافاصله پیام «کمبود آب در AF» را به مرکز فرماندهی خود مخابره می‌کردند. به این ترتیب رمزنگارها ثابت کردند که منظور از AF که قسمت مجهول پیام‌های رمز شده بود، همان جزیره‌ی میدوی است و آمریکایی‌ها هواپیماهای خود را به سوی جزیره ارسال کردند. نتیجه این شد که جزیره میدوی در دست آمریکایی‌ها باقی ماند و ژاپنی‌ها شکست سختی را متحمل شدند. حیله‌ای که رمزنگارهای آمریکایی در این جنگ بکار بردند نوع ضعیفی از حمله‌ی متن اصلی انتخابی است چرا که آن‌ها با این ترفند توانستند رمزشده‌ی پیامی که می‌خواستند یعنی «Midway» را به‌دست آورند. اگر الگوریتم‌های رمزنگاری ژاپنی‌ها در آن زمان تصادفی عمل می‌کرد این استراتژی آمریکایی‌ها ناکام می‌ماند و شاید تاریخ طور دیگری رقم می‌خورد!

حمله‌ی متن اصلی انتخابی

برای مدل کردن حمله‌ی متن اصلی انتخابی، فرض می‌کنیم که مهاجم به *اوراکل رمزنگاری* دسترسی دارد. مقصود از دسترسی اوراکلی [8] مهاجم به الگوریتم رمزنگاری این است که الگوریتم رمزنگاری برای مهاجم به مانند یک جعبه سیاه است. مهاجم بدون این که بداند درون جعبه سیاه چه می‌گذرد و از چه کلیدی در الگوریتم استفاده می‌شود، تنها می‌تواند به آن متن اصلی داده و رمزشده‌ی آن‌را دریافت کند. دسترسی اوراکلی الگوریتم (مهاجم) $\mathcal{A}$ به الگوریتمی یا تابعی مثل f را به صورت $\mathcal{A}^{f(\cdot)}$ نمایش می‌دهیم. در محاسبه‌ی زمان اجرای اوراکل $\mathcal{A}$ ، هر پرسمان و دریافت پاسخ آن یک واحد زمانی در نظر گرفته می‌شود، بنابراین یک مهاجم چند جمله‌ای تنها به تعداد چندجمله‌ای بر حسب طول ورودی می‌تواند اوراکل خود را فراخوانی کند.

آزمایش تمایزناپذیری در حمله‌ی متن اصلی انتخابی

این آزمایش را با $\mathsf{PrivK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n)$ نمایش می‌دهیم و شامل مراحل زیر است.

۱. چالشگر با اجرای $\mathsf{Gen}(1^n)$ کلید k را تولید می‌کند.

۲. مهاجم با دریافت پارامتر امنیتی 1^n ، در حالی که به $\mathsf{Enc}_k(\cdot)$ دسترسی اوراکلی دارد دو متن اصلی هم طول m_0 و m_1 را تولید می‌کند. که کلّ این مراحل را با $m_0, m_1 \leftarrow \mathcal{A}^{\mathsf{Enc}_k(\cdot)}$ نمایش می‌دهیم.

[8] oracle access

۳. چالشگر بیت تصادفی $b \in \{۰,۱\}$ را انتخاب کرده و متن رمزشده‌ی $c \leftarrow \mathsf{Enc}_k(m_b)$ را محاسبه و برای مهاجم ارسال می‌کند.

۴. مهاجم در حالی که به الگوریتم $\mathsf{Enc}_k(\cdot)$ دسترسی اوراکلی دارد ببیت b' را تولید می‌کند. $b' \leftarrow \mathcal{A}^{\mathsf{Enc}_k(\cdot)}$

۵. اگر $b = b'$ نتیجه‌ی آزمایش ۱ است که با $\mathsf{PrivK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n) = ۱$ نمایش می‌دهیم و می‌گوییم مهاجم مؤفق شده و در غیر این‌صورت نتیجه ۰ خواهد بود.

تعریف ۲۷.۱ (امنیت متن اصلی انتخابی). سامانه‌ی رمزنگاری متقارن $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ تمایزناپذیر تحت حمله‌ی متن اصلی انتخابی یا دارای امنیت متن اصلی انتخابی است هرگاه به ازای هر مهاجم چندجمله‌ای تصادفی نظیر $\mathcal{A}$ تابع ناچیز $\varepsilon(n)$ وجود داشته باشد بطوری که:

$$Pr\{\mathsf{PrivK}^{\mathrm{cpa}}_{\mathcal{A},\Pi}(n) = ۱\} \leq \frac{۱}{۲} + \varepsilon(n)$$

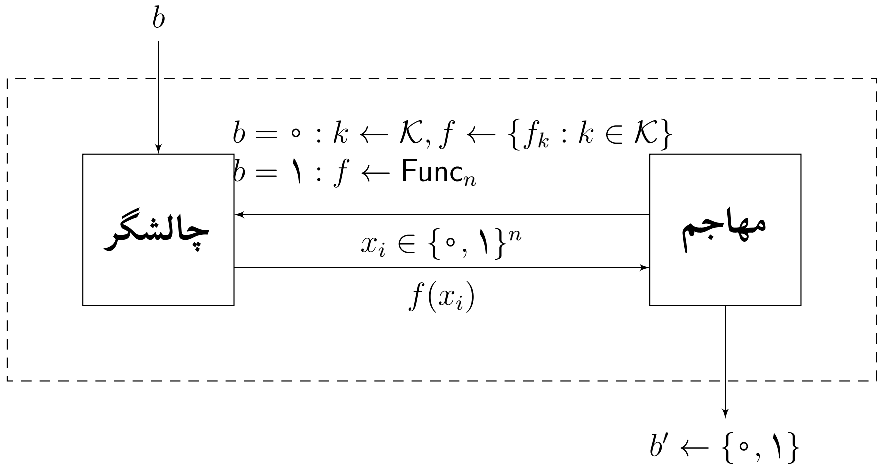
احتمال فوق به عوامل تصادفی الگوریتم $\mathcal{A}$ و الگوریتم رمزنگاری و بیت‌های تصادفی تولید شده در آزمایش بستگی دارد.

توابع شبه تصادفی که در ادامه با آن‌ها آشنا می‌شویم به ما کمک می‌کنند تا به امنیت چندپیامی و امنیت متن اصلی انتخابی دست یابیم.

توابع شبه تصادفی

مجموعه همه توابع از مجموعه $\{۰,۱\}^n$ به خودش را که با Func_n نمایش می‌دهیم در نظر بگیرید. به تابعی که به صورت تصادفی و با توزیع احتمال یکنواخت از این مجموعه انتخاب شود تابع *تصادفی* می‌گوییم، بنابراین تصادفی بودن تابع نه به رفتار آن بلکه به نحوه‌ی انتخاب آن بستگی دارد، تحت این تعریف حتّی تابع ثابت هم می‌تواند یک تابع تصادفی باشد.

تابع تصادفی از Func_n را می‌توانیم به صورت یک جدول جست‌وجو شامل یک ستون در نظر بگیریم که با درایه‌هایی که با توزیع یکنواخت از $\{۰,۱\}^n$ انتخاب شده‌اند پر شده است به طوری که شماره‌ی ردیف ورودی و محتوای ردیف خروجی را مشخص می‌کند. کدکردن Func_n به چنین روشی به $\log_۲(۲^{n \cdot ۲^n})$ بیت نیاز دارد که حتّی به ازای n های کوچک هم عدد بزرگی است، لذا به سراغ توابع شبه تصادفی می‌رویم که زیر مجموعه‌ای بسیار کوچکتر از Func_n است و در عین حال هیچ تمایزگر کارایی نباید بتواند با دسترسی اوراکلی به یک تابع که به صورت تصادفی از چنین خانواده‌ای انتخاب شده آن را از یک تابع تصادفی تمیز دهد. تعریف دقیق و مجانبی توابع شبه تصادفی به صورت زیر است.



شکل ۵.۱: تعامل بین چالشگر فرضی و تمایزگر در تمایز تابع شبه‌تصادفی

تعریف ۲۸.۱ (خانواده‌ی توابع شبه تصادفی). $\{۰, ۱\}^*$ را مجموعه‌ی همه‌ی رشته‌های دودویی با طول متناهی در نظر بگیرید. خانواده‌ی توابع $\{F_k : \{۰, ۱\}^{|k|} \rightarrow \{۰, ۱\}^{|k|}\}_{k \in \{۰,۱\}^*}$ را شبه تصادفی گوییم هر گاه:

۱. به ازای هر $k \in \{۰, ۱\}^*$ و هر $x \in \{۰, ۱\}^{|k|}$ الگوریتم چندجمله‌ای برای محاسبه‌ی $F_k(x)$ وجود داشته باشد.

۲. برای هر تمایزگر چندجمله‌ای تصادفی $\mathcal{D}$، تابع ناچیز $\varepsilon(n)$ وجود داشته باشد به طوری که

$$|Pr\{f \xleftarrow{unif} \mathsf{Func}_n : \mathcal{D}^{f(\cdot)}(۱^n) = ۱\} - Pr\{k \xleftarrow{unif} \{۰, ۱\}^n : \mathcal{D}^{F_k(\cdot)}(۱^n) = ۱\}| \leq \varepsilon(n).$$

حاصل قدر مطلق در عبارت فوق را مزیت تمایزگر می‌گوییم.

باید به این نکته دقت کرد در تعریف تابع شبه تصادفی فرض بر این است که تمایزگر ساختار خانواده‌ی توابع شبه تصادفی و نحوه‌ی عملکرد توابع این خانواده را می‌داند، تنها پارامتر مجهول برای تمایزگر کلید k است. در ضمن تمایزگر به تابع انتخابی دسترسی اوراکلی دارد و می‌تواند چندجمله‌ای بار (بر حسب طول کلید) خروجی تابع انتخابی را به ازای ورودی‌های دلخواه مورد پرسمان قرار دهد و در آخر باید حدس بزند، تابعی که مورد پرسمان قرار داده، بر اساس کدام توزیع انتخاب شده است. چنین تعاملی بین چالشگر فرضی و یک تمایزگر در شکل ۵.۱ نمایش داده شده است.

مثال ۲۹.۱. خانواده‌ی توابع $F_k(x) = k \oplus x$ که روی رشته‌های n بیتی تعریف می شوند را در نظر بگیرید، چنین خانواده‌ای شبه تصادفی نیست. برای اثبات تمایزگری را در نظر بگیرید که از اوراکل تابع انتخابی که با $\mathcal{O}(\cdot)$ نمایش می‌دهیم، خروجی تابع به ازای دو ورودی متمایز $x_۱$ و $x_۲$

را مورد پرسمان قرار دهد و مقادیر $y_۱ = \mathcal{O}(x_۱)$ و $y_۲ = \mathcal{O}(x_۲)$ را دریافت کند. از آن‌جایی که تمایزگر از ضابطه‌ی خانواده که به صورت $F_k(x) = k \oplus x$ است آگاه است، می‌داند که اگر تابع انتخابی از این خانواده باشد باید داشته باشیم $x_۱ \oplus x_۲ = y_۱ \oplus y_۲$ و در این صورت بیت $b' = ۱$ را تولید می‌کند. احتمال این که یک تابع تصادفی چنین ویژگی را ارضا کند برابر است با ۲^{-n} و لذا مزیت تمایزگر برابر است با $|۱ - ۲^{-n}|$ که تابع ناچیزی نیست.

در مثال بعد، با استفاده از توابع شبه تصادفی یک سامانه ی رمزنگاری می‌سازیم که دارای امنیت متن اصلی انتخابی و امنیت چند پیامی است.

مثال ۱.۳۰. فرض کنید $\{F_k : \{۰,۱\}^{|k|} \rightarrow \{۰,۱\}^{|k|}\}_{k\in\{۰,۱\}^*}$ یک خانواده از توابع شبه تصادفی باشد. سامانه‌ی رمزنگاری متقارن برای رمزکردن پیام‌های دودویی با طول n را به صورت زیر می‌سازیم.

- $\mathsf{Gen}(۱^n) : k \leftarrow \{۰,۱\}^n$
- الگوریتم رمزنگاری پس از دریافت کلید و پیام اصلی $m \in \{۰,۱\}^n$ ابتدا یک رشته‌ی تصادفی r از $\{۰,۱\}^n$ انتخاب کرده و سپس متن رمزی را به صورت زیر محاسبه می‌کند.

$$c = \mathsf{Enc}_k(m) := \langle r, F_k(r) \oplus m\rangle.$$

- در الگوریتم رمزگشایی با استفاده از متن رمزی $c = \langle r, s\rangle$ و کلید k، متن اصلی به صورت زیر محاسبه می‌شود. $m := F_k(r) \oplus s$.

سامانه‌ی فوق هم امنیت چندپیامی و هم امنیت متن اصلی انتخابی دارد. برای اثبات به روش تحویل فرض می‌کنیم مهاجمی وجود دارد که با مزیت قابل توجهی قادر است در آزمایش متن اصلی انتخابی مؤفق شود، در این‌صورت می‌توانیم تمایزگری با مزیت قابل توجه برای تمایز خانواده توابع F_k از توابع تصادفی بسازیم و این یعنی خانواده‌ی مذکور شبه تصادفی نیست، که با فرض در تناقض است. برای مشاهده‌ی جزئیات اثبات به [۴۰، ص.۸۳]، مراجعه کنید.

جایگشت شبه‌تصادفی

فرض کنید مجموعه‌ی همه‌ی جایگشت‌ها روی $\{۰,۱\}^n$ را با Perm_n نمایش دهیم. یاد آور می‌شویم که یک جایگشت تابعی دوسویی است. اگر بخواهیم Perm_n را نیز به همان روش توابع شبه تصادفی به صورت جدول‌های جست‌وجو کد کنیم، دوسویی بودن جایگشت این محدودیت که درایه‌های هر دو ردیف باید متمایز باشند را اعمال می‌کند، به همین خاطر در این حالت $\log_۲(۲^n!)$ بیت نیاز داریم که عدد بزرگی است. به همین دلیل از جایگشت‌های شبه تصادفی که به صورت زیر تعریف می‌شود استفاده می‌کنیم.

تعریف ۳۱.۱ (خانواده جایگشت شبه تصادفی). خانواده‌ی جایگشت‌های $\{E_k : \{0,1\}^{|k|} \to \{0,1\}^{|k|}\}_{k\in\{0,1\}^*}$ را شبه تصادفی گوییم هرگاه

۱. به ازی هر $k \in \{0,1\}^*$ و هر $x \in \{0,1\}^{|k|}$ الگوریتم چندجمله‌ای برای محاسبه‌ی $E_k(x)$ موجود باشد.

۲. به ازای هر تمایزگر چندجمله‌ای تصادفی مثل $\mathcal{D}$ تابع ناچیز $\varepsilon(n)$ موجود باشد به‌طوری که

$$|Pr\{E \xleftarrow{unif} \mathsf{Perm}_n : \mathcal{D}^{E(\cdot),E^{-1}(\cdot)}(1^n) = 1\} - Pr\{k \xleftarrow{unif} \{0,1\}^n : \mathcal{D}^{E_k(\cdot),E^{-1}(\cdot)}(1^n) = 1\}| \leq \varepsilon(n).$$

از آن‌جایی که گاهی لازم است تا طرفین مُجازی که از جایگشت‌های تصادفی در الگوریتم‌های رمزنگاری استفاده می‌کنند از معکوس جایگشت هم استفاده کنند و الگوریتم‌های مورد استفاده باید برای همه‌ی طرفین حتّی مهاجم مشخص باشد در قسمت دوم تعریف جایگشت شبه تصادفی تمایزگر علاوه بر اوراکل جایگشت به اوراکل معکوس آن‌هم دسترسی دارد.

جایگشت‌های شبه تصادفی کاربردهای زیادی در رمزنگاری دارند از جمله این که رمزهای قالبی که در ادامه تعریف می‌کنیم نمونه‌ای از جایگشت‌های شبه‌تصادفی به ازای طول کلید و طول ورودی ثابت هستند.

تعریف ۳۲.۱ (رمزهای قالبی). یک رمز قالبی روشی برای تولید خانواده‌ای از جایگشت‌های شبه تصادفی با مجموعه اندیس k است، به طوری که با تغییر کلید k جایگشت متفاوتی خواهیم داشت. رمزهای قالبی، همان‌طور که از نامشان برمی‌آید روی قالب‌هایی از داده با طول مشخص، عمل می‌کنند. دو پارامتر مهم رمزهای قالبی عبارت است از:

۱. طول ورودی و خروجی که به آن طول قالب هم می‌گویند

۲. اندازه‌ی کلید

در عمل رمزهای قالبی به تنهایی کاربرد خاصی ندارند، بلکه ابزاری مهم و پایه‌ای برای طراحی پروتکل‌های امنیتی محسوب می‌شوند. برای مثال مرحله‌ی بعد از طراحی یک رمز قالبی، این است که ساختارهایی را طراحی کنیم که با استفاده از آن رمز قالبی بتوان داده‌هایی را به صورت امن رمز کرد که طول آن‌ها بسیار فراتر از طول قالب باشد. امنیت سامانه‌های پیچیده‌تری که با استفاده از رمزهای قالبی ساخته می‌شوند مبنی بر شبه‌تصادفی بودن رمزهای قالبی است. از بین رمزهای قالبی معروف می‌توان به AES [9] و DES [10] اشاره کرد.

[9] Advanced Encryption Standard

[10] Data Encryption Standard

امنیت متن رمزی انتخابی

برای این که مهاجمی را مدل کنیم که علاوه بر دستگاه رمزنگاری به دستگاه رمزگشایی هم دسترسی دارد، در آزمایش تمایزناپذیری فرض می‌کنیم مهاجم هم به اوراکل رمزنگاری و هم به اوراکل رمزگشایی دسترسی دارد.

تعریف ۳۳.۱ (آزمایش تمایزناپذیری در حمله‌ی متن رمزی انتخابی). [۴۰] این آزمایش را با نماد $\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}$ نمایش می‌دهیم و متشکل از مراحل زیر است.

۱. چالشگر فرضی با اجرای $\mathsf{Gen}(1^n)$ کلید k را تولید می‌کند.

۲. مهاجم با دسترسی اوراکلی به الگوریتم‌های $\mathsf{Enc}_k(\cdot)$ و $\mathsf{Dec}_k(\cdot)$ دو متن اصلی هم طول m_0, m_1 را انتخاب می‌کند. کل این فرآیند را به صورت $m_0, m_1 \leftarrow \mathcal{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}$ نمایش می‌دهیم.

۳. چالشگر بیت تصادفی $b \in \{0, 1\}$ را انتخاب کرده و سپس متن رمز چالش $c \leftarrow \mathsf{Enc}_k(m_b)$ را محاسبه کرده و به مهاجم می‌دهد.

۴. مهاجم همچنان به الگوریتم‌های $\mathsf{Enc}_k(\cdot)$ و $\mathsf{Dec}_k(\cdot)$ دسترسی دارد ولی نمی‌تواند درخواست رمزگشایی متن رمز چالش c را از اوراکل رمزگشایی $\mathsf{Dec}_k(\cdot)$ داشته باشد. در نهایت مهاجم بیت تصمیم b' را تولید می‌کند.

۵. اگر $b = b'$ آن‌گاه $\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1$ و مهاجم مؤفق شده است.

تعریف ۳۴.۱ (امنیت متن رمزی انتخابی). سامانه‌ی رمزنگاری $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ دارای *امنیت متن رمزی انتخابی* است هر گاه برای هر مهاجم چندجمله‌ای تصادفی $\mathcal{A}$ ، تابع ناچیز $\varepsilon(n)$ موجود باشد که

$$Pr\{\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1\} \leq \frac{1}{2} + \varepsilon(n).$$

مثال ۳۵.۱. سامانه رمز ارائه شده در مثال ۳۰.۱ دارای امنیت متن رمزی انتخابی نیست. فرض کنید مهاجم در مرحله‌ی ۲ متن‌های اصلی $m_0 = 0^n$ و $m_1 = 1^n$ را انتخاب کند. سپس در مرحله‌ی ۴ پس از دریافت متن رمزچالش $c = \langle r, s\rangle$ متن رمزی جدید $c' = \langle r, s \oplus 10^{n-1}\rangle$ را ساخته و درخواست رمزگشایی آن را به اوراکل رمزگشایی ارسال می‌کند و $m' = \mathsf{Dec}_k(c')$ را به‌دست می‌آورد. اگر $m' = 10^{n-1}$ آن‌گاه $b' = 0$ و در غیر این صورت b' را برابر ۱ قرار می‌دهد. بنابراین $Pr\{\mathsf{PrivK}^{cca}_{\mathcal{A},\Pi}(n) = 1\}$ و سامانه امنیت متن رمزی انتخابی ندارد.

نه تنها سامانه‌ی مثال ۳۰.۱ بلکه هر سامانه‌ی رمز متقارن دیگری که بتوان متن رمزشده را به روش کنترل شده‌ای تغییر داد به طوری که متن رمزی به‌دست آمده معتبر باشد و بتوان راجع به متن اصلی متناظر با آن اطلاعاتی به‌دست آورد، دارای امنیت متن رمزی انتخابی نیست. در ادامه

یکی دیگر از اولیه‌های رمزنگاری را که در دست‌یابی به امنیت متن رمزی انتخابی به‌کار می‌آید معرفی می‌کنیم.

کد احراز اصالت پیام

همه‌ی مهاجم‌هایی که تا کنون مدل کردیم منفعل بودند و فقط سعی داشتند اطلاعاتی راجع به متن اصلی به‌دست آوردند، ولی مهاجم می‌تواند فعّال باشد به این معنی که می‌تواند با دست‌کاری پیام‌های رمزی ارسالی خراب‌کاری به بار آورد. به این ترتیب به ابزاری نیاز داریم تا مانع جعل و یا تغییر پیام‌های ارسالی شود.

تعریف ۳۶.۱ (کد احراز اصالت پیام). یک سامانه‌ی کد احراز اصالت پیام یک سه‌تایی از الگوریتم‌های چندجمله‌ای تصادفی روی فضای پیام $\mathcal{M}$ است به طوری که

- Gen الگوریتم تولید کلید است که ورودی آن پارامتر امنیتی 1^n و خروجی آن کلید k است.
- Mac الگوریتم تولید برچسب نام دارد. این الگوریتم با دریافت پیام $m \in \mathcal{M}$ و کلید $k \in \mathcal{K}$ برچسب $t \leftarrow \mathsf{Mac}_k(m)$ را تولید می‌کند.
- Vrfy الگوریتم تأیید برچسب نامیده می‌شود. ورودی آن پیام $m \in \mathcal{M}$ و کلید $k \in \mathcal{K}$ و برچسب t و خروجی آن یک (به معنی معتبر) یا صفر (به معنی نامعتبر) است.
- **شرط صحت** به ازای هر پیام m و هر کلید k داشته باشیم، $\mathsf{Vrfy}_k(\langle m, \mathsf{Mac}_k(m)\rangle) = 1$.

الگوریتم تولید برچسب می‌تواند تصادفی باشد، امّا الگوریتم تأیید همواره قطعی است.

سامانه کد احراز اصالت برای جلوگیری از جعل یک پیام ساخته شده و لذا حمله‌هایی که به چنین سامانه‌هایی اعمال می‌شود با هدف جعل پیام است. مانند قبل برای مدل کردن چنین حمله‌هایی آزمایشی ارائه می‌دهیم که توانایی مهاجم را در تولید یک برچسب معتبر برای پیامی که قبلاًبرچسب آن را ندیده و یا تولید برچسبی جدید برای پیامی که قبلاً برچسبی از آن را دیده، محک می‌زند.

تعریف ۳۷.۱ (آزمایش جعل‌ناپذیری قوی کد احراز اصالت پیام). این آزمایش را با $\mathsf{Mac\text{-}sforge}_{\mathcal{A},\Pi}$ نمایش می‌دهیم که شامل مراحل زیر است:

۱. چالشگر با اجرای $\mathsf{Gen}(1^n)$ کلید k را تولید می‌کند.

۲. مهاجم به $\mathsf{Mac}_{k(\cdot)}$ دسترسی اوراکلی دارد و می تواند برچسب پیام‌های مختلف را از اوراکل $\mathsf{Mac}_k(\cdot)$ پرسمان کند. در نهایت پس از چندجمله‌ای بار پرسمان، یک زوج $\langle m, t\rangle$ را تولید کرده و به چالشگر می‌دهد، $\langle m, t\rangle \leftarrow \mathcal{A}^{\mathsf{Mac}_k(\cdot)}$.

۳. اگر مجموعه‌ی همه‌ی زوج پرسمان‌های ارسالی و برچسب‌های دریافتی از اوراکل $\mathsf{Mac}_k(\cdot)$ را با Q نمایش دهیم خروجی آزمایش متغیر تصادفی $\mathsf{Mac\text{-}sforge}_{\mathcal{A},\Pi}(n)$ است که به صورت زیر تعریف می‌شود:

$$\mathsf{Mac\text{-}sforge}_{\mathcal{A},\Pi}(n) := \begin{cases} ۱ & \langle m,t\rangle \notin Q \wedge \mathsf{Vrfy}_k(\langle m,t\rangle) = ۱ \\ ۰ & \text{در غیر این‌صورت} \end{cases}$$

تعریف ۳۸.۱ (کد احراز اصالت به شدت امن). سامانه‌ی کد احراز اصالت $\Pi(\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ به شدت امن است، هرگاه برای هر مهاجم چندجمله‌ای تصادفی مثل $\mathcal{A}$ تابع ناچیز $\varepsilon(n)$ موجود باشد به‌طوری که

$$Pr\{\mathsf{Mac\text{-}sforge}_{\mathcal{A},\Pi}(n) = ۱\} \leq \varepsilon(n).$$

در ادامه آزمایشی را معرفی می‌کنیم که توانایی مهاجم در تولید متن رمزشده‌ی معتبر را محک می‌زند.

تعریف ۳۹.۱ (آزمایش رمزنگاری جعل‌ناپذیر). این آزمایش را با $\mathsf{Enc\text{-}Forge}_{\mathcal{A},\Pi}$ نمایش می‌دهیم و شامل مراحل زیر است:

۱. چالشگر $\mathsf{Gen}(۱^n)$ را اجرا کرده و کلید k را تولید می‌کند.

۲. مهاجم پارامتر امنیتی ۱^n را دریافت کرده و در حالی که به اوراکل رمزنگاری $\mathsf{Enc}_k(\cdot)$ دسترسی دارد، متن رمزشده‌ی c را تولید کرده و به چالشگر می‌دهد.

۳. اگر $m := \mathsf{Dec}_k(c)$ و Q مجموعه‌ی تمام پرسمان‌هایی باشد که مهاجم از اوراکل رمزنگاری داشته، آن‌گاه خروجی آزمایش ۱ است هرگاه $m \neq \perp$ و در ضمن $m \neq Q$. به عبارت دیگر اگر مهاجم بتواند متن رمزشده‌ی معتبری را که قبلاً ندیده تولید نماید موفق می‌شود.

تعریف ۴۰.۱. سامانه‌ی رمزنگاری متقارن Π *جعل‌ناپذیر* است اگر به ازای هر مهاجم چندجمله‌ای تصادفی $\mathcal{A}$، تابع ناچیز ε وجود داشته باشد به طوری که، $Pr\{\mathsf{Enc\text{-}Forge}_{\mathcal{A},\Pi}(n) = ۱\} \leq \varepsilon(n)$.

تعریف ۴۱.۱ (رمزنگاری احرازاصالت شده). سامانه‌ی رمزنگاری متقارن Π یک *سامانه‌ی رمزنگاری احراز اصالت‌شده* است، هر گاه علاوه بر امنیت متن رمزی انتخابی، جعل‌ناپذیر نیز باشد.

لازم به ذکر است که هر ترکیبی از چند سامانه‌ی امن منجر به یک سامانه‌ی امن نمی‌شود، آن‌چه باید در این مرحله به آن دقت کنیم نحوه‌ی ترکیب سامانه‌ها برای رسیدن به مقصود مورد نظر است. در ادامه سامانه‌ی رمزنگاری و سامانه‌ی کد احراز اصالت را به نحو مناسبی برای رسیدن به یک سامانه‌ی رمزنگاری احراز اصلات شده با هم ترکیب می‌کنیم.

قضیه ۴۲.۱. فرض کنید $\Pi_E(\mathsf{Enc},\mathsf{Dec})$ یک سامانه رمزنگاری با امنیت متن اصلی انتخابی و $\Pi_M(\mathsf{Mac},\mathsf{Vrfy})$ یک سامانه کد احراز اصالت به شدت امن باشد که تولید کلید در هر یک از آن‌ها با انتخاب تصادفی یکنواخت یک رشته‌ی n بیتی انجام می‌شود، در این صورت سامانه‌ی رمزنگاری متقارن $\Pi'(\mathsf{Gen}',\mathsf{Enc}',\mathsf{Dec}')$ که به صورت زیر ساخته می‌شود یک سامانه رمزنگاری احراز اصالت شده است.

۱. Gen' : با دریافت پارامتر امنیتی 1^n دو رشته‌ی تصادفی یکنواخت $k_E,k_M\in\{0,1\}^n$ را به صورت مستقل انتخاب می‌کند، خروجی این الگوریتم دوتایی (k_E,k_M) به عنوان کلید است.

۲. Enc' : الگوریتم رمزنگاری دوتایی کلید (k_E,k_M) و متن اصلی $m\in\{0,1\}^n$ را دریافت کرده و $c\leftarrow\mathsf{Enc}_{k_E}(m)$ و سپس $t\leftarrow\mathsf{Mac}_{k_M}(c)$ را محاسبه می‌کند، خروجی این الگوریتم دوتایی $\langle c,t\rangle$ است.

۳. Dec' : الگوریتم رمزگشایی به ازای ورودی کلید (k_E,k_M) و متن رمزشده‌ی $\langle c,t\rangle$ ، ابتدا درستی رابطه‌ی $\mathsf{Vrfy}_{k_M}(c,t)\stackrel{?}{=}1$ را بررسی می‌کند، اگر این رابطه برقرار باشد خروجی $\mathsf{Dec}_{k_M}(c)$ و در غیر این صورت $\perp$ (به معنای نامعتبر بودن متن رمزی) است.

برهان. به [۴۰] صفحه‌ی ۱۳۶ رجوع کنید. □

با توجه به تعریف رمزنگاری احراز اصالت شده، می‌توان نتیجه گرفت که هر سامانه‌ی رمزنگاری احراز اصالت شده، امنیت متن رمزی انتخابی نیز دارد امّا عکس این گزاره درست نیست.

اکنون به هدف خود یعنی احراز اصالت و حفظ محرمانگی با استفاده از سامانه‌های رمزنگاری متقارن دست‌یافته‌ایم. در بخش بعد رسیدن به همین اهداف را با استفاده از نوع دیگری از سامانه‌های رمزنگاری موسوم به سامانه‌های رمزنگاری نامتقارن یا کلید همگانی دنبال خواهیم کرد.

۴.۱ رمزنگاری نامتقارن یا کلید همگانی

انگیزه‌ی اصلی این که رمزنگارها به دنبال راه حلّی غیر از سامانه‌های رمزنگاری متقارن بودند، مسئله‌ی تبادل و مدیریت کلید در این سامانه‌ها است. اگر شبکه ارتباطی دارای n کاربر باشد و هر دو نفر بخواهند یک کلید به اشتراک بگذارند آن‌گاه $\frac{n(n-1)}{2}$ کلید باید به طور محرمانه مبادله شده و به صورت امن نگهداری شود که کار دشواری است، زیرا هر چه تعداد کلیدها بیشتر شود مدیریت آن‌ها دشوارتر و احتمال دست‌یابی مهاجم به تعدادی از آن‌ها بیشتر است. راه‌کار دیگر به کار گرفتن یک شخص ثالث معتمد در نقش مرکز توزیع کلید است. در این حالت لازم است تا هرکاربر فقط یک کلید را برای ارتباط امن با مرکز کلید، ذخیره سازد. در این مدل اگر A بخواهد

با B ارتباط امن برقرار کند، پیام را رمزکرده و برای مرکز کلید ارسال می‌کند، مرکز کلید که همه‌ی کلیدها را دارد، پیام ارسالی A را رمزگشایی کرده و سپس آن را با کلید مخفی B رمز کرده و برای او ارسال می‌کند، امّا این راه حل منصفانه نیست چرا که مرکز توزیع کلید از همه‌ی کلیدها آگاه است جدای از این اگر مهاجم به مرکز کلید دست‌یابد همه‌ی کلیدها فاش می‌شود.

تبادل کلید

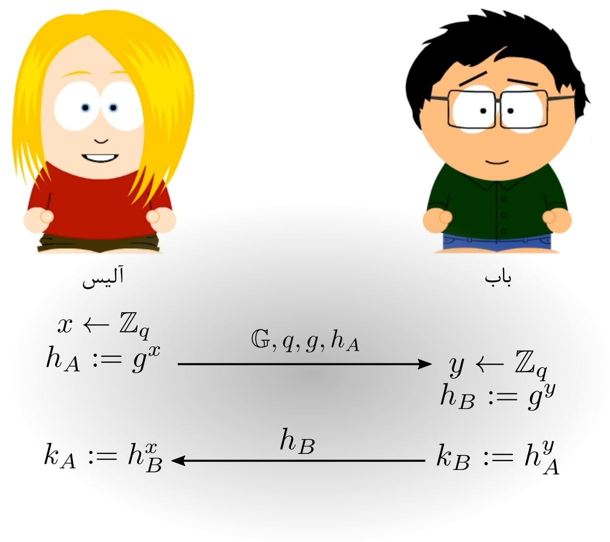
با وجود راه‌حل‌های ارائه شده در بند قبل هنوز هم تبادل کلید بین کاربران و یا کاربران و مرکز تولید کلید باید از طریق یک کانال امن صورت گیرد. تبادل کلید و داشتن یک ارتباط امن از طریق یک کانال ناامن نیازمند یک رویکرد کاملاً متفاوت در رمزنگاری بود. تا قبل از سال ۱۹۷۶ اعتقاد بر این بود که اساساً داشتن یک ارتباط امن، بدون به اشتراک گذاشتن یک سری اطلاعات مخفی از طریق یک کانال امن میسر نیست، تا این که دیفی [11] و هلمن [12] مقاله‌ای تحت عنوان «مسیری جدید در رمزنگاری» را در همان سال ارائه دادند. آن‌ها پی برده بودند که بسیاری از اتفاقات جهان ما طبیعتی نامتقارن دارد، به طور خاص برخی اتفاقات در جهت رفت سهل ولی در جهت عکس دشوار هستند. برای مثال شکستن یک لیوان کار آسانی است ولی تبدیل لیوان شکسته شده به همان لیوان اوّل کار دشواری است، به عنوان مثالی دیگر (و مرتبط با بحث ما) ضرب دو عدد کارآسانی است ولی تجزیه همان عدد به عواملش کار آسانی نیست. دیفی و هلمن پی بردند که این پدیده‌ها می توانند برای ساختن یک پروتکل تبادل کلید امن که به طرفین اجازه می‌دهد از طریق یک کانال ناامن کلیدی را به اشتراک بگذارند به کارآیند. در تأیید این عقیده که همیشه بخشی از علم مخصوصاً در رمزنگاری سری باقی می‌ماند و در زمان خودش منتشر نمی‌شود به این نکته اشاره می‌کنیم که مسأله‌ی تبادل کلید و سامانه‌های کلید همگانی از سال ۱۹۶۰ ذهن ریاضی‌دانان و رمزنگاران ستاد ارتباطات دولت انگلیس را به خود مشغول کرده بود و قبل از دیفی و هلمن آن‌ها مؤفق شده بودند سامانه‌هایی مشابه دیفی‐ هلمن را کشف کنند. این کشفیات تا سال ۱۹۹۷ محرمانه باقی‌مانده بود!

پروتکل تبادل کلید دیفی‐ هلمن

اکنون پروتکلی را که دیفی هلمن در مقاله‌ی اصلی خودشان [۲۸] آوردند شرح می‌دهیم. فرض کنید $\mathcal{G}$ یک الگوریتم چندجمله‌ای تصادفی باشد که ورودی آن پارامتر امنیتی 1^n و خروجی آن گروه دوری $\mathbb{G}$ ، و مرتبه‌ی آن q (که $\| q \|= n$) و یک مولد برای گروه مثل $g \in \mathbb{G}$ است.

تعریف ۴۳.۱ (پروتکل تبادل کلید دیفی‐ هلمن). تبادل کلید با استفاده از پروتکل دیفی‐ هلمن

[11] Whitfield Diffie
[12] Martin Hellman



شکل ۶.۱: پروتکل تبادل کلید دیفی‌ـ‌هلمن

بین A و B به ازای پارامتر امنیتی 1^n به صورت زیر انجام می‌شود.(شکل ۶.۱)

- A با اجرای $\mathcal{G}(1^n)$ سه‌تایی $(\mathbb{G}, q, g)$ را به‌دست می‌آورد و پس انتخاب تصادفی $x \in \mathbb{Z}_q$ با توزیع یکنواخت، $h_A := g^x$ را محاسبه می‌کند و د رنهایت $(\mathbb{G}, q, g, h_A)$ را برای B ارسال می‌کند.

- B پس از دریافت $(\mathbb{G}, q, g, h_A)$ ، $y \in \mathbb{Z}_q$ را با توزیع یکنواخت انتخاب کرده و پس از محاسبه $h_B := g^y$، آن را برای A ارسال می‌کند و در نهایت کلید را برابر با $k_B := h_A^y$ در نظر می‌گیرد.

- A پس از دریافت h_B کلید را برابر با $k_A := h_B^x$ در نظر می‌گیرد.

در عمل پارامترهای $(\mathbb{G}, q, g)$ در استانداردهای ارتباطی مشخص هستند و A کافی است h_A را برای B ارسال کند و نیازی نیست B برای محاسبه و ارسال h_B منتظر ارسال $(\mathbb{G}, g, q)$ از سوی A باشد. از آن‌جا که $k_A = g^{xy}$ و $_B = g^{xy}$ لذا $k_A = k_B$ و پروتکل درست کار می‌کند.
گروه دوری $\mathbb{G}$ از مرتبه‌ی q و مولدی از آن مثل $g \in \mathbb{G}$ را در نظر بگیرید. لگاریتم گسسته‌ی $h \in \mathbb{G}$ را با $\log_g(h)$ نشان می‌دهیم و برابر است با $x \in \mathbb{Z}_q$ به طوری که داشته باشیم $g^x = h$ محاسبه‌ی $\log_g(h)$ وقتی $g \in \mathbb{G}$ به تصادف انتخاب شده باشد به مسئله‌ی لگاریتم گسسته معروف است. به ازای $h_1, h_2 \in \mathbb{G}$ ، $\mathsf{DH}_g(h_1, h_2)$ را برابر با $g^{\log_g h_1 \cdot \log_g h_2}$ در نظر می‌گیریم، یعنی به ازای $h_1 = g^{x_1}$ و $h_2 = g^{x_2}$ داریم:

$$\mathsf{DH}_g(h_1, h_2) = g^{x_1 \cdot x_2} = h_1^{x_2} = h_2^{x_1}.$$

مسئله‌ی *دیفی هلمن محاسباتی* عبارت است از محاسبه‌ی $\mathsf{DH}_g(h_1, h_2)$ به ازای h_1 و h_2 که به

صورت تصادفی یکنواخت از $\mathbb{G}$ انتخاب شده باشند.

به مسئله‌ی تمایز $\mathsf{DH}_g(h_۱, h_۲)$ از یک عضو تصادفی یکنواخت گروه $\mathbb{G}$ مثل h' وقتی $h_۱$ و $h_۲$ هر دو به صورت تصادفی یکنواخت از $\mathbb{G}$ انتخاب شده باشند *مسئله‌ی دیفی هلمن تصمیمی* می گویند.

تعریف ۴۴.۱ (فرض دیفی هلمن تصمیمی). می‌گوییم مسأله‌ی دیفی هلمن تصمیمی نسبت به $\mathcal{G}$ سخت است یا اصطلاحاً فرض دیفی هلمن تصمیمی نسبت به $\mathcal{G}$ برقرار است هرگاه به ازای هر تمایزگر چندجمله‌ای تصادفی $\mathcal{D}$ تابع ناچیزی مثل $\varepsilon(n)$ موجود باشد که:

$$|Pr\{\mathcal{D}(\mathbb{G}, q, g, g^x, g^y, g^z) = ۱\} - Pr\{\mathcal{D}(\mathbb{G}, q, g, g^x, g^y, g^{xy}) = ۱\}| \leq \varepsilon(n),$$

که $x, y, z \in \mathbb{Z}_q$ به صورت تصادفی یکنواخت انتخاب شده‌اند.

در واقع فرض دیفی‌ـ هلمن تصمیمی روی گروه دوری $\mathbb{G}$ با مرتبه‌ی q و مولد $g \in \mathbb{G}$ بیانگر تمایزناپذیری محاسباتی دو متغیر تصادفی (g^x, g^y, g^z) و $(g^x, g^y, g^{x \cdot y})$ به ازای انتخاب تصادفی یکنواخت $x, y, z \in \mathbb{Z}_q$ است. در ادامه امنیت در برابر مهاجم منفعل در تبادل کلید را با یک آزمایش مدل می‌کنیم.

تعریف ۴۵.۱ (آزمایش تبادل کلید). این آزمایش را به ازای پارامتر امنیتی ۱^n با $\mathsf{KE}_{\mathcal{A},\Pi}(n)$ نمایش می‌دهیم که شامل مراحل زیر است:

۱. طرفین تبادل با تعیین پارامتر امنیتی ۱^n ، پروتکل تبادل کلید Π را اجرا می‌کنند. خروجی یک رونوشت مانند $trans$ شامل همه‌ی پیام‌های ارسالی طرفین، و یک کلید $k \in \{۰, ۱\}^n$ است.

۲. بیت $b \in \{۰, ۱\}$ به صورت تصادفی یکنواخت انتخاب می‌شود. اگر $b = ۰$ قرار می‌دهیم $k' = k$ و در غیر این‌صورت k' را به‌صورت تصادفی یکنواخت از $\{۰, ۱\}^n$ انتخاب می‌کنیم.

۳. مهاجم $\mathcal{A}$ رونوشت $trans$ و کلید k_b را دریافت کرده . بیت b' را تولید می‌کند.

۴. خروجی این آزمایش را با متغیر تصادفی $\mathsf{KE}_{\mathcal{A},\Pi}(n)$ نمایش می‌دهیم و برابر ۱ است هرگاه $b = b'$.

تعریف ۴۶.۱ (امنیت پروتکل تبادل کلید). پروتکل تبادل کلید Π ، در برابر مهاجم منفعل امن است، اگر برای هر مهاجم چندجمله‌ای تصادفی $\mathcal{A}$ ، تابع ناچیز ε وجود باشد که

$$Pr\{\mathsf{KE}_{\mathcal{A},\Pi}(n) = ۱\} \leq \frac{۱}{۲} + \varepsilon(n)$$

قضیه ۴۷.۱. اگر مسئله‌ی دیفی هلمن تصمیمی نسبت به $\mathcal{G}$ سخت باشد، آن‌گاه پروتکل تبادل کلید دیفی_ هلمن در برابر مهاجم منفعل امن خواهد بود.

برهان. به [۴۰] مراجعه کنید. □

تعریف سامانه‌های کلید همگانی

از رمزنگاری کلید همگانی تحت عنوان انقلابی در تاریخ رمزنگاری یاد می‌شود. در رمزنگاری کلید همگانی، برخلاف رمزنگاری کلید خصوصی نیازی به تبادل داده‌های مخفی نظیر کلید از طریق یک کانال امن، پیش از ارتباط رمز شده، نیست. شگفت انگیز است که دو نفر در دو طرف یک سالن و در حضور دیگران، که تنها می‌توانند صدای فریاد هم را بشنوند، بدون هیچ ملاقات قبلی بتوانند طوری با هم صحبت کنند که هیچ کس غیر از آن‌ها نتواند بفهمد آن‌ها راجع به چه چیزی صحبت می‌کنند! (البته همه‌ی افراد حاضر در سالن فارسی زبان هستند!) امّا رمزنگاری کلید همگانی این امکان را فراهم می‌کند، به این صورت که یکی از طرفین (گیرنده) یک زوج کلید (pk, sk)، که به ترتیب از سمت چپ، کلید همگانی و کلید خصوصی نامیده می‌شوند تولید می‌کند. فرستنده با استفاده از کلید همگانی داده‌ها را رمز کرده و گیرنده با استفاده از کلید خصوصی متناظر با آن داده‌ها را رمزگشایی می کند، در حالی که در رمزنگاری متقارن هم رمزنگاری و هم رمزگشایی با یک کلید صورت می‌گیرد، به همین دلیل به رمزنگاری کلید همگانی، رمزنگاری نامتقارن هم می‌گویند. امّا فرستنده چطور بدون هیچ ملاقات یا تبادل اطلاعات قبلی می‌تواند کلید عمومی گیرنده یعنی pk را به‌دست آورد؟ یک روش این است که گیرنده، کلید عمومی خود را به صورت کاملا آشکار و بدون نیاز به هیچ کانال امن به فرستنده بگوید. برای مثال آن را در وبگاه شخصی خود قرار دهد تا همه آن را بدانند یا در مثال فوق آن را در معرض عموم به کسی که آن طرف سالن است بگوید! در این که مهاجم هم کلید عمومی وی را به‌دست می‌آورد جای نگرانی نیست چون سامانه‌های کلید همگانی را طوری طراحی می‌کنند که دست‌یابی به کلید خصوصی با داشتن کلید عمومی کاری دشوار ولی عکس آن راحت باشد. بنابراین کلید عمومی چنان‌که از نام آن برمی‌آید عمومی و کلید خصوصی فقط نزد صاحب کلید می‌ماند و امنیت چنین سامانه‌هایی متکّی به مخفی ماندن کلید خصوصی است.

تعریف ۴۸.۱ (سامانه‌ی رمزنگاری نامتقارن). سامانه رمزنگاری نامتقارن یک سه‌تایی مرتب از الگوریتم‌های چندجمله‌ای تصادفی مثل $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ است به طوری که:

۱. Gen : الگوریتم تولید کلید، که ورودی آن پارامتر امنیتی 1^n و خروجی آن زوج مرتب (pk, sk) است. به pk کلید عمومی و به sk کلید خصوصی می‌گوییم.

۲. Enc : الگوریتم رمزنگاری است که به ازای ورودی کلید همگانی pk و متن اصلی m از

فضای پیام اصلی $\mathcal{M}$، متن رمزشده‌ی c را محاسبه می‌کند. از آن‌جایی که این الگوریتم می‌تواند تصادفی باشد عملکرد آن را به صورت $c \leftarrow \mathsf{Enc}_{pk}(m)$ نمایش می‌دهیم.

۳. Dec : الگوریتم رمزگشایی که یک الگوریتم قطعی است، ورودی آن متن رمزشده c و کلید خصوصی sk و خروجی آن متن اصلی $m \in \mathcal{M}$ یا نماد $\perp$ (به معنای نامعتبر بودن متن رمزی) است که به صورت $m := \mathsf{Dec}_{sk}(c) \in \mathcal{M} \cup \{\perp\}$ نمایش می‌دهیم.

۴. شرط صحت: $\forall\ (pk, sk) \leftarrow \mathsf{Gen}(1^n)\ \forall\ m \in \mathcal{M}\ \ \mathsf{Dec}_{sk}(\mathsf{Enc}_{pk}(m)) = m$.

امنیت در سامانه‌های کلید همگانی

مشابه رمزهای متقارن مفهوم امنیت در رمزهای کلید همگانی را نیز با استفاده از آزمایش‌ها تعریف می‌کنیم. از آن‌جایی که این آزمایش‌ها تا حد زیادی مشابه آزمایش‌های رمزنگاری متقارن است، تمرکز ما بیشتر روی تفاوت آن‌ها با آزمایش‌های رمزنگاری متقارن خواهد بود.

امنیت در مقابل حمله‌ی متن اصلی انتخابی

یک تفاوت اساسی رمزنگاری کلید همگانی با رمزنگاری متقارن در این است که، چون همه از جمله مهاجم به کلید عمومی دسترسی دارند، لذا مهاجم، خودبه‌خود به اوراکل رمزنگاری دسترسی دارد.

تعریف ۴۹.۱ (آزمایش تمایزناپذیری در حضور مهاجم شنودگر). این آزمایش را به ازای پارامتر امنیتی 1^n با $\mathsf{PubK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n)$ نمایش می‌دهیم که شامل مراحل زیر است.

۱. چالشگر با اجرای $\mathsf{Gen}(1^n)$ کلیدهای (pk, sk) را تولید می‌کند.

۲. مهاجم $\mathcal{A}$ که کلید عمومی pk را دارد دو متن اصلی هم طول $m_0, m_1 \in \mathcal{M}$ را انتخاب کرده، به چالشگر می‌دهد.

۳. چالشگر بیت تصادفی یکنواخت $b \in \{0,1\}$ را انتخاب کرده، سپس متن رمزشده‌ی $c \leftarrow \mathsf{Enc}_{pk}(m_n)$ را محاسبه می‌کند و به مهاجم می‌دهد. به c متن رمزی چالش می‌گوییم.

۴. مهاجم بیت b' را تولید می‌کند. خروجی آزمایش ۱ است هرگاه $b = b'$ که به معنای مؤفقیت مهاجم است و با $\mathsf{PubK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n) = 1$ نمایش می‌دهیم، در غر این‌صورت مهاجم شکست خورده و خروجی آزمایش ۰ خواهد بود.

تعریف ۵۰.۱ (امنیت تک‌پیامی). سامانه‌ی رمزنگاری کلید همگانی $\Pi(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ دارای امنیت تک‌پیامی یا تمایزناپذیر در برابر مهاجم شنودگر است هرگاه به ازای هر مهاجم چندجمله‌ای تصادفی نظیر $\mathcal{A}$ تابع ناچیز ε وجودد داشته باشد که، $Pr\{\mathsf{PubK}^{\mathrm{eav}}_{\mathcal{A},\Pi}(n) = 1\} \leq \frac{1}{2} + \varepsilon(n)$.

از آن‌جایی که مهاجم به کلید عمومی و در نتیجه اوراکل رمزنگاری دسترسی دارد، گزاره‌ی زیر برقرار است.

گزاره ۵۱.۱. اگر سامانه‌ی رمزنگاری کلیدهمگانی امنیت تک‌پیامی داشته باشد امنیت متن اصلی انتخابی هم دارد.

برهان. به [۴۰]، رجوع کنید. □

مثال ۲۳.۱ مؤید این است که گزاره فوق در رمزنگاری متقارن صادق نیست.

ناممکن بودن امنیت کامل در رمزنگاری نامتقارن

تعریف ۵۲.۱ (امنیت کامل در رمزنگاری نامتقارن). سامانه‌ی رمزنگاری کلیدهمگانی Π دارای امنیت کامل است هرگاه برای هرمهاجم (نه لزوماً چندجمله‌ای تصادفی) $\mathcal{A}$ تابع ناچیز $\varepsilon(n)$ وجود داشته باشد که

$$Pr\{\mathsf{PubK}^{\text{eav}}_{\mathcal{A},\Pi}\}(n) \leq \frac{۱}{۲}$$

گزاره ۵۳.۱. سامانه‌ی کلید همگانی با امنیت کامل وجود ندارد.

برهان. مهاجم با توانایی محاسباتی نامحدود $\mathcal{A}$ را در نظر بگیرید که در آزمایش $\mathsf{PubK}^{\text{eav}}_{\mathcal{A},\Pi}(n)$ شرکت کند. فرض کنید به ازای یک زوج کلید عمومی و خصوصی (pk, sk) رمز شده‌ی m با کلید عمومی pk (و یک مقدار تصادفی که احتمالاً در الگوریتم $\mathsf{Enc}_{pk}(\cdot)$ استفاده می‌شود) برابر با c باشد. به سبب این‌که الگوریتم رمزگشایی قطعی است، شرط صحت عملکرد سامانه‌ی رمزنگاری نامتقارن لازم می‌دارد تا حاصل رمزنگاری هر متن غیر از m با کلید عمومی pk و هر مقدار تصادفی ممکن برابر با c نباشد. از این رو کافی است مهاجم $\mathcal{A}$ هر دو متن اصلی $m_۰$ و $m_۱$ را به ازای تمام مقادیر تصادفی ممکن مورد استفاده در الگوریتم $\mathsf{Enc}_{pk}(\cdot)$ رمزکرده و با متن رمزی چالش c مقایسه کند. چنین مهاجمی با احتمال ۱ در آزمایش $\mathsf{PubK}^{\text{eav}}_{\mathcal{A},\Pi}(n)$ پیروز است و حکم ثابت می‌شود. □

همان طور که سامانه‌های رمزنگاری متقارن قطعی فاقد امنیت متن اصلی انتخابی بودند، سامانه‌های رمزگاری نامتقارن نیز چنین هستند.

قضیه ۵۴.۱. هر سامانه‌ی رمزنگاری کلیدهمگانی قطعی فاقد امنیت متن اصلی انتخابی است.

برهان. به [۴۰] مراجعه کنید. □

قضیه‌ی فوق اگر چه خیلی ساده است اما سامانه‌های کلیدهمگانی اولیه که در عمل به‌کار گرفته می‌شدند تا مدت‌ها از نوع سامانه‌های کلیدهمگانی قطعی بودند، دلیل این امر شاید این بود

که زمانی که رمزنگاری کلید همگانی معرفی شد هنوز اهمیت رمزنگاری تصادفی به طور کامل شناخته نشده بود.

در واقع رمزنگاری کلید همگانی قطعی بسیار آسیب پذیر است و نباید مورد استفاده قرار گیرد زیرا علاوه بر این که مانند سامانه‌های متقارن قطعی، به مهاجم اجازه می‌دهد ارسال مجدد یک پیام رمزشده را تشخیص دهد، بلکه به وی این امکان را می‌دهد تا در صورت کوچک بودن فضای پیام اصلی، متن رمز شده را با احتمال ۱ رمزگشایی کند! برای مثال فرض کنید یک استاد نمرات دانشجویان خود را که یکی از حروف مجموعه‌ی $\{A, B, C, D, F\}$ است با کلید عمومی هر یک از آن‌ها رمز کند. اگر رمزنگاری قطعی باشد مهاجم که می‌داند نمره‌ی هر دانشجو رمزشده‌ی یکی از اعضای مجموعه‌ی فوق است، چون کلید عمومی را دارد قادر است هر یک از ۵ حالت ممکن را بیازماید تا به نمره‌ی تک‌تک دانشجویان پی‌ببرد!

گزاره ۱.۵۵. سامانه‌ی رمزنگاری نامتقارن که امنیت متن اصلی انتخابی داشته باشد امنیت چندپیامی هم دارد.

برهان. به [۴۰] مراجعه کنید. □

سامانه‌ی رمزنگاری کلیدهمگانی الجمال

در سال ۱۹۸۵، طاهرالجمال [۱۳] پی‌برد که پروتکل تبادل کلید دیفی_هلمن را می‌توان طوری اصلاح کرد تا یک سامانه‌ی رمزنگاری کلید همگانی به‌دست آید. فرض کنید $\mathbb{G}$ یک گروه دوری از مرتبه‌ی q با مولد g است و $k \in \mathbb{G}$ کلیدی باشد که A و B با استفاده پروتکل تبادل کلید دیفی_ هلمن به اشتراک گذاشته‌اند. سامانه ی رمزنگاری متقارن با فضای پیام $\mathbb{G}$ را به این صورت در نظر بگیرید که برای رمزکردن پیام $m \in \mathbb{G}$ آن را در k ضرب و برای رمزگشایی متن رمزی $c \in \mathbb{G}$ آن را در k^{-1} ضرب کنیم. به‌راحتی ثابت می‌شود این سامانه‌ی متقارن امنیت کامل نیز دارد. اکنون ایده‌ی فوق را به نحوی اصلاح می‌کنیم تا به یک سامانه‌ی کلید همگانی تصادفی دست‌یابیم. مثل قبل فرض کنید $\mathcal{G}$ یک الگوریتم چندجمله‌ای تصادفی باشد که ورودی آن پارامتر امنیتی 1^n و خروجی آن یک گروه دوری $\mathbb{G}$ از مرتبه‌ی q (که $\| q \| = n$) و مولدی نظیر $g \in \mathbb{G}$ باشد.

تعریف ۱.۵۶ (سامانه‌ی رمزنگاری کلیدهمگانی الجمال). این سامانه متشکل از الگوریتم‌های زیر است.

- Gen : الگوریتم تولید کلید که با دریافت 1^n الگوریتم $\mathcal{G}(1^n)$ را اجرا کرده و سه‌تایی $(\mathbb{G}, q, g)$ را به‌دست می‌آورد. بعد از آن $x \in \mathbb{Z}_q$ را به صورت تصادفی یکنواخت انتخاب کرده و

[۱۳] رمزنگاری مصری، متولّد ۱۹۵۵ که پس از طی دوره‌ی کارشناسی در دانشگاه قاهره، دوره ی ارشد و دکتری خود را در رشته‌ی مهندسی برق دانشگاه استنفورد و تحت راهنمایی هلمن گذراند.

$h := g^x$ را محاسبه می‌کند. در نهایت خروجی آن، کلید عمومی $pk = \langle \mathbb{G}, q, g, h \rangle$ و کلید خصوصی $sk = \langle \mathbb{G}, q, g, x \rangle$ است. (فضای پیام عبارت است از $\mathbb{G}$)

- Enc : ورودی آن کلید عمومی $pk = \langle \mathbb{G}, q, g, h \rangle$ و پیام $m \in \mathbb{G}$ و خروجی آن متن رمز شده است که پس از انتخاب تصادفی یکنواخت $y \in \mathbb{Z}_q$، به‌صورت $c = \langle h^y \cdot m \rangle$، محاسبه می‌شود.

- Dec : ورودی آن کلید خصوصی $sk = \langle \mathbb{G}, q, g, x \rangle$ و متن رمزشده‌ی $\langle c_۱, c_۲ \rangle$ و خروجی آن عبارت است از: $m' = \mathsf{Dec}_{sk}(c_۱, c_۲) := \frac{c_۲}{c_۱^x}$.

فرض کنید $\langle c_۱, c_۲ \rangle = \langle g^y, h^y \cdot m \rangle$ و $h = g^x$. در این صورت:

$$m' = \mathsf{Dec}_{sk}(c_۱, c_۲) = \frac{c_۲}{c_۱^x} = \frac{h^y \cdot m}{(g^y)^x} = \frac{(g^x)^y \cdot m}{g^{xy}} = \frac{g^{xy} \cdot m}{g^{xy}} = m.$$

و رمزگشایی به درستی انجام می‌شود.

قضیه ۵۷.۱. اگر فرض دیفی ـ هلمن تصمیمی نسبت به $\mathcal{G}$ برقرار باشد، سامانه‌ی رمزنگاری الجمال دارای امنیت متن اصلی انتخابی است.

برهان. به [۴۰، ص.۴۰۲]، رجوع کنید. □

در رمزهای متقارن برای رسیدن به هدف احراز اصالت، از کدهای احراز اصالت استفاده می‌کردیم. در ادامه مشابه کلید همگانی کدهای احراز اصالت که امضای دیجیتال نام دارد را معرفی می‌کنیم.

تعریف ۵۸.۱ (امضای دیجیتال). یک سامانه‌ی امضای دیجیتال یک سه‌تایی از الگوریتم‌های چندجمله‌ای مثل (Gen, Sign, Vrfy) است که در شرایط زیر صدق کند.

- Gen: الگوریتم تولید کلید با دریافت پارامتر امنیتی ۱^n زوج کلید خصوصی و عمومی (pk, sk) را تولید می‌کند.

- Sign: الگوریتم امضا با دریافت پیام $m \in \mathcal{M}$ و کلید خصوصی sk ، امضای $\sigma \leftarrow \mathsf{Sign}_{sk}(m)$ را محاسبه می‌کند.

- Vrfy: الگوریتم تصدیق، که ورودی آن کلید همگانی pk ، پیام اصلی m و امضای σ و خروجی آن بیت $b = \mathsf{Vrfy}(m, \sigma) \in \{۰, ۱\}$ است به طوری که $b = ۱$ بیان‌گر معتبر بودن و $b = ۰$ نشان‌دهنده نامعتبر بودن امضا است.

- شرط صحت: $\forall (pk, sk) \leftarrow \mathsf{Gen}(۱^n)\ \forall m \in \mathcal{M}\ \ \mathsf{Vrfy}(m, \mathsf{Sign}_{sk}(m)) = ۱$.

همان‌طور که امضای سنتی باید طوری طراحی شود که جعل ناپذیر باشد، اولین انتظار از یک سامانه‌ی امضای دیجیتال نیز داشتن ویژگی جعل‌ناپذیری است. برای تعریف امنیت جعل‌ناپذیری امضای دیجیتال ابتدا آزمایش جعل‌ناپذیری امضای دیجیتال را تعریف می‌کنیم.

تعریف ۵۹.۱ (آزمایش جعل‌ناپذیری امضای دیجیتال). فرض کنید پارامتر امنیتی برابر با 1^n باشد. آزمایش جعل‌ناپذیری امضای دیجیتال با $\mathsf{PrivK}_{\mathcal{A},\Pi}$ داده می‌شود و شامل مراحل زیر است.

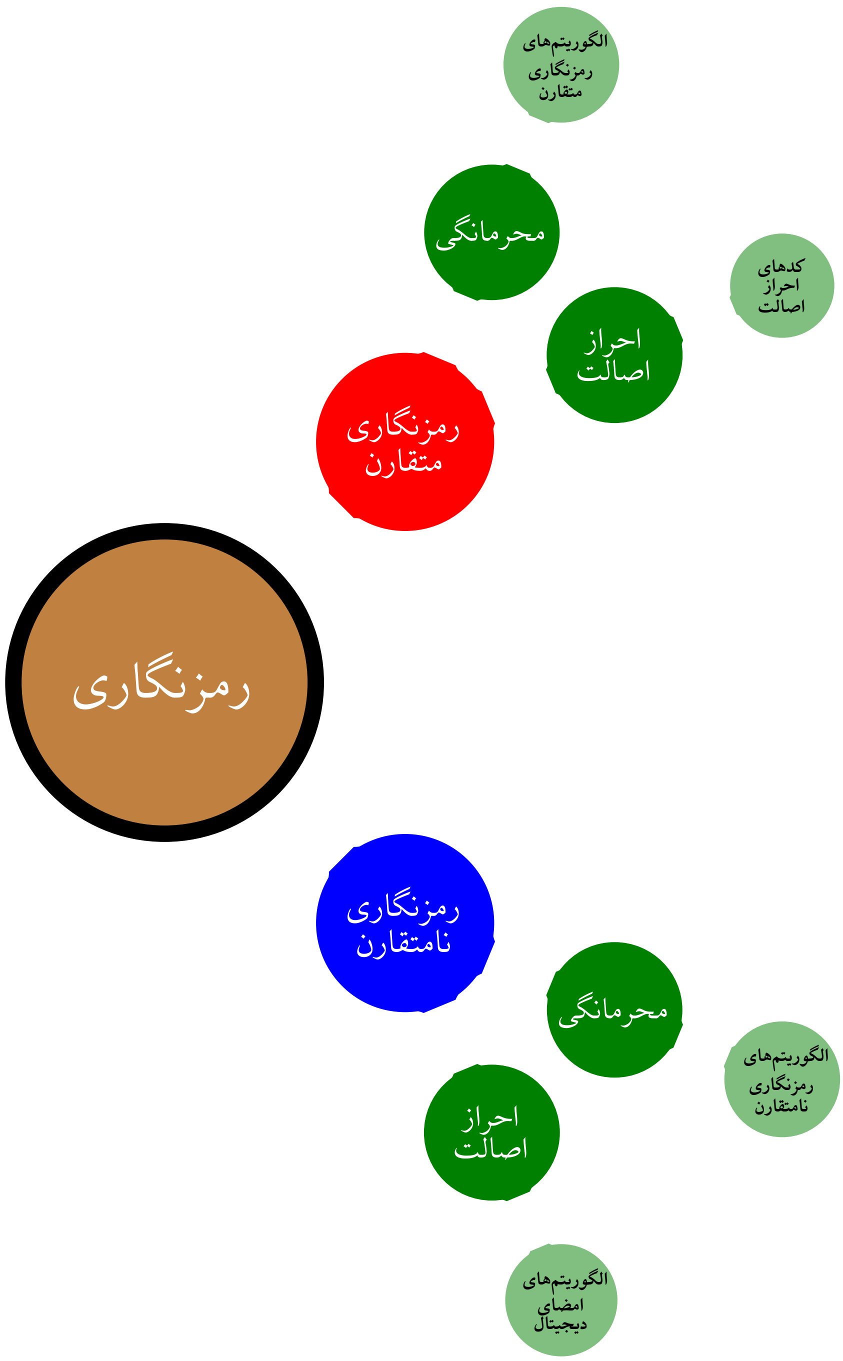
۱. $(pk, sk) \leftarrow \mathsf{Gen}(1^n)$

۲. مهاجم که به الگوریتم امضای $\mathsf{Sign}_{pk}(\cdot)$، دسترسی اوراکلی دارد، زوج متن اصلی و امضای (m, σ) را تولید می‌کند. $(m, \sigma) \leftarrow \mathcal{A}^{\mathsf{Sign}_{sk}(\cdot)}(1^n)$.

۳. فرض کنید Q، مجموعه‌ی همه پیام‌هایی باشد که امضای آن‌ها در مرحله‌ی ۲ توسط مهاجم مورد پرسمان قرار گرفته است. در این‌صورت خروجی آزمایش ۱ است هرگاه $m \notin Q$ و $\mathsf{Vrfy}(m, \sigma) = 1$.

تعریف ۶۰.۱ (امنیت جعل‌ناپذیری امضای دیجیتال). سامانه امضای دیجیتال $\Pi = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$، دارای امنیت جعل‌ناپذیری است هرگاه، به ازای هر مهاجم چندجمله‌ای تصادفی $\mathcal{A}$، تابع ناچیز $\varepsilon(n)$ وجود داشته باشد به‌طوری که، $Pr\{\mathsf{PrivK}_{\Pi,\mathcal{A}}(n) = 1\} \leq \varepsilon(n)$.

با ابزارهایی که تا کنون معرفی کرده‌ایم دست‌یابی به دو هدف اساسی رمزنگاری یعنی محرمانگی و احراز اصالت میسر خواهد بود. این ابزارهای اساسی به طور مختصر در دو دسته متقارن و نامتقارن در شکل ۷.۱، نمایش داده شده است.



شکل ۷.۱: رسیدن به اهداف محرمانگی و احراز اصالت در رمزهای متقارن و نامتقارن

فصل ۲

پایه گروبنر و پایه مرزی

۱.۲ پایه‌های گروبنر

پایه گروبنر مولدی با ویژگی‌های خوب برای یک ایده‌ال دلخواه از حلقه‌ی چندجمله‌ای‌های است. در این فصل با پایه‌های گروبنر و نحوه‌ی محاسبه‌ی آن به روش بوخبرگر آشنا می‌شویم. هدف نهایی ما استفاده از پایه گروبنر در حل دستگاه معادلات چندجمله‌ای به‌دست آمده از سامانه‌های رمزنگاری است که در فصل ۴ به آن می‌پردازیم. بیشتر مطالب این فصل برگرفته از مراجع [۴۵، ۴۶] و [۲۳]، است و برای مشاهده‌ی جزئیات بیشتر می‌توانید به آن‌ها مراجعه نمایید. در ابتدا برخی نمادهای پرکاربرد در این فصل را معرفی می‌کنیم.

- Kیک میدان (نه لزوماً بطور جبری بسته) است، که بستار جبری آن را با $\bar{K}$ نمایش می‌دهیم.

- $\mathbb{F}_p$ را یک میدان متناهی از مرتبه‌ی عدد اوّل p و $\mathbb{F}_{p^n}$ را یک توسیع متناهی $\mathbb{F}$ از درجه‌ی n در نظر می‌گیریم.

- حلقه‌ی چندجمله‌ای‌های n متغیره را با $P := K[x_1, ..., x_n]$ نمایش می‌دهیم.

- هر عبارت به‌صورت $m = x_1^{\alpha_1}...x_n^{\alpha_n}$ را که $\alpha_i \in \mathbb{Z}^n_{\geq 0}$، *یکجمله‌ای* و عباراتی به‌صورت $c \cdot m$ که m یک یکجمله‌ای و $c \in K$ را، یک *جمله* می‌نامیم. گاهی برای اختصار یکجمله‌ای $x_1^{\alpha_1}...x_n^{\alpha_n}$ را به‌صورت x^α نمایش می‌دهیم و $\alpha = (\alpha_1, ..., \alpha_n)$ در x^α را بردار نما یا توان x^α می‌نامیم.

- مجموعه همه یکجمله‌ای‌های n متغیره را با $\mathbb{T}^n$ نمایش می‌دهیم.

$$\mathbb{T}^n := \{x_1^{\alpha_1}, ..., x_n^{\alpha_n} \ : \ (\alpha_1, ..., \alpha_n) \in \mathbb{Z}^n_{\geq 0}\}.$$

- مجموعه همه یکجمله‌ای‌های، چندجمله‌ای $f = \sum_{\alpha \in \mathbb{Z}^n_{\geq 0}} a_\alpha x^\alpha \in P$ را با

$$\mathrm{Supp}(f) := \{x^\alpha \in \mathbb{T}^n : \ a_\alpha \neq 0\}.$$

و مجموعه همه جمله‌های آن‌را با

$$T(f) := \{a_\alpha x^\alpha : \ a_\alpha \neq 0\}.$$

نمایش می‌دهیم. این تعریف برای مجموعه‌ای از چندجمله‌ای‌ها مثل $F = \{f_1, ..., f_m\}$ به صورت زیر قابل تعمیم است:

$$\mathrm{Supp}(F) := \bigcup_{i=1}^{m} \mathrm{Supp}(f_i), \ T(f) := \bigcup_{i=1}^{m} T(f_i).$$

- درجه‌ی کلی یا به اختصار درجه‌ی یکجمله‌ای $m = x_1^{\alpha_1} ... x_n^{\alpha_n}$، عبارت است از $\deg(m) = \sum_{i=1}^n \alpha_i$ و آن را با $\deg(\alpha)$ یکسان در نظر می‌گیریم. این تعریف را برای چندجمله‌ای $f \in P$ به‌صورت زیر تعمیم می‌دهیم.

$$\deg(f) = \max\{\deg(m) \ : \ m \in M(f)\}.$$

- منظور ما از پایه یک ایده‌ال، یک مجموعه مولد متناهی آن ایده‌ال است.

ترتیب یکجمله‌ای

در حلقه چندجمله‌ای‌های یک متغیره $K[x]$ برای مقایسه یکجمله‌ای‌های یک متغیره ترتیبی به‌صورت

$$... > x^{m+1} > x^m > ... > x^2 > x > 1.$$

را در نظر می‌گرفتیم و با استفاده از آن دو یکجمله‌ای را مقایسه می‌کردیم، این مقایسه یکی از بخش‌های اساسی در تقسیم چندجمله‌ای‌ها و یا عملیات حذفی گاوس به شمار می‌رفت. در ادامه قصد داریم ترتیب را برای یکجمله‌ای‌ها با بیش از یک متغیر تعریف کنیم.

یکجمله‌ای‌های n متغیره، به همراه عمل ضرب یک تکواره یا نیم‌گروه [1] است که با تکواره‌ی

[1] تکواره یا نیم‌گروه یک مجموعه به همراه یک عمل دوتایی است، که تحت آن عمل، بسته، شرکت پذیر و دارای عضو همانی است.

$\mathbb{Z}^n_{\geq 0}$ با عمل جمع برداری یکریخت است. تابع یکریختی از $\mathbb{T}^n$ به $\mathbb{Z}^n_{\geq 0}$ همان تابع لگاریتم است:

$$\log(\cdot) : \mathbb{T}^n \to \mathbb{Z}^n_{\geq 0}$$

$$\log(x_1^{\alpha_1}...x_n^{\alpha_n}) = (\alpha_1, ..., \alpha_n).$$

بنابراین هر ترتیب روی $\mathbb{Z}^n_{\geq 0}$ یک ترتیب روی یکجمله‌ای‌ها را نتیجه می‌دهد. به همین دلیل در ادامه هر یکجمله‌ای $x^\alpha = x_1^{\alpha_1}...x_n^{\alpha_n}$ در حلقه‌ی $K[x_1, ..., x_n]$ را با بردار نمای آن، α یکسان در نظر می‌گیریم.

تعریف ۱.۲ (ترتیب یکجمله‌ای). رابطه $<$ روی $\mathbb{Z}^n_{\geq 0}$، (یا بطور معادل روی یکجمله‌ای‌های $\mathbb{T}^n$) را یک *ترتیب یکجمله‌ای* گوییم، هرگاه در شرایط زیر صدق کند:

- $<$ یک رابطه‌ی ترتیب تام (یا خطّی) باشد.
- اگر $\alpha > \beta$ و $\gamma \in \mathbb{Z}^n_{\geq 0}$ آن‌گاه $\alpha + \gamma > \beta + \gamma$.
- $\mathbb{Z}^n_{\geq 0}$ تحت رابطه‌ی $<$ خوش‌ترتیب باشد، یعنی هر زیرمجموعه‌ی ناتهی $\mathbb{Z}^n_{\geq 0}$ دارای کوچکترین عضو نسبت به ترتیب $<$ باشد.

فرض کنید $<$ یک رابطه ترتیب یکجمله‌ای روی $\mathbb{Z}^n_{\geq 0}$ باشد و $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$، منظور از $\alpha \geq \beta$ این است که $\alpha > \beta$ یا $\alpha = \beta$. در ضمن $\alpha \geq \beta$ را با $x^\alpha \geq x^\beta$ معادل در نظر می‌گیریم. لم زیر شرطی معادل برای خوش‌ترتیب بودن یک رابطه‌ی ترتیب است که به فهم معنای خوش‌ترتیبی کمک می‌کند.

لم ۲.۲. رابطه‌ی ترتیب $<$ روی $\mathbb{Z}^n_{\geq 0}$ یک رابطه‌ی خوش‌ترتیبی است اگر و تنها اگر هر دنباله‌ی اکیداً نزولی در $\mathbb{Z}^n_{\geq 0}$ مثل $\alpha_1 > \alpha_2 > \alpha_3 > ...$ سرانجام ایستا شود. یعنی $N \geq 1$ وجود داشته باشد که $\alpha_N = \alpha_{N+1} = \alpha_{N+2} = ...$.

برهان. رجوع کنید به [۲۳]. $\square$

در ادامه با چند ترتیب یکجمله‌ای پرکاربرد آشنا می‌شویم.

مثال ۳.۲. فرض کنیم $\alpha, \beta \in \mathbb{Z}^n_{\geq 0}$.

۱. **ترتیب الفبایی** گوییم $\alpha \underset{lex}{>} \beta$ هر گاه اولین درایه‌ی ناصفر از سمت چپ در بردار تفاضل $\alpha - \beta$ مثبت باشد. در ضمن $x^\alpha \underset{lex}{>} x^\beta$ هر گاه $\alpha \underset{lex}{>} \beta$.

۲. **ترتیب الفبایی مدرج** $\alpha \underset{deglex}{>} \beta$ هرگاه،

- $\deg(\alpha) > \deg(\beta)$،

- یا $\deg(\alpha)=\deg(\beta)$ و $\alpha \underset{lex}{>} \beta$.

۳. **ترتیب الفبایی مدرج معکوس** $\alpha \underset{degrevlex}{>} \beta$ هرگاه،

- $\deg(\alpha)>\deg(\beta)$ یا،
- $\deg(\alpha)=\deg(\beta)$ و اولین درایه‌ی ناصفر از سمت راست در بردار تفاضلبل $\alpha-\beta \in \mathbb{Z}^n_{\geq \circ}$ منفی باشد.

ترتیب یکجمله‌ای را می‌توانیم به ترتیبی برای چندجمله‌ای‌ها تبدیل کنیم، به این صورت که برای مقایسه‌ی دو چندجمله‌ای بزرگترین یکجمله‌ای آن‌ها را و اگر برابر بودند یکجمله‌ای‌های بعدی آن را با هم مقایسه می‌کنیم.
با استفاده از ترتیب یکجمله‌ای دیگری که درمثال بعدی آورده‌ایم، می‌توانیم دو ترتیب یکجمله‌ای را با هم ترکیب کنیم.

مثال ۴.۲ (ترتیب ضربی یا قالبی). فرض کنید $K[x,y]=K[x_۱,\dots,x_n,y_۱,\dots,y_m]$ و $>_x$ و $>_y$ به ترتیب رابطه‌های ترتیب یکجمله‌ای روی $K[x]$ و $K[y]$ باشند. ترتیب ضربی روی $K[x,y]$ که با $(>_x,>_y)$ نیز نمایش می‌دهیم، به‌صورت زیر تعریف می‌شود.

$$x^Ay^B > x^ay^b \iff x^A >_x x^a \ \vee (x^A = x^a \ \wedge \ y^B >_y y^b)$$

برای مثال می‌توان ترتیب الفبایی روی $K[x_۱,\dots,x_n]$ را نوعی ترتیب ضربی در نظر گرفت که از ضرب ترتیب‌های الفبایی روی قالب‌های کوچکتر $K[x_i]$ به ازای $i=۱,\dots,n$ به‌دست آمده.

تعریف ۵.۲. [۲۳] فرض کنید $f=\sum a_\alpha x^\alpha$ یک چندجمله‌ای ناصفر در $K[x_۱,\dots,x_n]$ و $>$ یک ترتیب یکجمله‌ای روی $\mathbb{T}^n$ باشد. در این‌صورت،

$$\begin{aligned}
\text{درجه‌ی مرکب } f &= \operatorname{multideg}(f) := \max_{\geq}\{\alpha\in\mathbb{Z}^n_{\geq\circ} :\ a_\alpha \neq \circ\},\\
\text{ضریب پیشرو } f &= \operatorname{LC}(f) := a_{\operatorname{multideg}(f)} \in K,\\
\text{یکجمله‌ای پیشرو } f &= \operatorname{LM}(f) := x^{\operatorname{multideg}(f)},\\
\text{جمله‌ی پیشرو } f &= \operatorname{LT}(f) := \operatorname{LC}(f)\cdot\operatorname{LM}(f).
\end{aligned}$$

ترتیب حذف، که در زیر آن را تعریف می‌کنیم، نقشی اساسی در حل دستگاه معادلات چندجمله‌ای ایفا خواهد کرد.

تعریف ۶.۲ (خاصیت حذف). فرض کنید $P=K[x_۱,\dots,x_n]$ و $L\subseteq\{x_۱,\dots,x_n\}$ و $\widehat{P}:=$ $K[x_i \mid x_i\notin L]$ حلقه‌ی چندجمله‌ای‌های متشکل از متغیرهایی باشد که در L نیستند. در این‌صورت

ترتیب یکجمله‌ای $<$ روی $\mathbb{T}^n$ دارای خاصیت حذف برای L است هرگاه

$$\forall\ f \in \boldsymbol{K}[x_۱, ..., x_n]\ \left\{\mathrm{LM}(f) \in \widehat{P} \Rightarrow f \in \widehat{P}\right\}.$$

تعریف ۷.۲. هر ترتیب یکجمله‌ای روی $\boldsymbol{K}[x_۱, ..., x_n]$ را که دارای خاصیت حذف برای $L \subseteq \{x_۱, ..., x_n\}$ باشد، *ترتیب حذف* برای L گوییم.

در واقع ترتیب حذف برای L به گونه‌ای است که اگر متغیرهای درون L در جمله‌ی پیشرو چندجمله‌ای f ظاهر نشده باشند با اطمینان می‌توان گفت که در سایر جملات آن نیز ظاهر نشده‌اند.

مثال ۸.۲. فرض کنید $۱ \leq j < n$، و $L = \{x_۱, ..., x_j\}$ ترتیب $>_{\mathrm{Elim(L)}}$ روی $\mathbb{T}^n$ را به‌صورت زیر تعریف می‌کنیم

$$\alpha >_{\mathrm{Elim(L)}}\ \beta \iff \begin{cases} \alpha_۱ + \cdots + \alpha_j > \beta_۱ + \cdots + \beta_j \\ \text{یا} \\ \alpha_۱ + \cdots + \alpha_j = \beta_۱ + \cdots + \beta_j\ \wedge\ \alpha \underset{degrevlex}{>} \beta. \end{cases}$$

این ترتیب یک ترتیب یکجمله‌ای و دارای خاصیت حذف برای $x_۱, ..., x_j$ است. در این ترتیب اگر x^α یک یکجمله‌ای از $\boldsymbol{K}[x_۱, ..., x_n]$ و حداقل شامل یکی از $x_۱, ..., x_j$ باشد، آن‌گاه به ازای هر یکجمله‌ای دیگر نظیر x^β که فقط از متغیرهای $x_{j+۱}, ..., x_n$ تشکیل شده است داریم، $x^\alpha >_{\mathrm{Elim(L)}} x^\beta$.

در نهایت ترتیبی را معرفی می‌کنیم که همه‌ی ترتیب‌های قبلی را می‌توان با استفاده از آن به‌دست آورد.

تعریف ۹.۲ (ترتیب وزنی). فرض کنید $w_۱, ..., w_m$ بردارهایی در $\mathbb{R}^n$ و $w \cdot \alpha$ نشان‌دهنده‌ی ضرب داخلی بردارهای w و α باشد و ماتریس

$$W = \begin{pmatrix} w_{۱۱} & \cdots & w_{۱n} \\ \vdots & \vdots & \vdots \\ w_{m۱} & \cdots & w_{mn} \end{pmatrix}_{m \times n}$$

ماتریسی باشد که سطرهای آن را بردارهای w_i تشکیل می‌دهند. در این صورت *ترتیب وزنی* برای مقایسه‌ی $\alpha, \beta \in \mathbb{Z}^n_{\geq ۰}$ نسبت به بردارهای وزن $w_۱, ..., w_m$ را با $>_W$ نمایش می‌دهیم و به‌صورت زیر تعریف می‌شود:

$$\alpha >_W \beta \iff \exists\, l \in \{۱, ..., m\}\ (\forall\ i < l\ \ w_i \cdot \alpha = w_i \cdot \beta)\ \wedge\ (w_l \cdot \alpha > w_l \cdot \beta).$$

مثال ۱۰.۲. می‌توان ثابت کرد همه‌ی ترتیب‌های یکجمله‌ای روی $\mathbb{Z}^n_{\geq 0}$ را می‌توان با استفاده از بردارهای وزن مناسب، به‌صورت یک ترتیب وزنی به‌دست آورد. برای مثال ماتریس وزنِ ترتیب وزنی متناظر با ترتیب‌های الفبایی، الفبایی مدرج و الفبایی مدرج معکوس در $K[x_1, x_2, x_3]$ با فرض $x_1 > x_2 > x_3$ عبارتند از:

$$W_{lex} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \; W_{deglex} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \; W_{degrevlex} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

به‌راحتی می‌توان دید که سطر آخر ماتریس میانی که متناظر با ترتیب الفبایی مدرج است، تأثیری در نتیجه ندارد و ماتریس بدون این سطر هم کار خود را درست انجام می‌دهد. این اتفاقی نیست بطور کلی از دل هر ماتریس وزن غیر مربعی متناظر با یک ترتیب یکجمله‌ای می‌توانیم یک ماتریس مربعی استخراج کنیم به‌طوری که ترتیب یکجمله‌ای متناظر با آن همان ترتیب قبلی باشد. در ضمن اگر $M_{n\times n}$ ماتریس مربعی استخراج شده باشد ترتیبی که به‌صورت زیر تعریف می‌شود، همان ترتیب یکجمله‌ای متناظر با ماتریس وزن اصلی است و گاهی به آن ترتیب نمایش داده شده با ماتریس M هم می‌گویند.

$$\alpha >_M \beta \iff \text{اولین درایه‌ی ناصفر بردار ستونی } M \times (\alpha - \beta)^{\mathrm{tr}} \text{ مثبت باشد}$$

در ادامه خواهیم دید که محاسبات (بخصوص محاسبه‌ی پایه‌ی گروبنر) با استفاده از ترتیب الفبایی مدرج معکوس کم‌هزینه‌تر و سریع‌تر از محاسبه با استفاده از ترتیب‌های حذف نظیر ترتیب الفبایی است، در عوض چون ترتیب الفبایی یک ترتیب حذف است، از آن برای استخراج جواب دستگاه معادلات چندجمله‌ای از روی پایه‌ی گروبنر محاسبه شده برای ایده‌ال دستگاه معادلات، استفاده خواهیم کرد.

الگوریتم تقسیم

قضیه ۱۱.۲ (الگوریتم تقسیم). فرض کنید $<$ یک ترتیب یکجمله‌ای روی $\mathbb{Z}^n_{\geq 0}$ و $f \in K[x_1, \dots, x_n]$ یک چندجمله‌ای دلخواه و $F = (f_1, \dots, f_s)$ یک s تایی مرتب از چندجمله‌ای‌های در $K[x_1, \dots, x_n]$ باشد. در این صورت مراحل اجرای الگوریتم تقسیم ۱، متناهی است و برای چندجمله‌ای‌های به‌دست آمده از خروجی آن یعنی $a_1, \dots, a_s, r$ داریم

$$f = a_1 f_1 + \dots + a_s f_s + r,$$

همچنین شرایط زیر برقرار خواهند بود:

۱. r یا برابر صفر است، یا ترکیبی K- خطی از یکجمله‌ای‌هایی است که هیچ‌یک از آن‌ها بر هیچ‌یک از یکجمله‌ای‌های $\mathrm{LM}(f_1), ..., \mathrm{LM}(f_s)$ بخش‌پذیر نیست.

۲. به ازای هر $1 \leq i \leq s$، اگر $a_i f_i \neq 0$ آن‌گاه، $\mathrm{LM}(f) \geq \mathrm{LM}(a_i f_i)$.

۳. به ازای هر $1 \leq i \leq s$ و هر یکجمله‌ای غیر صفر q_i مثل t ، جمله‌ی $t \cdot \mathrm{LT}(f_i)$ بر هیچ‌یک از $\mathrm{LT}(f_1), ..., \mathrm{LT}(f_s)$ بخش‌پذیر نیست.

به r باقی‌مانده‌ی تقسیم f بر F می‌گوییم و با $r = \bar{f}^F$ نمایش می‌دهیم. در ضمن چندجمله‌ای‌های $a_1, ..., a_s, r$ که در سه شرط فوق صدق کنند، به‌صورت یکتا توسط بردار $(f_1, ..., f_s, f) \in P^{s+1}$ تعیین می‌شوند.

الگوریتم ۱ الگوریتم تقسیم در $K[x_1, ..., x_n]$

Input: $(f_1, ..., f_s, f)$ یک $s+1$ تایی مرتب از چندجمله‌ای‌ها در P.
Output: $(a_1, ..., a_s, r)$ یک $s+1$ تایی مرتب از چندجمله‌ای‌ها در P.

```
a_i ← 0; r ← 0; p ← f
while p ≠ 0 do
  i ← 0
  divisionoccured ← False
  while i ≥ s  and divisionoccured = False do
    if LT(f_i) | LT(p) then
      a_i ← a_i + LT(p)/LT(f_i)
      p ← p − (LT(p)/LT(f_i)) f_i
      divisionoccured = True
    else
      i ← i + 1
    end if
  end while
  if divisionoccured = False then
    r ← r + LT(p)
    p ← p − LT(p)
  end if
end while
return (a_1, ..., a_s, r)
```

برهان. به [۴۶، ص، ۷۱] رجوع کنید. □

در الگوریتم تقسیم ۱ با تغییر ترتیب یکجمله‌ای $<$ و یا جایگشت f_i ها در بردار $F = (f_1, ..., f_s)$، خروجی الگوریتم نیز ممکن است تغییر کند.

ایده‌ال‌های یکجمله‌ای و لم دیکسون

تعریف ۱۲.۲ (ایده‌ال یکجمله‌ای). ایده‌ال $I \subseteq P$ را یک *ایده‌ال یکجمله‌ای* گوییم هرگاه زیرمجموعه‌ی $A \subseteq \mathbb{Z}^n_{\geq 0}$ موجود باشد به‌طوری که، $I = \langle x^\alpha \mid \alpha \in A \rangle$.

لم زیر روشی برای تشخیص عضویت یک یکجمله‌ای در یک ایده‌ال یکجمله‌ای را بیان می‌کند.

لم ۱۳.۲. فرض کنید $I = \langle x^\alpha \mid \alpha \in A \rangle$ یک ایده‌ال یکجمله‌ای باشد. در این‌صورت یکجمله‌ای x^β در I قرار دارد اگر و تنها اگر $\alpha \in A$ وجود داشته باشد به‌طوری که، $x^\alpha \mid x^\beta$.

برهان. رجوع کنید به [۲۳]. □

یکجمله‌ای x^β بر یکجمله‌ای x^α بخش‌پذیر است هرگاه $\gamma \in \mathbb{Z}_{\geq 0}$ وجود داشته باشد به‌طوری‌که داشته باشیم، $x^\beta = x^\alpha \cdot x^\gamma$ و یا به‌صورت معادل $\beta = \alpha + \gamma$. در نتیجه بردار نمای همه‌ی یکجمله‌ای‌هایی که بر x^α بخش‌پذیرند برابر است با:

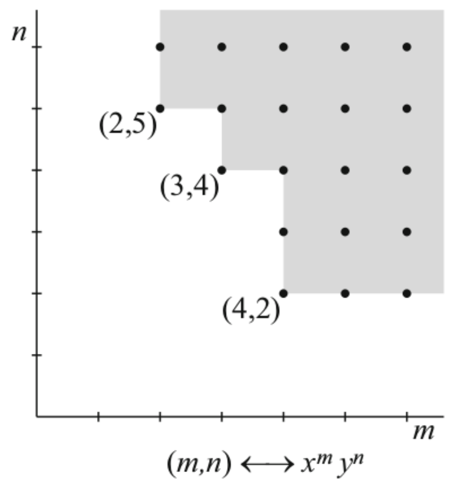
$$\alpha + \mathbb{Z}^n_{\geq 0} = \{\alpha + \gamma \mid \gamma \in \mathbb{Z}^n_{\geq 0}\}.$$

با توجّه به گزاره‌ی فوق و قضیه‌ی ۱۳.۲ می‌توانیم با داشتن مولّدی برای ایده‌ال یکجمله‌ای، همه‌ی یکجمله‌ای‌های آن‌را مشخص کنیم. به مثال زیر توجّه کنید.

مثال ۱۴.۲. فرض کنید $I = \langle x^۴y^۲, x^۳y^۴, x^۲y^۵ \rangle$، در این‌صورت مجموعه بردارهای توان همه یکجمله‌ای‌های I عبارت است از:

$$((۴,۲) + \mathbb{Z}^۲_{\geq 0}) \cup ((۳,۴) + \mathbb{Z}^۲_{\geq 0}) \cup ((۲,۵) + \mathbb{Z}^۲_{\geq 0}).$$

شکل ۱.۲ این نقاط را نشان می‌دهد.



شکل ۱.۲: نمایشی از یکجمله‌ای‌های ایده‌ال I

با استفاده از لم ۱۳.۲ می‌توانیم عضویت یک یکجمله‌ای در یک ایده‌ال یکجمله‌ای را تشخیص دهیم، امّا چطور می‌توانیم به عضویت یک چندجمله‌ای در ایده‌ال یکجمله‌ای پی‌ببریم؟ لم بعدی بیان می‌کند که فقط با بررسی عضویت یکجمله‌ای‌های یک چندجمله‌ای می‌توان به عضویت آن در یک ایده‌ال یکجمله‌ای پی‌برد.

لم ۱۵.۲. اگر $I \subseteq P$ یک ایده‌ال یکجمله‌ای باشد و $f \in K[x_1, ..., x_n]$، در این صورت گزاره‌های زیر معادل هستند.

۱. $f \in I$.

۲. هر جمله‌ی f در I قرار دارد.

۳. f ترکیبی K- خطی از یکجمله‌ای‌های عضو I است.

برهان. رجوع کنید به [۲۳]. □

لم زیر بیانگر این حقیقت است که هر ایده‌ال یکجمله‌ای، مجموعه مولّدی متناهی، متشکل از یکجمله‌ای‌ها دارد.

لم ۱۶.۲ (لم دیکسون). فرض کنید $I = \langle x^\alpha \mid \alpha \in A \rangle$ یک ایده‌ال یکجمله‌ای باشد، در این صورت عدد طبیعی s و $\alpha_1, ..., \alpha_s \in A$ وجود دارند به‌طوری که، $I = \langle x^{\alpha_1}, ..., x^{\beta} \rangle$.

برهان. رجوع کنید به [۲۳، ص، ۷۲]. □

یکی از نتایج لم دیکسون، به‌دست آوردن یک شرط معادل به‌صورت زیر، برای خوش‌ترتیبی یک رابطه روی $\mathbb{Z}^n_{\geq 0}$ است که کار ما را در بررسی خوش‌ترتیب بودن تحت یک رابطه‌ی ترتیب، راحت‌تر می‌کند.

نتیجه ۱۷.۲. فرض کنید $>$ یک رابطه روی $\mathbb{Z}^n_{\geq 0}$ باشد که در شرایط زیر صدق می‌کند

۱. $>$ یک رابطه‌ی ترتیب تام باشد.

۲. اگر $\alpha > \beta$ و $\gamma \in \mathbb{Z}^n_{\geq 0}$ آن‌گاه $\alpha + \gamma > \beta + \gamma$. در این‌صورت $>$ یک رابطه‌ی خوشترتیبی است اگر و تنها اگر، به ازای هر $\alpha \in \mathbb{Z}^n_{\geq 0}$ داشته باشیم، $\alpha \geq 0$.

برهان. رجوع کنید به [۲۳، ص، ۷۳]. □

یک ایده‌ال یکجمله‌ای می‌تواند پایه‌های یکجمله‌ای زیادی داشته باشد، ولی فقط یکی از آن‌ها فاقد هر گونه افزونگی و عنصر اضافی است که به‌صورت زیر تعریف می‌شود.

تعریف ۱۸.۲ (پایه‌ی یکجمله‌ای مینیمال). پایه‌ی یکجمله‌ای $\{x^{\alpha_1}, ..., x^{\alpha_s}\}$ برای یک ایده‌ال یکجمله‌ای مثل I را *مینیمال* گوییم، هر گاه به ازای هر $i \neq j$ داشته باشیم، $x^{\alpha_i} \nmid x^{\alpha_j}$.

گزاره ۱۹.۲. هر ایده‌ال یکجمله‌ای یک پایه‌ی یکجمله‌ای مینیمال یکتا دارد.

برهان. به [۲۳، ص، ۷۴] رجوع کنید. □

اکنون به اندازه‌ی کافی با یکجمله‌ای‌ها و ایده‌ال‌های یکجمله‌ای آشنا شده‌ایم، در ادامه به بررسی ایده‌ال‌های چندجمله‌ای و رابطه آن‌ها با ایده‌ال‌های یکجمله‌ای می‌پردازیم. پایه گروبنر که موضوع بخش بعدی است، یک پل ارتباطی خوب بین ایده‌ال‌های چندجمله‌ای و ایده‌ال‌های یکجمله‌ای است و به‌وسیله ی آن می توانیم بسیاری از مسائل مطرح در ایده‌ال‌های چندجمله‌ای را با ابزارهای یکجمله‌ای حل کنیم.

پایه‌های گروبنر

تعریف ۲۰.۲ (ایده‌ال یکجمله‌ای‌های پیشرو). فرض کنید $I \subseteq P$ یک ایده‌ال چندجمله‌ای باشد. ایده‌ال تولید شده توسط مجموعه‌ی

$$\mathrm{LM}(I) := \{\mathrm{LM}(f) \mid f \in I \backslash \{0\}\}$$

را *ایده‌ال یکجمله‌ای‌های پیشرو* یا به اختصار *ایده‌ال پیشرو* I نامیده و به‌صورت $\langle \mathrm{LM}(I) \rangle$ نمایش می‌دهیم.

مجموعه $\mathrm{LT}(I)$ نیز مجموعه همه‌ی جملات پیشرو چندجمله‌ای‌های موجود در I را تشکیل می‌دهد. گزاره‌ی زیر بیان می‌کند که $\langle \mathrm{LM}(I) \rangle$ و $\langle \mathrm{LT}(I) \rangle$ ایده‌ال‌های یکسانی هستند.

گزاره ۲۱.۲. فرض کنید $I \subseteq K[x_1, ..., x_n]$ یک ایده‌ال ناصفر باشد. در این‌صورت گزاره‌های زیر برقرارند.

۱. $\langle \mathrm{LM}(I) \rangle = \langle \mathrm{LT}(I) \rangle$.

۲. چندجمله‌ای‌های $g_1, ..., g_t \in I$ موجودند به‌طوری که $\langle \mathrm{LM}(I) \rangle = \langle \mathrm{LM}(g_1), ..., \mathrm{LM}(g_t) \rangle$. یا بطور معادل $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_1), ..., \mathrm{LT}(g_t) \rangle$.

برهان. رجوع کنید به [۲۳].

□

قضیه ۲۲.۲ (قضیه‌ی پایه‌ی هیلبرت). حلقه‌ی $P = K[x_1, ..., x_n]$، نوتری است، به عبارت دیگر هر ایده‌ال آن متناهی مولد است.

برهان. با توجه به گزاره‌ی ۲، قضیه‌ی قبل و الگوریتم تقسیم ۱، حکم واضح است. □

لازم به ذکر است که پایه یک ایده‌ال یکتا نیست و یک ایده‌ال می‌تواند نامتناهی پایه داشته باشد.

قضیه ۲۳.۲. در هر حلقه جابجایی مثل R گزاره‌های زیر معادل‌اند

۱. R نوتری است.

۲. اگر $\cdots \subseteq I_n \subseteq \cdots \subseteq I_۳ \subseteq I_۲ \subseteq I_۲ \subseteq I_۱$ یک زنجیر صعودی از ایده‌ال‌های R باشد، سرانجام ایستا خواهد بود، یعنی عدد طبیعی N وجود دارد به‌طوری که $I_N = I_{N+۱} = I_{N+۲} = \cdots$.

برهان. فرض کنید گزاره‌ی ۱ برقرار باشد و فرض کنید $\cdots \subseteq I_n \subseteq \cdots \subseteq I_۳ \subseteq I_۲ \subseteq I_۲ \subseteq I_۱$ یک زنجیر صعودی دلخواه از ایده‌ال‌های R باشد. چون دنباله‌ی ایده‌ال‌های I_n صعودی است لذا مجموعه‌ی $I = \bigcup_{n=۱}^{\infty}$ نیز یک ایده‌ال R خواهد بود. با توجه به برقراری گزاره‌ی ۱، اعضای $f_۱, ..., f_s \in I$ موجودند به‌طوری که $I = \langle f_۱, ..., f_s \rangle$. چون هر یک از f_i ها در I است در نتیجه به ازای هر f_i یک $N_i \in \mathbb{N}$ وجود دارد به‌طوری که $f_i \in I_{N_i}$. فرض کنید $N = \max\limits_{۱ \leq i \leq s} N_i$، در این‌صورت به ازای هر $i \in \{۱, ..., s\}$ داریم، $f_i \in I_N$ و در نتیجه $I = I_N$. چون زنجیر صعودی از ایده‌ال‌ها را دلخواه انتخاب کرده بودیم گزاره‌ی ۲ نتیجه می‌شود.

در جهت عکس، از برهان خلف استفاده می‌کنیم. فرض کنید با وجود برقراری گزاره‌ی ۲ ایده‌الی مثل I وجود دارد که متناهی مولّد نیست. فرض کنید $f_۱ \in I$ ، چون $I \neq \langle f_۱ \rangle$ لذا $f_۲ \in I$ وجود دارد به‌طوری که $f_۲ \notin \langle f_۱ \rangle$. بنابراین $\langle f_۱ \rangle \subsetneq \langle f_۱, f_۲ \rangle$. چون I متناهی مولهد نیست می‌توانیم به‌صورت متوالی و بی‌پایان روند فوق را ادامه دهیم و یک زنجیر صعودی ناایستا از ایده‌ال‌های R بسازیم، ولی این با گزاره‌ی ۲ در تناقض است. $\square$

اکنون این سؤال پیش می‌آید که، آیا یکجمله‌ای‌های پیشرو هر پایه از ایده‌ال I، مثل $\langle g_۱, ..., g_s \rangle$ می‌توانند پایه‌ی برای ایده‌ال یکجمله‌ای‌های پیشرو باشند، به عبارت دگر آیا رابطه‌ی $\langle \mathrm{LT}(I) \rangle = \langle \mathrm{LT}(g_i) \rangle_{i=۱}^{s}$ برقرار است؟ پاسخ منفی است و چنان‌چه در مثال زیر خواهیم دید هر پایه‌ای لزوماً چنین ویژگی را ندارد.

مثال ۲۴.۲. دو چندجمله‌ای $f = x^۲ - y$ و $g = x^۳ - x$ را با ترتیب الفبایی روی $\mathbb{R}[x, y]$ در نظر بگیرید. فرض کنید $I = \langle f, g \rangle$. در این‌صورت داریم:

$$xy - x = -x \cdot f + g \Rightarrow xy - x \in I \Rightarrow xy \in \langle \mathrm{LM}(I) \rangle.$$

از طرفی چون

$$\mathrm{LM}(f) = x^۲ \nmid xy, \ \mathrm{LM}(g) = x^۳ \nmid xy$$

طبق لم ۱۳.۲، نتیجه می‌گیریم، $xy \notin \langle \mathrm{LM}(f), \mathrm{LM}(g) \rangle$، و بنابراین $\langle \mathrm{LM}(f), \mathrm{LM}(g) \rangle \neq \langle \mathrm{LM}(I) \rangle$.

تعریف ۲.۲۵ (پایه‌ی گروبنر). [۲۳] فرض کنید I یک ایده‌ال حلقه‌ی $P = \boldsymbol{K}[x_1, ..., x_n]$ و $>$ یک رابطه‌ی ترتیب یکجمله‌ای روی P باشد. $G = \{g_1, ..., g_m\}$ را *پایه‌ی گروبنر* یا *پایه‌ی استاندارد* ایده‌ال I گوییم، هرگاه

$$\langle \mathrm{LM}(g_1), ..., \mathrm{LM}(g_m)\rangle = \langle \mathrm{LM}(I)\rangle.$$

در ضمن با توجه به قرارداد $\{0\} = \langle\emptyset\rangle$ مجموعه‌ی $\emptyset$ را پایه‌ی گروبنر ایده‌ال $\{0\}$ در نظر می‌گیریم.

با توجه به این‌که $\langle \mathrm{LT}(I)\rangle = \langle \mathrm{LM}(I)\rangle$ لذا در تعریف پایه‌ی گروبنر می‌توانیم جای $\mathrm{LM}(\cdot)$ را با $\mathrm{LM}(\cdot)$ عوض کنیم.

نتیجه ۲.۲۶. اگر G یک پایه‌ی گروبنر ایده‌ال $I \subseteq P$ باشد، طبق لم ۲.۱۳ به ازای هر $f_i \in I$ یک $g_i \in G$ وجود دارد به‌طوری که، $\mathrm{LM}(g_i) \mid \mathrm{LM}(f_i)$.

سؤالی که پیش می‌آید این است که آیا هر ایده‌ال، پایه‌ی گروبنر دارد، در این‌صورت آیا پایه‌ی گروبنر می‌تواند مجموعه‌ی مولّدی برای ایده‌ال باشد؟ قضیه‌ی زیر به این سؤال پاسخ مثبت می دهد.

قضیه ۲.۲۷. هر ایده‌ال $I \subseteq P$ نسبت به هر ترتیب یکجمله‌ای دلخواه روی P، دارای پایه‌ی گروبنر است. در ضمن هر پایه‌ی گروبنر I ، پایه‌ای برای I نیز است.

برهان. رجوع کنید به [۲۳]. □

همان‌طور که قبلاً ذکر شد، در تقسیم چندجمله‌ای f بر $F = (f_1, ..., f_s) \in P^s$ با اعمال یک جایگشت روی F باقی‌مانده‌ی تقسیم $r = \bar{f}^F$ نیز ممکن بود تغییر کند، ولی یکی از ویژگی‌های خوب پایه‌ی گروبنر که در قضیه‌ی بعد نشان داده شده، این است که باقی‌مانده‌ی تقسیم بر پایه‌ی گروبنر تحت جایگشت مقسوم‌علیه‌ها ناوردا است.

قضیه ۲.۲۸. فرض کنید $G = \{g_1, ..., g_s\}$ یک پایه‌ی گروبنر برای ایده‌ال $I \subseteq P$ و $f \in P$ یک چندجمله‌ای دلخواه باشد. در این‌صورت باقی‌مانده‌ی تقسیم f بر $(g_1, ..., g_s)$ مستقل از ترتیب g_i ها، یکتا است.

برهان. فرض کنید r و r' باقی‌مانده‌های تقسیم f بر دو جایشگت دلخواه از اعضای G باشند. در این‌صورت چون $r - r' \in I$، داریم:

$$\mathrm{LT}(r - r') \in \langle \mathrm{LM}(I)\rangle = \langle \mathrm{LM}(g_1), ..., \mathrm{LM}(g_s)\rangle.$$

فرض کنید $r \neq r'$، طبق لم ۲.۱۳ ، $\mathrm{LM}(r - r')$ بر یکی از $\mathrm{LM}(g_i)$ ها بخش‌پذیر خواهد بود، که با باقی‌مانده بودن r و r' در الگوریتم تقسیم ۱ در تناقض است. در نتیجه $r = r'$. □

باقی‌مانده‌ی تقسیم f بر پایه‌ی گروبنری مثل G را *فرم نرمال* f نسبت به G نامیده و با $NF_G(f)$ یا اگر ابهامی نباشد با $NF(f)$ نمایش می‌دهیم. در این‌حالت چون باقی‌مانده تحت جایگشت مقسوم علیه‌ها ناوردا است، G را به‌جای یک لیست یا s تایی مرتب به‌صورت یک مجموعه در نظر می‌گیریم.

نکته ۲۹.۲. اگرچه باقی‌مانده‌ی تقسیم f بر G وقتی G پایه‌ی گروبنر باشد، یکتا است ولی، خارج‌قسمت‌ها در الگوریتم تقسیم ۱ با تغییر ترتیب مقسوم‌علیه‌ها تغییر خواهند کرد.

قضیه‌ی بعد نشان می‌دهد که چطور با استفاده از پایه‌ی گروبنر و الگوریتم تقسیم می‌توانیم مسئله‌ی عضویت در ایده‌ال را حل کنیم.

قضیه ۳۰.۲. فرض کنید $G = \{g_1, ..., g_s\}$ یک پایه‌ی گروبنر ایده‌ال $I \subseteq P$ و $f \in P$ یک چندجمله‌ای دلخواه باشد. در این صورت:

$$f \in I \iff \bar{f}^G = NF_G(f) = 0.$$

برهان. به [۲۳، ص.۸۴] رجوع کنید. □

قضیه‌ی ۲۷.۲ وجود پایه‌ی گروبنر برای هر ایده‌ال چندجمله‌ای را تضمین می‌کند، ولی هیچ روشی برای به‌دست آوردن آن ارائه نمی‌دهد. در بخش بعدی الگوریتمی معرفی می‌کنیم که به ازای یک مولّد داده شده از ایده‌ال، پایه‌ی گروبنر آن را محاسبه می‌کند.

الگوریتم بوخبرگر

فرض کنید پایه‌ای از یک ایده‌ال داده شده است، چگونه می‌توان فهمید که پایه‌ی مورد نظر، پایه‌ی گروبنر است یا خیر؟ برای پاسخ به این سؤال باید ببینیم چه مشکلی می‌تواند خاصیت پایه‌ی گروبنر بودن یک مولّد ایده‌ال را از آن سلب کند. فرض کنید $G = \{f_1, ..., f_m\} \subseteq P$ مولّدی برای ایده‌ال I باشد. طبق تعریف پایه‌ی گروبنر اگر $m \in \langle \mathrm{LM}(I) \rangle$ وجود داشته باشد به‌طوری که:

$$m \notin \langle \mathrm{LM}(f_1), ..., \mathrm{LM}(f_m) \rangle,$$

آن‌گاه G پایه‌ی گروبنر ایده‌ال I نخواهد بود. امّا چندجمله‌ای‌هایی مثل m چطور ممکن است بوجود بیایند؟ قبلاً در مثال ۲۴.۲ مشاهده شد که ممکن است جملات پیشرو دو چندجمله‌ای مولّد I، نظیر f_i و f_j در جملاتی مانند ax^α و bx^β ضرب شوند و سپس در ترکیب، $s = ax^\alpha f_i - bx^\beta f_j$، قرینه هم شده و حذف شوند. می‌دانیم که $s \in I$ و در نتیجه $\mathrm{LM}(s) \in \langle \mathrm{LM}(I) \rangle$. ولی در صورتی که آن‌چه گفته شد رخ دهد، و جملات پیشروی دو چندجمله‌ای f_i و f_j در s حذف

شوند، آن‌گاه جمله‌ی پیشرو s توسط هیچ یک از جملات پیشرو f_i و f_j بخش نمی‌شود و لذا $s \notin \langle \mathrm{LM}(f_1), ..., \mathrm{LM}(f_s)\rangle$ و G پایه‌ی گروبنر نخواهد بود. بوخبرگ ثابت کرده است که فقط با حلّ همین مشکل، هر مولّدی را می‌توان به یک پایه‌ی گروبنر گسترش داد. S- چندجمله‌ای‌ها روشی برای به‌دست آوردن ترکیباتی با خاصیت فوق هستند.

تعریف ۳۱.۲ (S ـ چندجمله‌ای). فرض کنید f و g چندجمله‌ای‌های ناصفری در $K[x_1, ..., x_n]$ و به ترتیب با یکجمله‌ای‌های پیشروی x^α و x^β باشند و x^γ کوچکترین مضرب مشترک x^α و x^β باشد. یعنی به ازای هر $1 \leq i \leq n$ داشته باشیم، $\gamma_i = max(\alpha_i, \beta_i)$ که به‌صورت $x^\gamma = \mathrm{LCM}(x^\alpha, x^\beta)$ نمایش می‌دهیم. در این‌صورت چندجمله‌ای

$$S(f, g) := \frac{x^\gamma}{\mathrm{LT}(f)} f - \frac{x^\gamma}{\mathrm{LT}(g)} g$$

را S- چندجمله ای f و g می‌نامیم.

تعریف $S(f, g)$ به‌گونه‌ای است که f و g بعد از ضرب شدن در جملات مناسب دارای جملات پیشروی یکسان می‌شوند و سپس این جملات یکسان در ترکیب $S(f, g)$ حذف خواهند شد. نمونه‌ای از این فرآیند در مثال زیر نشان داده شده.

مثال ۳۲.۲. $f = x^3y^2 - x^2y^3 + x$ و $g = 3x^4y + yy^2$ در $\mathbb{Q}[x, y]$ را با ترتیب الفبایی مدرج معکوس در نظر بگیرید، در این‌صورت $\gamma = (4, 2)$ و داریم:

$$\begin{aligned} S(f, g) =& \frac{x^4y^2}{x^3y^2} \cdot f - \frac{x^4y^2}{3x^4y} \cdot g \\ =& x \cdot f - \frac{1}{3} \cdot y \cdot g \\ =& - x^3y^3 - \frac{1}{3}y^3 + x^2. \end{aligned}$$

لم زیر نشان می‌دهد، هرگاه ترکیبی از چندجمله‌ای‌ها منجر به حذف جملات پیشرو شود، چنین حذفی، با S- چندجمله‌ای‌های آن‌ها نیز قابل دست‌یابی است.

لم ۳۳.۲. فرض کنید همه‌ی جمعوندهای مجموع $s = \sum_{i=1}^{s} g_i$ دارای درجه‌ی مرکب یکسان δ باشند. اگر $\mathrm{multideg}(s) < \delta$ ، آن‌گاه s ترکیبی خطی از $S(g_i, g_j)$ ها با $\mathrm{multideg}(S(g_i, g_j)) < \delta$ به ازای $1 \leq j, k \leq s$ خواهد بود.

برهان. به [۲۳، ص، ۸۵] رجوع کنید. $\square$

اگر $g_۱, ..., g_s$ در شرایط لم فوق صدق کنند در این‌صورت داریم:

$$\sum_{i=۱}^{s} g_i = \sum_{j,k} c_{jk} S(g_i, g_k).$$

در سمت چپ رابطه‌ی فوق هر جمعوند دارای درجه‌ی مرکب δ است و لذا حذف جملات قرینه و کاهش درجه مرکب بعد از عمل جمع رخ می‌دهد، در حالی که درجه‌ی مرکب هر جمعوند عبارت سمت راست کمتر از δ است و این‌یعنی حذف جملات، قبل از جمع و در خود S- چندجمله‌ای‌ها رخ داده است. در نتیجه همه‌ی حذف‌های ممکن با استفاده از S- چندجمله‌ای‌ها میسر است و به همین دلیل بوخبرگر برای اطمینان از پایه‌ی گروبنر بودن یک مولد داده شده بررسی S- چندجمله‌ای‌ها را کافی می‌دانست و آزمون زیر را ارائه داد.

قضیه ۳۴.۲ (محک بوخبرگر). پایه‌ی $G = \{g_۱, ..., g_s\}$ برای ایده‌ال چندجمله‌ای I یک پایه‌ی گروبنر I است، اگر و تنها اگر برای هر زوج (i, j) که $i \neq j$ ، باقی‌مانده‌ی تقسیم $S(g_i, g_j)$ بر G صفر باشد.

برهان. به [۲۳، ص، ۸۶] مراجعه کنید. □

برای معرفی محک دیگری، برای تشخیص پایه‌ی گروبنر، مفهومی تحت عنوان تحویل یا تقلیل یافتن به صفر را به‌صورت زیر تعریف می‌کنیم.

تعریف ۳۵.۲. مجموعه‌ی چندجمله‌ای‌های $G = \{g_۱, ..., g_s\}$ در $K[x_۱, ..., x_n]$ را با یک ترتیب یکجمله‌ای دلخواه در نظر بگیرید. می‌گوییم چندجمله‌ای f به پیمانه‌ی G *به صفر تحویل می‌یابد* و با $f \underset{G}{\rightarrow} ۰$ نمایش می‌دهیم هرگاه:

$$\exists\, a_۱, ..., a_s \in K[x_۱, ..., x_n] \;\; f = a_۱ g_۱ + \cdots + a_۱ g_s$$

به‌طوری که به ازای هر $a_i g_i \neq ۰$ داشته باشیم، $\mathrm{LM}(f) \geq \mathrm{LM}(a_i g_i)$.

می‌توانیم تعریف فوق را حالت خاصّی از تعریف زیر در نظر بگیریم.
با توجه به تعریف فوق اگر $\bar{f}^G = ۰$ آن‌گاه $f \underset{G}{\rightarrow} ۰$ ولی مثال زیر نشان می‌دهد عکس آن درست نیست.

مثال ۳۶.۲. چندجمله‌ای $f = xy^۲ - x$ و دوتایی مرتب $G = (xy + ۱, y^۲ - ۱)$ را در $\mathbb{Q}[x, y]$ با ترتیب الفبایی مدرج معکوس در نظر بگیرید. با استفاده از الگوریتم تقسیم ۱ ، f نمایشی به‌صورت $f = y \cdot (xy + ۱) + ۰ \cdot (y^۲ - ۱) + (-x - y)$ دارد و در نتیجه $\bar{f}^G \neq ۰$. با این‌حال نمایشی دیگر به‌صورت زیر نیز دارد

$$f = ۰ \cdot (xy + ۱) + x \cdot (y^۲ - ۱),$$

که نشان می‌دهد $f \xrightarrow[G]{} 0$.

اکنون محک دیگری را برای پایه‌ی گروبنر با استفاده از مفهوم تحویل یافتگی معرفی می‌کنیم.

قضیه ۳۷.۲. پایه‌ی $G = \{g_1, ..., g_s\}$ برای ایده‌ال I ، یک پایه‌ی گروبنر است، اگر و تنها اگر برای هر $i \neq j$ داشته باشیم، $S(g_i, g_j) \xrightarrow[G]{} 0$.

برهان. به [۲۳، ص.۱۰۶] رجوع کنید. □

با استفاده از محک بوخبرگر می‌توانیم پایه‌ی گروبنر بودن یک مولّد داده شده را تشخیص دهیم. امّا از این روش برای تبدیل یک مولّد به پایه‌ی گروبنر نیز می‌توانیم استفاده کنیم. روش بوخبرگر برای ساختن یک پایه‌ی گروبنر برای ایده‌ال I با استفاده از یک مولّد داده شده این است که نخست S- چندجمله‌ای‌های دوبدوی مولّدها را محاسبه می‌کند و سپس فرم نرمال آن‌ها نسبت به مجموعه‌ی مولّد را به‌دست آورده و آن‌گاه، فرم‌های نرمال غیر صفر را به مجموعه‌ی مولّد اضافه می‌کند. این روند تا جایی ادامه پیدا می‌کند که دیگر هیچ S- چندجمله‌ای با فرم نرمال غیر صفر نسبت به مولّد جدید وجود نداشته باشد.آن‌چه در نهایت به‌دست می‌آید پایه‌ی گروبنر ایده‌ال I است. جزئیات این روش در قضیه‌ی بعد آمده است.

قضیه ۳۸.۲ (الگوریتم بوخبرگر). فرض کنید $F = (f_1, ..., f_s) \in P^s$ و I ایده‌ال تولید شده توسط $\{f_1, ..., f_s\}$ باشد. در این صورت الگوریتم بوخبرگر ۲ پس از اجرای تعداد متناهی مرحله با محاسبه‌ی پایه‌ی گروبنر ایده‌ال I (نسبت به ترتیب یکجمله‌ای در نظر گرفته شده)، به پایان می‌رسد.

برهان. رجوع کنید به [۲۳]. □

پایه‌ی گروبنری که توسط الگوریتم ساده‌ی بوخبرگر ۲ محاسبه می‌شود معمولاً دارای افزونگی و بزرگتر از حد نیاز است، به این معنی که دارای عناصر اضافی است و همان‌طور که لم زیر نشان می‌دهد با حذف کردن چنین عناصری، مجموعه‌ی باقی‌مانده باز هم پایه‌ی گروبنر باقی خواهد ماند.

لم ۳۹.۲. فرض کنید G پایه‌ی گروبنر ایده‌ال $I \subseteq K[x_1, ..., x_n]$ باشد. اگر $p \in G$ یک چندجمله‌ای باشد به‌طوری که $\mathrm{LT}(p) \in \langle \mathrm{LT}(G \backslash \{p\}) \rangle$ ، آن‌گاه $G \backslash \{p\}$ نیز یک پایه‌ی گروبنر I است.

برهان. می‌دانیم که $\langle \mathrm{LT}(G) \rangle = \langle \mathrm{LT}(I) \rangle$. بنابراین اگر $\mathrm{LT}(p) \in \langle \mathrm{LT}(G \backslash \{p\}) \rangle$ ، آن‌گاه داریم $\langle \mathrm{LT}(G \backslash \{p\}) \rangle = \langle \mathrm{LT}(G) \rangle$ و طبق تعریف پایه‌ی گروبنر، $G \backslash \{p\}$ نیز یک پایه‌ی گروبنر است. □

فرض کنید G یک پایه‌ی گروبنر برای ایده‌ال I باشد، در این‌صورت با ضرب ثابت‌های مناسب در اعضای G و تبدیل ضرایب پیشرو آن به ثابت ۱ و حذف عناصری مثل p که $\mathrm{LT}(p) \in \mathrm{LT}(G \backslash \{p\})$ به پایه‌ای می‌رسیم که به *پایه‌ی گروبنر مینیمال* معروف است. پایه‌ی گروبنر مینیمال

الگوریتم ۲ الگوریتم بوخبرگر برای محاسبه‌ی پایه گروبنر

Input: $F = (f_1, ..., f_s) \in P^s$ (یک مولّد برای ایده‌ال مورد نظر که به‌صورت یک s تایی مرتب داده شده است.)
Output: $G \supseteq F$ (یک پایه‌ی گروبنر برای ایده‌ال که شامل مولّد داده شده نیز است.)

```
G ← F
G' ← ∅
while G' ≠ G do
  G' ← G
  for (f, g) ∈ G' × G' do
    if LM(f) < LM(g) then
      r ← S(f, g)^{G'}   (remainder of S(f,g) modulo G')
      if r ≠ 0 then
        G ← G ∪ {r}
      end if
    end if
  end for
end while
return G
```

یک ایده‌ال ناصفر را می‌توانیم با استفاده از الگوریتم بوخبرگر ۲ و سپس اعمال لم ۳۹.۲ برای حذف عناصر زائد، محاسبه کنیم. اگر چه پایه‌ی مینیمال یکتا نیست ولی همه‌ی پایه‌های مینیمال دارای ویژگی مشترکی هستند که در نتیجه‌ی زیر آمده است.

نتیجه ۴۰.۲. ایده‌ال دلخواه $I \subseteq K[x_1, ..., x_n]$ و یک ترتیب یکجمله‌ای دلخواه را در نظر بگیرید. فرض کنید $G = \{g_1, ..., g_s\}$ و $H = \{h_1, ..., h_t\}$ دو پایه‌ی گروبنر مینیمال برای I باشند. در این‌صورت $t = s$ و بعد از یک اندیس‌گذاری مجدد (در صورت نیاز)، به ازای هر $i = 1, ..., s$ داریم، $\mathrm{LT}(g_i) = \mathrm{LT}(h_i)$.

برهان. چون $\mathrm{LT}(G)$ و $\mathrm{LT}(H)$ پایه‌های گروبنر مینیمال هستند، هر دو آن‌ها، پایه‌ی یکجمله‌ای مینیمال برای ایده‌ال $\mathrm{LT}(I)$ هستند. از طرفی طبق قضیه‌ی ۱۹.۲ پایه‌ی یکجمله‌ای مینیمال یکتاست، در نتیجه $\mathrm{LT}(G) = \mathrm{LT}(H)$. □

می‌توان با کمی تغییر در تعریف پایه‌ی گروبنر مینیمال، پایه‌ی گروبنر تحویل یافته را به‌صورت زیر تعریف کرد که خود یک پایه‌ی گروبنر مینیمال و خوشبختانه **یکتاست**.

تعریف ۴۱.۲ (پایه‌ی گروبنر تحویل‌یافته). پایه‌ی گروبنر G برای ایده‌ال چندجمله‌ای I را *پایه‌ی گروبنر تحویل یافته* گوییم هرگاه در شرایط زیر صدق کند:

۱. به ازای هر $p \in G$ داشته باشیم $\mathrm{LC}(p) = 1$.

۲. به ازای هر $p \in G$ اگر $m \in \mathrm{Supp}(p)$ آن‌گاه $m \notin \langle \mathrm{LT}(G\backslash\{p\})\rangle$. یعنی هیچ‌یک از یکجمله‌ای‌های یک عضو از G بر هیچ‌یک از جملات پیشرو سایر اعضای G بخش‌پذیر نباشد.

قضیه ۴۲.۲. به ازای هر ایده‌ال ناصفر I از P پایه‌ی گروبنر تحویل‌یافته وجود دارد و یکتاست.

برهان. [۲۳]. □

لازم به ذکر است که با تغییر ترتیب یکجمله‌ای، نه تنها پایه‌ی گروبنر بلکه پایه‌ی گروبنر تحویل‌یافته یک ایده‌ال می‌تواند دستخوش تغییر شود. برخی از ایده‌ال‌ها دارای مولّدی هستند که نسبت به هر ترتیب یکجمله‌ای دلخواه، یک پایه‌ی گروبنر است. چنین مولّدی را پایه‌ی گروبنر عام گویند و هر ایده‌ال لزوماً دارای چنین مولّدی نیست.

امروزه توابع کتاب‌خانه‌ای برای محاسبه‌ی پایه‌ی گروبنر در بسیاری از *سامانه‌های جبری کامپیوتری* گنجانده شده. خروجی چنین توابعی پایه‌ی گروبنری است که اعضای آن مضرب ثابتی از اعضای پایه‌ی گروبنر تحویل یافته هستند و لذا پاسخ این سامانه‌ها به یک مسئله‌ی واحد جز در یک‌سری ضرایب ثابت تفاوت چندانی نداشته و به‌راحتی می‌توان پاسخ به‌دست آمده از یک سامانه را با پاسخ سامانه‌ی دیگر تطبیق داد.

عوامل مؤثر در پیچیدگی الگوریتم بوخبرگر

اگرچه الگوریتم بوخبرگر، الگوریتمی دقیق و قطعی برای محاسبه‌ی پایه‌ی گروبنر یک ایده‌ال است، ولی متأسفانه همان‌طور که در [۶۷، ص.۵۱۱] نیز اشاره شده، پیچیدگی محاسباتی آن بر حسب اندازه‌ی ورودی چندجمله‌ای نیست. ورودی الگوریتم بوخبرگر یک مولّد داده شده برای ایده‌ال مورد نظر است و ما اندازه آن را با پارامترهایی نظیر ماکزیمم درجه چندجمله‌ای‌های ظاهر شده در آن و تعداد متغیرها و ماکزیمم ضریب ثابت بکار رفته در آن می‌سنجیم. از بین این موارد تعداد متغیرها و سپس ماکزیمم درجه‌ی ظاهر شده در میان چندجمله‌ای‌های ورودی، تأثیر بیشتری روی پیچیدگی الگوریتم می‌گذارند.

پیچیدگی محاسباتی الگوریتم بوخبرگر بسیار بالا است و این الگوریتم به هیچ‌وجه یک الگوریتم کارا برای محاسبه‌ی پایه‌ی گروبنر به‌شمار نمی‌رود. یکی از علت‌های رشد سریع زمان و حافظه‌ی مورد نیاز در این الگوریتم رشد سریع اندازه‌ی پایه‌ی در حال گسترش در حین اجرای الگوریتم است. این اندازه با ماکزیمم درجه‌ی چندجمله‌ای‌های ظاهر شده در محاسبه ارتباط مستقیم دارد. قضیه‌ی زیر کرانی برای این درجه ارائه می‌دهد.

قضیه ۴۳.۲. فرض کنید I یک ایده‌ال صفر بعدی از $\mathbb{F}_q[x_1, \dots, x_n]$ باشد، و $f_1, \dots, f_n$ که درجه‌ی هر یک از آن‌ها به ترتیب برابر $d_1, \dots, d_n$ است، مولد داده شده برای این ایده‌ال باشد. در این‌صورت اگر D ماکزیمم درجه‌ی چندجمله‌ای‌های ظاهر شده در محاسبه‌ی پایه‌ی گروبنر باشد آن‌گاه

۱. به ازای ترتیب الفبایی، $D \leq \Pi_{i=1}^{n} d_i$.

۲. به ازای ترتیب الفبایی مدرج معکوس، $D \leq 1 - n + \sum_{i=1}^{n} d_i$.

برهان. رجوع کنید به [۳۲]. □

بنابراین با توجه به قضیه‌ی فوق محاسبه‌ی پایه‌ی گروبنر با استفاده از ترتیب الفبایی مدرج معکوس سریع‌تر است.

مهمترین و البته پرهزینه‌ترین بخش الگوریتم بوخبرگر ۲، بخش محاسبه‌ی S- چندجمله‌ای و فرم نرمال آن است. در واقع عامل اصلی پیچیدگی الگوریتم بوخبرگر تعداد زیاد محاسبات S- چندجمله‌ای‌ها و فرم‌های نرمال، است. الگوریتم بوخبرگر ۲ معرفی شده در بخش‌های قبل به جهت فهم بهتر، خیلی ساده بیان شد و لذا قابلیت های زیادی برای بهینه‌تر شدن دارد. برای مثال وقتی فرم نرمال یک S- چندجمله‌ای در یک مرحله نسبت به مولد به‌دست آمده، صفر می‌شود، بدیهی است که در سایر مراحل هم فرم نرمال آن نسبت به مولد به‌دست آمده در آن مراحل صفر است، زیرا مولد هر مرحله شامل مولد مراحل قبل است. در نتیجه به عنوان اولین بهبود، وقتی چندجمله‌ای f_j را به عنوان یک عضو جدید به پایه‌ی در حال گسترش اضافه می‌کنیم، تنها فرم‌های نرمالی که باید ناصفر بودن آن‌ها را بررسی کنیم عبارتند از $\overline{S(f_i, f_j)}^{G'}$ به ازای $i \leq j - 1$، یا یک روش معادل این است که یک مجموعه شامل همه‌ی دوتایی‌هایی از چندجمله‌ای‌ها که S- چندجمله‌ای آن‌ها باید محاسبه شود، در الگوریتم در نظر بگیریم و در هر مرحله، دوتایی‌هایی را که S- چندجمله‌ای آن‌ها بررسی می‌شود را از آن مجموعه خارج کنیم تا در مرحله‌ی بعد لازم نباشد دوباره S- چندجمله‌ای و فرم نرمال آن‌ها را محاسبه کنیم.

از عوامل دیگر مؤثر در بهبودی الگوریتم بوخبرگر می‌توانیم به موارد زیر اشاره کنیم

- ترتیب انتخاب دوتایی‌های (f, g) که فرم نرمال S- چندجمله‌ای آن‌ها باید محاسبه شود در زمان اجرا مؤثر است.

- الگوریتم‌هایی وجود دارند، [۳۳، ۱۵] که پایه‌ی گروبنر به‌دست آمده تحت یک ترتیب یکجمله‌ای را به پایه‌ی گروبنر آن ایده‌ال تحت یک ترتیب یکجمله‌ای دیگر، تبدیل می‌کنند. بنابراین می‌توانیم ابتدا پایه‌ی گروبنر را نسبت به ترتیب الفبایی مدرج معکوس که سریع‌تر است، محاسبه کنیم، سپس با استفاده از این الگوریتم‌ها نتیجه‌ی به‌دست آمده را به پایه‌ی گروبنر نسبت به ترتیبی که می‌خواهیم، تبدیل کنیم.

- می‌توان با استفاده از محک‌های دیگر، از محاسبه‌های غیر ضروری فرم‌های نرمال اجتناب کرد.

در حال حاضر، بهترین الگوریتم‌های شناخته شده برای محاسبه‌ی پایه‌ی گروبنر الگوریتم‌های F4 [۳۰] و F5 [۳۱]، از فوجر [2] هستند. این الگوریتم‌ها اگرچه چندجمله‌ای نیستند، ولی زمان اجرای آن‌ها در مقایسه با سایر الگوریتم‌های پایه گروبنر کمتر است.

[2] Jean-Charles Faugère

۲.۲ پایه‌های مرزی

در بخش قبل دانستیم که یک ایده‌ال می‌تواند پایه‌های متفاوتی داشته باشد، و از بین آن‌ها پایه‌ی گروبنر را که دارای ویژگی‌های خوبی بود، مورد بررسی قرار دادیم. امّا پایه‌ی گروبنر تنها پایه‌ی خوبی نیست که می‌شناسیم، در این بخش قصد داریم تا پایه‌ای دیگر از ایده‌ال را که *پایه‌ی مرزی* نام دارد مورد بررسی قرار دهیم، خواهیم دید این پایه در واقع تعمیمی از پایه‌ی گروبنر است که در آن به‌جای ترتیب‌های یکجمله‌ای از ایده‌ال ترتیبی استفاده می‌کنیم. پایه‌های مرزی از این جهت که از لحاظ عددی بهتر از پایه‌های گروبنر رفتار می‌کنند [۶۳]، نقشی کلیدی در جبر محاسباتی دارد. افرادی همچون آوزینگر [3] و استتر [4] [۶]، مولر [5] [۵۱] و مورِین [6] [۵۳] تجربیات مؤفقی در استفاده از این پایه برای حل دستگاه معادلات چندجمله‌ای که ایده‌ال تولید شده توسط معادلات آن صفر بعدی باشد، داشته‌اند. این موضوع از یک سو و اعتقاد ما بر این‌که الگوریتم‌های یافتن پایه‌ی مرزی،میزان حافظه کم‌تری نسبت به الگوریتم‌های محاسبه‌ی پایه‌ی گروبنر مصرف می‌کنند، از سوی دیگر، انگیزه‌ی اصلی ما در مطالعه‌ی پایه‌های مرزی است. مباحث این بخش تا حد زیادی برگرفته از [۴۵] و [۴۱] است و خواننده می‌تواند برای مشاهده‌ی جزئیات بیشتر به آن‌ها رجوع کند.

وجود و یکتایی

در پایه‌های گروبنر با مفهوم ترتیب‌های یکجمله‌ای آغاز کردیم که یک ابزار کلیدی ما در الگوریتم‌های تقسیم و الگوریتم بوخبرگر و سایر الگوریتم‌های محاسبه پایه گروبنر بود. در این‌جا به‌جای ترتیب یکجمله‌ای از یک مفهوم دیگر که ایده‌ال ترتیبی نام دارد استفاده می‌کنیم.

تعریف ۴۴.۲ (ایده‌ال ترتیبی). مجموعه‌ی ناتهی $\mathcal{O} \subset \mathbb{T}^n$ را در نظر بگیرید.

۱. **بستار** $\mathcal{O}$ را به‌صورت زیر تعریف می‌کنیم:

$$\overline{\mathcal{O}} := \{t \in \mathbb{T}^n \mid \exists t' \in \mathcal{O} \ : \ t \mid t'\}.$$

به عبارت دیگر $\overline{\mathcal{O}}$ شامل همه یکجمله‌ای‌هایی است که عضوی از $\mathcal{O}$ را بخش می‌کنند.

۲. $\mathcal{O}$ یک ***ایده‌ال ترتیبی*** نامیده می‌شود هر گاه، $\overline{\mathcal{O}} = \mathcal{O}$.

از تعریف فوق مشخص است که ایده‌ال ترتیبی واقعا یک ایده‌ال از حلقه‌ی $P = K[x_1, ..., x_n]$ نیست بلکه مجموعه‌ای از یکجمله‌ای‌هاست که تحت عامل‌های سازنده‌ی اعضایش بسته است.

[3] Auzinger
[4] Stetter
[5] Möller
[6] Mourrain

مثال ۴۵.۲. به‌راحتی می‌توان دید که مجموعه‌های زیر ایده‌ال ترتیبی هستند.

$$\mathcal{O}_1 = \{1\},\ \mathcal{O}_2 = \{x_1^i :\ 0 \le i \le k,\ k \in \mathbb{N}\},\ \mathcal{O}_3 = \{1, x_1, x_1^2, x_1x_2, x_2\}.$$

با استفاده از یک ایده‌ال ترتیبی داده شده می‌توانیم به‌صورت زیر ایده‌ال‌های ترتیبی جدیدی بسازیم.

تعریف ۴۶.۲. فرض کنید مجموعه همه یکجمله‌ای‌های n متغیره با درجه کمتر یا مساوی d را با $\mathbb{T}^n_{\le d}$ و یکجمله‌ای‌های با درجه‌ی دقیقاً مساوی d را با $\mathbb{T}^n_d$ نمایش دهیم. اکنون فرض کنید $\mathcal{O} \subseteq \mathbb{T}^n$ یک ایده‌ال ترتیبی باشد.

۱. **مرز** ایده‌ال ترتیبی $\mathcal{O}$ را به‌صورت زیر تعریف می‌کنیم:

$$\partial\mathcal{O} := \mathbb{T}^n_1\mathcal{O}\backslash\mathcal{O} = \{x_it|\ 1 \le i \le n, t \in \mathcal{O}\}\backslash\mathcal{O}.$$

در ضمن قرارداد می‌کنیم که، $\partial\emptyset := \{1\}$.

۲. **اولین بستار مرزی** $\mathcal{O}$ عبارت است از، $\overline{\partial\mathcal{O}} := \mathcal{O} \cup \partial\mathcal{O}$.

۳. به ازای هر $k \ge 1$، $k+1$ **امین مرز** $\mathcal{O}$ را به‌صورت $\partial^{k+1}\mathcal{O} = \partial(\overline{\partial^k\mathcal{O}})$, و $k+1$ **امین بستار مرزی** $\mathcal{O}$ را به‌صورت $\overline{\partial^{k+1}\mathcal{O}} = \overline{\partial^k\mathcal{O}} \cup \partial^{k+1}\mathcal{O}$, تعریف می‌کنیم.

نتیجه ۴۷.۲. k امین بستار ایده‌ال ترتیبی $\mathcal{O}$ یعنی $\overline{\partial^k\mathcal{O}}$ به ازای هر $k \ge 0$ یک ایده‌ال ترتیبی است.

با دیدن مثال زیر وجه تسمیه‌ی در نظر گرفتن نام مرز برای $\partial\mathcal{O}$ روشن می‌شود.

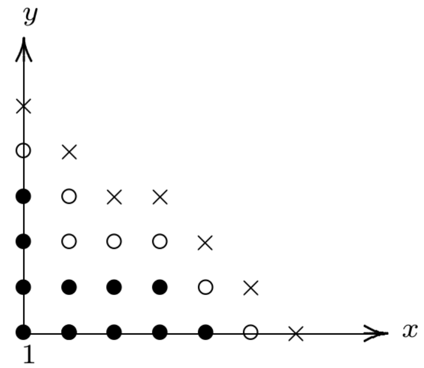
مثال ۴۸.۲. فرض کنید $\mathcal{O} = \{1, x, y, x^2, xy, y^2, x^3, x^2y, y^3, x^4, x^3y\} \subseteq \mathbb{T}^n$. در این‌صورت به‌راحتی می‌توان دید که $\mathcal{O}$ یک ایده‌ال ترتیبی است. یک بصری‌سازی از این ایده‌ال ترتیبی در شکل ۲.۲ نمایش داده شده است.

گزاره ۴۹.۲ (ویژگی‌های اولیه‌ی مرز). فرض کنید $\mathcal{O} \subseteq \mathbb{T}^n$ یک ایده‌ال ترتیبی باشد.

۱. به ازای هر $k \ge 0$، $\overline{\partial^k\mathcal{O}}$ اجتماع مجزای $\partial^i\mathcal{O}$ هاست، به طوری که، $\overline{\partial^k\mathcal{O}} = \bigcup_{i=0}^k \partial^i\mathcal{O}$. در ضمن، $\mathbb{T}^n = \bigcup_{i=0}^\infty \partial^i\mathcal{O}$.

۲. به ازای هر $k \ge 1$، داریم، $\partial^k\mathcal{O} = \mathbb{T}^n_k \cdot \mathcal{O}\backslash\mathbb{T}^n_{<k} \cdot \mathcal{O}$.

۳. یکجمله‌ای $t \in \mathbb{T}^n$ توسط عضوی از $\partial\mathcal{O}$ بخش می‌شود اگر و تنها اگر $t \in \mathbb{T}^n\backslash\mathcal{O}$.



شکل ۲.۲: ایده‌ال ترتیبی $\mathcal{O}$ و مرز اول و دوم آن

برهان. ۱. با استقرا روی k ثابت می‌کنیم. به ازای $k=1$ طبق تعریف داریم، $\overline{\partial\mathcal{O}}=\mathcal{O}\cup\partial\mathcal{O}$ که $\mathcal{O}$ و $\partial\mathcal{O}$ مجزا نیز هستند. فرض کنید حکم به ازای k برقرار باشد. طبق تعریف داریم:

$$\overline{\partial^{k+1}\mathcal{O}}=\overline{\partial^{k}\mathcal{O}}\cup\partial^{k+1}\mathcal{O},$$

که در آن $\overline{\partial^{k}\mathcal{O}}$ و $\partial^{k+1}\mathcal{O}$ نیز مجزا هستند. طبق فرض استقرا داریم $\overline{\partial^{k}\mathcal{O}}=\bigcup_{i=0}^{k}\partial^{i}\mathcal{O}$ و در نتیجه $\overline{\partial^{k+1}\mathcal{O}}=\bigcup_{i=0}^{k+1}\partial^{i}\mathcal{O}$. قسمت دوم گزاره از این حقیقت نتیجه می‌شود که به ازای هر $t\in\mathbb{T}^n$ یک $k\geq 0$ وجود دارد به‌طوری که $t\in\partial^{k}\mathcal{O}$.

۲. بر اساس تعریف $\overline{\partial\mathcal{O}}$ می‌دانیم که $\overline{\partial\mathcal{O}}=\mathcal{O}\cup\mathbb{T}_1^n\cdot\mathcal{O}$. به‌راحتی می‌توان به‌صورت استقرایی ثابت کرد که،

$$\overline{\partial^{k+1}\mathcal{O}}=\overline{\partial^{k}\mathcal{O}}\cup\mathbb{T}_1^n\cdot\overline{\partial^{k}\mathcal{O}}=\overline{\partial^{k}\mathcal{O}}\cup\mathbb{T}_{k+1}^n\mathcal{O}.$$

از طرفی می‌دانیم که، $\partial^{k+1}\mathcal{O}=\overline{\partial^{k+1}\mathcal{O}}\backslash\overline{\partial^{k}\mathcal{O}}$. در نتیجه داریم،

$$\begin{aligned}\partial^{k}\mathcal{O}&=\overline{\partial^{k}\mathcal{O}}\backslash\partial^{k-1}\mathcal{O}=(\mathbb{T}_k^n\cdot\mathcal{O}\cup\overline{\partial^{k-1}\mathcal{O}})\backslash\overline{\partial^{k-1}\mathcal{O}}=\mathbb{T}_k^n\cdot\mathcal{O}\backslash\overline{\partial^{k-1}\mathcal{O}}\\&=\mathbb{T}_k^n\cdot\mathcal{O}\backslash(\mathbb{T}_{k-1}^n\cdot\mathcal{O}\cup\overline{\partial^{k-2}\mathcal{O}})=\cdots=\mathbb{T}_k^n\cdot\mathcal{O}\backslash\mathbb{T}_{<k}^n\cdot\mathcal{O}.\end{aligned}$$

۳. فرض کنید، t' عضوی از $\partial\mathcal{O}$ باشد که t را بخش می‌کند، به برهان خلف فرض کنید $t\in\mathcal{O}$ در این‌صورت چون $\mathcal{O}$ یک ایده‌ال ترتیبی است، باید $t'\in\mathcal{O}$ که با فرض $t'\in\partial\mathcal{O}$ در تناقض است.

□

طبق قسمت (۱) گزاره‌ی ۴۹.۲ چون $\mathbb{T}^n$ اجتماع مجزا از $\partial^{k}\mathcal{O}$ هاست، هر یکجمله‌ای در $\mathbb{T}^n$ دقیقاً در یکی از $\partial^{k}\mathcal{O}$ ها قرار می‌گیرد، در نتیجه می‌توانیم به‌صورت زیر کمیتی را ترعیف کنیم که در واقع فاصله‌ی یک یکجمله‌ای از یک ایده‌ال ترتیبی را بیان می‌کند.

تعریف ۵۰.۲ (اندیس یک یکجمله‌ای). به ازای هر یکجمله‌ای $t \in \mathbb{T}^n$ ، عدد یکتای $k \in \mathbb{N}$ به طوری که $t \in \partial^k \mathcal{O}$ ، را اندیس t نسبت به $\mathcal{O}$ نامیده و با نماد $\mathrm{ind}_{\mathcal{O}}(t)$ و اگر ابهامی نباشد با $\mathrm{ind}(t)$ نمایش می‌دهیم. این تعریف به‌صورت زیر برای هر چندجمله‌ای $f \in P \backslash \{0\}$، نیز قابل تعمیم است.

$$\mathrm{ind}_{\mathcal{O}}(f) := max\{\mathrm{ind}_{\mathcal{O}}(t) \mid t \in \mathrm{Supp}(f)\}.$$

در گزاره‌ی بعدی برخی از خصوصیات مفید اندیس را بیان می‌کنیم.

گزاره ۵۱.۲. فرض کنید $\mathcal{O} \subseteq \mathbb{T}^n$ یک ایده‌ال ترتیبی باشد.

۱. به ازای هر $t \in \mathbb{T}^n$ عدد $k = \mathrm{ind}_{\mathcal{O}}(t)$ کوچکترین عددی است که به ازای آن $t = t't''$ به‌طوری که $t' \in \mathbb{T}^n_k$ و $t'' \in \mathcal{O}$.

۲. به ازای هر دو یکجمله‌ای مثل $t, t' \in \mathbb{T}^n$ داریم، $\mathrm{ind}_{\mathcal{O}}(tt') \leq \deg(t) + \mathrm{ind}_{\mathcal{O}}(t')$.

۳. به ازای هر دو چندجمله‌ای ناصفر که $f+g \neq 0$ نابرابری، $\mathrm{ind}_{\mathcal{O}}(f+g) \leq \max\{\mathrm{ind}_{\mathcal{O}}(f), \mathrm{ind}_{\mathcal{O}}(g)\}$، برقرار است.

۴. به ازای هر دو چندجمله‌ای ناصفر $f, g \in P$ نامساوی زیر برقرار است:

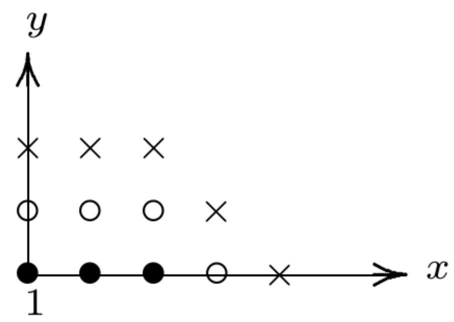
$$\mathrm{ind}_{\mathcal{O}}(f \cdot g) \leq \min\{\deg(f) + \mathrm{ind}_{\mathcal{O}}(g), \deg(g) + \mathrm{ind}_{\mathcal{O}}(f)\}.$$

برهان. گزاره‌ی (۱) نتیجه‌ی مستقیم قسمت (۲) گزاره‌ی ۴۹.۲ است. برای اثبات گزاره‌ی (۲) فرض کنید $\mathrm{ind}(t') = k'$، آن‌گاه می‌توان گفت، $t' = x^{\beta_1}x^{\beta_2}$ به‌طوری که $x^{\beta_2} \in \mathcal{O}$ و $x^{\beta_1} \in \mathbb{T}^n_{k'}$. همچنین فرض کنید $t = x^{\alpha}$ در این‌صورت داریم، $tt' = x^{\alpha+\beta_1}x^{\beta_2}$. فرض کنید $\mathrm{ind}(tt') = k$، با توجه به گزاره‌ی (۱) باید داشته باشیم، $k < \deg(\alpha) + \deg(\beta_1)$ ولی توجه کنید که $\deg(\beta_1) = k'$ و $\deg(\alpha) = \deg(t)$ در نتیجه گزاره‌ی (۲) نیز برقرار است.

گزاره‌ی (۳) نتیجه‌ی مستقیم این حقیقت است که $\mathrm{Supp}(f+g) \subseteq \mathrm{Supp}(f) \cup \mathrm{Supp}(g)$. گزاره‌ی (۴) نیز از گزاره‌ی (۲) و با استفاده از رابطه‌ی $\mathrm{Supp}(fg) \subseteq \{t't'' \mid t' \in \mathrm{Supp}(f), t'' \in \mathrm{Supp}(g)\}$، به‌راحتی نتیجه می‌شود. □

مثال زیر نشان می‌دهد که $\mathcal{O}$ ـ اندیس با ضرب یکجمله‌ای‌ها سازگار نیست یعنی نمی‌توان در حالت کلی از برقراری $\mathrm{ind}_{\mathcal{O}}(t) \leq \mathrm{ind}_{\mathcal{O}}(t')$ رابطه‌ی $\mathrm{ind}_{mO}(tt'') \leq \mathrm{ind}_{mO}(t't'')$ را نتیجه گرفت و بنابراین، نمی‌توان آن را به عنوان یک ترتیب روی یکجمله‌ای‌ها در نظر گرفت.

مثال ۵۲.۲. فرض کنید $\mathcal{O} = \{1, x, x^2\} \subseteq \mathbb{T}^2$. در این‌صورت $\mathcal{O}$ یک ایده‌ال ترتیبی با مرزی برابر با مجموعه‌ی $\partial\mathcal{O} = \{y, xy, x^2y, x^3\}$ است. نمودار ۳.۲ مرز اول و دوم این ایده‌ال ترتیبی را نمایش داده است.



شکل ۳.۲: مرز اول و دوم ایده‌ال ترتیبی $\{۱, x, x^۲\}$

$\operatorname{ind}(x^۲) = ۰$ و $\operatorname{ind}(y) = ۱$ در نتیجه $\operatorname{ind}(x^۲) < \operatorname{ind}(y)$ با ضرب $x^۲$ در طرفین بر خلاف آن‌چه از یک ترتیب انتظار داریم، خواهیم داشت، $\operatorname{ind}(x^۲ \cdot x^۲) > \operatorname{ind}(x^۲ \cdot y^۲)$. بطور مشابه باوجود این‌که $\operatorname{ind}(y) = \operatorname{ind}(x^۲y)$ ولی بعد از ضرب طرفین در x داریم، $\operatorname{ind}(x \cdot y) < \operatorname{ind}(x \cdot x^۲y)$.

چون موضوع بحث ما ایده‌ال‌های صفر بعدی هستند، لذا فرض می‌کنیم ایده‌ال‌های ترتیب و در نتیجه مرز آن‌ها متناهی هستند. البته دلیل این فرض را در ادامه خواهیم دید. بنابراین ایده‌ال ترتیبی را معمولاً به‌صورت $\mathcal{O} = \{t_۱, ..., t_\mu\}$ و مرز آن را به‌صورت $\partial\mathcal{O} = \{b_۱, ..., b_\nu\}$ نمایش می‌دهیم.

تعریف ۵۳.۲ (پیش‌پایه‌ی مرزی). فرض کنید $\mathcal{O} = \{t_۱, ..., t_\mu\}$ یک ایده‌ال ترتیبی و $\partial\mathcal{O} = \{b_۱, ..., b_\nu\}$ مرز آن باشد. مجموعه‌ی چندجمله‌ای‌های $G = \{g_۱, ..., g_\nu\}$ را $\mathcal{O}$- *پیش‌پایه‌ی مرزی* گوییم هرگاه، $\alpha_{ij} \in K$ به ازای $۱ \leq i \leq \mu$ و $۱ \leq j \leq \nu$ موجود باشد به‌طوری که:

$$g_j = b_j - \sum_{i=۱}^{\mu} \alpha_{ij} t_i, \ ۱ \leq j \leq \nu.$$

اکنون الگوریتمی مشابه به الگوریتم تقسیم در بحث پایه‌ی گروبنر را معرفی می‌کنیم. همان طور که الگوریتم تقسیم، در محاسبه‌ی باقی‌مانده‌ی S- چندجمله‌ای‌ها برای یافتن پایه‌ی گروبنر بکار می‌رفت، این الگوریتم نیز نقشی اساسی در آن‌چه در ادامه مطرح می‌کنیم خواهد داشت.

گزاره ۵۴.۲ (الگوریتم تقسیم مرزی). فرض کنید $\mathcal{O} = \{t_۱, ..., t_\mu\} \subseteq \mathbb{T}^n$ یک ایده‌ال ترتیبی و $\partial\mathcal{O} = \{b_۱, ..., b_\nu\}$ مرز آن باشد. فرض کنید $\{g_۱, ..., g_\nu\}$ یک $\mathcal{O}$- پیش‌پایه و $f \in P$ یک چندجمله‌ای دلخواه باشد. در این‌صورت خروجی الگوریتم ۳ با دریافت ورودی‌های $f, \mathcal{O}, \partial\mathcal{O}, G$ پس از متناهی مرحله اجرا، چندتایی مرتب $(f_۱, ..., f_\nu, c_۱, ..., c_\mu) \in P^\nu \times K^\mu$ خواهد بود به‌طوری که :

$$f = f_۱g_۱ + \cdots + f_\nu + c_۱t_۱ + \cdots + c_\mu t_\mu$$

و به ازای هر $i \in \{۱, ..., \nu\}$ اگر $f_ig_i \neq ۰$ آن‌گاه، $\deg(f_i) \leq \operatorname{ind}_{\mathcal{O}}(f) - ۱$. در ضمن نمایش فوق مستقل از انتخاب $h_۱$ در مرحله‌ی $(\star)$ است.

الگوریتم ۳ الگوریتم تقسیم مرزی

Input: $f, \mathcal{O}, \partial\mathcal{O}, \{g_1, ..., g_\nu\}$
Output: $(f_1, ..., f_\nu, c_1, ..., c_\mu)$
$(f_1, ..., f_\nu, c_1, ..., c_\mu) \leftarrow (0, ..., 0, 0, ..., 0)$
$h \leftarrow f$
while $h \neq 0$ **do**
  if $\mathrm{ind}_{\mathcal{O}}(h) = 0$ **then**
    $(c_1, ..., c_\mu) \leftarrow \{c_i|\ h = c_1t_1 + \cdots + c_\mu t_\mu, c_i \in K, t_i \in \mathcal{O}\}$
    print
  else if $\mathrm{ind}_{\mathcal{O}}(h) > 0$ **then**
    $\{a_1h_1, ..., a_sh_s\} \leftarrow \{a_ih_i|\ h = a_1h_1 + \cdots + a_sh_s, a_i \in K, h_i \in \mathbb{T}^n, \mathrm{ind}_{\mathcal{O}}(h_1) = \mathrm{ind}_{\mathcal{O}}(h)\}(\star)$
    $i \leftarrow \min\{1 \leq i \leq \nu|\ \exists t' \in \mathbb{T}^n, \deg(t') = \mathrm{ind}_{\mathcal{O}}(h) - 1, h_1 = t'b_i\}$
    $h \leftarrow h - a_1t'g_i$
    $f_i \leftarrow f_i + a_1t'$
  end if
end while
return $(f_1, ..., f_\nu, c_1, ..., c_\mu)$

برهان. ابتدا نشان می‌دهیم، الگوریتم ۳ قابل اجرا است. در حالتی که $\mathrm{ind}_{\mathcal{O}}(h) = ۰$ است باید داشته باشیم، $\mathrm{Supp}(h) \subseteq \mathcal{O}$، در نتیجه h نمایشی به‌صورت $h = c_۱t_۱ + \cdots + c_\mu t_\mu$ دارد، که ازای هر $۱ \leq i \leq \mu$، داریم $t_i \in \mathcal{O}$. در حالتی که $\mathrm{ind}_{\mathcal{O}}(h) > ۰$ طبق تعریف $\mathrm{ind}_{\mathcal{O}}(h) := max\{\mathrm{ind}_{\mathcal{O}}(h_i) \mid h_i \in \mathrm{Supp}(h)\}$. در نتیجه به ازای یکی از یکجمله‌ای‌های h که ما آن را $h_۱$ نام می‌گذاریم، داریم $k = \mathrm{ind}_{\mathcal{O}}(h) = \mathrm{ind}_{\mathcal{O}}(h_۱)$. طبق قسمت (۱) گزاره‌ی ۵۱.۲، $h_۱$ تجزیه‌ی به‌صورت $h_۱ = \tilde{t}t_i$ که $\tilde{t} \in \mathbb{T}^n_k$ و $t_i \in \mathcal{O}$، دارد، ضمن اینکه دیگر هیچ نمایشی به این صورت که در آن درجه‌ی $\tilde{t}$ کوچکتر از درجه‌ی $\tilde{t}$ فعلی باشد وجود ندارد. چون $k > ۰$ ، $t' \in \mathbb{T}^n$ وجود دارد به‌طوری که به ازای یک $j \in \{۱, ..., n\}$ داریم، $\tilde{t} = t'x_j$، که نتیجه می‌دهد، $\deg(t') = k - ۱$. از طرفی می‌دانیم که $\tilde{t}$ طوری انتخاب شده که کوچکترین درجه‌ی ممکن را دارد که نتیجه می‌دهد، $x_jt_i \in \partial\mathcal{O}$. بنابراین $b_k \in \partial\mathcal{O}$ وجود دارد به‌طوری که، $h_۱ = t'(x_jti) = t'b_k$.

اکنون متناهی بودن مراحل اجرای الگوریتم را ثابت می‌کنیم. نشان می‌دهیم که شرط $\mathrm{ind}_{\mathcal{O}} > ۰$ فقط به تعداد متناهی بار می‌تواند رخ دهد و لذا پس از متناهی مرحله یا $\mathrm{ind}_{\mathcal{O}}(h) = ۰$ و یا $h = ۰$.
در حلقه‌ی اصلی هر بار که شرط $\mathrm{ind}_{\mathcal{O}}(h) > ۰$ برقرار شود h با $h - a_۱t'g_i$ به ازای یک $i \in \{۱, ..., n\}$ جایگزین می‌شود. g_i یک عنصر از پیش‌پایه است و طبق تعریف پیش‌پایه، $g_i = b_i - \sum_{k=۱}^{\mu} \alpha_{ki}t_k$ به‌طوری که $\alpha_{ki} \in K$. بنابراین داریم:

$$h - a_۱t'g_i = a_۱h_۱ + \cdots + a_sh_s - a_۱t'b_i + a_۱t' \sum_{k=۱}^{\mu} \alpha_{ki}t_k.$$

ولی با توجه به نحوه‌ی انتخاب $h_۱$ می‌دانیم که، $a_۱h_۱ = a_۱t'b_i$، در نتیجه $a_۱h_۱$ که اندیسش برابر

با $\mathrm{ind}_{\mathcal{O}}(h)$ است از h حذف می‌شود و یک جمله به شکل $t't_l \in \overline{\partial^{k-1}\mathcal{O}}$ جایگزین آن می‌شود که طبق قسمت (۲) گزاره‌ی ۵۱.۲ نتیجه می‌گیریم، $\mathrm{ind}_{\mathcal{O}} \leq k-1$، که اکیداً از اندیس قبلی h کمتر است. بدیهی است که اگر h بیش از یک جمله با $\mathrm{ind}_{\mathcal{O}}(h)$ داشته باشد، این‌جملات در مراحل متولی حذف می‌شوند. در نتیجه پس از متناهی بار اجرای الگوریتم یا $\mathrm{ind}_{\mathcal{O}}(h)$ نزول کرده و به صفر می‌رسد یا بطور کلی h برابر با صفر می‌شود که در هر دو حالت، الگوریتم خاتمه می‌یابد.

اکنون درستی خروجی الگوریتم و این‌که در شرایط ذکر شده صدق می‌کند را نشان می‌دهیم. نشان می‌دهیم شکل کلّی رابطه‌ی

$$f = h + f_1g_1 + \cdots + f_\nu g_\nu + c_1 + \cdots + c_\mu t_\mu. \tag{۱.۲}$$

در هر مرحله از اجرای الگوریتم برقرار است یا اصطلاحاً این رابطه تحت اجرای الگوریتم ناورداست. این رابطه در ابتدا و قبل از اجرای حلقه‌ی اصلی، که انتساب $f \to h$ صورت می‌گیرد و همه‌ی f_i و c_i ها صفر هستند بوضوح برقرار است. نشان می‌دهیم اگر رابطه‌ی فوق در یک مرحله از اجرا برقرار باشد در مرحله‌ی بعدی هم برقرار است و لذا، طبق استقرا حکم ثابت می‌شود. فرض کنید شرط $\mathrm{ind}_{\mathcal{O}}(h) > 0$ برقرار باشد در این‌صورت رابطه فوق وقتی دست‌خوش تغییر می‌شود که انتساب‌های $h \leftarrow h - a_1t'g_i$ و $f_i \leftarrow f_i + a_1t'$ در پایان حلقه‌ی اصلی رخ دهند. فرض کنید مقادیر h و f_i جدید به ازای i مورد نظر را با $\tilde{h}$ و $\tilde{f_i}$ نمایش دهیم، فرض کنید رابطه در مرحله‌ی قبل برقرار بوده، در نتیجه با جایگذاری مقادیر مرحله‌ی فعلی داریم:

$$\begin{aligned} f &= \tilde{h} + a_1t'g_i + f_1g_1 + \cdots + (\tilde{f_i} - a_1t')g_i + \cdots + f_\nu g_\nu + c_1t_1 + \cdots + c_\mu t_\mu \\ &= \tilde{h} + f_1g_1 + \cdots + \tilde{f}_ig_i + \cdots + f_\nu g_\nu + c_1 + \cdots + c_\mu t_\mu \end{aligned}$$

در نتیجه شکل کلّی معادله‌ی ۱.۲، در این حالت ناوردا است و تنها h و f_i با مقدارهای جدیدشان جایگزین شده‌اند. حالت دیگر این است که، $\mathrm{ind}_{\mathcal{O}}(h) = 0$. در این صورت c_i ها با مقادیر جدید طوری جایگزین می‌شوند که داشته باشیم، $h = c_1t_1 + \cdots + c_\mu t_\mu$ که در این‌حالت هم شکل کلی رابطه‌ی ۱.۲ برقرار می‌ماند و فقط مقادیر c_i در آن عوض می شوند. وقتی الگوریتم متوقف شود داریم $h = 0$ و لذا خروجی به شکل زیر است:

$$f = f_1g_1 + \cdots + f_\nu g_\nu + c_1t_1 + \cdots + c_\mu t_\mu.$$

در بندهای قبل نشان دادیم که در هر مرحله، وقتی انتساب $f_i \leftarrow f_i + a_1t'$ صورت می‌گیرد، t' در شرط $\deg(t') = \mathrm{ind}_{\mathcal{O}}(h) - 1$ صدق می‌کند، از طرفی $\mathrm{ind}_{\mathcal{O}}(h)$ در ابتدا برابر $\mathrm{ind}_{\mathcal{O}}(f)$ است و در مرحله نزول می‌کند، بنابراین به ازای هر $1 \leq i \leq \nu$ اگر $f_i \neq 0$، رابطه‌ی $\deg(f_i) \leq \mathrm{ind}_{\mathcal{O}}(f) - 1$ نیز برقرار است.

تنها ادعای باقی‌مانده که باید ثابت کنیم این است که خروجی مستقل از نحوه‌ی انتخاب h_1

است. فرض کنید در حالتی که $\mathrm{ind}_{\mathcal{O}}(h) > ۰$ است چند h_i در رابطه‌ی $\mathrm{ind}_{\mathcal{O}}(h) = \mathrm{ind}_{\mathcal{O}}(h_i)$ صدق کنند. در این‌صورت فرقی ندارد کدام را به عنوان $h_۱$ انتخاب کنیم، چون در نهایت $h_۱$ با جمله‌ای جایگزین می‌شود که اندیسش اکیداً کمتر از $\mathrm{ind}_{\mathcal{O}}(h_۱)$ است، و تداخلی در حذف سایر h_i ها در مراحل بعدی ایجاد نمی‌کند. به این ترتیب نتیجه‌ی نهایی، بعد از این‌که همه‌ی جملات بازنویسی می‌شوند، مستقل از ترتیب انتخاب کاندیدهای $h_۱$ است. □

نکته ۵۵.۲. نمایش به‌دست آمده از خروجی الگوریتم تقسیم مرزی ۳ از این لحاظ که به ازای هر $t \in \mathrm{Supp}(f_i)$ داریم، $\mathrm{ind}_{\mathcal{O}}(tb_i) = \deg(t) + ۱$ و $\mathrm{ind}_{\mathcal{O}}(t(g_i - b_i)) \leq \deg(t)$، بهینه عمل می‌کند.

مثال ۵۶.۲. فرض کنید $\mathcal{O} = \{t_۱, t_۲, t_۳\} \subseteq \mathbb{T}^۲$ یک ایده‌ال ترتیبی باشد به‌طوری که، $t_۱ = ۱, t_۲ = x, t_۳ = y$. مرز این ایده‌ال ترتیبی عبارت است از، $\partial\mathcal{O} = \{b_۱, b_۲, b_۳\}$ به‌طوری که، $b_۱ = x^۲, b_۲ = xy, b_۳ = y^۲$. مجموعه‌ی $G = \{g_۱ = x^۲ + x + ۱, g_۲ = xy + y, g_۳ = y^۲ + x + ۱\}$ نیز یک $\mathcal{O}$ ـ پیش‌پایه‌ی مرزی است. فرض کنید $f = x^۳y^۲ - xy^۲ + x^۲ + ۲$، در زیر مراحل تقسیم f بر G توسط الگوریتم تقسیم مرزی ۳، نشان داده شده است.

برای سادگی در محاسبه‌ی اندیس مرزهای دوم تا چهارم ایده‌ال ترتیبی $\mathcal{O}$ در زیر نشان داده شده.

$$\partial^۲\mathcal{O} = \{x^۳, x^۲y, xy^۲, y^۳\},\ \partial^۳\mathcal{O} = \{x^۴, x^۳y, x^۲y^۲, xy^۳, y^۴\},\ \partial^۴\mathcal{O} = \{x^۵, x^۴y, x^۳y^۲, x^۲y^۳, xy^۴, y^۵\}.$$

گام‌های الگوریتم تقسیم مرزی را مطابق مراحل زیر طی می‌کنیم.

۱. قرار می‌دهیم، $f_۱ = f_۲ = f_۳ = ۰, c_۱ = c_۲ = c_۳ = ۰$ و $h = x^۳y^۲ - xy^۲ + x^۲ + ۲$. همچنین داریم:

$$\mathrm{Supp}(f) = \{h_۱ = x^۳y^۲, h_۲ = xy^۲, h_۳ = x^۲, h_۴ = ۱\}$$
$$\mathrm{ind}_{\mathcal{O}}(h_۱) = ۴, \mathrm{ind}_{\mathcal{O}}(h_۲) = ۲, \mathrm{ind}_{\mathcal{O}}(h_۳) = ۱, \mathrm{ind}_{\mathcal{O}}(h_۴) = ۰$$

۲. $\mathrm{ind}_{\mathcal{O}}(h) > ۰$ و داریم $x^۳y^۲ = xy^۲ \cdot b_۱$ به‌طوری که، $\deg(xy^۲) = \mathrm{ind}(h) - ۱$. بنابراین قرار می‌دهیم $f_۱ = xy^۲$ و $h = x^۳y^۲ - xy^۲ + x^۲ + ۲ - xy^۲(x^۲ + x + ۱)$. یکجمله‌ای‌های $h = -x^۲y^۲ - ۲xy^۲ + x^۲ + ۲$ به ترتیب دارای اندیس‌های $۳, ۲, ۱, ۰$ هستند.

۳. همچنان شرط $\mathrm{ind}(h) > ۰$ برقرار است و داریم، $x^۲y^۲ = y^۲b_۱$ به‌طوری که، $\deg(y^۲) = \mathrm{ind}(h) - ۱$. بنابراین $-y^۲$ را به $f_۱$ اضافه می‌کنیم که $f_۱ = xy^۲ - y^۲$ به‌دست می‌آید، برای h هم داریم،

$$h = -x^۲y^۲ - ۲xy^۲ + x^۲ + ۲ + y^۲(x^۲ + x + ۱).$$

یکجمله‌ای‌های $h = -xy^۲ + x^۲ + y^۲ + ۲$ به‌ترتیب دارای $\mathcal{O}$ ـ اندیس‌های $۲, ۱, ۱, ۰$ هستند.

۴. داریم، $xy^۲ = y \cdot b_۲$ به‌طوری که $\deg(y) = \mathrm{ind}(h) - ۱$. قرار می‌دهیم،

$$h = -xy^۲ + x^۲ + y^۲ + ۲ + y(xy + y), f_۲ = -y.$$

یکجمله‌ای‌های $h = x^۲ + ۲y^۲ + ۲$ به ترتیب دارای $\mathcal{O}$- اندیس‌های $۱, ۱, ۰$ هستند.

۵. در این مرحله داریم $x^۲ = ۱ \cdot b_۱$ به‌طوری که، $\deg(۱) = \mathrm{ind}(h) - ۱$. با افزودن ۱ به f و کاستن $۱ \cdot g_۱$ از h به‌دست می‌آید، $f_۱ = xy^۲ - y^۲ + ۱$ و $h = x^۲ + ۲y^۲ + ۲ - ۱(x^۲ + x + ۱) = ۲y^۲ - x + ۱$. یکجمله‌ای‌های h به ترتیب دارای $\mathcal{O}$ ــ اندیس $۱, ۰, ۰$ هستند.

۶. داریم $y^۲ = ۱ \cdot b_۳$ به‌طوری‌که $\deg(۱) = \mathrm{ind}(h) - ۱$. با افزودن ۲ به $f_۳$ و کاستن $۲ \cdot g_۳$ از h به‌دست می‌آید، $f_۳ = ۲$ و $h = ۲y^۲ - x + ۱ - ۲(y^۲ + x + ۱) = -۳x - ۱$. در نتیجه $\mathcal{O}$ ــ اندیس یکجمله‌ای‌های h به ترتیب برابر است با $۰, ۰$.

۷. در این مرحله $\mathrm{ind}_{\mathcal{O}}(h) = ۰$ ، و داریم $h = -۱ \cdot t_۱ - ۳ \cdot t_۲ + ۰ \cdot t_۳$. بنابراین الگوریتم متوقف شده و چندتایی $(xy^۲ - y^۲ + ۱, ۲, ۱, -۳, ۰)$ را به‌عنوان خروجی تولید می‌کند.

به این‌ترتیب نمایشی به‌صورت زیر برای f به‌دست آمد

$$f = (xy^۲ - y^۲ + ۱)g_۱ - yg_۲ + ۲g_۳ - ۱t_۱ - ۳t_۲ + ۰t_۳.$$

اگر ترتیب g_i ها در ورودی به‌صورت $\mathcal{G}' = (g'_۱, g'_۲, g'_۳) = (g_۳, g_۲, g_۱)$ باشد، آن‌گاه خروجی زیر حاصل می‌شود

$$\begin{aligned} f =&(x^۳ + x)g'_۱ - ۱g'_۲ + (x^۲ + ۲)g'_۳ + ۱t_۱ - ۳t_۲ - ۱t_۳ \\ =&(x^۲ + ۲)g_۱ - ۱g_۲ + (x^۳ + x)g_۳ + ۰t_۱ + ۱t_۲ - ۱t_۳. \end{aligned}$$

که نشان می‌دهد با جایگشت روی مقسوم‌علیه‌ها خروجی تغییر خواهد کرد.

تعریف ۵۷.۲ ($\mathcal{O}$ **ــ مانده نرمال یک چندجمله‌ای).** فرض کنید $\mathcal{O} = \{t_۱, ..., t_\mu\}$ یک ایده‌ال ترتیبی و $G = \{g_۱, ..., g_\nu\}$ یک پیش‌پایه نسبت به $\mathcal{O}$ باشد. در این‌صورت فرم نرمال چندجمله‌ای دلخواه f نسبت به $\mathcal{G} = (g_۱, ..., g_\nu)$ عبارت است از

$$\mathrm{NR}_{\mathcal{O},\mathcal{G}} = c_۱t_۱ + \cdots + c_\mu t_\mu$$

که عبارت فوق همان بخش دوم، خروجی الگوریتم تقسیم مرزی در تقسیم f بر $(g_۱, ..., g_\nu)$ است.

نکته ۵۸.۲. به زبان حلقه‌ها f و $\mathrm{NR}_{\mathcal{O},\mathcal{G}}(f)$ هر دو متعلق به یک کلاس مانده‌ها در حلقه‌ی خارج قسمتی $\frac{P}{\langle g_۱, ..., g_\nu \rangle}$ هستند. در ضمن اگر $G = \{g_۱, ..., g_\nu\}$ یک $\mathcal{O}$ پیش‌پایه باشد، مجموعه‌ی

کلاس‌های مانده $\mathcal{O}$ همواره یک مولد برای K ـ فضای برداری $\frac{P}{\langle g_1,\ldots,g_\nu\rangle}$ است. زیرا به ازای هر $\bar{f}\in\frac{P}{\langle g_1,\ldots,g_\nu\rangle}$، طبق الگوریتم تقسیم مرزی ۳ داریم:

$$f = f_1g_1+\cdots+f_\nu g_\nu + c_1t_1+\cdots+c_\mu t_\mu \Rightarrow \bar{f} = c_1\bar{t}_1+\cdots+c_\mu\bar{t}_\mu.$$

با این وجود نمی‌توان گفت $\mathcal{O}$ همواره یک پایه‌ی این فضای برداری است. برای نمونه در مثال ۵۶.۲ داریم، $\mathrm{NR}_{\mathcal{O},\mathcal{G}'}(f)-\mathrm{NR}_{\mathcal{O},\mathcal{G}}(f) = 4x-y+1\in\langle g_1,\ldots,g_\nu\rangle$ که نشان‌دهنده‌ی وابستگی خطی اعضای $\mathcal{O}$ است. در ادامه توجه خود را به ایده‌ال‌های ترتیبی معطوف می‌کنیم که پایه‌ای برای فضای برداری $\frac{P}{\langle g_1,\ldots,g_\nu\rangle}$ هستند.

در قضیه‌های بعدی فرض می‌کنیم $\mathcal{O}$ یک ایده‌ال ترتیبی به‌صورت $\mathcal{O}=\{t_1,\ldots,t_\mu\}$، و $\partial\mathcal{O}=\{b_1,\ldots,b_\nu\}$ مرز آن باشد و از ذکر مجدد این موضوع خودداری می‌کنیم.

تعریف ۵۹.۲ **($\mathcal{O}$– پایه‌ی مرزی).** فرض کنید $G=\{g_1,\ldots,g_\nu\}$ یک $\mathcal{O}$– پیش‌پایه باشد و قرار دهید $\mathcal{G}=(g_1,\ldots,g_\nu)\in P^\nu$. در ضمن فرض کنید K– فضای خطی (برداری) تولید شده توسط $\mathcal{O}$ را با $\langle\mathcal{O}\rangle_K$ نمایش دهیم. ایده‌ال $I\subseteq P$ که شامل G است را در نظر بگیرید. مجموعه‌ی G یا ν تایی مرتب $\mathcal{G}$ را یک $\mathcal{O}$– *پایه‌ی مرزی* ایده‌ال I گوییم هر گاه یکی از شرط‌های معادل زیر برقرار باشد.

۱. کلاس‌های مانده $\overline{O}=\{\bar{t}_1,\ldots,\bar{t}_\mu\}$ یک پایه‌ی برای K– فضای برداری $\frac{P}{I}$ باشد.

۲. $I\cap\langle\mathcal{O}\rangle_K=\{0\}$.

۳. P به عنوان K ـ فضای برداری، جمع مستقیم $\langle\mathcal{O}\rangle_K$ و K ـ فضای برداری I باشد. $P = I\oplus\langle\mathcal{O}\rangle_K$.

اولین سؤال‌ها بعد از تعریف یک شئ ریاضی، وجود و یکتایی آن شئ است. در فصل قبل مشاهده کردیم هر ایده‌ال نسبت به هر ترتیب یکجمله‌ای دارای پایه‌ی گروبنر است، آیا هر ایده‌ال نسبت به هر ایده‌ال ترتیبی دارای پایه‌ی مرزی است؟ آیا پایه‌ی مرزی هم همچون پایه‌ی گروبنر، مولدی برای ایده‌ال خواهد بود؟ همچنین دانستیم که پایه‌ی گروبنر تحویل‌یافته‌ی هر ایده‌ال یکتا است، آیا پایه‌ی مرزی ایده‌ال در صورت وجود یکتا است؟ این‌ها سؤالاتی است که پاسخ آن‌ها را در قضیه‌ی های بعدی خواهیم داد.
می‌دانیم پایه‌ی گروبنر هر ایده‌ال مولید برای آن ایده‌ال است قضیه‌ی بعد نشان می‌دهد پایه‌ی مرزی هم چنین است.

قضیه ۶۰.۲. فرض کنید G یک $\mathcal{O}$ پایه‌ی مرزی ایده‌ال چندجمله‌ای $I\subseteq P$ باشد. در این‌صورت G مولدی برای ایده‌ال I است.

برهان. فرض کنید $G = \{g_۱, ..., g_\nu\}$ در این‌صورت طبق تعریف می‌دانیم که $\langle g \rangle \subseteq I$. برای اثبات شمول در جهت عکس، $f \in I$ را دلخواه در نظر بگیرید. با استفاده از الگوریتم تقسیم مرزی ۳ داریم:

$$f = f_۱g_۱ + \cdots + f_\nu g_\nu + c_۱t_۱ + \cdots + c_\mu t_\mu,$$

به‌طوری که، $f_۱, ..., f_\nu \in P$ و $c_۱, ..., c\mu \in K$. در نتیجه کلاس مانده‌ی f در $\frac{P}{I}$ عبارت است از، $\bar{f} = c_۱\bar{t}_۱ + \cdots + c_\mu \bar{t}_\mu = ۰$. از طرفی فرض کردیم که G یک $\mathcal{O}$ پایه‌ی مرزی است، در نتیجه کلاس‌های مانده‌ی $\bar{t}_۱, ..., \bar{t}_\mu$ باید K مستقل خطی باشند. بنابراین $c_۱ = \cdots = c_\mu = ۰$ و $f = f_۱g_۱ + \cdots + f_\nu g_\nu \in \langle G \rangle$. □

روشن است که شرط لازم برای این‌که ایده‌ال I ، نسبت به ایده‌ال ترتیبی $\mathcal{O}$ دارای پایه‌ی مرزی باشد این است که $|\mathcal{O}| = \dim_K(\frac{P}{I})$. ولی مثال زیر نشان می‌دهد که این شرط کافی نیست.

مثال ۶۱.۲. فرض کنید $P = \mathbb{Q}[x, y]$ و $\mathbb{X} = \{(۰, ۰), (۰, -۱), (۱, ۰), (۱, ۱), (-۱, ۱)\} \subseteq \mathbb{Q}^۲$. ایده‌ال $I = I(\mathbb{X})$ را در نظر بگیرید، می‌دانیم که $\dim_K(\frac{P}{I}) = ۵$. ایده‌ال ترتیبی $\mathcal{O} = \{۱, x, x^۲, x^۳, x^۴\}$ گرچه ۵ عضو دارد ولی $x^۳ - x \in I$ و لذا اعضایش مستقل خطی نیستند. ایده‌ال ترتیبی $\mathcal{O} = \{۱, y, y^۲, y^۳, y^۴\}$ نیز به علت این‌که $y^۳ - y \in I$ اعضایش مستقل خطی نیستند. بنابراین، چنین نیست که یک ایده‌ال چندجمله‌ای نسبت به هر ایده‌ال ترتیبی دارای پایه‌ی مرزی باشد.

فرض کنید یک ایده‌ال صفر بعدی دلخواه مثل $I \subseteq P$ داده شده است، این سؤال پیش می‌آید که اگر یک ایده‌ال ترتیبی بیابیم که کلاس‌های مانده آن در حلقه‌ی خارج قسمتی $\frac{P}{I}$ پایه‌ای برای فضای برداری $\frac{P}{I}$ باشد، آیا وجود $\mathcal{O}$- پایه‌ی مرزی تضمین خواهد شد؟ قضیه‌ی زیر به این سؤال پاسخ مثبت می‌دهد.

قضیه ۶۲.۲ (وجود و یکتایی پایه‌ی مرزی). فرض کنید $I \subseteq P$ یک ایده‌ال صفر بعدی باشد و مجموعه کلاس‌های مانده اعضای $\mathcal{O}$ تشکیل یک پایه برای K ـ فضای برداری $\frac{P}{I}$ دهد. در این‌صورت گزاره‌های زیر برقراراند.

۱. I یک $\mathcal{O}$ ـ پایه‌ی مرزی یکتا خواهد داشت.

۲. اگر G یک $\mathcal{O}$- پیش‌پایه‌ی مرزی و مشمول در I باشد، آن‌گاه G، $\mathcal{O}$- پایه‌ی مرزی I نیز هست.

برهان. ۱. فرض کنید $\mathcal{O} = \{t_۱, ..., t_\mu\}$ و $\partial\mathcal{O} = \{b_۱, ..., b_\nu\}$. چون $\overline{\mathcal{O}}$ یک پایه‌ی K- فضای برداری $\frac{P}{I}$ است، بنابراین کلاس مانده‌ی هر $b_i \in \partial\mathcal{O}$ را می‌توان به‌صورت ترکیب خطی از اعضای $\overline{\mathcal{O}}$ و به‌صورت زیر نوشت.

$$\bar{b_i} = \sum_{j=۱}^{\mu} \alpha_{ij}\bar{t_j}$$

که $\alpha_{ij} \in K$. در نتیجه به ازای هر $1 \le i \le \nu$ چندجمله‌ای $g_i \in I$ وجود دارد به‌طوری که داریم، $g_i = b_i - \sum_{j=1}^{\mu} \alpha_{ij} t_j$. به این ترتیب $G = \{g_1, ..., g_\nu\}$ یک $\mathcal{O}$- پیش‌پایه‌ی مرزی I و با در نظر گرفتن فرض‌های قضیه یک $\mathcal{O}$ پایه‌ی مرزی I است.

برای اثبات یکتایی با برهان خلف، فرض کنید $G' = \{g'_1, ..., g'_\nu\}$ یک $\mathcal{O}$ پایه‌ی مرزی دیگر I باشد. فرض کنید یک $i \in \{1, ..., \nu\}$ وجود داشته باشد به‌طوری که، $g'_i = b_i - \sum_{j=1}^{\mu} \alpha'_{ij} t_j$ و به ازای یک $j \in \{1, ..., \mu\}$ داشته باشیم، $\alpha'_{ij} \neq \alpha_{ij}$. در این صورت $g_i - g'_i$ یک چندجمله‌ای ناصفر در I است، به‌طوری که $\mathrm{Supp}(g_i - g'_i) \subseteq \mathcal{O}$. ولی این با فرض پایه بودن و در نتیجه مستقل خطی بودن کلاس‌های مانده $\mathcal{O}$ در $\frac{P}{I}$ در تناقض است.

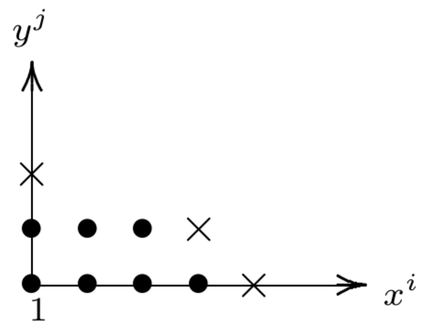
۲. نتیجه‌ی مستقیم تعریف پایه‌ی مرزی و فرض قضیه است.

□

قضیه‌ی ۶۲.۲ این اطمینان را می‌دهد که پایه‌ی مرزی در صورت وجود یکتا خواهد بود، اما سؤال مهم‌تر این‌است که آیا هر ایده‌ال پایه‌ی مرزی دارد؟ یا به عبارت دیگر آیا به ازای هر ایده‌ال صفر بعدی می‌توان یک ایده‌ال ترتیبی مثل $\mathcal{O}$ یافت که در شرایط قضیه‌ی ۶۲.۲ صدق کند و در نتیجه وجود یک $\mathcal{O}$ پایه‌ی مرزی تضمین شود؟پاسخ این سؤال که مثبت است، در بخش بعد ضمن بررسی رابطه‌ی پایه‌ی گروبنر و پایه‌ی مرزی داده می‌شود.

رابطه‌ی پایه‌ی گروبنر و پایه‌ی مرزی

فرض کنید $\mathcal{O}$ یک ایده‌ال ترتیبی باشد، در این صورت $\mathbb{T}^n \backslash \mathcal{O}$ مجموعه‌ی شامل همه‌ی یکجمله‌ای‌های یک ایده‌ال یکجمله‌ای است. طبق گزاره‌ی ۱۹.۲ می‌دانیم که هر ایده‌ال یکجمله‌ای، دارای مولّد مینیمال یکتا است. اعضای این مولد مینیمال را ***گوشه‌های*** $\mathcal{O}$ می‌نامیم. در شکل ۴.۲ یک ایده‌ال ترتیبی به همراه گوشه‌هایش نمایش داده شده، همان طور که در شکل هم مشاهده می‌شود گوشه نام با مُسمّایی برای چنین اعضایی است.



شکل ۴.۲: گوشه‌های یک ایده‌ال ترتیبی

تعریف ۶۳.۲. فرض کنید σ یک ترتیب یکجمله‌ای روی $\mathbb{T}^n$ و $I \subseteq P$ یک ایده‌ال باشد. در این‌صورت *ایده‌ال ترتیبی متناظر با* I را به‌صورت زیر تعریف می‌کنیم.

$$\mathcal{O}_\sigma(I) = \mathbb{T}^n \setminus \mathrm{LM}_\sigma(I) = \mathbb{T}^n \setminus \{\mathrm{LM}_\sigma(f) |\ f \in I\}$$

به‌راحتی می‌توان دید که $\mathcal{O}_\sigma(I)$ یک ایده‌ال ترتیبی است.

قضیه ۶۴.۲ (قضیه‌ی پایه‌ی مکالی). فرض کنید $P = K[x_1, ..., x_n]$ و σ یک ترتیب یکجمله‌ای روی $\mathbb{T}^n$ و $I \subseteq P$ یک ایده‌ال دلخواه باشد. در این‌صورت مجموعه‌ی کلاس‌های مانده‌ی اعضای $\mathcal{O}_\sigma(I)$ در $\frac{P}{I}$ یعنی $\overline{\mathcal{O}_\sigma(I)} = \{t + I | t \in \mathcal{O}_\sigma(I)\}$ یک پایه برای K ـ فضای برداری $\frac{P}{I}$ خواهد بود.

برهان. رجوع کنید به [۴۶، ص.۶۲]. □

در قضیه‌ی بعد هم به رابطه‌ی پایه‌ی مرزی با پایه‌ی گروبنر پی می‌بریم، و هم به این سؤال پاسخ می‌دهیم که، آیا برای هر ایده‌ال صفر بعدی، می‌توان ایده‌ال ترتیبی یافت که وجود پایه‌ی مرزی نسبت به آن ایده‌ال ترتیبی تضمین شده باشد؟

قضیه ۶۵.۲. فرض کنید σ یک ترتیب یکجمله‌ای روی $\mathbb{T}^n$ باشد و I یک ایده‌ال صفر بعدی باشد، در این‌صورت I دارای $\mathcal{O}_\sigma(I)$- پایه‌ی مرزی یکتا مثل G است، که این پایه شامل پایه‌ی گروبنر تحویل یافته‌ی I نیز است. در ضمن پایه‌ی گروبنر تحویل‌یافته‌ی I را اعضایی از G تشکیل می‌دهند که متناظر با گوشه‌های $\mathcal{O}_\sigma(I)$ هستند.

برهان. بر اساس قضیه‌ی پایه‌ی مکالی ۶۴.۲ مجموعه‌ی کلاس‌های مانده اعضای $\mathcal{O}_\sigma(I)$ تشکیل یک پایه برای K- فضای برداری $\frac{P}{I}$ می‌دهد. به این ترتیب طبق قضیه‌ی ۶۲.۲، I دارای یک $\mathcal{O}_\sigma$ ـ پایه‌ی مرزی یکتا است. برای اثبات قسمت دوم، فرض کنید $b \in \mathbb{T}^n \mathcal{O}_\sigma(I)$ یکی از گوشه‌های $\mathcal{O}_\sigma(I)$ باشد. عضوی از پایه‌ی گروبنر که یکجمله‌ای پیشرو آن برابر b است را در نظر بگیرد، این عضو به‌صورت $f = \mathrm{LT}(f) + h$ است که $\mathrm{LT}(f) = \mathrm{LM}(f) = b$. از طرفی داریم، $\mathrm{NF}_I(f) = 0 = \mathrm{NF}_I(b) + \mathrm{NF}_I(h)$، در نتیجه $f = b - \mathrm{NF}_I(b)$. از طرفی طبق الگوریتم تقسیم ۱، هیچ‌یک از یکجمله‌ای‌های $\mathrm{NF}_I(b)$ بر هیچ‌یک از یکجمله‌ای‌های پیشرو اعضای پایه‌ی گروبنر بخش‌پذیر نیست و لذا هر یک از یکجمله‌ای‌های آن در $\mathcal{O}_\sigma(I)$ قرار دارند و در نتیجه $\mathrm{NF}_I(b)$ در K ـ فضای تولید شده توسط $\mathcal{O}_\sigma(I)$ قرار دارد.
از آن‌جایی‌که $\mathcal{O}_\sigma(I)$- پایه‌ی مرزی I یکتا است، f همان عضوی از $\mathcal{O}_\sigma(I)$ ـ پایه‌ی مرزی متناظر با b است. □

نباید تصور کرد که فقط به ازای ایده‌ال‌های ترتیبی $\mathcal{O} = \mathcal{O}_\sigma(I)$ می‌توانیم $\mathcal{O}$- پایه‌ی مرزی داشته باشیم. مثال زیر نشان می‌دهد، به ازای ایده‌ال ترتیبی $\mathcal{O}$ که از یک ترتیب یکجمله‌ای به‌دست نیامده هم امکان داشتن $\mathcal{O}$ ـ پایه‌ی مرزی وجود دارد.

مثال ۶۶.۲. فرض کنید $I \subseteq \mathbb{F}_2[x,y]$ ایده‌ال تولید شده توسط $\{x^2+xy, y^2+xy, x^3\}$ باشد. فرض کنید ایده‌ال ترتیبی را برابر با $\mathcal{O}=\{1,x,y,xy\}$ انتخاب کنیم که نتیجه می‌شود، $\partial\mathcal{O}=\{x^2, x^2y, xy^2, y^2\}$. نشان می‌دهیم که $G=\{x^2+xy, x^2y, xy^2, y^2+xy\}$ یک $\mathcal{O}$ پایه‌ی مرزی برای I است که به ازای هر ترتیب یکجمله‌ای، نمی‌تواند شامل یک پایه‌ی گروبنر I باشد.

از آن‌جایی که G یک پیش‌پایه‌ی مرزی است، طبق الگوریتم تقسیم مرزی ۳، $\mathcal{O}$ یک مولد $\mathbb{F}_2$ ـ فضای برداری $\frac{\mathbb{F}_2[x,y]}{I}$ است. بنابراین کافی است نشان دهیم مجموعه‌ی کلاس‌های مانده $\mathcal{O}$ مستقل خطی است، یا بطور معادل نشان دهیم هر ترکیب خطی از اعضای $\mathcal{O}$ که در I باشد، تمام ضرایبش صفر هستند.

فرض کنید $L=c_1+c_2x+c_3y+c_4xy \in I$ که $c_1,c_2,c_3,c_4 \in \mathbb{F}_2$. یک ترکیب خطی دلخواه از اعضای $\mathcal{O}$ باشد. ترتیب یکجمله‌ای الفبایی را در نظر بگیرید. پایه‌ی گروبنر تحویل‌یافته‌ی ایده‌ال I نسبت به این ترتیب عبارت است از:

$$GB=\{x^2+y^2, xy+y^2, y^3\}$$

$$\mathrm{LM}(GB)=\{x^2, xy, y^3\}$$

می‌دانیم که $L \in I$ اگر و تنها اگر $\mathrm{NF}_I(L)=0$. با محاسبه‌ی فرم نرمال L نسبت به I داریم:

$$L'=\mathrm{NF}_I(L)=c_1+c_1x+c_3y+c_4y^2$$

از آن‌جایی که $\mathrm{LM}(L')=c_2x$ و $x \notin \mathrm{LM}(GB)$ پس $c_2=0$ به این‌ترتیب $L'=c_1+c_3y+c_4y^2$، باز هم چون $\mathrm{LM}(L')=y^2 \notin \mathrm{LM}(GB)$ نتیجه می‌گیریم، $c_4=0$. با تکرار این روند c_1 و c_3 هم باید صفر باشند. بنابراین $\overline{\mathcal{O}}$ در فضای برداری $\frac{\mathbb{F}_2[x,y]}{I}$ مستقل خطی است و طبق قضیه‌ی ۶۲.۲ ، G یک $\mathcal{O}$- پایه‌ی مرزی است.

اکنون نشان می‌دهیم G تحت هر ترتیب یکجمله‌ای نمی‌تواند شامل یک پایه‌ی گروبنر باشد. فرض کنید σ یک ترتیب یکجمله‌ای دلخواه باشد. اگر $x >_\sigma y$ در این‌صورت $xy=\mathrm{LM}_\sigma(y^2+xy) \in \mathrm{LM}_\sigma(I)$. اگر $x <_\sigma y$ آن‌گاه $xy=\mathrm{LM}_\sigma(x^2+xy) \in \mathrm{LM}_\sigma(I)$. در نتیجه ایده‌ال ترتیبی $\mathcal{O}$ تحت هر ترتیب یکجمله‌ای مثل σ نمی‌تواند برابر با $\mathbb{T}^2 \setminus \mathrm{LM}_\sigma(I)$ باشد و در نتیجه شامل پایه‌ی گروبنر نیست.

یک جمع‌بندی از مباحث فوق نشان می‌دهد که لزومی ندارد یک ایده‌ال مثل I نسبت به هر ایده‌ال ترتیبی مثل $\mathcal{O}$ دارای پایه‌ی مرزی باشد، امّا وقتی ایده‌ال I صفر بعدی است، اگر ایده‌ال ترتیبی را بر اساس یک ترتیب یکجمله‌ای دلخواه مثل σ به‌صورت $\mathcal{O}=\mathcal{O}_\sigma(I)$ انتخاب کنیم، آن‌گاه $\mathcal{O}$ ـ پایه‌ی مرزی به‌صورت یکتا وجود خواهد داشت و جالب‌تر این‌که این پایه‌ی مرزی شامل پایه‌ی گروبنر تحویل یافته‌ی I نسبت به ترتیب یکجمله‌ای σ نیز هست. این یعنی در حالتی که ایده‌ال صفر بعدی باشد، پایه‌های مرزی تعمیمی از پایه‌های گروبنر محسوب می‌شوند. از سوی

دیگر ایده‌ال‌های ترتیبی وجود دارند که پایه‌ی مرزی متناظر با آن‌ها شامل پایه‌ی گروبنر تحویل یافته نیست. در واقع تعداد پایه‌های مرزی یک ایده‌ال از تعداد پایه‌های گروبنر آن خیلی بیشتر است و این یعنی قابلیت انعطاف بیشتری در کار با پایه‌های مرزی وجود دارد.

تعریف ۶۷.۲ (ایده‌ال ترتیبی پذیرفتنی). فرض کنید I یک ایده‌ال و $\mathcal{O}$ یک ایده‌ال ترتیبی باشد. $\mathcal{O}$ را یک *ایده‌ال ترتیبی پذیرفتنی* برای I گوییم، هرگاه I دارای یک $\mathcal{O}$- پایه‌ی مرزی باشد.

یکی از ویژگی‌های خوب پایه‌ی گروبنر، یکتایی باقی‌مانده مستقل از جایگشت مقسوم‌علیه‌ها در الگوریتم تقسیم ۱ بود، قضیه‌ی زیر نشان می‌دهد، این ویژگی بطور مشابه برای الگوریتم تقسیم مرزی و پایه‌ی مرزی نیز برقرار است.

قضیه ۶۸.۲. فرض کنید $\mathcal{G} = (g_۱, ..., g_\nu)$ یک $\mathcal{O}$ ـ پایه‌ی مرزی برای ایده‌ال $I \subseteq P$ باشد. فرض کنید $\pi : \{۱, ..., \nu\} \to \{۱, ..., \nu\}$ یک جایگشت و $\mathcal{G}' = (g_{\pi(۱)}, ..., g_{\pi(\nu)})$ یک جایگشت از $\mathcal{G}$ باشد. در این‌صورت به ازای هر چندجمله‌ای $f \in P$ داریم $\mathrm{NR}_{\mathcal{O},\mathcal{G}}(f) = \mathrm{NR}_{\mathcal{O},m\mathcal{G}'}(f)$.

برهان. اگر f را با استفاده از الگوریتم تقسیم مرزی ۳ بر $\mathcal{G}$ و $\mathcal{G}'$ تقسیم کنیم به ترتیب خواهیم داشت:

$$f = f_۱ g_۱ + \cdots + f_\nu g_\nu + \mathrm{NR}_{\mathcal{O},\mathcal{G}}(f) = f'_۱ g_۱ + \cdots + f'_\nu g_\nu + \mathrm{NR}_{\mathcal{O},\mathcal{G}'}(f),$$

که $f_i, f'_i \in P$. بنابراین داریم، $\mathrm{NR}_{\mathcal{O},\mathcal{G}}(f) - \mathrm{NR}_{\mathcal{O},\mathcal{G}'}(f) \in \langle \mathcal{O} \rangle_K \cap I$. از طرفی فرض کردیم که $\mathcal{G}$ یک $\mathcal{O}$ پایه‌ی مرزی I است که در این‌صورت داریم، $\langle \mathcal{O} \rangle_K \cap I = \{۰\}$، و در نتیجه $\mathrm{NR}_{\mathcal{O},\mathcal{G}}(f) = \mathrm{NR}_{\mathcal{O},\mathcal{G}'}(f)$. □

طبق قضیه‌ی ۶۲.۲، $\mathcal{O}$- پایه‌ی مرزی یک ایده‌ال صفربعدی نسبت به یک ایده‌ال ترتیبی در صورت وجود یکتا است. لذا می‌توانیم فرم نرمال یک چندجمله‌ای نسبت به یک ایده‌ال ترتیبی را به‌صورت زیر تعریف کنیم.

تعریف ۶۹.۲ (فرم نرمال چندجمله‌ای نسبت به ایده‌ال ترتیبی). فرض کنید $G = \{g_۱, ..., g_\nu\}$ یک $\mathcal{O}$- پایه‌ی مرزی ایده‌ال صفر بعدی I باشد. *فرم نرمال* چندجمله‌ای $f \in P$ نسبت به ایده‌ال ترتیبی $\mathcal{O}$ عبارت است از $\mathrm{NF}_{\mathcal{O},I}(f) = \mathrm{NR}_{\mathcal{O},\mathcal{G}}(f)$.

نتیجه ۷۰.۲. اگر I یک ایده‌ال صفر بعدی و $\mathcal{O}$ یک ایده‌ال ترتیبی پذیرفتنی برای I باشد. در این‌صورت $f \in I$ اگر و تنها اگر $\mathrm{NF}_{\mathcal{O},\mathcal{G}}(f) = ۰$.

در گزاره‌ی بعدی ویژگی‌های اولیه‌ی فرم نرمال بیان شده است.

گزاره ۷۱.۲. فرض کنید I یک ایده‌ال صفربعدی و $\mathcal{O}$ یک ایده‌ال ترتیبی پذیرفتنی برای I باشد. در این صورت به ازای هر $f, f_۱, f_۲ \in P$ و $a_۱, a_۲ \in K$، گزاره‌های زیر برقرار هستند.

۱. اگر یک ترتیب یکجمله‌ای نظیر σ وجود داشته باشد به‌طوری که $\mathcal{O} = \mathcal{O}_\sigma(I)$ ، آن‌گاه $\mathrm{NF}_{\mathcal{O},I}(f) = \mathrm{NF}_{\sigma,I}(f)$.

۲. $\mathrm{NF}_{\mathcal{O},I}(a_1 f_1 + a_2 f_2) = a_1 \mathrm{NF}_{\mathcal{O},I}(f_1) + a_2 \mathrm{NF}_{\mathcal{O},I}(f_2)$.

۳. $\mathrm{NF}_{\mathcal{O},I}(\mathrm{NF}_{\mathcal{O},I}(f)) = \mathrm{NF}_{\mathcal{O},I}(f)$.

۴. $\mathrm{NF}_{\mathcal{O},I}(f_1 f_2) = \mathrm{NF}_{\mathcal{O},I}(\mathrm{NF}_{\mathcal{O},I}(f_1)\, \mathrm{NF}_{\mathcal{O},I}(f_2))$.

برهان. می‌دانیم که همه‌ی ویژگی‌های ذکر شده از گزاره‌ی ۲ به بعد برای $\mathrm{NF}_{\sigma,I}(f)$ برقرار است، پس کافی است گزاره‌ی ۱ را ثابت کنیم. طبق تعریف $\mathcal{O}$- پایه‌ی مرزی، چون $\mathcal{O}$ یک ایده‌ال ترتیبی پذیرفتنی برای I است لذا داریم $P = I \oplus \langle \mathcal{O} \rangle_K$. در نتیجه هر چندجمله‌ای مثل f نمایشی یکتا به‌صورت $f = g + h$ دارد به‌طوری که $g \in I$ و $h \in \langle \mathcal{O} \rangle_K$. از طرفی اگر $G = \{g_1, ..., g_\nu\}$ یک $\mathcal{O}$ــ پیش‌پایه‌ی مرزی باشد، طبق الگوریتم‌های تقسیم ۱ و تقسیم مرزی ۳، f نمایش‌هایی به‌صورت زیر دارد:

$$f = a_1 g_1 + \cdots + a_s g_\nu + \mathrm{NF}_{\sigma,I}(f),\ a_1 g_1 + \cdot + a_\nu g_\nu \in I,\ \mathrm{NF}_{\sigma,I}(f) \in \langle \mathcal{O} \rangle_K.$$

$$f = f_1 g_1 + \cdots + f_s g_\nu + \mathrm{NF}_{\sigma,I}(f),\ f_1 g_1 + \cdot + f_\nu g_\nu \in I,\ \mathrm{NF}_{\mathcal{O},I}(f) \in \langle \mathcal{O} \rangle_K.$$

چون نمایش f به‌صورت ذکر شده در بند قبل، باید یکتا باشد لذا $\mathrm{NF}_{\sigma,I}(f) = \mathrm{NF}_{\mathcal{O},I}(f)$. □

مثال زیر دو وجه برتری مهم پایه‌های مرزی نسبت به پایه‌ی گروبنر را نشان می‌دهد.

مثال ۷۲.۲. فرض کنید $P = \mathbb{Q}[x, y]$ و $I = \langle \frac{1}{4}x^2 + y^2 - 1, x^2 + \frac{1}{4}y^2 - 1 \rangle$. فرض کنید ε یک عدد کوچک باشد و ایده‌ال $\tilde{I}$ را به‌صورت زیر در نظر بگیریم:

$$\tilde{I} = \langle \frac{1}{4}x^2 + y^2 + \varepsilon xy - 1, x^2 + \frac{1}{4}y^2 + \varepsilon xy - 1 \rangle.$$

در این‌صورت ایده‌ال ترتیبی $\mathcal{O} = \{1, x, y, xy\}$ یک ایده‌ال ترتیبی پذیرفتنی برای هر دو ایده‌ال I و $\tilde{I}$ است. می‌خواهیم بین پایه‌های گروبنر و مرزی I و $\tilde{I}$ مقایسه‌ای داشته باشیم. فرض کنید برای محاسبه‌ی پایه‌ی گروبنر تحویل‌یافته از ترتیب یکجمله‌ای الفبایی مدرج معکوس استفاده کنیم. در این‌صورت پایه‌ی گروبنر I و $\tilde{I}$ را که به ترتیب با GB و $\tilde{GB}$ نمایش می‌دهیم عبارتند از:

$$GB = \{x^2 - \frac{4}{5}, y^2 - \frac{4}{5}\}$$

$$\tilde{GB} = \{x^2 - y^2, xy + \frac{5}{4\varepsilon}y^2 - \frac{1}{\varepsilon}, y^3 - \frac{16\varepsilon}{16\varepsilon^2 - 25}x + \frac{20}{16\varepsilon^2 - 25}y\}.$$

همان طور که مشاهده می‌شود وقتی ε از مقدار ۰ در ایده‌ال I به یک مقدار ناچیز در ایده‌ال $\tilde{I}$ تغییر می‌کند، پایه‌ی گروبنر به سبب وجود جمله‌ی $\frac{5}{4\varepsilon}$ دچار تغییرات بسیار بزرگی خواهد شد که نشان دهنده‌ی ناپایداری پایه‌ی گروبنر نسبت به تغییرات کوچک در مولد داده شده است. اکنون $\mathcal{O}$-پایه‌ی مرزی I و $\tilde{I}$ را که به ترتیب با B و $\tilde{B}$ نمایش داده‌ایم در زیر مشاهده کنید.

$$B = \{x^2 - \frac{4}{5}, x^2y - \frac{4}{5}y, xy^2 - \frac{4}{5}x, y^2 - \frac{4}{5}\}.$$

$$\begin{aligned}\tilde{B} = \{&x^2 + \frac{4}{5}\varepsilon xy - \frac{4}{5}, xy^2 - \frac{16\varepsilon}{16\varepsilon^2 - 25}x + \frac{20}{16\varepsilon^2 - 25}y,\\ &xy^2 + \frac{20}{16\varepsilon^2 - 25}x - \frac{16\varepsilon}{16\varepsilon^2 - 25}y, y^2 + \frac{4}{5}\varepsilon xy - \frac{4}{5}\}.\end{aligned}$$

در این‌حالت وقتی ضرایب xy از مقدار صفر در ایده‌ال I به یک مقدار ناچیز در $\tilde{I}$ تغییر می‌کند، پایه‌ی مرزی به‌طور پیوسته تغییر می‌کند، یعنی اگر تغییرات ε ناچیز باشد، تغییرات پایه‌ی مرزی هم ناچیز خواهد بود، به همین جهت است که می‌گویند پایه‌های مرزی از لحاظ عددی رفتاری پایدارتر از پایه‌های گروبنر دارند. وجه دیگر برتری پایه‌ی مرزی نسبت به پایه‌ی گروبنر حفظ تقارن است. مولد ایده‌ال I دارای تقارن است، یعنی اگر جای x ، y را در آن عوض کنیم، شکل مولد تغییر نمی‌کند، همان طور که مشاهده می‌شود پایه‌ی مرزی تقارن را حفظ کرده به‌طوری که B نیز متقارن است، ولی پایه‌ی گروبنر تقارن را از بین برده و می‌بینیم که GB متقارن نیست.

با وجود این‌که حفظ تقارن و پایداری دو وجه برتری پایه‌های مرزی نسبت به پایه‌های گروبنر هستند، امّا این دو ویژگی انگیزه‌ی اصلی ما از استفاده از پایه‌های مرزی در رمزنگاری نیست، بلکه ما ویژگی‌های دیگری از پایه‌ی مرزی را مد نظر داریم که در ادامه آن‌ها را بیان خواهیم کرد.

نکته ۷۳.۲. وقتی $\mathcal{O}$ یک ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال I باشد، داریم $P = I \oplus \langle\mathcal{O}\rangle_K$. در نتیجه به ازای هر ایده‌ال ترتیبی پذیرفتنی دیگر مثل $\mathcal{O}'$ داریم،

$$|\mathcal{O}| = \dim_K(\langle\mathcal{O}\rangle_K) = \dim_K(\langle\mathcal{O}'\rangle_K) = |\mathcal{O}'|.$$

وقتی I یک ایده‌ال صفر بعدی است یعنی $\dim_K(\frac{P}{I})$ متناهی است ولی می‌دانیم که $\dim_K(\frac{P}{I}) = \dim(\langle\mathcal{O}\rangle_K)$ ، بنابراین صفر بعدی بودن I متناهی بودن $|\mathcal{O}|$ و در نتیجه متناهی بودن $|\partial\mathcal{O}|$ را به‌دنبال دارد و بالعکس. ایده‌ال‌های مورد مطالعه‌ی ما در رمزنگاری ایده‌ال‌های صفر بعدی هستند به همین دلیل است که ایده‌ال‌های ترتیبی را متناهی در نظر می‌گیریم.

شناسایی پایه‌ی مرزی

در این بخش ابتدا با تقلید از روش‌های شناسایی پایه‌های گروبنر که در فصل قبل بیان شد، روش‌هایی را برای شناسایی پایه‌های مرزی ارائه می‌کنیم. سپس با روش‌هایی برای شناسایی پایه‌های مرزی آشنا می‌شویم که مشابهی در پایه‌های گروبنر ندارند.

گزاره ۷۴.۲. فرض کنید $\mathcal{O}$ یک ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال صفر بعدی $I \subseteq P$ باشد. در ضمن فرض کنید $G = \{g_1, ..., g_\nu\}$ یک $\mathcal{O}$- پیش‌پایه‌ی I باشد. در این‌صورت G یک $\mathcal{O}$- پایه‌ی مرزی I است، اگر وتنها اگر یکی از شرط های معادل زیر برقرار باشد.

۱. به ازای هر $f \in I\backslash\{0\}$ ، چندجمله‌ای‌های $f_1, ..., f_\nu \in P$ وجود داشته باشند به‌طوری که:

$$f = f_1 g_1 + \cdots + f_\nu g_\nu \ \wedge \ (f_i \neq 0 \Rightarrow \deg(f_i) \leq \mathrm{ind}_{\mathcal{O}}(f) - 1).$$

۲. به ازای هر $f \in I\backslash\{0\}$ ، چندجمله‌ای‌های $f_1, ..., f_\nu \in P$ وجود داشته باشند، به‌طوری که:

$$f = f_1 g_1 + \cdots + f_\nu g_\nu, \ \max\{\deg(f_i) \mid i \in \{1, ..., \nu\}, f_i \neq 0\} = \mathrm{ind}_{\mathcal{O}}(f) - 1.$$

برهان. به [۴۵، ص.۴۳۱] ، رجوع کنید. □

ما پایه‌های گروبنر را به عنوان پایه‌هایی از ایده‌ال می‌شناختیم که مجموعه‌ی جملات پیشرو اعضای آن مولّدی برای ایده‌ال پیشرو بود. در پایه‌های مرزی مفهوم مشابهی تحت عنوان فرم مرزی جایگزن یکجمله‌ای‌های پیشرو می‌شود و به‌دنبال آن ایده‌الی تحت عنوان ایده‌ال تولید شده توسط فرم‌های مرزی را تعریف می‌کنیم، تا بوسله‌ی آن ویژگی‌ای مشابه با پایه‌های گروبنر، برای پایه‌های مرزی تعریف کنیم.

تعریف ۷۵.۲. چند جمله‌ای دلخواه $f = a_1u_1 + \cdots + a_su_s$ را که $a_1, ..., a_s \in K\backslash\{0\}$ و $u_1, ..., u_s \in \mathbb{T}^n$ را طوری مرتب کرده‌ایم که، $\mathrm{ind}_{\mathcal{O}}(u_1) \geq \cdots \geq \mathrm{ind}_{\mathcal{O}}(u_s)$.

۱. چند جمله‌ای $\mathrm{BF}_{\mathcal{O}}(f) = \sum_{\{i \mid \ \mathrm{ind}_{\mathcal{O}}(u_i) = \mathrm{ind}_{\mathcal{O}}(f)\}} a_i u_i$ را **فرم مرزی** f نسبت به $\mathcal{O}$ می‌نامیم و برای $f = 0$ قرارداد می‌کنیم که $\mathrm{BF}_{\mathcal{O}}(f) = 0$.

۲. به ازای ایده‌ال $I \subseteq P$ ، ایده‌ال $\langle \mathrm{BF}_{m\mathcal{O}} \rangle$ را **ایده‌ال فرم‌های مرزی** I نسبت به $\mathcal{O}$ می‌نامیم.

برای مثال اعضای یک $\mathcal{O}$- پیش‌پایه‌ی مرزی I مثل $G = \{g_1, ..., g_\nu\}$ دارای فرم‌های مرزی $\mathrm{BF}_{\mathcal{O}}(g_i) = b_i$ هستند.

گزاره ۷۶.۲. فرض کنید $\mathcal{O} = \{t_1, ..., t_\mu\}$ یک ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال صفر بعدی I و $G = \{g_1, ..., g_\nu\}$ یک $\mathcal{O}$ ـ پیش‌پایه‌ی ایده‌ال I باشد. در این‌صورت G یک $\mathcal{O}$ ـ پایه‌ی مرزی I است، اگر و تنها اگر یکی از شرایط معادل زیر برقرار باشد.

۱. $\forall\, f \in I:\ \mathrm{Supp}(\mathrm{BF}_{\mathcal{O}}) \subseteq \mathbb{T}^n \backslash \mathcal{O}$.

۲. $\langle \mathrm{BF}_{\mathcal{O}}(I)\rangle = \langle \mathrm{BF}_{\mathcal{O}}(g_1), ..., \mathrm{BF}_{\mathcal{O}}(g_\nu)\rangle = \langle b_1, ..., b_\nu\rangle$

برهان. ابتدا نشان می‌دهیم که $\mathcal{O}$- پایه‌ی مرزی بودن G گزاره‌ی (۱) را نتیجه می‌دهد. با برهان خلف فرض کنید چندجمله‌ای $f \in I\backslash\{0\}$ وجود دارد به‌طوری که، $\mathrm{Supp}(\mathrm{BF}_{\mathcal{O}}(f)) \cap \mathcal{O} \neq \emptyset$ در این صورت $\mathrm{Supp}(f) \subseteq \mathcal{O}$ یا به عبارت دیگر $c_1, ..., c_\mu \in K$ وجود دارند که $f = c_1t_1 + \cdots + c_\mu t_\mu$. در این‌صورت فرض $\mathcal{O}$- پایه‌ی مرزی بودن G لازم می‌دارد که $c_1 = \cdots = c_\mu = 0$ که با فرض $f \neq 0$ در تناقض است.

در این مرحله ثابت می‌کنیم گزاره‌ی (۱) گزاره‌ی (۲) را نتیجه می‌دهد. به ازای هر $i \in \{1, ..., \nu\}$ چون $g_i \in I$ خواهیم داشت، $b_i \in \langle \mathrm{BF}_{\mathcal{O}}(I)\rangle$، و در نتیجه $\langle b_1, ..., b_\nu\rangle \subseteq \langle \mathrm{BF}_{\mathcal{O}}(I)\rangle$. برای اثبات شمول در جهت عکس فرض کنید $f \in I\backslash\{0\}$ دلخواه باشد.طبق گزاره‌ی (۱)، $\mathrm{Supp}(\mathrm{BF}_{\mathcal{O}}(f)) \subseteq \mathbb{T}^n \backslash \mathcal{O}$. در نتیجه هر یکجمله‌ای از $\mathrm{BF}_{\mathcal{O}}(f)$ توسط یکی از اعضای $\partial\mathcal{O}$ بخش می‌شود. بنابراین $\mathrm{BF}_{\mathcal{O}} \in \langle b_1, ..., b_\nu\rangle$.

برای کامل شدن اثبات نشان می‌دهیم که گزاره‌ی (۲)، $\mathcal{O}$- پایه‌ی مرزی بودن G را نتیجه می‌دهد. کافی است نشان دهیم مجموعه‌ی کلاس‌های مانده اعضای $\mathcal{O}$ به پیمانه‌ی ایده‌ال I مستقل خطی است. فرض کنید $c_1, ..., c_\mu \in K$ وجود داشته باشند به‌طوری که $f = c_1t_1 + \cdots c_\mu t_\nu \in I$. در این‌صورت همه‌ی یکجمله‌ای‌های f دارای اندیس صفر هستند و لذا $f = \mathrm{BF}_{\mathcal{O}}(f)$. طبق گزاره‌ی (۲) داریم $f \in \langle b_1, ..., b_\nu\rangle$، از این‌رو به ازای هر $c_i \neq 0$ یکجمله‌ای t_i توسط یکی از b_j ها بخش می‌شود که این امکان پذیر نیست و در نتیجه باید داشته باشیم، $c_1 = \cdots = c_\mu = 0$. □

فرض کنید $\mathcal{O} = \{t_1, ..., t_\mu\}$ یک ایده‌ال ترتیبی و $G = \{g_1, ..., g_\nu\}$ یک $\mathcal{O}$- پیش‌پایه باشد. فرض کنید $f \in P$ و $t \in \mathrm{Supp}(f)$ مضربی از یک یکجمله‌ای مرزی باشد یعنی $b_i \in \partial\mathcal{O}$ وجود داشته باشد به‌طوری که $t = t'b_i$. اگر $c \in K$ ضریب t در f باشد، آن‌گاه چندجمله‌ای $h = f - ct'g_i$ فاقد یکجمله‌ای t خواهد بود، در این‌صورت می‌گوییم f در **یک گام** به به‌وسیله ی g_i به h **تحویل یافته** ، و با $f \xrightarrow{g_i} h$ نمایش می‌دهیم. به بیان ساده تحت این رابطه‌ی تحویل هر جایی که یک یکجمله‌ای مرزی مثل b_j وجود داشت به جای آن $\sum_{i=1}^{\mu} \alpha_{ij} t_i$ را قرار می‌دهیم.

بستار تعدی و بازتابی رابطه‌های $\xrightarrow{g_i}$ به ازای $i \in \{1, ...,\}$ را، **رابطه‌ی بازنویسی** وابسته به G گوییم و با $\xrightarrow{G}$ نمایش می‌دهیم. به عبارت دیگر $f \xrightarrow{G} h$ اگر و تنها اگر $i_1, ..., i_t \in \{1, ..., \nu\}$ و $h_0, ..., h_t \in P$ وجود داشته باشند به‌طوری که،

$$f = h_0 \xrightarrow{g_{i_1}} h_1 \xrightarrow{g_{i_2}} \cdots \xrightarrow{g_{i_t}} h_t = h.$$

مثال ۷۷.۲. فرض کنید $P = \mathbb{Q}[x, y]$ و $\mathcal{O} = \{1, x, y, x^2, y^2\}$. در این‌صورت $\mathcal{O}$ یک ایده‌ال ترتیبی با مرز $\partial\mathcal{O} = \{xy, x^3, x^2y, xy^2, y^3\}$ است. در ضمن $G = \{g_1, ..., g_5\}$ که $g_1 = xy - x^2 - y^2, g_2 =$

$g_5 = y^3$ و $x^3, g_3 = x^2y, g_4 = xy^2$، یک $\mathcal{O}$- پیش‌پایه‌ی مرزی است. زنجیری از تحویل‌ها در زیر نشان داده شده.

$$x^2y \xrightarrow{g_1} x^3 + xy^2 \xrightarrow{g_2} xy^2 \xrightarrow{g_1} x^2y + y^3 \xrightarrow{g_5} x^2y.$$

این فرآیند به‌صورت نامتناهی می‌تواند ادامه یابد.

گزاره ۷۸.۲. فرض کنید G یک $\mathcal{O}$- پیش‌پایه برای ایده‌ال صفر بعدی I باشد، در این صورت G یک $\mathcal{O}$- پایه‌ی مرزی I است اگر و تنها اگر گزاره‌ی زیر برقرار باشد.

$$\forall\, f \in P: \ (f \xrightarrow{G} 0 \iff f \in I).$$

برهان. به [۴۵، ص.۴۳۳]، رجوع کنید. □

در ادامه قصد داریم راهی برای شناسایی پایه‌ی مرزی ارائه دهیم که مشابهی در پایه‌ی گروبنر ندارد. برای این‌کار لازم است تا یک شیٔ جدید تعریف کنیم.

تعریف ۷۹.۲ (ماتریس ضربی استاندارد). فرض کنید $\mathcal{O} = \{t_1, ..., t_\mu\}$ یک ایده‌ال ترتیبی و $G = \{g_1, ..., g_\nu\}$ یک $\mathcal{O}$- پیش‌پایه‌ی مرزی باشد به‌طوری که به ازای $i = 1, ..., \mu$ و $j = 1, ..., \nu$ داشته باشیم، $g_j = b_j - \sum_{i=1}^{\mu} \alpha_{ij} t_i$. به ازای $r \in \{1, ..., n\}$ داده شده، که n تعداد متغیرهای حلقه‌ی چندجمله‌ای است، r امین *ماتریس ضربی استاندارد* G را با $\mathcal{X}_r = (\xi_{kl}^{(r)})$ نمایش داده و به‌صورت زیر تعریف می‌کنیم:

$$\xi_{kl}^{(r)} = \begin{cases} \delta_{ki}, & x_r t_l = t_i \\ \alpha_{kj}, & x_r t_l = b_j \end{cases}$$

در رابطه‌ی فوق هر گاه $k = i$ آن‌گاه $\delta_{ki} = 1$ و در غیر این‌صورت $\delta_{ki} = 0$. به عبارت دیگر برای به‌دست آوردن r امین ماتریس ضربی استاندارد G متغیر x_r را به ازای هر $i \in \{1, ..., \mu\}$، در $t_i \in \mathcal{O}$ ضرب می‌کنیم، اگر $x_r t_i = b_j$، و حاصل یک یکجمله‌ای مرزی بود و $g_j = b_j - \sum_{k=1}^{\mu} \alpha_{kj} t_k$ آن‌گاه ستون i ام ماتریس $\mathcal{X}_r$ را با $(\alpha_{1j}, ..., \alpha_{\mu j})$ پر می‌کنیم. اگر $x_r t_i = t_j$ آن‌گاه ستون i ام $\mathcal{X}_r$ را با بردار یکه‌ی j ام پر می‌کنیم.

قضیه ۸۰.۲. فرض کنید $\mathcal{O} = \{t_1, ..., t_\mu\}$ یک ایده‌ال و $G = \{g_1, ..., g_\nu\}$ یک $\mathcal{O}$- پیش‌پایه‌ی ایده‌ال $I = \langle g_1, ..., g_\nu \rangle$ باشد. در این‌صورت G یک $\mathcal{O}$ ـ پایه‌ی مرزی I است اگر و تنها اگر ضرب دوبه‌دو ماتریس‌های ضربی استاندارد G خاصیت جابجایی داشته باشد، یعنی

$$\forall\, 1 \le i \le j \le n: \ \mathcal{X}_i \mathcal{X}_j = \mathcal{X}_j \mathcal{X}_i.$$

برهان. به [۴۵، ص.۴۳۴] ، مراجعه کنید. □

مثال ۸۱.۲. فرض کنید $P=\mathbb{Q}[x,y]$ و $\mathcal{O}=\{۱,x,y,x^۲,y^۲\}$. مرز $\mathcal{O}$ عبارت‌است از $\partial\mathcal{O}=\{xy,x^۳,y^۳,x^۲y,xy^۲\}$. مجموعه‌ی چندجمله‌ای‌های $G=\{g_۱,...,g_۵\}$ که $g_۱=xy-x^۲-y^۲, g_۲=x^۳-x^۲, g_۳=y^۳-y^۲$ و $g_۴=x^۲y-x^۲$ و $g_۵=xy^۲-y^۲$، یک $\mathcal{O}$- پیش‌پایه‌ی مرزی ایده‌ال $I=\langle g_۱,...,g_۵\rangle$ است. ماتریس‌های ضربی G عبارتند از

$$\mathcal{X}=\begin{pmatrix} ۰&۰&۰&۰&۰\\ ۱&۰&۰&۰&۰\\ ۰&۰&۰&۰&۰\\ ۰&۱&۱&۱&۰\\ ۰&۰&۱&۰&۱ \end{pmatrix}\quad \mathcal{Y}=\begin{pmatrix} ۰&۰&۰&۰&۰\\ ۰&۰&۰&۰&۰\\ ۱&۰&۰&۰&۰\\ ۰&۱&۰&۱&۰\\ ۰&۱&۱&۰&۱ \end{pmatrix}$$

که ضرب آن‌ها خاصیت جابجایی ندارد، زیرا

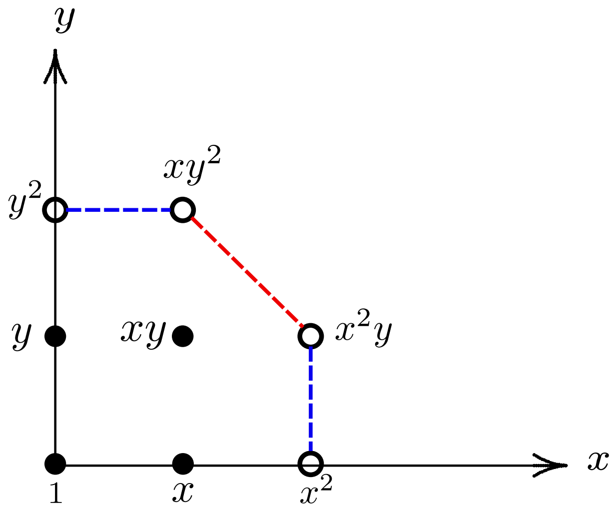
$$\mathcal{X}\cdot\mathcal{Y}=\begin{pmatrix} ۰&۰&۰&۰&۰\\ ۰&۰&۰&۰&۰\\ ۰&۰&۰&۰&۰\\ ۱&۱&۰&۱&۰\\ ۱&۱&۱&۰&۱ \end{pmatrix}\neq\ \mathcal{Y}\cdot\mathcal{X}=\begin{pmatrix} ۰&۰&۰&۰&۰\\ ۰&۰&۰&۰&۰\\ ۰&۰&۰&۰&۰\\ ۱&۱&۱&۱&۰\\ ۱&۰&۱&۰&۱ \end{pmatrix}$$

در نتیجه طبق قضیه‌ی ۸۰.۲ ، یک $\mathcal{O}$- پایه‌ی مرزی نیست.

در ادامه قصد داریم تا محکی مشابه محک بوخبرگر برای شناسایی پایه‌های گروبنر را، برای پایه‌های مرزی ارائه کنیم، ولی قبل از آن لازم است تا دو مفهوم جدید را تعریف کنیم.

تعریف ۸۲.۲ (همسایگی). فرض کنید $\mathcal{O}$ یک ایده‌ال ترتیبی و $b_i,b_j\in\partial\mathcal{O}$ دو یکجمله‌ای مرزی آن باشند. b_i و b_j را *همسایه* گوییم هرگاه x_k و x_l وجود داشته باشند به‌طوری که، $b_i=x_kb_j$ یا $x_kb_i=x_lb_j$.

مثال ۸۳.۲. فرض کنید $\mathcal{O}=\{۱,x,y,xy\}$، در نتیجه داریم $\partial\mathcal{O}=\{x^۲,x^۲y,xy^۲,y^۲\}$. همان طور که در شکل ۵.۲ مشاهده می‌شود $(x^۲,x^۲y)$ و $(y^۲,xy^۲)$ و $(x^۲y,xy^۲)$ همسایه هستند.



شکل ۵.۲: یکجمله‌ای‌های مرزی همسایه

تعریف ۸۴.۲ (S**- چندجمله‌ای).** فرض کنید G یک $\mathcal{O}$- پیش‌پایه‌ی مرزی باشد و $g_i, g_j \in G$، به‌طوری که $g_i = b_i - \sum_k^{\mu} \alpha_{ki} t_k$ و $g_j = b_j - \sum_k^{\mu} \alpha_{kj} t_k$. در این‌صورت S- چند جمله‌ای g_i و g_j به‌صورت زیر تعریف می‌شود.

$$S_{ij} = (\frac{\mathrm{LCM}(b_i, b_j)}{b_i}) \cdot g_i - (\frac{\mathrm{LCM}(b_i, b_j)}{b_j}) \cdot g_j.$$

نکته ۸۵.۲. فرض کنید G یک $\mathcal{O}$- پیش‌پایه‌ی مرزی باشد و $g_i, g_j \in G$ و b_i و b_j همسایه باشند.

۱. اگر به ازای یک x_k داشته باشیم $b_j = x_k b_i$ آن‌گاه $S_{ij} = g_j - x_k g_i$.

۲. اگر به ازای x_k, x_l داشته باشیم، $x_k b_i = x_l b_j$ آن‌گاه $S_{ij} = x_k g_i - x_l g_j$.

همان طور که مشاهده می‌شود، در هر دو حالت داریم $\mathrm{Supp}(S_{ij}) \subseteq \mathcal{O} \cup \partial\mathcal{O}$. بنابراین $a_۱, ..., a_\mu \in K$ وجود دارند به‌طوری که

$$\mathrm{NR}_{\mathcal{O},G}(S_{ij}) = S_{ij} - \sum_{m=۱}^{\mu} a_m g_m \in I \ \wedge \ \mathrm{Supp}(\mathrm{NR}_{\mathcal{O},G}(S_{ij})) \subseteq \mathcal{O}.$$

در این‌صورت اگر G یک $\mathcal{O}$- پایه‌ی مرزی باشد، نتیجه می‌گیریم که $\mathrm{NR}_{\mathcal{O},G}(S_{ij}) = ۰$.

قضیه ۸۶.۲ (محک بوخبرگر برای پایه‌های مرزی). فرض کنید $\mathcal{O} = \{t_۱, ..., t_\mu\}$ یک ایده‌ال ترتیبی و $G = \{g_۱, ..., g_\nu\}$ یک $\mathcal{O}$- پیش‌پایه‌ی مرزی ایده‌ال I باشد. در این‌صورت $G \subseteq I$ یک $\mathcal{O}$- پایه‌ی مرزی I است اگر و تنها اگر یکی از شرط‌های معادل زیر برقرار باشد.

۱. به ازای هر $۱ \leq i \leq j \leq \nu$ ، داشته باشیم $S(g_i, g_j) \xrightarrow{G} ۰$.

۲. به ازای هر $\{i, j\}$، اگر b_i و b_j همسایه باشند آن‌گاه $S(g_i, g_j) \xrightarrow{G} ۰$

۳. به ازای هر $\{i,j\}$ اگر b_i و b_j همسایه باشند آن‌گاه $\mathrm{NR}_{\mathcal{O},G}(S_{ij}) = 0$.

۴. به ازای هر $\{i,j\}$ ، که b_i و b_j همسایه هستند، ثابت‌های $c_1, \dots, c_\nu \in K$ وجود دارند به‌طوری که

$$S(g_i, g_j) = c_1 g_1 + \cdots + c_\nu g\nu.$$

برهان. به [۴۵، ص.۴۳۸] ، رجوع کنید. □

الگوریتم‌های محاسبه‌ی پایه‌ی مرزی

این بخش از نظر ما مهمترین بخش مرتبط با پایه‌های مرزی است چرا که دلیل ترجیح ما در استفاده از پایه‌ی مرزی نسبت به پایه‌ی گروبنر در الگوریتم‌های محاسبه‌ی پایه‌ی مرزی نهفته است.

همان طور که قضیه‌ی ۶۵.۲ بیان می‌کند، پایه‌های مرزی تعمیمی از پایه‌های گروبنر برای ایده‌ال‌های صفربعدی هستند. در ابتدای این قسمت الگوریتمی تحت عنوان الگوریتم تغییر پایه را معرفی می‌کنیم، که با استفاده از محاسبه‌ی پایه‌ی گروبنر قادر است پایه‌ی مرزی را نیز محاسبه کند. نتیجه‌ی ضمنی این الگوریتم این است که محاسبه‌ی پایه‌ی مرزی نباید سخت‌تر از محاسبه‌ی پایه‌ی گروبنر باشد.

قضیه ۸۷.۲ (الگوریتم تغییر پایه). فرض کنید $I \subseteq P$ یک ایده‌ال چندجمله‌ای صفربعدی و $\mathcal{O}$ یک ایده‌ال ترتیبی باشد. الگوریتم ۴ با دریافت $\mathcal{O}$ ابتدا مجاز بودن $\mathcal{O}$ برای ایده‌ال I را بررسی می‌کند، در صورتی که $\mathcal{O}$ مجاز نباشد توقف کرده و مجاز نبودن $\mathcal{O}$ را به عنوان خروجی تولید می‌کند در غیر این‌صورت، پس از متناهی مرحله اجرا، $\mathcal{O}$- پایه‌ی مرزی یکتای I را محاسبه می‌کند.

الگوریتم ۴ الگوریتم تغییر پایه برای محاسبه پایه مرزی

Input: I و مولدی برای ایده‌ال $\mathcal{O} = \{t_1, ..., t_\mu\}$

Output: $G = \{g_1, ..., g_\nu\}$ (G یک $\mathcal{O}$- پایه‌ی مرزی برای ایده‌ال I است.)

$\sigma \leftarrow$ یک ترتیب یکجمله‌ای

$\mathcal{O}_\sigma(I) \leftarrow \mathbb{T}^n \setminus \mathrm{LM}_\sigma(I)$

if $|\mathcal{O}_\sigma(I)| \neq \mu$ **then**

print اندازه‌ی $\mathcal{O}$ برای داشتن پایه‌ی مرزی کافی نیست

return

end if

$\{s_1, ..., s_\mu\} \leftarrow \mathcal{O}_\sigma(I)$

for $m \in \{1, ..., \mu\}$ **do**

$\sum_{i=1}^{\mu} \tau_{im} s_i \leftarrow \mathrm{NF}_{\sigma,I}(t_m)$

end for

$\mathcal{T} \leftarrow [\tau_{im}]_{1\leq i,m\leq\mu}$. ماتریسی که با τ_{im} ها پر شده

if $\det(\mathcal{T}) = 0$ **then**

print $\mathcal{O}$ یک ایده‌ال ترتیبی پذیرفتنی نیست

return

end if

$\{b_1, ..., b_\nu\} \leftarrow \partial\mathcal{O}$

for $j \in \{1, ..., \nu\}$ **do**

$\sum_{i=1}^{\mu} \beta_{ij} s_i \leftarrow \mathrm{NF}_{\sigma,I}(b_j)$

end for

$\mathcal{B} \leftarrow [\beta_{ij}]_{1\leq i\leq\mu, 1\leq j\leq\nu}$.

$[\alpha_{ij}]_{\mu\times\nu} \leftarrow \mathcal{T}^{-1}\mathcal{B}$

for $j \in \{1, ..., \nu\}$ **do**

$g_j \leftarrow \sum_{i=1}^{\mu} \alpha_{ij} t_i$

end for

return $\{g_1, ..., g_\nu\}$

برهان. متناهی بودن مراحل اجرا بدیهی است و فقط صحت خروجی الگوریتم را ثابت می‌کنیم. طبق قضیه‌ی پایه‌ی مکالی ۶۴.۲، $|\mathcal{O}_\sigma(I)| = \dim_K(\frac{P}{I})$. بنابراین وقتی شرط $|\mathcal{O}| \neq |\mathcal{O}_\sigma(I)|$ برقرار نیست الگوریتم با چاپ پیام «اندازه‌ی $\mathcal{O}$ کافی نیست» خاتمه می‌یابد. همچنین طبق قضیه‌ی پایه‌ی مکالی کلاس‌های مانده اعضای $\mathcal{O}_\sigma(I)$ یک پایه برای فضای برداری $\frac{P}{I}$ است. در گامی که $\mathrm{NF}_{\sigma,I}(t_m)$ ها محاسبه می‌شوند، در واقع نمایش $\bar{t_i}$ ها نسبت به پایه‌ی $\overline{\mathcal{O}}_\sigma(I) = \{\bar{s_1}, ..., \bar{s_\mu}\}$ برای

$\frac{P}{I}$ به‌صورت زیر به‌دست می‌آید:

$$\forall\ 1 \leq i \leq \mu\ :\ \bar{t_i} = \tau_{1i}\bar{s_1} + \cdots + \tau_{\mu i}\bar{s_\mu}$$

نمایش ماتریسی رابطه‌ی فوق به‌صورت $(\bar{t_1}\ \cdots\ \bar{t_\mu}) = (\bar{s_1}\ \cdots\ \bar{s_\mu})\mathcal{T}$ است. بنابراین $\overline{\mathcal{O}}$ یک پایه برای فضای برداری $\frac{P}{I}$ است، اگر و تنها اگر $\mathcal{T}$ معکوس‌پذیر باشد. از طرفی روابط زیر در $\frac{P}{I}$ برقرار است.

$$\bar{b_j} = (\bar{s_1}\ \cdots\ \bar{s_\mu})\begin{pmatrix}\beta_{1j}\\ \vdots\\ \beta_{\mu j}\end{pmatrix} = (\bar{t_1}\ \cdots\ \bar{t_\mu})\mathcal{T}^{-1}\begin{pmatrix}\beta_{1j}\\ \vdots\\ \beta_{\mu j}\end{pmatrix}.$$

که نشان می‌دهد خروجی الگوریتم یک $\mathcal{O}$ پایه‌ی مرزی برای I است. $\square$

الگوریتم فوق که توسط کروزِر و کِرین در سال ۲۰۰۶ در [۴۱] معرفی شد، شبیه الگوریتم شناخته شده‌ی FGLM [۳۳] است، چرا که هر دو الگوریتم به ازای یک پایه‌ی گروبنر داده شده، با استفاده از تغییر پایه‌ی فضای برداری، پایه‌ای دیگر برای ایده‌ال محاسبه می‌کنند. با این‌حال بین الگوریتم تغییر پایه و FGLM یک تفاوت اساسی وجود دارد، الگوریتم FGLM از پایه‌ی گروبنر داده شده تحت یک ترتیب یکجمله‌ای مثل σ یکجمله‌ای‌های ایده‌ال ترتیبی $\mathcal{O}_\tau$ نسبت به ترتیب یکجمله‌ای جدید τ را جمله به جمله محاسبه می‌کند، در حالی‌که الگوریتم تغییر پایه در فوق ایده‌ال ترتیبی $\mathcal{O}$ را بطور کامل از ورودی دریافت می‌کند.

الگوریتم تغییر پایه ۴، گرچه روشی سرراست برای محاسبه‌ی پایه‌ی مرزی تحت یک ایده‌ال ترتیبی پذیرفتنی، محسوب می‌شود ولی بدلیل محاسبه‌ی پایه‌ی گروبنر برای محاسبه‌ی فرم‌های نرمال و یا $\mathcal{O}_\sigma(I)$ ، تمام پیچیدگی‌های محاسبه‌ی پایه‌ی گروبنر را همراه دارد، که اصلاً مطلوب نیست، ضمن این‌که ایده‌ال ترتیبی به‌جای این‌که در الگوریتم محاسبه شود، یکی از ورودی‌ها است و ما تمایل داریم الگوریتمی طراحی کنیم که هم ایده‌ال ترتیبی و هم پایه‌ی مرزی متناظر با آن را محاسبه کند.

فرض کنید یک مجموعه‌ی مولد مثل $F = \{f_1, ..., f_m\}$ برای ایده‌ال صفربعدی I داده شده است، هدف ما این است که الگوریتمی داشته باشیم که با دریافت F یک ایده‌ال ترتیبی پذیرفتنی برای I مثل $\mathcal{O}$ و یک $\mathcal{O}$- پایه‌ی مرزی برای I تولید کند.

در سال ۱۹۹۹ مورِین الگوریتمی ارائه داد [۵۳]، که قادر بود به ازای یک مولد داده شده از یک ایده‌ال صفر بعدی، پایه‌ای موسوم به پایه‌ی مورین که مفهومی کلی‌تر از پایه‌ی مرزی است را برای ایده‌ال مورد نظر محاسبه کند. این الگوریتم برای محاسبه‌ی پایه‌ی مرزی طراحی نشده بود و

خروجی آن لزوماً یک پایه‌ی مرزی نبود. هفت سال بعد از آن یعنی سال ۲۰۰۶، مارتین کِروزِر [7] و آکِم کِرین [8] با الهام گرفتن از الگوریتم مورِین، الگوریتمی برای محاسبه‌ی پایه‌ی مرزی ایده‌ال های صفر بعدی ارائه دادند [۴۱]، که در ادامه قصد داریم الگوریتم آن‌ها را مورد بررسی قرار دهیم.

تعریف ۸۸.۲ (گسترش همسایگی). فرض کنید $P = K[x_1, ..., x_n]$ و $V \subseteq P$ یک زیر فضای برداری از P باشد. *گسترش همسایگی* V را به‌صورت زیر تعریف می‌کنیم.

$$V^+ := V + x_1V + \cdots + x_nV.$$

واضح است که V^+ نیز یک فضای برداری در P است. برای یک مجموعه‌ی متناهی از چندجمله‌ای‌ها نظیر $\mathcal{F} = \{f_1, ..., f\}$ نیز گسترش همسایگی، به‌صورت زیر تعریف می‌شود.

$$\mathcal{F}^+ := \mathcal{F} \cup x_1\mathcal{F} + \cup \cdots x_n\mathcal{F}.$$

نکته ۸۹.۲. فرض کنید $V \subseteq P = K[x_1, ..., x_n]$ یک زیرفضای برداری و $\mathcal{F}$ یک مجموعه از چندجمله‌ای‌ها باشد. اگر $V = \langle\mathcal{F}\rangle_K$ آن‌گاه، $\langle\mathcal{F}^+\rangle_K = \langle\mathcal{F}\rangle_K^+ = V^+$. زیرا عمل ضرب در یک متغیر نظیر x_i در فضای برداری P، یک تبدیل K- خطی است. در نتیجه برای گسترش همسایگی یک زیرفضای برداری مثل V کافی است همسایگی یک مجموعه‌ی مولد K- خطی آن را گسترش دهیم.

برای سادگی و فهم بهتر، ابتدا الگوریتم اصلی را به الگوریتم‌های کوچکتر تبدیل می‌کنیم و پس از این‌که هر یک از الگوریتم‌های کوچکتر را شرح دادیم، در نهایت الگوریتم اصلی برای محاسبه‌ی پایه‌ی مرزی را شرح خواهیم داد. ولی قبل از آن بهتر است ایده‌ی اصلی و نقشه راه را بدانیم. هدف ما محاسبه‌ی پایه‌ی مرزی یک ایده‌ال صفر بعدی است، بنابراین اولین گام یافتن یک ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال مورد نظر است. به همین دلیل ابتدا الگوریتم‌هایی را معرفی می‌کنیم که به ازای یک مجموعه‌ی مولد داده شده برای ایده‌ال مورد نظر، قادرند یک ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال تولید کنند. ایده‌ی

تعریف ۹۰.۲ (فضای برداری L ــ پایا). فرض کنید $F \subseteq L$ زیرفضاهای برداری $P = K[x_1, ..., x_n]$ باشند. در این‌صورت فضای برداری F را L ــ پایا گوییم هرگاه $F^+ \cap L = F$.

تعریف ۹۱.۲ (پوشش L- پایا). فرض کنید $F \subseteq L$ زیرفضاهای برداری P باشند. پوشش L ــ پایای F، عبارت است از کوچکترین فضای برداری شامل F مثل V به‌طوری که V، L- پایا باشد، یعنی $V^+ \cap L = V$.

[7] Martin Kreuzer
[8] Achim Kehrein

گاهی اوقات یک فضای برداری L- پایا را فضای برداری پایا نسبت به L و به همین ترتیب پوشش L ـ پایای یک فضای برداری را پوشش پایای آن فضا نسبت به فضای L نیز می‌گوییم.

مثال ۹۲.۲. در زیر مثال‌هایی از فضاهای پایا نسبت به یک فضای برداری، آمده است.

۱. L به ازای هر زیرفضای P مثل L ، خود L یک فضای برداری L- پایا است.

۲. فرض کنید $S = \{۱, x, y, x^۲y^۲, y^۳\} \subseteq P[x, y]$ و قرار دهید، $. = \langle S\rangle_K$ در این‌صورت $\langle x+y, x^۲, y^۲, y^۳\rangle_K$ یک فضای L ـ پایا است ولی $\langle x+y, x^۲\rangle_K$ چنین نیست.

یک روش سرراست برای ساختن یک پوشش L ـ پایا برای یک زیرفضای برداری نظیر F به‌صورت زیر است:

$$F_۰ := F \ \wedge\ \forall k \in \mathbb{Z}_{\geq ۰}:\ F_{k+۱} = F_k^+ \cap L$$

در این صورت $F_L = \bigcup_{k\geq ۰} F_K$ یک پوشش L ـ پایا برای F خواهد بود. در حالتی که $\mathcal{F} :=$ $\{f_۱, ..., f_s\}$ یک مجموعه‌ی متناهی از چندجمله‌ای‌ها باشد، آن‌گاه $F = \langle \mathcal{F}\rangle_K$ ، $\boldsymbol{K}$- فضای خطی تولید شده توسط $\mathcal{F}$ است که یک زیرفضای P است، در این حالت پوشش L ـ پایای F را با $\mathcal{F}_L$ هم نمایش می‌دهیم.

در حالت خاصی که $P = L$ باشد آن‌گاه، پوشش L ـ پایای $\langle \mathcal{F}\rangle_K$ همان ایده‌ال تولید شده توسط چندجمله‌ای‌های $\mathcal{F}$ خواهد بود، یعنی $F_P = \langle f_۱, ..., f_r\rangle$. البته ما برای محاسبات واقعی به همیشه L را به‌جای یک زیرفضای P با بعد متناهی در نظر می‌گیرم. در ادامه خواهید دید که الگوریتم‌های محاسبه‌ی پایه‌ی مرزی طوری طراحی شده‌اند که تمام محاسبات در فضای با بعد متناهی L انجام می‌شود و هیچ‌گاه عضوی خارج از فضای L در محاسبات ظاهر نخواهد شد، به همین خاطر به L فضای محاسباتی هم می‌گویند. در ادامه با برخی از ویژگی‌های اولیه‌ی پوشش‌های پایا آشنا خواهیم شد.

لم ۹۳.۲. فرض کنید $F \subseteq G \subseteq\subseteq L$ زیرفضاهایی از فضای برداری $P = \boldsymbol{K}[x_۱, ..., x_n]$ باشند. در این‌صورت داریم:

$$F \subseteq F_L,\ F_L = (F_L)_L,\ F_L \subseteq G_L,\ F_U \subseteq F_L,\ F_L = (F_U)_L.$$

برهان. به لم ۱۱ از [۴۱] ، رجوع کنید. □

ویژگی آخر از لم ۹۳.۲ ، یعنی $F_L = (F_U)_L$ نشان می‌دهد که به‌جای محاسبه‌ی پوشش U- پایا از همان ابتدا می‌توانیم پوشش L ـ پایا F را محاسبه کنیم.

یکی از مراحل الگوریتم اصلی برای محاسبه‌ی پایه‌ی مرزی، توسیع پایه‌ی یک فضای برداری است. فرض کنید $\mathcal{V}$ پایه‌ی فضای برداری $\langle \mathcal{V}\rangle_K \in P$ و $\mathcal{G}$ مجموعه‌ای از چندجمله‌ای‌ها باشد و

بخواهیم این پایه را به پایه‌ای برای فضای $\langle \mathcal{V} \cup \mathcal{G} \rangle_K$ گسترش دهیم، برای این‌کار می‌توانیم از الگوریتم حذف گاوس که جزئیات آن در لم بعد توضیح داده شده استفاده کنیم.

لم ۹۴.۲ (الگوریتم حذف گاوس در K- فضای برداری $K[x_1, \dots, x_n]$). فرض کنید σ یک ترتیب یکجمله‌ای و $\mathcal{V} = \{v_1, \dots, v_r\} \subseteq P \backslash \{0\}$ مجموعه‌ای متناهی از چندجمله‌ای‌های باشد به‌طوری که، ضریب پیشرو همه‌ی چندجمله‌ای‌های آن ۱ باشد و هر دو چندجمله‌ای از آن دارای یکجمله‌ای‌های پیشرو متمایز باشند. فرض کنید $\mathcal{G} = \{g_1, \dots, g_s\} \subseteq P$. الگوریتم GaussEL مجموعه‌ی $\mathcal{W} \subseteq P$ را طوری محاسبه می‌کند که ضریب پیشرو همه‌ی چندجمله‌ای‌های آن ۱ است و هر دو چندجمله‌ای از $\mathcal{V} \cup \mathcal{W}$ دارای یکجمله‌ای پیشرو متمایز هستند و در ضمن $\langle \mathcal{V} \cup \mathcal{W} \rangle_K = \langle \mathcal{V} \cup \mathcal{G} \rangle_K$. (مجموعه‌ی $\mathcal{V}$ یا $\mathcal{W}$ می‌تواند تهی باشد.)

الگوریتم ۵ الگوریتم حذف گاوس برای چندجمله‌ای‌ها ـ GaussEL

ورودی $\mathcal{V}$ و $\mathcal{G}$ به صورتی که در لم شرح داده شده.

خروجی $\mathcal{W}$ به‌صورتی که در لم شرح داده شده.

۱: قرار دهید $\mathcal{H} := \mathcal{G}$ و $\eta := 0$

۲: اگر $\mathcal{H} = \emptyset$ آن‌گاه $\mathcal{W} := \{v_{r+1}, \dots, v_{r+\eta}\}$ و الگوریتم متوقف می‌شود.

۳: $f \in \mathcal{H}$ را انتخاب کرده و آن را از $\mathcal{H}$ حذف می‌کنیم و قرار می‌دهیم $i := 1$.

۴: اگر $f = 0$ یا $i > r + \eta$ آن‌گاه به گام ۷ می‌رویم.

۵: اگر $\mathrm{LM}_\sigma(f) = \mathrm{LM}_\sigma(v_i)$ آن‌گاه f را با $f - \mathrm{LM}_\sigma(f) \cdot v_i$ جایگزین کرده و قرار می‌دهیم، $i := 1$ و به گام ۴ می‌رویم.

۶: مقدار i را یک‌واحد افزایش داده و به گام ۴ می‌رویم.

۷: اگر $f \neq 0$ آن‌گاه η را یک‌واحد افزایش داده و قرار می‌دهیم $v_{r+\eta} := \frac{f}{\mathrm{LM}_\sigma(f)}$ و با گام ۲ ادامه می‌دهیم.

برهان. به لم ۱۲ [۴۱] ، رجوع کنید. □

دلیل این‌که در الگوریتم GaussEL شرط کردیم که اعضای پایه‌ی $\mathcal{V}$ دوبه‌دو دارای یکجمله‌ای پیشرو متمایز باشند این است که اگر $\mathcal{V} = \{v_1, \dots, v_r\}$ پایه‌ای برای زیرفضای $V \subseteq P$ با خصوصیت ذکر شده باشد، آن‌گاه $\mathrm{LM}_\sigma(V) := \{\mathrm{LM}_\sigma(v) | \, v \in V \backslash \{0\}\}$ با $\mathrm{LM}_\sigma(\mathcal{V}) := \{\mathrm{LM}_\sigma(v_1), \dots, \mathrm{LM}_\sigma(v_r)\}$ برابر خواهد شد.

تعریف ۹۵.۲ (ترتیب یکجمله‌ای سازگار با درجه). ترتیب یکجمله‌ای σ روی $\mathbb{T}^n$ را *سازگار با درجه* گوییم هرگاه به ازای هر $t_1, t_2 \in \mathbb{T}^n$ ، اگر $t_1 \geq_\sigma t_2$ آن‌گاه $\deg(t_1) \geq \deg(t_2)$.

برای مثال ترتیب الفبایی مدرج و الفبایی مدرج معکوس ترتیب‌های یکجمله‌ای سازگار با درجه هستند.

با استفاده از الگوریتم GaussEL، در گزاره‌ی بعدی الگوریتمی برای محاسبه‌ی پوشش L ـ پایای یک فضای برداری وقتی $L = \langle \mathbb{T}^n_{\leq d} \rangle_K$ است ارائه می‌کنیم.

گزاره ۹۶.۲ (محاسبه‌ی پوشش پایا). فرض کنید $P \supseteq \{f_۱, ..., f_r\} =: \mathcal{F}$ و $L := \langle \mathbb{T}^n_{\leq d} \rangle_K$ به‌طوری که $f_۱, ..., f_r \in L$ ، یعنی داشته باشیم $d \geq \max\{\deg(f_i) | ۱ \leq i \leq r\}$. فرض کنید σ یک ترتیب یکجمله‌ای سازگار با درجه باشد، در این‌صورت الگوریتم ۶ پایه‌ی پوشش L پایای $\mathcal{F}$ یعنی پایه‌ی $\mathcal{F}_L$ را محاسبه می‌کند. علاوه بر این چندجمله‌ای‌های پایه‌ی محاسبه شده دارای یکجمله‌ای‌های پیشرو متمایز هستند.

الگوریتم ۶ الگوریتم محاسبه‌ی پوشش پایای یک فضای برداری

Input $\mathcal{F} = \{f_1, ..., f_r\}, \ L = \langle \mathbb{T}^n_{\leq d} \rangle_K$.
Output $\mathcal{V}$ که پایه‌ای برای فضای برداری $\mathcal{F}_L$ است
$\mathcal{V} \leftarrow GaussEL(\emptyset, \mathcal{F})$
repeat
$\mathcal{W}' \leftarrow GaussEL(\mathcal{V}, \mathcal{V}^+ \backslash \mathcal{V})$
$\mathcal{W} \leftarrow \{w \in \mathcal{W}' | \ \deg(w) \leq d\} = \{w \in \mathcal{W}' | \ \mathrm{Supp}(w) \subseteq L\}$
until $\mathcal{W} \neq \emptyset$
return $\mathcal{V}$

برهان. به گزاره‌ی ۱۳ در [۴۱] ، مراجعه کنید. □

نتیجه ۹۷.۲. طبق تعریف، پوشش L ـ پایای $\mathcal{F}$ یعنی $\mathcal{F}_L$ مشمول در فضای برداری L است. از طرفی بدیهی است که $\mathcal{F}_L$ مشمول در ایده‌ال تولید شده به‌وسیله‌ی $\mathcal{F}$ نیز قرار دارد در نتیجه $\mathcal{F}_L \subseteq L \cap \langle \mathcal{F} \rangle$.

مثال ۹۸.۲. فرض کنید $\mathcal{F} = \{f_۱, f_۲, f_۳\}$ که $f_۱ = x^۲y^۲ + ۱, f_۲ = x^۴$ و $f_۳ = y^۴$. همچنین فرض کنید $\mathcal{H} = \{۱\}$. هر دو مجموعه ایده‌ال $\langle ۱ \rangle$ را تولید می‌کنند زیرا $۱ = f_۲ \cdot f_۳ - f_۱^۲ + ۲f_۱$.

۱. فرض کنید $L = \langle \mathbb{T}^n_{\leq ۴} \rangle_K$. در این‌صورت $\mathcal{F}_L = \langle \mathcal{F} \rangle$ و $\dim_K(\mathcal{F}_L) = ۳$، در حالی که $\mathcal{H} = L$ و $\dim_K(\mathcal{H}_L) = ۱۰$.

۲. اگر $L = \langle \mathbb{T}^n_{\leq ۵} \rangle_K$، آن‌گاه $\mathcal{F}_L = L = \mathcal{H}_L$.

اولین هدف ما یافتن یک ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال داده شده است و گزاره‌ی بعدی نشان می‌دهد که پوشش پایایی که در الگوریتم ۶ محاسبه می‌شود، حاوی اطلاعاتی در مورد کاندیدهای ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال تولید شده توسط چندجمله‌ای‌های مجموعه‌ی $\mathcal{F}$ است.

گزاره ۹۹.۲. فرض کنید $\mathcal{F} = \{f_1, ..., f_s\} \subseteq P$ و $L = \langle \mathbb{T}^n_{\leq d} \rangle_K$ به‌طوری که $\mathcal{F} \subseteq L$. در این‌صورت یک ایده‌ال ترتیبی مثل $\mathcal{O}$ وجود دارد به‌طوری که،

$$L = \mathcal{F}_L \oplus \langle \mathcal{O} \rangle_K$$

. یعنی، اگر σ یک ترتیب یکجمله‌ای سازگار با درجه و $\mathcal{V} = \{v_1, ..., v_r\}$ پایه‌ی محاسبه شده برای $\mathcal{F}_L$ توسط الگوریتم ۶ باشد، آن‌گاه چندجمله‌ای‌های $\mathcal{V}$ دوبه‌دو دارای یکجمله‌ای‌های پیشرو متمایز بوده و $\mathrm{LM}_\sigma(\mathcal{F}_L)$ تحت مضارب اعضای خود در L بسته است. یعنی اگر به ازای یک $v \in \mathcal{F}_L$ و یک یکجمله‌ای $t \in \mathbb{T}^n_{\leq d}$ داشته باشیم، $t \mid \mathrm{LM}_\sigma(v)$ ، آن‌گاه یک $w \in \mathcal{F}_L$ وجود دارد به‌طوری که $t = \mathrm{LM}_\sigma(w)$. بطور معادل $\mathcal{O} = \mathbb{T}^n_{\leq d} \backslash \{\mathrm{LM}_\sigma(v_1), ..., \mathrm{LM}_\sigma(v_r)\}$ یک ایده‌ال ترتیبی است که در رابطه‌ی جمع مستقیم فوق نیز صدق می‌کند.

برهان. به گزاره‌ی ۱۵ [۴۱] ، مراجعه کنید. $\square$

گزاره‌ی بعدی به عنوان شرط توقف در الگوریتم محاسبه‌ی پایه‌ی مرزی که در ادامه ارائه می‌کنیم به‌کار خواهد رفت. در واقع توسط گزاره‌ی بعدی می‌توان پی‌برد که کدام‌یک از کاندیدهای به‌دست آمده برای ایده‌ال ترتیبی ، می‌توانند ایده‌ال ترتیبی پذیرفتنی برای ایده‌ال داده شده باشند.

گزاره ۱۰۰.۲. فرض کنید L یک زیرفضای برداری P و $\tilde{I}$ یک زیرفضای برداری ایده‌ال صفر بعدی $I \subseteq P$ باشد، به‌طوری که $\tilde{I}^+ \cap L = \tilde{I}$ و $\langle \tilde{I} \rangle = I$ ، به‌عبارت دیگر $\tilde{I}$ یک فضای L- پایا باشد که ایده‌ال تولید شده توسط آن برابر I است. علاوه بر این فرض کنید $\mathcal{O}$- یک ایده‌ال ترتیبی باشد که در رابطه‌ی $L = \tilde{I} \oplus \langle \mathcal{O} \rangle_K,$ صدق می‌کند. در این‌صورت اگر $\partial\mathcal{O} \subseteq L$ آن‌گاه $\mathcal{O}$ یک ایده‌ال ترتیبی پذیرفتنی برای I خواهد بود.

برهان. به ازای هر یکجمله‌ای مرزی $b_j \in \partial\mathcal{O} \subseteq L$ بر اساس رابطه‌ی جمع مستقیم فوق، چندجمله‌ای $g_j \in \tilde{I}$ وجود دارد به‌طوری که

$$b_j = g_j + \sum_{i=1}^{m} \alpha_{ij} t_i \in \tilde{I} \oplus \langle \mathcal{O} \rangle_K.$$

به این ترتیب می‌توان مجموعه‌ی $G = \{g_1, ..., g_\nu\}$ را به‌دست آورد که یک $\mathcal{O}$- پیش‌پایه‌ی مرزی است.

فرض کنید g_k و g_l دو همسایه‌ی دلخواه از پیش‌پایه‌ی G باشند. در این‌صورت یکجمله‌ای‌های S- چندجمله‌ای آن‌ها یعنی $S(g_k, g_l) \in \tilde{I}^+$ در $\mathcal{O}^+$ قرار می‌گیرند. بنابراین ضرایب $c_j \in K$ وجود دارند به‌طوری که، یکجمله‌ای‌های عبارت $h := S(g_k, g_l) - \sum_{j=1}^{\nu} c_j g_j$ در $\mathcal{O}$ قرار می‌گیرند. در نتیجه $h \in \tilde{I}^+ \cap \langle \mathcal{O} \rangle_K = \tilde{I}^+ \cap L \cap \langle \mathcal{O} \rangle_K = \{0\}$. در نهایت طبق محک بوخبرگر برای پایه‌ی مرزی ۸۶.۲ ، G یک $\mathcal{O}$- پایه‌ی مرزی خواهد بود. $\square$

دقت کنید، اگر در قضیه‌ی فوق جای به‌جای $\tilde{I}$، ایده‌ال I و به‌جای L کل فضای P را قرار دهیم، به تعریف پایه‌ی مرزی خواهیم رسید.

یک جمع بندی از مباحث قبل نشان می‌دهد که می‌توانیم در ابتدا با استفاده از الگوریتم ۶ پوشش پایای فضای K- خطی تولید شده توسط مجموعه‌ی مولد داده شده را نسبت به یک فضای برداری مثل $\langle \mathbb{T}^n_{\leq d} \rangle_K$ به‌دست آوریم. سپس این مجموعه کاندیدهایی را برای ایده‌ال ترتیبی پذیرفتنی معرفی می‌کند که با استفاده از گزاره‌ی ۱۰۰.۲ می‌توانیم، کاندیدی که یک ایده‌ال ترتیبی پذیرفتنی است تشخیص دهیم. بعد از این مراحل یک ایده‌ال ترتیبی پذیرفتنی به‌دست می‌آید و به یک الگوریتم نیاز داریم که پایه‌ی مرزی ایده‌ال نسبت به ایده‌ال ترتیبی به‌دست آمده را استخراج کند. این الگوریتم در گزاره‌ی بعدی ارائه شده است.

گزاره ۱۰۱.۲. فرض کنید $\mathcal{F} = \{f_1, ..., f_s\} \subseteq P$ یک مجموعه‌ي مولد داده شده برای ایده‌ال صفربعدی I باشد. فرض کنید σ یک ترتیب یکجمله‌ای سازگار با درجه و L یک ایده‌ال ترتیبی (برای مثال $L = \mathbb{T}^n_{\leq d}$) باشد. در ضمن فرض کنید $\mathcal{V}$ یک پایه برای پوشش L پایای $\mathcal{F}$ یعنی فضای برداری $\mathcal{F}_L$ باشد که یکجمله‌ای‌های پیشرو هر دو عضو آن متمایز باشند و $\mathcal{O} = L \backslash \mathrm{LM}_\sigma(\mathcal{V})$ به‌طوری که

$$L = \mathcal{F}_L \oplus \langle \mathcal{O} \rangle_K \ \wedge \ \partial\mathcal{O} \subseteq L.$$

در این‌صورت الگوریتم ۷، $\mathcal{O}_\sigma(I)$- پایه‌ی مرزی I را محاسبه می‌کند.

الگوریتم ۷ الگوریتم تحویل نهایی– Final Reduction

ورودی $\mathcal{F}, L$ و σ و $\mathcal{O}, \mathcal{V}$ با ویژگی‌های ذکر شده در لم.

خروجی $\mathcal{O}_\sigma(I)$ – پایه‌ی مرزی I.

۱: $\mathcal{V}_R = \emptyset$

۲: اگر $\mathcal{V} = \emptyset$ آن‌گاه به گام ۸ می‌رویم.

۳: عضو $v \in \mathcal{V}$ را برابر با عضوی از $\mathcal{V}$ که دارای یکجمله‌ای پیشرو مینیمال است قرار داده و آن‌را از $\mathcal{V}$ حذف می‌کنیم.

۴: $\mathcal{H} := \mathrm{Supp}(v) \backslash (\{\mathrm{LM}_\sigma(v)\} \cup \mathcal{O})$

۵: اگر $\mathcal{H} = \emptyset$ آن‌گاه $\frac{v}{\mathrm{LC}_\sigma(v)}$ را به $\mathcal{V}_R$ ضمیمه کرده و به گام ۲ می‌رویم.

۶: برای هر $h \in \mathcal{H}$، $w_h \in \mathcal{V}_R$ و $c_h \in K$ را طوری محاسبه می‌کنیم که $h = \mathrm{LM}_\sigma(w_h)$ و

۷: $h \notin \mathrm{Supp}(v - c_h \cdot w_h)$.

۸: v را با مقدار جدید $v - \sum_h c_h \cdot w_h$ جایگزین کرده و $\frac{v}{\mathrm{LC}_\sigma(v)}$ را به $\mathcal{V}_R$ ضمیمه می‌کنیم، سپس به گام ۲ می‌رویم.

۹: فرض کنید $\partial\mathcal{O} = \{b_1, ..., b_\nu\}$. به ازای هر $b_j \in \partial\mathcal{O}$ چندجمله‌ای $g_j \in \mathcal{V}_R$ را طوری انتخاب می‌کنیم که $b_j = \mathrm{LM}_\sigma(g_j)$، در این‌صورت، خروجی عبارت‌است از $g_1, ..., g_\nu$.

برهان. به گزاره‌ی ۱۷ از [۴۱] ، مراجعه کنید. □

الگوریتم ۸ الگوریتم محاسبه‌ی پایه مرزی‌ــBBA

ورودی $\mathcal{F}, \sigma$ آن‌گونه که در لم گفته شده.
خروجی $\mathcal{O}_\sigma(I)$ ــ پایه‌ی مرزی برای I.
۱: $d := \max\{\deg(f_i) \mid 1 \le i \le s\},\ L := \langle \mathbb{T}^n_{\le d} \rangle_K$
۲: پایه‌ی $\mathcal{V} = \{v_1, \ldots, v_r\}$ را برای $\langle \mathcal{F} \rangle_K$ محاسبه می‌کنیم طوری که یکجمله‌ای‌های پیشرو اعضایش متمایز باشند.
۳: پایه‌ی توسیع یافته $\mathcal{W}' := \{v'_{r+1}, \ldots, v'_{r+\eta'}\}$ برای $\langle \mathcal{V} \rangle_K \supseteq \langle \mathcal{V}^+ \rangle_K$ را طوری محاسبه می‌کنیم که یکجمله‌ای‌های پیشرو اعضای $\mathcal{V} \cup \mathcal{W}'$ متمایز باشند.
۴: $\mathcal{W} = \{v_{r+1}, \ldots, v_{r+\eta}\} = \{v \in \mathcal{W}' \mid \deg(v) \le d\}.$
۵: اگر $\eta > 0$ آن‌گاه $\mathcal{V}$ را با $\mathcal{V} \cup \mathcal{W}$ جایگزین کرده، قرار می‌دهیم $r = r + \eta$ و سپس به گام ۳ می‌رویم.
۶: $\mathcal{O} := \mathbb{T}^n_{\le d} \backslash \{\mathrm{LM}_\sigma(v_1), \ldots, \mathrm{LM}_\sigma(v_r)\}$.
۷: اگر $\partial\mathcal{O} \nsubseteq L$ ، d را یک‌واحد افزایش داده و قرار می‌دهیم $L := \langle \mathbb{T}^n_{\le d} \rangle_K$ و سپس به گام ۳ می‌رویم.
۸: با فراخوانی الگوریتم تحویل نهایی [۷] چندجمله‌ای‌های $g_1, \ldots, g_\nu$ را که این الگوریتم محاسبه می‌کند به عنوان خروجی نهایی الگوریتم در نظر می‌گیریم.

اکنون با کنار هم قرار دادن الگوریتم‌های قبلی الگوریتمی برای محاسبه‌ی پایه‌ی مرزی یک ایده‌ال صفر بعدی به ازای یک مولد داده شده، ارائه می‌دهیم.

گزاره ۱۰.۲.۲ (الگوریم محاسبه‌ی پایه‌ی مرزی). فرض کنید $\mathcal{F} = \{f_1, \ldots, f_s\} \subseteq P$ مولدی برای ایده‌ال صفربعدی $I = \langle \mathcal{F} \rangle$ باشد. فرض کنید σ یک ترتیب یکجمله‌ای سازگار با درجه باشد. در این‌صورت الگوریتم ۸، $\mathcal{O}_\sigma(I)$ ــ پایه‌ی مرزی I را محاسبه می‌کند.

برهان. رجوع کنید به [۴۵]. □

این بحث را در این‌جا خاتمه می‌دهیم. در فصل ۴ که روش‌های حل دستگاه چندجمله‌ای را معرفی می‌کنیم، الگوریتمی بهینه‌تر برای محاسبه‌ی پایه‌ی مرزی ارائه می‌دهیم که برای حل دستگاه‌های به‌دست آمده از سامانه‌های رمزنگاری مناسب‌تر است.

فصل ۳

جبری‌سازی و استخراج معادلات سامانه‌های رمزنگاری

در این فصل ثابت می‌کنیم که نه تنها نگاشت‌های رمزنگاری بلکه هر نگاشت از یک مجموعه‌ی متناهی به یک مجموعه‌ی متناهی دیگر را می‌توان به صورت یک نگاشت چندجمله‌ای روی یک میدان متناهی بیان کرد، به این ترتیب شکستن یک سامانه‌ی رمزنگاری به مسئله‌ی حلّ دستگاه معادلات چندجمله‌ای تبدیل می‌شود. پس از تعریف دقیق حمله جبری با سناریوهای مختلف حمله‌ی جبری روی انواع سامانه‌های رمزنگاری آشنا می‌شویم و در انتها مثال‌هایی واقعی از تبدیل سامانه‌های رمزنگاری به دستگاه معادلات چندجمله‌ای روی میدان متناهی را خواهیم دید. بخش زیادی از مباحث این فصل برگرفته از [۷]، است که یکی از مراجع مفید در زمینه حمله‌های جبری محسوب می‌شود.

۱.۳ نگاشت‌های عام

هدف از این بخش این است که ثابت کنیم، هر نگاشت رمزنگاری را می‌توان به صورت نگاشتی چندجمله‌ای روی یک میدان متناهی بیان کرد، امّا نحوه به‌دست آوردن آن موضوعی است که در بخش‌های بعدی بررسی می‌شود.

لم ۱.۳. فرض کنید $\mathbb{F}$ یک میدان متناهی از مرتبه‌ی عدد اوّل p باشد، در این صورت هر نگاشت از $\mathbb{F}$ ـ فضای برداری متناهی بعد V به میدان $\mathbb{F}$ را می توان به صورت یک نگاشت چندجمله‌ای با ضرایب در $\mathbb{F}$ نوشت.

برهان. می‌دانیم مجموعه‌ی همه‌ی نگاشت‌ها از V به $\mathbb{F}$ یک $\mathbb{F}$ ـ فضای برداری است، که آن را با Φ نمایش می‌دهیم. برای اثبات حکم، پایه‌ای برای این فضای برداری ارائه می‌دهیم که همه‌ی

اعضایش چندجمله‌ای‌هایی با ضرایب در $\mathbb{F}$ هستند. مجموعه‌ی $B := \{\varphi_x : V \to \mathbb{F} \mid x \in V\}$ که برای هر $x \in V$ نگاشت $\varphi_x : V \to \mathbb{F}$ را به صورت زیر تعریف می‌کنیم:

$$\varphi_x(y) := \begin{cases} ۱ & y = x \\ ۰ & y \neq x \end{cases}$$

در نظر بگیرید. نگاشت دلخواه $\varphi \in \Phi$ را می‌توان به‌صورت زیر نوشت:

$$\varphi = \sum_{x \in V} \varphi(x) \cdot \varphi_x$$

بنابراین B مولدی برای V است، از طرفی به ازای $x_۰ \in V$ دلخواه، $B \backslash \{\varphi_{x_۰}\}$ یک مولّد Φ نیست، زیرا $\varphi_{x_۰}$ را نمی‌توان به صورت ترکیب $\mathbb{F}$ خطّی از اعضای $B \backslash \{x_۰\}$ نوشت. بنابراین B یک مولّد مینیمال و لذا یک پایه $\mathbb{F}$ ـ فضای برداری Φ است. چون میدان $\mathbb{F}$ و در نتیجه فضای برداری V متناهی است، کافی است نشان دهیم هر یک از φ_x ها یک $\mathbb{F}$ ـ چندجمله‌ای است. بُعد V برابر با n و $x = (x_۱, ..., x_n) \in V$ را دلخواه و از این پس ثابت فرض کنید. $\mathbb{F}$ یک میدان است، بنابراین $(x_i - y_i)^{p-۱} - ۱ \neq ۰$ اگر و تنها اگر $x_i = y_i$. در نتیجه چندجمله‌ای زیر همه‌جا به جز در x برابر صفر است:

$$\forall\ y = (y_۱, ..., y_n) \in V :\ \psi_x(y_۱, ..., y_n) := \Pi_{i=۱}^{n}((x_i - y_i)^{p-۱} - ۱)$$

با توجّه به تعریف ψ_x داریم:

$$\forall y = (y_۱, ..., y_n) \in V :\ \varphi_x(y) = \frac{\psi_x(y_۱, ..., y_n)}{\psi_x(x_۱, ..., x_n)}.$$

چون $x \in V$ ثابت فرض شده بود مخرج نگاشت فوق یک مقدار ثابت و لذا φ_x یک چندجمله‌ای n متغیره بر حسب $y_۱, ..., y_n$ است و در نهایت چون $x \in V$ را دلخواه انتخاب کرده بودیم حکم ثابت می‌شود. □

لم ۲.۳. اگر $\mathbb{F}$ یک میدان متناهی با مشخصه‌ی عدد اوّل p باشد، هر نگاشت از یک $\mathbb{F}$- فضای برداری متناهی بعد مثل V به $\mathbb{GF}(p)$ را می‌توان به صورت چندجمله‌ای با ضرایب در $\mathbb{GF}(p)$ نوشت.

برهان. $\mathbb{F}$ یک میدان متناهی با مشخصه‌ی p است، لذا بدون کاستن از کلیت فرض می‌کنیم $\mathbb{F} = \mathbb{GF}(p^n)$ که $n \in \mathbb{N}$. بنابراین $\mathbb{F}$ یک فضای برداری از بعد n روی $\mathbb{GF}(p)$ نیز خواهد بود که ضرب اسکالر آن به صورت زیر تعریف می‌شود.

$$\cdot : \mathbb{GF}(p) \to \mathbb{F}$$

$$(k, x) \mapsto k.x = x + x + \cdots + x \ (k \text{بار})$$

بُعد V را برابر عدد طبیعی m در نظر می‌گیریم. در نتیجه V یک فضای برداری $m \cdot n$ بعدی روی میدان $\mathbb{GF}(p)$ است و طبق لم قبل حکم ثابت می‌شود. □

لم ۳.۳. اگر $\mathbb{F}$ یک میدان متناهی با مشخصه‌ی عدد اوّل p باشد، هر نگاشت از $\mathbb{F}$- فضای برداری متناهی بُعد V به $\mathbb{GF}(p)$- فضای برداری متناهی بُعد U یک نگاشت چندجمله‌ای با ضرایب در $\mathbb{GF}(p)$ است، یعنی با فرض $dim(V) = n$ و $dim(U) = m$ برای هر نگاشت $f : V \to U$ داریم

$$\exists\ f_۱, ..., f_m \in \mathbb{GF}(p)[x_۱, ..., x_n] \ \ f(x_۱, ..., x_n) = (f_۱(x_۱, ..., x_n), ..., f_m(x_۱, ..., x_n)).$$

که $f_i \in \mathbb{GF}(p)[x_۱, ..., x_n]$ برای هر $i = ۱, ..., m$.

برهان. چون V یک $\mathbb{F}$- فضای برداری با بعد n و $\mathbb{F}$ یک میدان متناهی با مشخصه‌ی عدد اوّل p است، طبق لم قبل هر یک از توابع مؤلفه‌ای f_i که $i = ۱, ..., m$ یک چندجمله‌ای است و حکم ثابت می‌شود. □

لم ۴.۳. فرض کنید $\mathbb{F}$ و $\mathbb{G}$ دو میدان متناهی با مشخصه‌ی عدد اوّل p باشند، در این‌صورت هر نگاشت از $\mathbb{F}$- فضای برداری متناهی بُعد V به $\mathbb{G}$- فضای برداری متناهی بُعد U یک نگاشت چندجمله‌ای است.

برهان. بدون کاستن از کلیت فرض می‌کنیم $\mathbb{G} = \mathbb{G}(p^m)$ و $U = \mathbb{GF}(p^m)^n$ که m و n اعدادی طبیعی هستند. در نتیجه U یک فضای برداری $m \cdot n$ بعدی روی $\mathbb{GF}(p)$ است و طبق لم قبل حکم ثابت می‌شود. □

قضیه ۵.۳ (نگاشت عام). هر نگاشت از مجموعه‌ی متناهی S به مجموعه‌ی متناهی T را می‌توان به‌صورت یک نگاشت چند جمله‌ای روی $\mathbb{GF}(p)$ که p یک عدد اوّل دلخواه است نوشت.

برهان. p را یک عدد اوّل دلخواه در نظر بگیرید. اعداد طبیعی m و n را طوری در نظر می‌گیریم که $|S| \leq p^m$ و $|T| \leq p^n$. سپس S را در $\mathbb{GF}(p^n)$ می‌نشانیم یا به بیان ساده‌تر اعضای S را با اعضای $\mathbb{GF}(p^n)$ برچسب‌گذاری می‌کنیم به همین ترتیب T را در $\mathbb{GF}(p^m)$ می‌نشانیم. اکنون می‌توانیم هر نگاشت دلخواه مثل $f : S \to T$ را به صورت زیر به یک نگاشت روی $\mathbb{GF}(p^n)$ توسعه می‌دهیم.

$$F : \mathbb{GF}(p^n) \to \mathbb{GF}(p^m)$$

$$F(x) = \begin{cases} f(x) & x \in S \\ y_۰ & o.w \end{cases}$$

که $y_\circ \in \mathbb{GF}(p^m)$ یک عضو دلخواه و ثابت است. طبق لم ۴.۳ نگاشت فوق چندجمله‌ای است و در نتیجه تحدید آن به S (یا مجموعه‌ی متناظر با آن در $\mathbb{GF}(p^n)$) که آن را با $F|_S = f$ نمایش می‌دهیم نیز چندجمله‌ای است. □

قضیه‌ی فوق این اطمینان را می‌دهد که هر نگاشت (رمزنگاری) را می‌توان به‌صورت چندجمله‌ای بیان کرد ولی روش به‌دست آوردن این نمایش را در اختیار ما قرار نمی‌دهد. در ادامه پس از معرفی حمله‌های جبری الگوریتمی را معرفی می‌کنیم که می‌تواند در تبدیل نگشات‌های رمزنگاری به نگاشت چندجمله‌ای مورد استفاده قرار گیرد.

۲.۳ حمله‌ی جبری چیست؟

پروتکل‌های رمزنگاری که هسته‌ی اصلی تشکیل‌دهنده آن‌ها سامانه‌های رمزنگاری هستند نقشی حیاتی در زندگی نوین ایفا می‌کنند. امنیت این پروتکل‌ها متّکی به امنیت سامانه‌های رمزنگاری سازنده‌ی آن‌هاست، امّا خود این سامانه‌ها از الگوریتم‌هایی تشکیل می‌شوند که در حالت کلّی نگاشتی از فضای متن اصلی و کلید به فضای متن رمز شده هستند. در بخش قبل ثابت کردیم که هر نگاشت رمزنگاری را می‌توان به یک نگاشت چندجمله‌ای روی یک میدان متناهی تبدیل کرد. در نتیجه مسأله‌ی شکستن یک سامانه‌ی رمزنگاری به مسأله‌ی حلّ دستگاه معادلات چندجمله‌ای چندمتغیره روی میدان متناهی تبدیل می‌شود. چنین رویکردی برای شکستن سامانه‌های رمزنگاری را، **حمله‌های جبری** می‌نامیم.

هدف حمله‌ی جبری می‌تواند به‌دست آوردن کلید یا متن اصلی باشد و از سه مرحله‌ی اصلی زیر تشکیل می‌شود.

۱. **جبری‌سازی سامانه رمزنگاری:** مدل کردن نگاشت‌های رمزنگاری مورد استفاده در سامانه‌ی مورد نظر به صورت نگاشت‌های چندجمله‌ای روی یک میدان متناهی. این مرحله قابلیت پیش‌پردازش دارد، یعنی می‌توان یک‌بار آن را انجام داد و در حملات بعدی به دفعات از آن استفاده کرد.

۲. **جایگذاری مقادیر معلوم:** مقادیر معلوم حاصل از اطلاعات به‌دست آمده، نظیر شنود یک پیام رمز شده یا داشتن یک یا چند زوج متن اصلی و رمزشده‌ی متناظر با آن‌را، در چندجمله‌ای‌های به‌دست آمده از مرحله‌ی قبل جایگذاری می‌کنیم تا به یک دستگاه معادلات چندجمله‌ای روی میدان متناهی برسیم.

۳. **حلّ دستگاه معادلات** با حل دستگاه چندجمله‌ای، متغیرهای مجهول که می‌توانند بیانگر کلید سامانه یا متن اصلی متناظر با یک متن رمزشده باشند را به‌دست می‌آوریم.

در ادامه فضای متن اصلی $\mathcal{M}$ و فضای متن رمزشده $\mathcal{C}$ را به عنوان فضاهای برداری متناهی بعد روی یک میدان متناهی (معمولا با مشخصه‌ی ۲) در نظر می گیریم که فرض خوبی است و تقریباً همه‌ی سامانه‌های عملی را می‌توان این‌گونه مدل کرد. میدان‌هایی که در رمزنگاری با آن‌ها کار می‌کنیم میدان‌های متناهی هستند. در این بخش میدان متناهی $\mathbb{F}_q$ که $q = p^e$ و p یک عدد اوّل است را با K نمایش می‌دهیم.

نکته ۶.۳. از قضیه‌ی ۵.۳ نتیجه می‌گیریم که هر نگاشت مانند $\varphi : K^n \to K^m$ که K یک میدان متناهی است را می‌توان به‌صورت

$$\varphi(a_۱, ..., a_n) = (f_۱(a_۱, ..., a_n), ..., f_m(a_۱, ..., a_n))$$

نوشت که f_i ها چندجمله‌ای‌هایی از حلقه‌ی $K[x_۱, ..., x_n]$ هستند. امّا باید دقت کرد که این نمایش برای φ تنها نمایش چندجمله‌ای نیست. چرا که اگر فرض کنیم $\mathbb{X} = K^n$ آن‌گاه با افزودن هر عضو دلخواه از ایده‌ال

$$I(\mathbb{X}) = \{g \in K[x_۱, ..., x_n] \mid \forall\ (a_۱, ..., a_n) \in \mathbb{X} :\ \ g(a_۱, ..., a_n) = ۰\}$$

به چندجمله‌ای‌های نمایش‌دهنده‌ی φ در عین حال که هیچ تغییری در عملکرد نگاشت ایجاد نمی‌شود، نمایش آن را تغییر می‌دهد.

بنابراین مدل چندجمله‌ای متناظر با یک سامانه‌ی رمزنگاری، یکتا نیست و مهاجم هوشمند می‌تواند به نحوی این مدلسازی را انجام دهد که پیچیدگی حلّ دستگاه در مرحله‌ی سوّم به مقدار زیادی کاهش پیدا کرده و منجر به یک حمله‌ی مؤفق و عملی شود به همین دلیل مرحله‌ی اوّل حمله‌ی جبری، یعنی جبری‌سازی از اهمیت زیادی برخوردار است . در ادامه این بخش به بحث پیرامون جبری‌سازی می‌پردازیم و بحث راجع به حل دستگاه معادلات به‌دست آمده را به فصل بعدی واگذار می کنیم.

۳.۳ الگوریتم بوخبرگر– مولر

قضیه‌ی نگاشت‌های عام ۵.۳ تضمین می‌کند که هر نگاشت رمزنگاری را می‌توان به‌صورت یک نگاشت چندجمله‌ای نوشت امّا هیچ روشی برای به‌دست آوردن آن ارائه نمی‌دهد. در این بخش می‌خواهیم الگوریتمی برای استخراج معادلات چندجمله‌ای حاکم بر یک نگاشت رمزنگاری معرفی کنیم.

در اوایل دهه‌ی ۸۰ میلادی بوخبرگر با میشل مولر [1] آشنا شد و در سال ۱۹۸۲ مقاله‌ی مشترکی

[1] Michael Muller

با وی در مورد استفاده از پایه‌های گروبنر در یافتن مولّدهای ایده‌الی که در مجموعه‌ای متناهی از نقاط مشخص صفر می‌شود و همچنین محاسبه‌ی چندجمله‌ای درونیاب این نقاط نوشت [۵۲]. الگوریتمی که آن دو ارائه دادند، به الگوریتم بوخبرگر_مولر معروف است. فرض کنید K یک میدان و $V \subset K^n$ در این صورت هدف این الگوریتم محاسبه پایه گروبنر ایده‌ال

$$I(V) = \{f \in K[x_1, ..., x_n] |\ \forall p \in V\ \ f(p) = 0\}$$

است. این الگوریتم بر خلاف سایر الگوریتم‌های پایه گروبنر که به ازای یک مولد داده شده سعی در پیدا کردن پایه گروبنر دارند، بر حسب تعداد نقاط داده شده و تعداد متغیرها، زمان چندجمله‌ای است. ما قصد داریم از این الگوریتم برای استخراج دستگاه معادلات چندجمله‌ای حاکم بر نگاشت‌های رمزنگاری استفاده کنیم.

معمولاً برای به‌دست آوردن معادلات سامانه‌های متقارن که از عناصر غیر خطی نظیر s-box ها استفاده می‌کنند، ابتدا باید معادلات حاکم بر s-box ها را بیابیم. الگوریتم زیر که نسخه اصلی الگوریتم بوخبرگر_مولر است قادر است این‌کار را انجام دهد.

گزاره ۷.۳ (الگوریتم بوخبرگر_مولر۱). فرض کنید K یک میدان و σ یک ترتیب یکجمله‌ای روی $\mathbb{T}^n$ باشد. مجموعه متناهی از نقاط $\mathbb{X} = \{p_1, ..., p_s\} \subseteq K^n$ که نقاط آن را بصورت $p_i = (c_{i1}, ..., c_{in})$ نمایش می‌دهیم، را در نظر بگیرید. در این صورت الگوریتم ۹ ، پس از متناهی مرحله اجرا دوتایی $(\mathcal{G}, O)$ را بازمی‌گرداند بطوری که $\mathcal{G}$، σ- پایه‌ی گروبنر ایده‌ال $I(\mathbb{X}) \subseteq K[x_1, ..., x_n]$، است و $O = \mathbb{T}^n \setminus \mathrm{LM}_\sigma(I(\mathbb{X}))$.

برهان. به [۴۵، ص.۳۹۲]، مراجعه کنید. □

مثال ۸.۳. فرض کنید $\mathbb{X} = \{p_1, p_2, p_3, p_4, p_5\} \subseteq \mathbb{Q}^2$ که $p_1 = (0, 0), p_2 = (0, -1), p_3 = (1, 0)$ و $p_4 = (1, 1), p_5 = (-1, 1)$ ، و ترتیب یکجمله‌ای را ترتیب $\sigma = deglex$ در نظر می‌گیریم. با استفاده از الگوریتم بوخبرگر مولر ۹، طی چندگام $I(\mathbb{X})$ یا به‌عبارت دیگر چندجمله‌ای‌های دو متغیره که روی این نقاط صفر می‌شوند را می‌یابیم.

۱. $\mathcal{G} = \emptyset, O = \emptyset, \mathcal{S} = \emptyset$ و $L = \{1\}$.

۲. $t = 1$ را انتخاب کرده و پس از حذف آن از L داریم $L = \emptyset$.

۳. $(v_1, ..., v_5) = (t(p_1), ..., t(p_5)) = (1, 1, 1, 1, 1)$

۴. $\mathcal{M} = (11111), \mathcal{S} = (1), O = \{1\}$ و $L = \{y, x\}$.

۵. $t = y$ را انتخاب می‌کنیم و پس از حذف آن از L داریم $L = \{x\}$.

۶. $(t(p_1), ..., t(p_5)) = (0, -1, 0, 1, 1) = (v_1, ..., v_5)$.

الگوریتم ۹ الگوریتم بوخبرگر- مولر۱

ورودی $\mathbb{X}=\{p_1,\dots,p_s\}\subseteq K^n$

خروجی $(\mathcal{G},O)$ (با شرایط ذکر شده در صورت قضیه.)

۱: قرار می‌دهیم $\mathcal{G}=\emptyset, O=\emptyset, \mathcal{S}=\emptyset, L=\{1\}$ و $\mathcal{M}=(m_{ij})\in \mathrm{Mat}_{0,s}(K)$ را ماتریسی در نظر می‌گیریم که در ابتدا دارای s ستون و ۰ ردیف است و به گام ۲ می‌رویم.

۲: اگر $L=\emptyset$، زوج $(\mathcal{G},O)$ را بازگردانده و متوقف می‌شویم. در غیر این‌صورت یکجمله‌ای $t=\min_\sigma(L)$ را انتخاب کرده و سپس آن را از L حذف می‌کنیم و به گام ۳ می‌رویم.

۳: بردار مقادیر $(t(p_1),\dots,t(p_s))\in K^s$ را محاسبه کرده و آن را نسبت به سطرهای ماتریس $\mathcal{M}$ تحویل می‌کنیم تا بردار $(v_1,\dots,v_s)$ به‌صورت

$$(v_1,\dots,v_s)=(t(p_1),\dots,t(p_s))-\sum_i a_i(m_{i1},\dots,m_{is})$$

به‌دست آید بطوری که $a_i\in K$.

۴: اگر $(v_1,\dots,v_s)=(0,\dots,0)$ آن‌گاه چندجمله‌ای $t-\sum_i a_is_i$ را که s_i ، iامین عضو $\mathcal{S}$ است، به مجموعه‌ی $\mathcal{G}$ اضافه می‌کنیم. سپس همه مضارب t را از L حذف کرده و به گام ۲ می‌رویم.

۵: در غیر این‌صورت اگر $(v_1,\dots,v_s)\neq 0$ ، بردار $(v_1,\dots,v_s)$ را به عنوان یک سطر جدید به $\mathcal{M}$ و چندجمله‌ای $t-\sum_i a_is_i$ را به عنوان یک عضو جدید به $\mathcal{S}$ ، اضافه می‌کنیم. یکجمله‌ای t را به O اضافه می‌کنیم و سپس اعضایی از $\{x_1t,\dots,x_nt\}$ را که مضرب هیچ عضوی از $L\cup \mathrm{LM}_\sigma(\mathcal{G})$ نیستند، به L اضافه می‌کنیم و به گام ۲ می‌رویم.

۷. $O=\{1,y\}, \mathcal{S}=(1,y)$, $\mathcal{M}\begin{pmatrix}1&1&1&1&1\\0&-1&0&1&1\end{pmatrix}$ و $L=\{x,y^2\}$

۸. با انتخاب $t=x$ و حذف آن از L داریم $L=\{y^2\}$.

۹. $(t(p_1),\dots,t(p_5))=(0,0,1,1,-1)=(v_1,\dots,v_5)$.

۱۰. $O=\{1,y,x\}, \mathcal{S}=(1,y,x)$, $\mathcal{M}=\begin{pmatrix}1&1&1&1&1\\0&-1&0&1&1\\0&0&1&1&-1\end{pmatrix}$ و $L=\{y^2,xy,x^2\}$.

۱۱. با انتخاب $t=y^2$ و حذف آن از L داریم $L=\{xy,x^2\}$.

۱۲. $(t(p_1),\dots,t(p_5))=(0,1,0,1,1)$. با تحویل بردار به‌دست آمده نسبت به سطرهای ماتریس $\mathcal{M}$ داریم $(v_1,\dots,v_5)=(0,1,0,1,1)+(0,-1,0,1,1)=(0,0,0,2,2)$.

۱۳. $O = \{۱, y, x, y^۲\}$، $\mathcal{S} = (۱, y, x, y^۲+y)$، $\mathcal{M} = \begin{pmatrix} ۱ & ۱ & ۱ & ۱ & ۱ \\ ∘ & -۱ & ∘ & ۱ & ۱ \\ ∘ & ∘ & ۱ & ۱ & -۱ \\ ∘ & ∘ & ∘ & ۲ & ۲ \end{pmatrix}$ و $L = \{xy, x^۲, y^۳\}$.

۱۴. $t = xy \to L = \{x^۲, y^۳\}$

۱۵. $(t(p_۱), ..., t(p_۵)) = (∘, ∘, ∘, ۱, -۱)$، که پس از تحویل آن نسبت به سطرهای ماتریس $\mathcal{M}$ داریم

$$(v_۱, ..., v_۵) = (∘, ∘, ∘, ∘, ۲).$$

۱۶. $\mathcal{S} = (۱, y, x, y^۲ + y, xy - \frac{۱}{۲}y^۲ - \frac{۱}{۲}y)$، $\mathcal{M} = \begin{pmatrix} ۱ & ۱ & ۱ & ۱ & ۱ \\ ∘ & -۱ & ∘ & ۱ & ۱ \\ ∘ & ∘ & ۱ & ۱ & -۱ \\ ∘ & ∘ & ∘ & ۲ & ۲ \\ ∘ & ∘ & ∘ & ∘ & ۲ \end{pmatrix}$ و همچنین

$$O = \{۱, y, x, y^۲, xy\}, L = \{x^۲, y^۳, xy^۲\}.$$

۱۷. $t = x^۲ \to L = \{y^۳, xy^۲\}$.

۱۸. با محاسبه‌ی $(t(p_۱), ..., t(p_۵)) = (∘, ∘, ۱, ۱, ۱)$ و تحویل آن نسبت به سطرهای $\mathcal{M}$ داریم

$$(v_۱, ..., v_۵) = (∘, ∘, ∘, ∘, ∘).$$

۱۹. $\mathcal{G} = (x^۲ + xy - \frac{۱}{۲}y^۲ - x - \frac{۱}{۲}y)$ و $L = \{y^۳, xy^۲\}$.

۲۰. $t = y^۳ \to L = \{xy^۲\}$.

۲۱. با محاسبه‌ی $(t(p_۱), ..., t(p_۵)) = (∘, -۱, ∘, ۱, ۱)$ و تحویل آن نسبت به سطرهای $\mathcal{M}$ داریم

$$(v_۱, ..., v_۵) = (∘, ∘, ∘, ∘, ∘).$$

۲۲. $\mathcal{G} = (x^۲ + xy - \frac{۱}{۲}y^۲ - x - \frac{۱}{۲}y, y^۳ - y)$، و $L = \{xy^۲\}$.

۲۳. $t = xy^۲ \to L = \emptyset$.

۲۴. با محاسبه‌ی $(t(p_1), ..., t(p_5)) = (0, 0, 0, 1, -1)$ و تحویل آن نسبت به سطرهای $\mathcal{M}$ داریم

$$(v_1, ..., v_5) = (0, 0, 0, 0, 0).$$

۲۵. $\mathcal{G} = (x^2 + xy - \frac{1}{2}y^2 - x - \frac{1}{2}y, y^3 - y, xy^2 - xy)$، و $L = \emptyset$.

۲۶. به گام ۲ الگوریتم ۹ رسیده‌ایم در حالی که $L = \emptyset$ ، لذا شرط توقف برقرار است و $(\mathcal{G}, O)$ خروجی الگوریتم است.

$\mathcal{G} = (x^2 + xy - \frac{1}{2}y^2 - x - \frac{1}{2}y, y^3 - y, xy^2 - xy)$، پایه گروبنر تحویل یافته‌ی ایده‌ال $I(\mathbb{X})$ است و $O = \{1, x, y, xy, y^2\}$ پایه‌ای برای $\mathbb{Q}$ ـ فضای برداری $\frac{\mathbb{Q}[x,y]}{I(\mathbb{X})}$ است.

ورودی یک نگاشت رمزنگاری کلید و متن اصلی و خروجی آن متن رمز شده است، فرض کنید در موقعیت حمله متن اصلی معلوم قرار داریم، یعنی تعدادی از ورودی‌ها و خروجی‌های متناظرشان در نگاشت رمزنگاری را در اختیار داریم، در این‌صورت با یک تغییر کوچک در الگوریتم بوخبرگر می‌توانیم الگوریتمی به‌دست آوریم که چندجمله‌ای‌های حاکم بر این سامانه‌ی رمزنگاری را برای ما محاسبه کند.

گزاره ۹.۳. فرض کنید K یک میدان و $\mathbb{Y} = \{(p_1, k_1), ..., (p_s, k_s)\} \subseteq K^n \times K^l$ یک مجموعه‌ی متناهی از نقاط باشد. (در بحث ما p_i متناظر با متن اصلی و k_i متناظر با کلید رمزنگاری است.) علاوه بر این فرض کنید به ازای هر $i = 1, ..., s$ داشته باشیم $q_i = \mathsf{Enc}_{k_i}(p_i)$. حلقه‌ی $Q = K[x_1, ..., x_n, y_1, ..., y_l]$ را در نظر بگیرید، در این‌صورت الگوریتم ۱۰، پس از متناهی مرحله، چندجمله‌ای‌های $(f_1, ..., f_m) \in Q^m$ و مجموعه‌ی $\mathcal{G}$ را محاسبه می‌کند به‌طوری که $\mathcal{G}$ پایه‌ی گروبنر تحویل‌یافته‌ی $I(\mathbb{Y})$ نسبت به ترتیب یکجمله‌ای در نظر گرفته شده در گام اول الگوریتم است. همچنین به ازای هر $i \in 1, ..., s$ رابطه‌ی $(f_1(p_i, k_i), ..., f_m(p_i, k_i)) = q_i$ برقرار است.

چندجمله‌ای‌هایی که الگوریتم ۱۰ ، محاسبه می‌کند، فقط به ازای آن‌دسته از متن‌های اصلی و کلیدهای متناظر با آن‌ها که به الگوریتم داده‌ایم دقیق هستند. بنابراین اگر بخواهیم همه‌ی متن‌های اصلی و کلیدهای متناظر با آن‌ها در روابط چندجمله‌ای به‌دست آمده صدق کنند باید تمام ورودی‌ها و خروجی‌های الگوریتم رمزنگاری را به الگوریتم ۱۰، بدهیم. اگرچه الگوریتم بوخبرگر_ مولر یک الگوریتم کارا محسوب می‌شود، ولی باید توجه کرد که این الگوریتم زمانی سریع عمل می‌کند که ورودی آن از مرتبه چندجمله‌ای و دارای اندازه‌ای معقول باشد. بنابراین نمی‌توانیم کل الگوریتم رمزنگاری را به عنوان یک جعبه سیاه در نظر بگیریم و با دادن تمام ورودی و خروجی‌های آن به الگوریتم بوخبرگر_ مولر ۱۰، معادلات آنرا استخراج کنیم، چرا که اندازه مجموعه شامل تمام ورودی و خروجی‌های الگوریتم خیلی بزرگ و از مرتبه نمایی بر حسب طول کلید خواهد بود. رویکرد بهتر این است که توجه خود را به بلوک‌های کوچکتر سازنده الگوریتم

رمزنگاری معطوف کنیم و به‌صورت گام‌به‌گام معادلات حاکم بر هر یک از این بخش‌های کوچکتر را بیابیم و سپس با کنار هم قرار دادن معادلات حاکم بر این بلوک‌های سازنده به معادلاتی برای کل سامانه دست‌یابیم. در چنین رویکردی الزاماً فقط از الگوریتم بوخبرگر_ مولر استفاده نمی‌شود، بلکه روش‌های ابتکاری و متنوع دیگری نیز به‌کار می‌رود که در ادامه این فصل با برخی از آن‌ها آشنا می‌شویم.

۴.۳ جبری‌سازی رمزهای قالبی

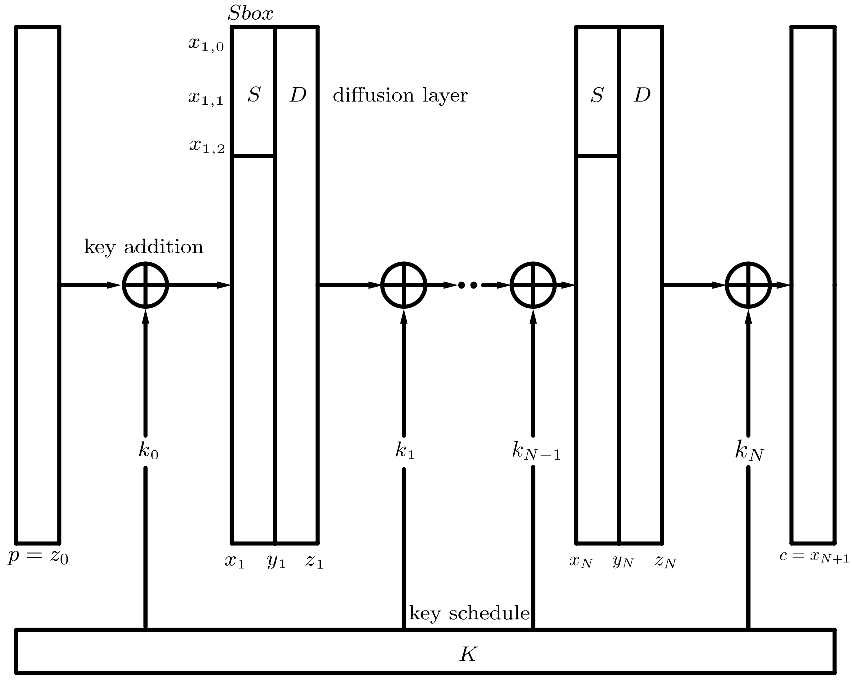
رمزهای قالبی مدرن از جمله AES ساختاری مطابق شکل ۱.۳ دارند. این ساختار متشکّل از چند دور است که هر دور شامل یک جانشینی و یک جایگشت و همچنین اضافه‌کردن کلید دور است . برای انجام عمل جانشینی از واحدهایی تحت عنوان جعبه‌های جانشینی استفاده می‌شود. جعبه‌های جانشینی و لایه‌های جایگشت با هدف توزیع همگن اطّلاعات واحدهای متن اصلی روی واحدهای متن رمز شده طرّاحی می‌شوند تا رمز قالبی در نهایت مدلی تقریبی از یک جایگشت شبه‌تصادفی باشد.

رمزهای قالبی معماری‌های متفاوتی دارند برای مثال غیر از شبکه‌ی جانشینی_ جایگشتی که در شکل ۱.۳ نشان داده شده، معماری دیگری تحت عنوان ساختار فیستل هم وجود دارد که برای مثال در طرّاحی رمز قالبی DES مورد استفاده قرار گرفته است. در ضمن در ساختار رمزهای قالبی یک الگوریتم وجود دارد که از روی کلید اصلی کلیدهای فرعی یا کلید هر یک از دورها را استخراج می‌کند.

تقریباً همه‌ی اجزای شبکه‌های SPN به جز جعبه‌های جانشینی را می‌توان با چندجمله‌ای‌های خطی مدل کرد. جعبه‌های جانشینی را نیز می‌توان به صورت چندجمله‌ای‌های درجه دو (مربّعی) یا درجات بالاتر، مدل کرد که در بخش‌های بعدی مثال‌هایی از آن را خواهیم دید. بنابراین مدل‌های پایه برای حمله به رمزهای قالبی به شرح زیر است.

حمله‌ی جبری متن اصلی معلوم

در این مدل مهاجم یک یا چند زوج متن اصلی و متن رمزشده را در اختیار دارد و حمله از نوع متن اصلی معلوم و هدف به‌دست آوردن کلید است. برای سادگی فرض کنید طول قالب و طول کلید هر دو برابر n، و تعداد دورها برابر N باشد و مهاجم فقط یک زوج متن اصلی و رمزشده به‌صورت $(\mathbf{p},\mathbf{c}) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$ را در اختیار دارد. او در ابتدا نگاشت‌های چندجمله‌ای متناظر با هر یک از واحدهای بکار رفته در سامانه را به صورت جداگانه به‌دست می‌آورد، سپس با ترکیب این نگاشت‌ها، روابط چندجمله‌ای بین متغیرهای بکار رفته در کل سامانه $\mathsf{Enc}_k(\cdot)$ را به‌صورت زیر



شکل ۱.۳: ساختار شبکه‌ی جانشینی جایگشتی (SPN) در رمزهای قالبی

به‌دست می‌آورد.

$$\mathsf{Enc}_k : \mathbb{F}_2^n \times \mathbb{F}_2^n \to \mathbb{F}_2^n \Rightarrow \begin{cases} f_1(k_0, ..., k_N, x_1, \dots, x_{N+1}, y_1, ..., y_N, z_0, \dots, z_N) = 0 \\ \vdots \\ f_s(k_0, ..., k_N, x_1, \dots, x_{N+1}, y_1, ..., y_N, z_0, \dots, z_N) = 0. \end{cases}$$

که در آن k_i نمایش فشرده‌ی بیت‌های کلید در دور i است و داریم $k_i = k_{i0}, ..., k_{in-1}$. متغیر x_i نمایش فشرده بیت‌های ورودی به دور i ام است و داریم $x_i = x_{i0}, ..., x_{in-1}$، به همین ترتیب متغیر y_i نمایش فشرده‌ی بیت‌های میانی دور iام و z_i نمایش فشرده بیت‌های خروجی دور iام است. تحت نمادگذاری فوق z_0 همان متن اصلی p و x_{N+1}، متن رمز شده c است. بنابراین مهاجم با جایگذاری مقادیر معلوم $\mathbf{p}$ و $\mathbf{c}$ در معادلات به‌دست آمده از مرحله‌ی قبل به دستگاه معادلات زیر می‌رسد.

$$\mathcal{S} = \begin{cases} f_1(k_0, ..., k_N, x_1, \dots, x_{N+1} = \mathbf{c}, y_1, ..., y_N, z_0 = \mathbf{p}, \dots, z_N) = 0 \\ \vdots \\ f_s(k_0, ..., k_N, x_1, \dots, x_{N+1} = \mathbf{c}, y_1, ..., y_N, z_0 = \mathbf{p}, \dots, z_N) = 0. \end{cases}$$

مجهولات این دستگاه بیت‌های متناظر با متغیرهای میانی و از همه مهم‌تر بیت‌های کلید است.

در اکثر مواقع دستگاه به‌دست آمده جوابی یکتا دارد. بدیهی است که بخشی از جواب متعلق به بیت‌های کلید است و به این ترتیب کلید سامانه به‌دست خواهد آمد.

برخلاف حملاتی مثل تحلیل تفاضلی یا خطّی که اعمال آن‌ها مشروط به داشتن تعداد زیادی زوج متن اصلی و رمزشده‌آن است، در این نوع حمله معمولاً داشتن فقط یک یا دو زوج متن اصلی و رمزشده آن کافی است تا دستگاه به‌دست آمده یک جواب یکتا داشته باشد و حمله اعمال شود.

نکته ۱۰.۳. حمله متن اصلی معلوم جبری که در فوق تشریح شد دور از انتظار نیست، تصوّر کنید فرستنده یک فایل با فرمت pdf را بدون هیچ اقدام احتیاطی رمز کرده باشد و مهاجم از این موضوع (نوع فایل) مطّلع باشد از آن‌جایی که چهار بایت اول فایل PDF همواره عبارت است از %PDF بنابراین مهاجم می‌تواند حمله‌ی فوق را اعمال کند.

نکته ۱۱.۳. فرآیند جبری سازی و به‌دست آوردن معادلات چندجمله‌ای حاکم بر یک سامانه، می‌تواند یک‌بار انجام شده و سپس به دفعات و در سناریوهای مختلف مورد استفاده قرار گیرد.

سناریویی را در نظر بگیرید که چند متن اصلی با یک کلید یکسان رمز شده باشند و مهاجم از این امر مطلع باشد. در این صورت مهاجم ابتدا مانند مرحله‌ی قبل معادلات حاکم بر سامانه را استخراج می‌کند. فرض کنید مهاجم t زوج متن اصلی و رمزشده $(\mathbf{p}_i, \mathbf{c}_i)$ که همه آن‌ها با یک کلید رمزشده‌اند را در دست دارد. در این صورت با جایگذاری مقادیر معلوم، در معادلات استخراج شده به معادلاتی مانند معادلات زیر دست می‌یابد.

$$\mathcal{S} = \begin{cases} \mathcal{S}_1 = \begin{cases} f_1(k_0, ..., k_N, x_1^1, ..., x_N^1, x_{N+1}^1 = \mathbf{c}_1, y_1^1, ..., y_N^1, z_0^1 = \mathbf{p}_1, ..., z_N^1) = 0 \\ \qquad\vdots \qquad\qquad\qquad \vdots \\ f_s(k_0, ..., k_N, x_1^1, ..., x_N^1, x_{N+1}^1 = \mathbf{c}_1, y_1^1, ..., y_N^1, z_0^1 = \mathbf{p}_1, ..., z_N^1) = 0. \end{cases} \\ \vdots \\ \mathcal{S}_t = \begin{cases} f_1(k_0, ..., k_N, x_1^t, ..., x_N^t, x_{N+1}^t = \mathbf{c}_t, y_1^t, ..., y_N^t, z_0^t = \mathbf{p}_t, ..., z_N^t) = 0 \\ \qquad\vdots \qquad\qquad\qquad \vdots \\ f_s(k_0, ..., k_N, x_1^t, ..., x_N^t, x_{N+1}^t = \mathbf{c}_t, y_1^t, ..., y_N^t, z_0^t = \mathbf{p}_t, ..., z_N^t) = 0. \end{cases} \end{cases}$$

نکته قابل توجه در معادلات فوق این است که متغیر متناظر با کلید در همه آن‌ها یکسان است، چرا که t متن اصلی با کلیدی یکسان رمز شده‌اند. مهاجم با حل دستگاه معادلات فوق قادر است کلید سامانه را به‌دست آورد. بدیهی است که مهاجم با داشتن تعداد بیشتری متن رمز شده تحت همان کلید، می‌تواند معادلات بیشتری به‌دست آورد که این حل دستگاه را راحت‌تر و در صورت یکتا نبودن جواب (که احتمال آن کم است) تعداد جواب‌هایی را که شامل کلید اصلی نیستند کمتر می‌کند. در حالتی که مهاجم اطّلاعاتی راجع به متن اصلی ندارد و فقط متن رمز شده را در اختیار دارد حمله ی بعدی را به کار می‌گیرد.

حمله‌ی جبری فقط متن رمزشده

فرض کنید، مهاجم به تعدادی متن رمزشده که تحت یک کلید رمز شده‌اند دسترسی دارد. مانند روش‌های قبل مهاجم می‌تواند دستگاه معادلات چندجمله‌ای تشکیل دهد که متغیرهای متناظر با کلید در همه معادلات آن مشترک، ولی متغیرهای متناظر با متن اصلی و متغیرهای میانی در آن‌ها متفاوت هستند. پس از آن با جایگذاری متن‌های رمزشده در هر یک از معادلات به‌دست‌آمده به دستگاهی می‌رسد که با حل آن متن اصلی و کلید به‌دست می‌آید.

۱.۴.۳ جبری سازی KeyLoq

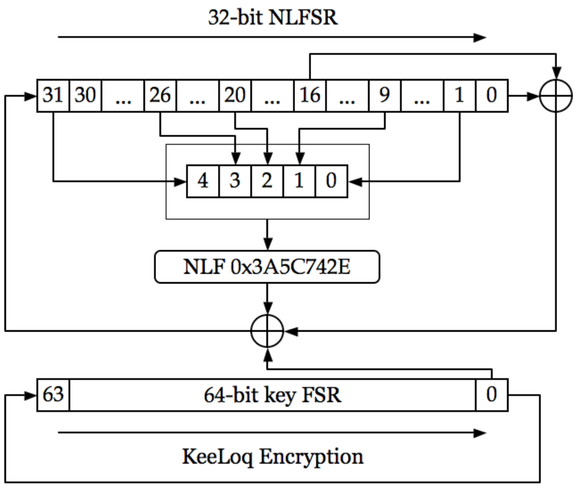
معرفی KeeLoq

KeeLoq، یک رمز قالبی با طول قالب متن اصلی و رمزشده ۳۲ بیت و طول کلید ۶۴ بیت است که در سامانه‌های کنترل از راه دور درب خودروها و منازل از آن استفاده می‌شود. همان‌طور که در شکل ۲.۳ نمایش داده شده، بخش اصلی این الگوریتم رمزنگاری از یک ثبات انتقال با بازخورد غیرخطی ۳۲ بیتی که تابع بازخورد آن یک تابع غیر خطی ۵ متغیره است تشکیل شده است. علاوه بر این یک ثبات انتقال خطی ۳۲ بیتی دارد که بیت‌های کلید روی آن بارگذاری می‌شوند.

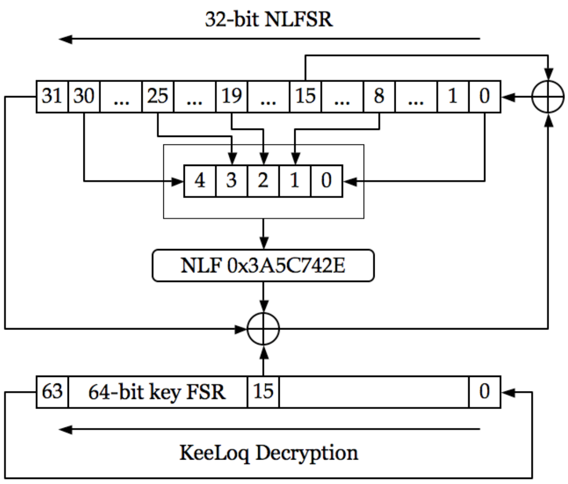
تابع بازخورد غیرخطی NLF در این الگوریتم را با $NLF_{0x3A5C742E}$، نمایش می‌دهیم. این نمایش ضابطه‌ی این تابع را نیز مشخص می‌کند و به این معنی است که، هرگاه معادل ده‌دهی رشته‌ی باینری ورودی عدد $i \in \{۰, ..., ۳۱\}$ باشد، خروجی تابع NLF بیت iام از معادل دودویی عدد هگزادسیمال $0x3A5C742E$ است. با استفاده از جدول درستی این تابع بولی می‌توان ضابطه‌ی آن را به‌صوررت زیر استخراج کرد.

$$NLF : \mathbb{F}_۲ \to \mathbb{F}_۲$$
$$NLF(a,b,c,d,e) = d + e + ac + ae + bc + be + cd + de + ade + ace + abd + abc.$$

فرض کنید بیت‌های متن اصلی را با $P_۰, ..., P_{۳۱}$، و بیت‌های متن رمزشده را با $C_۰, ..., C_{۳۱}$، که $P_۰$ و $C_۰$، به‌ترتیب بیت‌های کم‌ارزش متن اصلی و متن رمزشده هستند، نمایش دهیم. همچنین فرض کنید بیت‌های کلید را با $K_۰, ..., K_{۶۳}$ که $K_۰$، بیت کم‌ارزش کلید است، نمایش دهیم. فرآیند رمزنگاری مطابق الگوریتم ۱۱، و به صورت نشان داده شده در شکل ۲.۳، و نحوه رمزگشایی طبق الگوریتم ۱۲، و به‌صورت نمایش داده شده در شکل ۳.۳ صورت می‌گیرد.



شکل ۲.۳: رمزنگاری در Keeloq



شکل ۳.۳: رمزگشایی در Keeloq

روابط بین ورودی و خروجی

با وجود این‌که تابع بازخورد NLF در کی‌لاک (Keeloq) یک چندجمله‌ای از درجه‌ی ۳ است، می‌توان رابطه‌ای چندجمله‌ای با درجه‌ی جبری ۲ بین ورودی و خروجی این تابع پیدا کرد، که یک نمونه از آن در زیر آمده است.

$$(e+b+a+y)(c+d+y)=0,$$

که y خروجی و a,b,c,d,e ورودی تابع بازخورد هستند. در واقع رابطه‌ی چندجمله‌ای فوق رابطه‌ای است که همه‌ی رشته‌های $(a,b,c,d,e,y)\in\mathbb{F}_2^6$ که $y=NLF(a,b,c,d,e)$ در آن صدق می‌کنند.

ثبات انتقال حاوی بیت‌های کلید، در هر دور به اندازه‌ی یک واحد انتقال چرخشی پیدا می‌کند،

سپس بیت کم ارزش وارد فرآیند رمزنگاری می‌شود. بنابراین اگر بیت‌های کلید مخفی را با $k_{۳۱}, ..., k_{۰}$ نمایش دهیم آن‌گاه بیت کلیدی که در دور tام وارد فرآیند رمزنگاری می‌شود برابر است با $k_{(t-۱) \mod ۶۴}$.

فرض کنید بیت‌های حالت اولیه ثبات انتقال با بازخورد غیرخطی را با $L_{۳۱}, ..., L_{۰}$ و بیت حالت تولید شده در دور i را با $L_{۳۱+i}$ نمایش دهیم. در نتیجه آخرین بیت حالتی که تولید می‌شود بیت $L_{۵۵۹}$ متعلق به دور ۵۲۸ام است. با توجه به نمادگذاری فوق رابطه زیر بین متغیرهای L_i و بیت‌های متن اصلی و رمزشده برقرار است.

$$L_{۵۵۹}, ..., L_{۵۲۸} = C_{۳۱}, ..., C_{۰} \text{ متن رمز شده},$$

$$L_{۳۱}, ..., L_{۰} = P_{۳۱}, ..., P_{۰} \text{ متن اصلی}.$$

برای یکی‌کردن اندیس‌ها در k_t و L_i، توجه کنید که بیت حالت L_i به ازای $i \in \{۳۲, ..., ۵۵۹\}$ در دور $t = i - ۳۱$ تولید می‌شود. بنابراین بیت کلیدی که برای محاسبه‌ی L_i به‌کار می‌رود عبارت است از $k_{(i-۳۲) \mod ۶۴}$.

اکنون می‌توانیم روابط چندجمله‌ای بین متغیرهای حالت L_i و متغیرهای k_i متناظر با بیت‌های کلید را به‌صورت زیر به‌دست آوریم.

$$\begin{aligned}
L_i &= P_i && \forall i \in \{۰, ..., ۳۱\}\\
L_i &= k_{(i-۳۲) \mod ۶۴} + L_{(i-۳۲)} + L_{(i-۱۶)} && \forall i \in \{۳۲, ..., ۵۵۹\}\\
&\quad + NLF(L_{i-۱}, L_{i-۶}, L_{i-۱۲}, L_{i-۲۳}, L_{i-۳۰}) &&\\
C_{i-۵۲۸} &= L_i && \forall i \in \{۵۲۸, ۵۵۹\}
\end{aligned}$$

با جایگذاری رابطه‌ی درجه‌ی ۳ تابع NLF در معادلات فوق به روابط چندجمله‌ای از درجه‌ی ۳ به‌صورت زیر می‌رسیم.

$$\begin{aligned}
L_i &= P_i && \forall i \in \{۰, ..., ۳۱\}\\
L_i &= k_{(i-۳۲) \mod ۶۴} + L_{(i-۳۲)} + L_{(i-۱۶)} + L_{i-۲۳} + L_{i-۳۰} && \forall i \in \{۳۲, ..., ۵۵۹\}\\
&\quad + L_{i-۱}L_{i-۱۲} + L_{i-۱}L_{i-۳۰} + L_{i-۶}L_{i-۱۲} + L_{i-۶}L_{i-۳۰} &&\\
&\quad + L_{i-۱۲}L_{i-۲۳} + L_{i-۲۳}L_{i-۳۰} + L_{i-۱}L_{i-۲۳}L_{i-۳۰} &&\\
&\quad + L_{i-۱}L_{i-۱۲}L_{i-۳۰} + L_{i-۱}L_{i-۶}L_{i-۲۳} + L_{i-۱}L_{i-۶}L_{i-۱۲} &&\\
C_{i-۵۲۸} &= L_i && \forall i \in \{۵۲۸, ۵۵۹\}
\end{aligned}$$

اکنون فرض کنید یک زوج متن اصلی و رمزشده مثل (P, C) را داشته باشیم در این صورت با جایگذاری مقادیر $L_i = P_i$ به ازای $i \in \{۰, ..., ۳۱\}$ و $L_j = C_{j-۵۲۸}$ به ازای $j \in \{۵۲۸, ..., ۵۵۹\}$، در روابط فوق، یک دستگاه معادلات از درجه‌ی ۳ که مجهولات آن بیت‌های کلید و متغیرهای حالت داخلی L_i به ازای $i \in \{۳۲, ..., ۵۲۷\}$، هستند به‌دست می‌آید. در نتیجه در مجموع $۵۶۰ = ۶۴ + ۴۹۶$، مجهول و ۵۲۸ معادله‌ی درجه‌ی ۳ خواهیم داشت. اگر فرض کنیم μ تا زوج متن اصلی و رمز شده در دست باشد آن‌گاه به ازای هر زوج، دستگاهی شبیه به دستگاه قبل داریم، که متغیرهای متناظر با بیت‌های کلید در همه‌ی این دستگاه‌ها یکسان است. بنابراین به ازای μ زوج متن اصلی و رمز شده، دستگاه معادلاتی شامل $۴۹۶\mu + ۶۴$ مجهول و ۵۲۸μ معادله‌ی درجه‌ی ۳ خواهیم داشت.

می‌توانیم با تغییر متغیر زیر درجه‌ی دستگاه را از ۳ به ۲ کاهش دهیم.

$$
\begin{aligned}
NLF(a,b,c,d,e) =& d + e + ac + \beta + bc + be + cd + de + d\beta + c\beta + \alpha\beta + \alpha c \\
\alpha =& ab \\
\beta =& ae.
\end{aligned}
$$

تغییر متغیر فوق سبب می‌شود که هر یک از معادلات قبلی با سه معادله‌ی جدید جایگزین شوند که هر یک از این معادلات، شامل دو متغیر جدید α_i و β_i نیز هستند. دستگاه حاصل در زیر نمایش داده شده.

$$
\begin{aligned}
&L_i = P_i && \forall i \in \{۰, ..., ۳۱\} \\
&L_i = k_{(i-۳۲) \bmod ۶۴} + L_{(i-۳۲)} + L_{(i-۱۶)} + L_{i-۲۳} + L_{i-۳۰} && \forall i \in \{۳۲, ..., ۵۵۹\} \\
&\quad L_{i-۱}L_{i-۱۲} + \beta_i + L_{i-۶}L_{i-۱۲} + L_{i-۶}L_{i-۳۰} + L_{i-۱۲}L_{i-۲۳} && \\
&\quad + L_{i-۲۳}L_{i-۳۰} + \beta_i L_{i-۲۳} + \beta_i L_{i-۱۲} + \alpha_i L_{i-۲۳} + \alpha_i L_{i-۱۲} && \\
&\alpha_i = L_{i-۱}L_{i-۶} && \forall i \in \{۳۲, ..., ۵۵۹\} \\
&\beta_i = L_{i-۱}L_{i-۳۰} && \forall i \in \{۳۲, ..., ۵۵۹\} \\
&C_{i-۵۲۸} = L_i && \forall i \in \{۵۲۸, ..., ۵۵۹\}
\end{aligned}
$$

در نتیجه اگر μ زوج متن اصلی و رمزشده را داشته باشیم، دستگاهی به‌دست می‌آید که شامل ۱۵۸۴μ معادله و $۱۵۵۲\mu + ۶۴$ مجهول است. این دستگاه قطعاً دارای جواب خواهد بود چون بطور قطعی از سامانه‌ی رمزنگاری استخراج شده است، ولی ممکن است جواب آن یکتا نباشد. هر چه تعداد زوج متن اصلی و رمز شده‌ای که در اختیار داریم بیشتر باشد، قیدها یا تعداد معادلات بیشتر شده و جواب به سمت یکتایی پیش می‌رود. برای مثال به ازای $\mu = ۲$ تعداد معادلات به ۳۱۶۸ (معادله درجه‌ی ۲) و تعداد مجهولات به ۳۱۶۸ می‌رسد. حل دستگاه به‌دست آمده از این روش با

استفاده از روش‌های موجود بسیار دشوار و حتی نشدنی است! بنابراین صرف جبری سازی یک سامانه‌ی رمزنگاری نمی‌تواند منجر به شکسته شدن آن شود. هدف ما از آوردن این مثال ارائه‌ی یک روش ساده و قابل فهم در تبدیل یک سامانه‌ی رمزنگاری به دستگاه معادلات چندجمله‌ای بود. روش مذکور تنها روش حمله‌ی جبری به این سامانه‌ی رمزنگاری نیست، برای مشاهده‌ی روش‌های جبری که منجر به شکست عملی Keeloq شده‌اند، می‌توانید به [۷، ۱۷] مراجعه کنید.

۲.۴.۳ استخراج معادلات جعبه‌های جانشینی

جعبه‌های جانشینی یکی از بخش‌های کلیدی بسیاری از الگوریتم‌های رمزنگاری نوین، به‌خصوص رمزهای قالبی با معماری شبکه جانشینی‌ـجایگشتی هستند. جعبه جانشینی در واقع نگاشتی غیرخطی است، که یک m بیتی را به یک n بیتی می‌نگارد که در این‌صورت آن‌را یک جعبه جانشینی $m \times n$ می‌نامند. در اغلب موارد جعبه‌های جانشینی تنها عامل غیرخطی در ساختار الگوریتم‌های رمزنگاری هستند که در این بخش قصد داریم روشی برای استخراج معادلات مستقل خطی و با درجه‌ی مشخص برای جعبه‌های جانشینی معرفی کنیم. یکی از روش‌هایی که برای به‌دست آوردن روابط جبری بین متغیرهای ورودی و خروجی یک جعبه جانشیی وجود دارد، به‌دست آوردن تابع بولی هر یک از متغیرهای خروجی بر حسب متغیرهای متناظر با بیت‌های ورودی است. برای مثال جعبه جانشینی ۳ × ۳، با ضابطه

x	۰	۱	۲	۳	۴	۵	۶	۷
$S[x]$	۷	۶	۰	۴	۲	۵	۱	۳

را در نظر بگیرید. اگر ۳ بیت ورودی را با x_0, x_1, x_2، و ۳ بیت خروجی را با y_0, y_1, y_2، که x_0 و y_0، به‌ترتیب بیت‌های کم‌ارزش ورودی و خروجی هستند نمایش دهیم آن‌گاه با استفاده از قطعه کد زیر در نرم‌افزار سیج تابع بولی y_0, y_1, y_2 را به‌صورت صریح بر حسب متغیرهای x_0, x_1, x_2، به‌دست می‌آوریم.

```
S = mq.SBox(7,6,0,4,2,5,1,3)
n = S.n
Y = [0]*n
for i in range(n - 1, -1, -1):
    f = S.component_function(2^i)
    Y[i] = f.algebraic_normal_form()
```

و در نتیجه داریم

$$
\begin{aligned}
y_2 &= x_0x_1 + x_0x_2 + x_1x_2 + x_1 + x_2 + 1 \\
y_1 &= x_0x_2 + x_1 + 1 \\
y_0 &= x_0x_1 + x_0 + x_1 + x_2 + 1
\end{aligned}
$$

در صورتی که نگاشت جعبه جانشینی یک‌به‌یک و در نتیجه معکوس‌پذیر باشد، می‌توان تابع بولی هر یک از ورودی‌ها بر حسب خروجی‌ها را نیز به‌دست آورد تا به این‌ترتیب معادلات بیشتری از جعبه جانشینی داشته باشیم. برای نمونه در مثال قبل می‌توانیم به‌صورت زیر معادلات معکوس را استخراج کنیم.

```
m = S.m; n = S.n
P = BooleanPolynomialRing(m + n, ['x' + str(i) for i in range(m)]
                                      + ['y' + str(i) for i in range(n)])
vars = P.gens()
T = S.inverse()
X = [0]*m
for i in range(m - 1, -1, -1):
        f = T.component_function(2^i)
        f = f.algebraic_normal_form()
        X[i] = f.subs(dict(zip(vars[:m], vars[m:])))
```

تابع بولی هر یک از ورودی‌ها بر حسب خروجی‌ها، که از اجرای کد فوق به‌دست آمده، به صورت زیر است.

$$x_2 = y_0y_1 + y_0 + y_1y_2 + y_1$$
$$x_1 = y_0y_1 + y_0y_2 + y_1 + 1$$
$$x_0 = y_0y_1 + y_2$$

اگر جعبه جانشینی $n \times n$، معکوس‌پذیر باشد، می‌توانیم با کنار هم قرار دادن معادلات جعبه جانشینی و معادلات معکوس آن دستگاهی شامل $2n$ معادله و $2n$ مجهول برای جعبه جانشینی تشکیل دهیم. چنان‌چه در [۴] نیز گزارش شده است، گاهی حل معادلات به‌دست آمده از این روش سریع‌تر از حل معادلاتی است که از سایر روش‌ها به‌دست می‌آیند، اما این روش یک محدودیت اساسی دارد و آن درجه بالای جبری توابع بولی است که از این روش به‌دست می‌آید.

جعبه‌های جانشینی در رمزهای نوین طوری طراحی می‌شوند که درجه توابع بولی مؤلفه‌ای آن تا حد امکان بالا باشد تا به این‌ترتیب حل آن‌ها توسط حل‌کننده‌ها سخت‌تر گردد. به همین دلیل در ادامه روش دیگری را معرفی می‌کنیم که درجه معادلات استخراج شده در آن تا حدی قابل کنترل است. این روش که در [۱۱]، معرفی شده و نوع تعمیم‌یافته آن در نرم‌افزار سِیج SAGE[2] [۶۲] پیاده‌سازی شده است را روش ماتریس افزوده می‌نامیم، و با ذکر یک مثال شرح می‌دهیم.

همان جعبه جانشینی قبلی را در نظر بگیرید، این بار نیز فرض کنید بیت‌های ورودی و خروجی را به ترتیب با x_0, x_1, x_2 و y_0, y_1, y_2، که x_0 و y_0، به‌ترتیب بیت‌های *کم‌ارزش* ورودی و خروجی هستند نمایش دهیم. فرض کنید هدف ما به‌دست آوردن روابط چندجمله‌ای از درجه حداکثر ۲ بین متغیرهای ورودی و خروجی است، برای این‌کار ابتدا یک ماتریس افزوده به صورت زیر

[2] System for Algebra and Geometry Experimentation

تشکیل می‌دهیم.

$$
\begin{array}{c}
\begin{array}{ccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 &
\end{array}\\
\begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1\\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & x_2\\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & x_1\\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & x_0\\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & y_2\\
1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & y_1\\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & y_0\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & x_1x_2\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & x_0x_2\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & x_2y_2\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & x_2y_1\\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & x_2y_0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & x_0x_1\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & x_1y_2\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_1y_1\\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & x_1y_0\\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & x_0y_2\\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & x_0y_1\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & x_0y_0\\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & y_1y_2\\
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & y_0y_2\\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & y_0y_1
\end{pmatrix}
\end{array}
$$

ستون آخر این ماتریس افزوده، شامل همه یکجمله‌ای‌های از درجه حداکثر ۲ (به انضمام ۱) تحت یک ترتیب یکجمله‌ای است. سایر درایه‌های هر سطر با مقادیری که یکجمله‌ای متناظر با آن سطر به ازای تمام ورودی‌های ممکن، اختیار می‌کند، پر شده‌اند.

بعد از تشکیل ماتریس، با عملیات حذفی گاوس ماتریس صفر و یک سمت چپ را به صورت سطری‌ـ پلکانی تحویل‌یافته درمی‌آوریم و در طول این فرآیند، هر عملیاتی که روی ماتریس سمت چپ ماتریس افزوده اعمال می‌شود را روی سطرهای ماتریس سمت راست، یعنی ستون شامل

یکجمله‌ای‌ها نیز اعمال می‌کنیم که در نتیجه آن ماتریس زیر حاصل می‌شود.

$$\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_0+x_1+x_2y_2+y_1+y_2+1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & x_1+x_2y_2+x_2+y_0+1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & x_2y_2+x_2+y_2+1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & x_0+x_2y_2+x_2+y_0+y_1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & x_0+x_1+x_2y_2+x_2+y_0+y_1+y_2+1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & x_2y_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & x_0+x_2y_2+y_0+y_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & x_1+x_2y_2+y_1+1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0x_2+x_1+y_1+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+x_1x_2+x_1+y_0+y_1+y_2+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+x_2y_1+x_2+y_0+y_2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+x_1+x_2y_0+x_2y_2+y_0+y_1+y_2+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0x_1+x_0+x_1+x_2+y_0+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+x_1y_2+x_2y_2+x_2+y_0+y_1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_1y_1+x_1+x_2y_2+y_1+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+x_1y_0+x_1+y_0+y_1+y_2+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0y_2+x_0+x_1+x_2y_2+y_1+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0y_1+x_2+y_0+y_1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0y_0+x_1+y_1+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+x_2+y_0+y_1y_2+y_1+y_2} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+x_1+y_0y_2+y_1+y_2+1} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{x_0+y_0y_1+y_2}
\end{pmatrix}$$

از آن‌جایی که مقدار هر یک از روابط چندجمله‌ای متناظر با سطرهای تمام صفر، به ازای همه‌ی مقادیر ورودی همواره صفر است لذا تمام ورودی و خروجی‌های جعبه جانشینی در معادلاتی که از مساوی صفر قرار دادن این روابط به‌دست می‌آیند صدق می‌کنند. در نتیجه ۱۴ معادله به‌صورت زیر خواهیم داشت.

$$0 = x_0x_2+x_1+y_1+1,$$
$$0 = x_0+x_1x_2+x_1+y_0+y_1+y_2+1,$$
$$0 = x_0+x_2y_1+x_2+y_0+y_2,$$
$$0 = x_0+x_1+x_2y_0+x_2y_2+y_0+y_1+y_2+1,$$
$$0 = x_0x_1+x_0+x_1+x_2+y_0+1,$$
$$0 = x_0+x_1y_2+x_2y_2+x_2+y_0+y_1,$$
$$0 = x_1y_1+x_1+x_2y_2+y_1+1,$$
$$0 = x_0+x_1y_0+x_1+y_0+y_1+y_2+1,$$
$$0 = x_0y_2+x_0+x_1+x_2y_2+y_1+1,$$
$$0 = x_0y_1+x_2+y_0+y_1,$$
$$0 = x_0y_0+x_1+y_1+1,$$
$$0 = x_0+x_2+y_0+y_1y_2+y_1+y_2,$$
$$0 = x_0+x_1+y_0y_2+y_1+y_2+1,$$
$$0 = x_0+y_0y_1+y_2.$$

با استفاده از دو دستور زیر در نرم‌افزار سِیج، به‌راحتی می‌توانیم معادلات فوق را استخراج کنیم.

```
S = mq.SBox(7,6,0,4,2,5,1,3)
S.polynomials([x2,x1,x0],[y2,y1,y0])
```

این ۱۴ معادله، مستقل خطی و شامل ۲۲ یکجمله‌ای متمایز هستند. دلیل استقلال خطی این است که هر معادله شامل یکجمله‌ای‌هایی است که دیگر معادلات فاقد آن هستند. لازم به‌ذکر است که گاهی ممکن است ماتریس تحویل شده، فاقد سطر تمام‌صفر باشد و در نتیجه نتوانیم معادلات ضمنی درجه دو برای جعبه جانشینی به‌دست آوریم. در این صورت می‌توانیم همین روش را برای به‌دست‌آوردن معادلات درجه ۳ و بالاتر تعمیم دهیم. برای این‌کار کافی است تا ستون آخر ماتریس افزوده را با یکجمله‌ای‌های از درجه حداکثر ۳ (یا بالاتر) پر کنیم و مانند قبل عمل کنیم.

یک روش دیگر برای استخراج معادلات جعبه‌های جانشینی استفاده از الگوریتم بوخبرگر مولر است. یک جعبه جانشینی $m \times n$ را در نظر بگیرید، نقاط ورودی و خروجی این جعبه‌جانشینی را می‌توانیم به‌صورت نقاط دودویی از فضای $\mathbb{F}_۲^{m+n}$ در نظر بگیریم. فرض کنید نقاطی از این فضا که در جعبه جانشینی داده شده صدق می‌کنند را با V نمایش دهیم در این‌صورت اندازه این مجموعه برابر است با ۲^m. با در نظر گرفتن V به عنوان ورودی الگوریتم بوخبرگر–مولر، خروجی آن چندجمله‌ای‌هایی از حلقه $\mathbb{F}_۲[x_۰, \dots, x_{m-۱}, y_۰, \dots, y_{n-۱}]$ خواهند بود که پایه گروبنر $I(V)$ را تشکیل می‌دهند.

۳.۴.۳ جبری‌سازی CTC

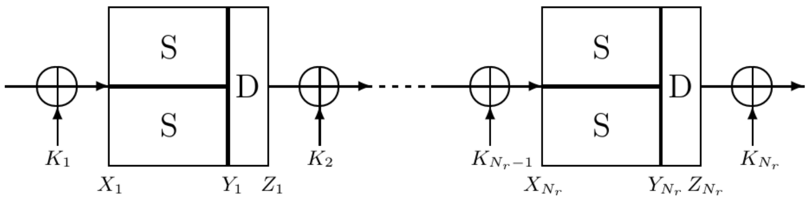
به عنوان یک نمونه دیگر از رمزهای قالبی، نحوه جبری‌سازی الگوریتم CTC [۳]، را بررسی می‌کنیم. این الگوریتم توسط کورتوا، در دو نسخه CTC1 [۲۰]، و CTC2 [۱۸]، و فقط برای آزمودن حمله‌های جبری ارائه شده است و یک سامانه‌ی رمزنگاری کاربردی محسوب نمی‌شود. البته ساختار این الگوریتم به ساختار بسیاری از الگوریتم‌های رمز قالبی واقعی نظیر سرپنت [۴] و AES، شباهت دارد. این الگوریتم از چند دور مشابه تشکیل می‌شود. هر دور شامل سه عملیات اصلی است که عبارتند از، جمع به پیمانه‌ی ۲ با کلید دور، عبور از لایه جعبه‌های جانشینی و سپس عبور بیت‌های خروجی لایه‌ی جعبه‌های جانشینی از یک لایه‌ی انتشار خطی. الگوریتم استخراج کلید هر یک از دورها، شبیه الگوریتم استخراج کلید در DES و فقط شامل یک جایگشت ساده روی بیت‌های کلید اصلی (مخفی) است.

پارامترهای اصلی این سامانه عبارتند از

۱. جعبه‌های جانشینی یکسان با طول قالب $s = ۳$. جعبه‌ی جانشینی این سامانه یک نگاشت روی $\mathbb{F}_{۲^۳}$ است که توسط لیست مرتب $[۷, ۶, ۰, ۴, ۲, ۵, ۱, ۳]$ نیز نمایش داده می‌شود، این نمایش به این معنی است که به ازای ورودی ۳ بیتی x که معادل ده‌دهی آن عددی است بین ۰ تا ۷، خروجی y، عضو xام از لیست تعریف شده است.

[3] Courtois Toy Cipher

[4] serpent



شکل ۴.۳: رمز CTC با ۲ جعبه‌ی جانشینی در هر دور

۲. تعداد جعبه‌های جانشینی در هر دور، که با B نمایش داده می‌شود. این پارامتر می‌تواند بین ۱ تا ۱۲۸ متغیر باشد. برای مثال شکل ۴.۳ یک دور از CTC با $B = ۲$ را نشان می‌دهد.

۳. طول قالب متن اصلی و متن رمز شده (بر حسب بیت) برابر $B \cdot s$، و طول کلید اصلی برابر است با $H_k = ۱, ۲, ۳, ...$، که بیت‌های کلید اصلی را با $k = (k_۰, ..., k_{H_k - ۱})$، نمایش می دهیم. در CTC1، $H_k = B \cdot s$، اما در CTC2، طول قالب الزاما برابر با $B \cdot s$ نیست و می‌تواند کوچکتر، مساوی و یا بزرگتر از آن باشد.

۴. تعداد دورها، که با N_r نمایش داده می‌شود.

فرض کنید بیت‌های ورودی و خروجی دور iام را به ترتیب با X_{ij} و Z_{ij} نمایش دهیم که $i = ۱, ..., N_r$ و $j = ۱, ..., Bs - ۱$. با این نحوه‌ی نمایش، $Z_۰$ همان متن اصلی و $X_{N_r+۱}$ همان متن رمز شده است که در حمله از نوع متن اصلی معلوم یا متن رمزی انتخابی فرض می‌کنیم هر دو معلوم هستند. در CTC1 کلید هر یک از دورها با یک جایگشت روی بیت‌های کلید اصلی و به‌صورت زیر به‌دست می‌آید.

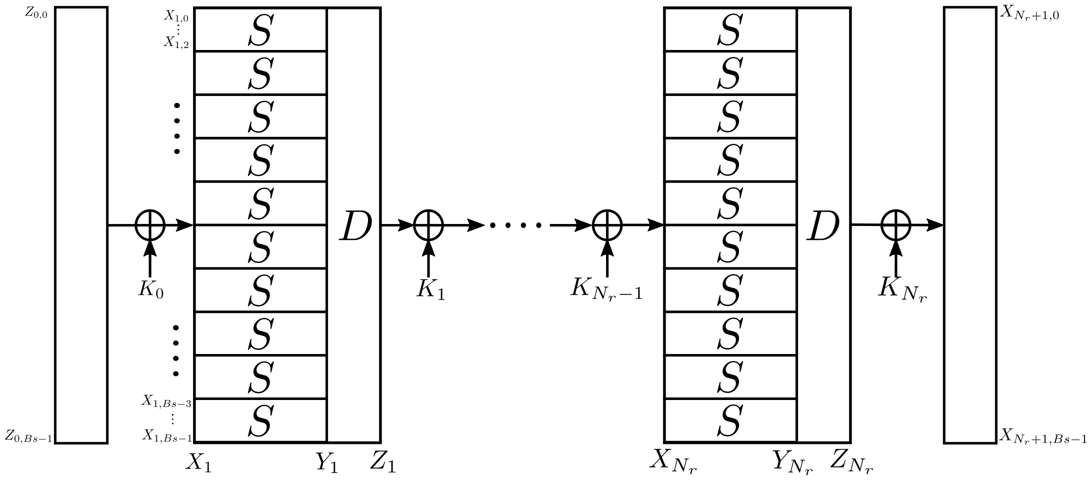
$$K_{ij} := K_{۰(j+i \mod Bs)}.$$

که $K_۰ = (K_{۰۰}, ..., K_{۰(B \cdot s - ۱)})$ همان کلید اصلی است. اما در CTC2 داریم

$$K_{ij} = k_{j+i \cdot ۵۰۹ \mod H_k}.$$

که $k = (k_۰, ..., k_{H_k - ۱})$ کلید اصلی است.
به ازای هر دور $i = ۱, ..., Nr$، عملکرد لایه‌ی انتشار D در CTC1 به صورت

$$\begin{cases} Z_{i(j \cdot ۱۹۸۷ + ۲۵۷ \mod Bs)} = Y_{i۰} & j = ۰ \\ Z_{i(j \cdot ۱۹۸۷ + ۲۵۷ \mod Bs)} = Y_{ij} \oplus Y_{i(j+۱۳۷ \mod Bs)} & \forall\, j \neq ۰. \end{cases}$$



شکل ۵.۳: رمز CTC با ۱۰ جعبه‌ی جانشینی در هر دور

و در CTC2 به‌صورت زیر است.

$$\begin{cases} Z_{i(j\cdot ۱۹۸۷+۲۵۷ \mod Bs)} = Y_{ij} \oplus Y_{i(j+۱۳۷ \mod B\cdot s)} \oplus Y_{i(j+۲۷۴ \mod B\cdot s)} & j = ۲۵۷ \mod B\cdot s \\ Z_{i(j\cdot ۱۹۸۷+۲۵۷ \mod Bs)} = Y_{ij} \oplus Y_{i(j+۱۳۷ \mod Bs)} & \text{در غیر این‌صورت} \end{cases}$$

که Y_{ij} نمایش ورودی لایه‌ی انتشار است.

شکل ۵.۳ یک رمز CTC با پارامتر $B = ۱۰$ را نشان می‌دهد. در ادامه نحوه استخراج معادلات حاکم بر یک دور از این الگوریتم را نشان می‌دهیم که به هر تعداد دور دلخواه نیز قابل تعمیم خواهد بود. همان طور که مشاهده شد تنها بخش غیر خطی هر دور از الگوریتم رمزنگاری CTC، جعبه‌های جانشینی هستند. معادله‌های حاکم بر دور اول را به‌صورت گام‌به‌گام و در جهت عبور بیت‌های متن اصلی از لایه‌های رمزنگاری به‌دست می‌آوریم. برای سادگی فرض کنید فقط یک جعبه جانشینی در هر دور به‌کار ببریم. بنابراین پارامترهای CTC، در این مثال عبارتند از $N_r = ۱$ و $B = ۱$. بیت‌های متن اصلی که عبارتند از $(Z_{۰۰}, Z_{۰۱}, Z_{۰۲})$، ابتدا با بیت‌های کلید بر اساس رابطه‌ی $X_{i+۱j} = Z_{ij} \oplus K_{ij}$، به ازای $i = ۰$ و $j = ۰, ۱, ۲$، ترکیب می‌شوند، بنابراین سه رابطه به‌صورت زیر خواهیم داشت.

$$\begin{aligned} ۰ =& K_{۰۰} + Z_{۰۰} + X_{۱۰}, \\ ۰ =& K_{۰۱} + Z_{۰۱} + X_{۱۱}, \\ ۰ =& K_{۰۲} + Z_{۰۲} + X_{۱۲}. \end{aligned}$$

بیت‌ها پس از ترکیب اولیه با کلید اصلی، از لایه جعبه‌های جانشینی عبور می‌کنند. اگر بیت‌های ورودی و خروجی لایه جعبه جانشینی در دور اول را طبق قرارداد به ترتیب با $X_{۱۲}, X_{۱۱}, X_{۱۰}$ و $Y_{۱۲}, Y_{۱۱}, Y_{۱۰}$، که $X_{۱۰}$ و $Y_{۱۰}$، به‌ترتیب بیت‌های کم‌ارزش ورودی و خروجی هستند نمایش دهیم،

آن‌گاه بر اساس روش ماتریس افزوده که در بخش قبل معرفی شد، ۱۴ رابطه چندجمله‌ای درجه ۲ به‌صورت زیر بین متغیرهای ورودی و خروجی لایه جانشینی به‌دست می‌آید.

$$\begin{aligned}
0 &= X_{10}X_{12} + X_{11} + Y_{11} + 1, & 0 &= X_{10} + X_{11}Y_{10} + X_{11} + Y_{10} + Y_{11} + Y_{12} + 1,\\
0 &= X_{10} + X_{11}X_{12} + X_{11} + Y_{10} + Y_{11} + Y_{12} + 1, & 0 &= X_{10}Y_{12} + X_{10} + X_{11} + X_{12}Y_{12} + Y_{11} + 1,\\
0 &= X_{10} + X_{12}Y_{11} + X_{12} + Y_{10} + Y_{12}, & 0 &= X_{10}Y_{11} + X_{12} + Y_{10} + Y_{11},\\
0 &= X_{10} + X_{11} + X_{12}Y_{10} + X_{12}Y_{12} + Y_{10} + Y_{11} + Y_{12} + 1, & 0 &= X_{10}Y_{10} + X_{11} + Y_{11} + 1,\\
0 &= X_{10}X_{11} + X_{10} + X_{11} + X_{12} + Y_{10} + 1, & 0 &= X_{10} + X_{12} + Y_{10} + Y_{11}Y_{12} + Y_{11} + Y_{12},\\
0 &= X_{10} + X_{11}Y_{12} + X_{12}Y_{12} + X_{12} + Y_{10} + Y_{11}, & 0 &= X_{10} + X_{11} + Y_{10}Y_{12} + Y_{11} + Y_{12} + 1,\\
0 &= X_{11}Y_{11} + X_{11} + X_{12}Y_{12} + Y_{11} + 1, & 0 &= X_{10} + Y_{10}Y_{11} + Y_{12}.
\end{aligned}$$

مرحله بعدعبور از لایه انتشار (خطی) است که با استفاده از نحوه‌ی تعریف لایه‌ی خطی D، سه معادله به‌صورت زیر خواهیم داشت.

$$\begin{aligned}
0 &= Z_{10} + Y_{11} + Y_{10},\\
0 &= Z_{11} + Y_{12} + Y_{11},\\
0 &= Z_{12} + Y_{10}.
\end{aligned}$$

مرحله‌ی آخر ترکیب بیت‌های خروجی لایه انتشار با بیت‌های کلید دور اول است که سه رابطه زیر را به‌دست می‌دهد.

$$\begin{aligned}
0 &= K_{10} + Z_{10} + X_{20},\\
0 &= K_{11} + Z_{11} + X_{21},\\
0 &= K_{12} + Z_{12} + X_{22}.
\end{aligned}$$

در ضمن با توجه به نحوه عمل‌کرد الگوریتم استخراج کلید، روابط زیر بین بیت‌های کلید اصلی و بیت‌های کلید دور اول برقرار است.

$$\begin{aligned}
0 &= K_{10} + K_{01},\\
0 &= K_{11} + K_{02},\\
0 &= K_{12} + K_{00}.
\end{aligned}$$

اکنون یک حمله متن رمزی معلوم را در نظر بگیرید، فرض کنید متن اصلی $P = (0, 1, 0)$، و متن رمزشده متناظر با آن که برابر است با $C = (1, 1, 1)$، را در اختیار داشته باشیم و هدف ما به‌دست آوردن کلید باشد. در این‌صورت با کنار هم قرار دادن معادله‌های هر یک از لایه‌های رمزنگاری و جایگذاری مقادیر متناظر با متن اصلی و رمز شده که به‌ترتیب برابرند با $(Z_{00}, Z_{01}, Z_{02}) = (0, 1, 0)$

و $(X_{20}, X_{21}, X_{22}) = (1, 1, 1)$، به دستگاه معادلاتی که در ادامه آمده است می‌رسیم.

$$
\begin{aligned}
0 &= K_{00} + X_{10}, & 0 &= X_{10}Y_{10} + X_{11} + Y_{11} + 1,\\
0 &= K_{01} + X_{11} + 1, & 0 &= X_{10} + X_{12} + Y_{10} + Y_{11}Y_{12} + Y_{11} + Y_{12},\\
0 &= K_{02} + X_{12}, & 0 &= X_{10} + X_{11} + Y_{10}Y_{12} + Y_{11} + Y_{12} + 1,\\
0 &= X_{10}X_{12} + X_{11} + Y_{11} + 1, & 0 &= X_{10} + Y_{10}Y_{11} + Y_{12}\\
0 &= X_{10} + X_{11}X_{12} + X_{11} + Y_{10} + Y_{11} + Y_{12} + 1, & 0 &= Y_{10} + Z_{12},\\
0 &= X_{10} + X_{12}Y_{11} + X_{12} + Y_{10} + Y_{12}, & 0 &= Y_{10} + Z_{10} + Y_{11},\\
0 &= X_{10} + X_{11} + X_{12}Y_{10} + X_{12}Y_{12} + Y_{10} + Y_{11} + Y_{12} + 1, & 0 &= Y_{11} + Z_{11} + Y_{12},\\
0 &= X_{10}X_{11} + X_{10} + X_{11} + X_{12} + Y_{10} + 1, & 0 &= K_{10} + Z_{10} + 1,\\
0 &= X_{10} + X_{11}Y_{12} + X_{12}Y_{12} + X_{12} + Y_{10} + Y_{11}, & 0 &= K_{11} + Z_{11} + 1,\\
0 &= X_{11}Y_{11} + X_{11} + X_{12}Y_{12} + Y_{11} + 1, & 0 &= K_{12} + Z_{12} + 1,\\
0 &= X_{10} + X_{11}Y_{10} + X_{11} + Y_{10} + Y_{11} + Y_{12} + 1, & 0 &= K_{01} + K_{10},\\
0 &= X_{10}Y_{12} + X_{10} + X_{11} + X_{12}Y_{12} + Y_{11} + 1, & 0 &= K_{02} + K_{11},\\
0 &= X_{10}Y_{11} + X_{12} + Y_{10} + Y_{11}, & 0 &= K_{00} + K_{12}.
\end{aligned}
$$

این دستگاه دارای ۲۶ معادله و ۱۵ مجهول و همچنین ۳۱ یکجمله‌ای متمایز است. پایه گروبنر ایده‌ال تولید شده توسط دستگاه معادلات فوق را با استفاده از نرم‌افزار سِیج محاسبه کرده‌ایم که آن‌را در زیر مشاهده می‌کنید.

$$GB = \{K_{00}, K_{01}, K_{02}+1, Y_{10}+1, X_{10}, K_{10}, Z_{10}+1, Y_{11}, X_{11}+1, K_{11}+1, Z_{11}, Y_{12}, X_{12}+1, K_{12}, Z_{12}+1\}$$

همان‌طور که مشاهده می‌شود، به‌راحتی می‌توان پاسخ یکتای دستگاه را از روی پایه گروبنر محاسبه شده به‌دست آورد. به این ترتیب جواب دستگاه عبارت‌است از

$$\{K_{00} = 0, K_{01} = 0, K_{02} = 1, Y_{10} = 1, X_{10} = 0, K_{10} = 0, Z_{10} = 1, Y_{11} = 0, X_{11} = 1, K_{11} = 1,$$

$$Z_{11} = 0, Y_{12} = 0, X_{12} = 1, K_{12} = 0, Z_{12} = 1\}$$

و در نتیجه کلید سامانه برابر است با $(K_{00}, K_{01}, K_{02}) = (0, 0, 1)$. باید توجه کرد که به ازای برخی از زوج متن‌های اصلی و رمز شده ممکن است دستگاهی به‌دست آید که جواب یکتا نداشته باشد، برای مثال اگر متن اصلی $P = (0, 0, 0)$ را با همان کلید $K = (0, 0, 1)$، رمز کنیم به متن رمزشده $C = (0, 0, 1)$، می‌رسیم، حال فرض کنید فقط زوج (P, C)، را در اختیار داشته باشیم. این‌بار دستگاهی که از جایگذاری مقادیر متناظر با P و C، به‌دست می‌آید، دارای جواب یکتا نیست بلکه ۴ جواب دارد که بیت‌های کلید در هر چهار جواب، متمایز هستند و لذا برای

	طول کلید	طول قالب داده	تعداد دورها
AES-128	۱۲۸ بیت	۱۲۸ بیت	۱۰
AES-192	۱۹۲ بیت	۱۲۸ بیت	۱۲
AES-256	۲۵۶ بیت	۱۲۸ بیت	۱۴

جدول ۱.۳: پارامترهای نسخه‌های مختلف AES

تشخیص کلید صحیح به زوج متن‌های اصلی و رمزشده بیشتری نیاز داریم. در فصل بعد راجع به روش‌های حل چنین دستگاه‌هایی بیشتر بحث می‌کنیم.

۴.۴.۳ جبری‌سازی AES و SR

معرفی AES

مؤسسه ملی فناوری و استانداردهای ایالات متحده [۵]، با برگزاری یک رقابت چندمرحله‌ای به نام AES، که از سال ۱۹۹۷ تا سال ۲۰۰۰ ادامه داشت، تلاش کرد تا استاندارد جدیدی برای رمزهای قالبی برگزینند. الگوریتم پیروز این مسابقه قرار بود استاندارد رمزنگاری پیشرفته و جایگزین DES باشد. طی یک فراخوان عمومی ۱۵ الگوریتم به مسابقه ارسال شد، از این میان ۵ الگوریتم به مرحله نهایی راه‌یافتند تا این که سرانجام الگوریتمی تحت عنوان راین‌دال [۶]، که توسط دو جوان بلژیکی به نام‌های وینسنت رایمِن [۷] و یوان دیمِن [۸]، به مسابقه ارسال شده بود به عنوان برنده مسابقه اعلام شد. الگوریتم راین‌دال پس از پیروزی در مسابقه‌ی AES، توسط NIST، در نوامبر سال ۲۰۰۱ استانداردسازی و در سند FIPS 197 تحت نام AES عرضه شد. AES، مثل بسیاری از رمزهای قالبی دیگر، دارای دو بخش جداگانه است که عبارت‌اند از الگوریتم استخراج کلید و الگوریتم رمزنگاری. الگوریتم رمزنگاری شامل چند دور است که هر دور آن نیازمند کلیدی مختص آن دور و هم‌اندازه با طول قالب است. وظیفه الگوریتم استخراج کلید این است که با دریافت شاه‌کلید اصلی، کلید هر یک از دورها را تولید کند. ما ابتدا فرض می‌کنیم الگوریتم استخراج کلید کار خود را انجام داده و کلید هر یک از دورها را تولید کرده است، با این فرض الگوریتم رمزنگاری را توضیح می‌دهیم و سپس الگوریتم استخراج کلید را توضیح خواهیم داد. AES، دارای سه نسخه است که در هرسه، طول قالب داده ثابت و برابر ۱۲۸ بیت، ولی طول کلید و تعداد دورها در این سه نسخه متفاوت، ومطابق جدول ۱.۳ است. در ادامه خواهیم دید، برخلاف این‌که طول کلید در نسخه‌های مختلف AES متفاوت است ولی، هر یک از زیرکلیدها یا کلید دورها که توسط الگوریتم استخراج کلید تولید می‌شوند هم‌طول با قالب داده یعنی ۱۲۸ بیتی هستند. برای شرح

[5] NIST
[6] Rijndeal
[7] Vincent Rijmen
[8] Joan Daemen

۰	۴	۸	۱۲
۱	۵	۹	۱۳
۲	۶	۱۰	۱۴
۳	۷	۱۱	۱۵

آرایش زیرکلیدها در AES

۰	۴	۸	۱۲
۱	۵	۹	۱۳
۲	۶	۱۰	۱۴
۳	۷	۱۱	۱۵

ماتریس حالت در AES

شکل ۶.۳: آرایش داده و کلید در AES

عمل‌کرد AES، قالب داده و زیرکلیدها را به صورت آرایه‌های 4×4 از کلمه‌های یک بایتی و با آرایش نشان داده شده در شکل ۶.۳ در نظر می‌گیریم و ماتریس شامل داده در حال رمزنگاری را ماتریس حالت می‌نامیم.

بین مجموعه‌ی بایت‌ها و میدان $\mathbb{GF}(2^8)$ تناظری یک‌به‌یک به‌صورت زیر برقرار است.

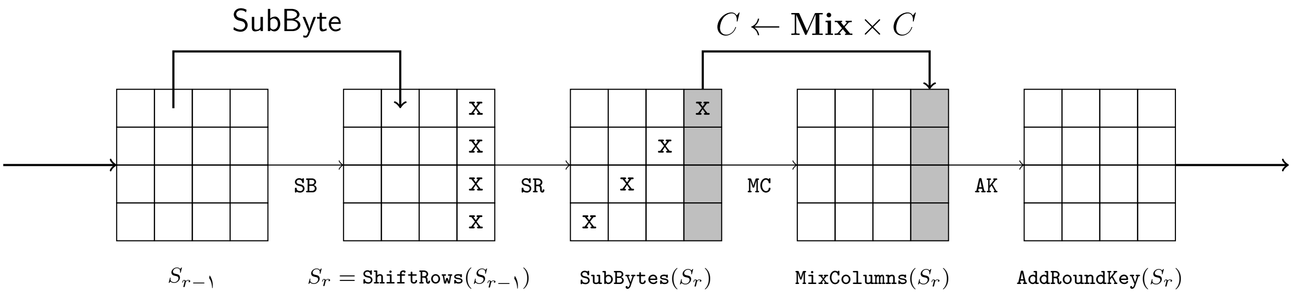
$$\mathbb{F}_2^8 \to \mathbb{GF}(2^8) = \frac{\mathbb{GF}(2)[X]}{\langle X^8 + X^4 + X^3 + X + 1 \rangle} = \mathbb{GF}(2)(\theta)$$
$$(b_7, \ldots, b_0) \mapsto \sum_{i=0}^{7} b_i X^i \mod m(X) = \sum_{i=0}^{7} b_i \theta^i$$

که $m(X) = X^8 + X^4 + X^3 + X + 1 \in \mathbb{GF}(2)[X]$ یک چندجمله‌ای تحویل‌ناپذیر از حلقه $\mathbb{GF}(2)[X]$، موسوم به چندجمله‌ای راین‌دال، و θ یک ریشه آن است. با استفاده از این تناظر می‌توانیم هر بایت را به صورت یک چندجمله‌ای از میدان $\mathbb{GF}(2)(\theta)$ در نظر بگیریم، برای مثال $\texttt{0xD1} = \theta^7 + \theta^6 + \theta^4 + 1$. به این ترتیب ماتریس حالت، عضوی از فضای $\mathrm{Mat}_4(\mathbb{F}_{2^8})$ است. به همین ترتیب کلید هر یک از دورها را نیز می‌توانیم به‌صورت یک ماتریس از $\mathrm{Mat}_4(\mathbb{F}_{2^8})$ در نظر بگیریم. در ادامه با استفاده از این تناظر، یک بیان جبری از سامانه رمزنگاری AES ارائه خواهیم کرد. فرض کنید تعداد دورها را با n نمایش دهیم. در این‌صورت فرآیند رمزنگاری مطابق الگوریتم ۱۳ است. همان‌طور که در شکل ۷.۳ نیز نمایش داده شده، هر دور از حلقه اصلی الگوریتم ۱۳، (به استثنا دور آخر) شامل ۴ عملیات مختلف است که در ادامه هر یک را توضیح می‌دهیم.

`AddRoundKey` در این تابع ماتریس حالت با کلید دور ترکیب می‌شود، به‌این‌صورت که هر بایت از ماتریس ورودی با بایت متناظر از کلید دور، xor می‌شود، و یا به بیان جبری، ماتریس حالت و ماتریس کلید به عنوان دو ماتریس از فضای $\mathrm{Mat}_4(\mathbb{F}_{2^8})$ با هم جمع می‌شوند.

قبل از توضیح مرحله بعد دو مفهوم جبری را تعریف می‌کنیم. در تعاریف زیر فرض کنید $\mathbb{F}$ یک میدان از مرتبه q و K یک توسیع از $\mathbb{F}$ از درجه d باشد.

تعریف ۱۲.۳ (مزدوج‌های یک عضو میدان). فرض کنید $a \in K$، در این‌صورت مزوج‌های a نسبت به $\mathbb{F}$ عبارت‌اند از اعضای مجموعه $\{a^{q^i} : 0 \le i \le d-1\}$.



شکل ۷.۳: نمایش تصویری یک دور از الگوریتم AES

تعریف ۱۳.۳ (q- چندجمله‌ای). چندجمله‌ای $f(x) = \sum_{i=۰}^{d-۱} a_i x^{q^i} \in K[x]$ را که $a_i \in K$، یک q- چندجمله‌ای گوییم. بنابراین وقتی $f(x)$ یک q- چندجمله‌ای باشد، آن‌گاه به ازای هر $a \in K$ مقدار $f(a)$ ترکیب K- خطی از مزدوج‌های a خواهد بود.

یک q ـ چندجمله‌ای مثل $f(x) \in K[x]$ یک نگاشت خطی روی K به عنوان فضای برداری روی $\mathbb{F}$ است. همچنین می‌توان نشان داد که هر نگاشت خطی روی K به عنوان یک فضای برداری روی $\mathbb{F}$ را می‌توان به‌صورت یک q ـ چندجمله‌ای بیان کرد. برای مثال هر نگاشت خطی روی $\mathbb{GF}(۲^۸)$ به عنوان یک فضای برداری روی $\mathbb{GF}(۲)$ را می‌توان به‌صورت $f(X) = \lambda_۰ X^{۲^۰} + \lambda_۱ X^{۲^۱} + \ldots + \lambda_۷ X^{۲^۷}$ نوشت که در آن $\lambda_i \in \mathbb{GF}(۲^۸)$.

`SubByte` در این مرحله هر بایت از ماتریس ورودی از جعبه جانشینی AES عبور کرده و بایت متناظر در ماتریس خروجی را تشکیل می‌دهد. جعبه جانشینی شامل سه عملیات مختلف است که آن‌ها را توضیح می‌دهیم.

- اولین عملیات جعبه جانشینی معکوس گیری توسیع یافته است. به این‌خاطر آن‌را توسیع یافته گوییم که صفر نیز عضو دامنه این نگاشت است و طبق قرارداد آن‌را به‌خودش می‌نگارد. با توجه به این‌که مرتبه میدان $\mathbb{GF}(۲^۸)$ برابر ۲۵۶ است لذا به ازای هر $w \in \mathbb{GF}(۲^۸)\backslash\{۰\}$ داریم $w^{-۱} = w^{۲۵۴}$. همچنین $۰ = ۰^{۲۵۴}$ بنابراین می‌توان این مرحله را با استفاده از نگاشت زیر توصیف کرد.

$$I : \mathbb{GF}(۲^۸) \to \mathbb{GF}(۲^۸)$$
$$w = \sum_{i=۰}^{۷} w_i \theta^i \mapsto x = I(w) = w^{۲۵۴}.$$

- گام دوم اِعمال یک نگاشت $\mathbb{GF}(2)$ ـ خطی به‌صورت زیر است.

$$x = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}$$

می‌دانیم که هر نگاشت خطی روی $\mathbb{GF}(2^8)$ به عنوان یک فضای برداری روی $\mathbb{GF}(2)$ را می‌توان به‌صورت یک ۲ ـ چندجمله‌ای نوشت. ۲ ـ چندجمله‌ای متناظر با نگاشت خطی فوق عبارت‌است از

$$x \mapsto A(x) = \mathtt{05}x^{2^0} + \mathtt{09}x^{2^1} + \mathtt{F9}x^{2^2} + \mathtt{25}x^{2^3} + \mathtt{F4}x^{2^4} + \mathtt{01}x^{2^5} + \mathtt{B5}x^{2^6} + \mathtt{8F}x^{2^7}$$

- گام آخر در جعبه جانشینی افزودن مقدار ثابت است. در این مرحله خروجی گام دوم و مقدار ثابت $\mathbb{GF}(2^8) \ni 1 + \theta + \theta^5 + \theta^6 = \mathtt{63}$ به عنوان اعضای میدان $\mathbb{GF}(2^8)$ با هم جمع می‌شوند و حاصل خروجی جعبه جانشینی خواهد بود.

بنابراین اگر نگاشت جعبه جانشینی را با $S_{RD} : \mathbb{GF}(2^8) \to \mathbb{GF}(2^8)$ نمایش دهیم در این‌صورت داریم $S_{RD}(w) = A(I(w)) + \mathtt{63}$.

`ShiftRows` همان‌طور که در شکل ۷.۳ نمایش داده شده، در این مرحله کلمه‌های سطر iام از آرایه داده به اندازه $0 \le i \le 3$ به سمت چپ انتقال چرخشی پیدا می‌کنند. همان‌طور که مشخص است این مرحله مستقل از تعداد سطرها است و سطر اول در این مرحله بدون تغییر باقی می‌ماند.اگر این انتقال چرخشی سطرها وجود نداشته باشد، کل عملیات رمزنگاری روی ستون‌های ۳۲ بیتی انجام می‌شود، و ما در عمل یک رمز قالبی با طول قالب ۳۲ بیت خواهیم داشت!

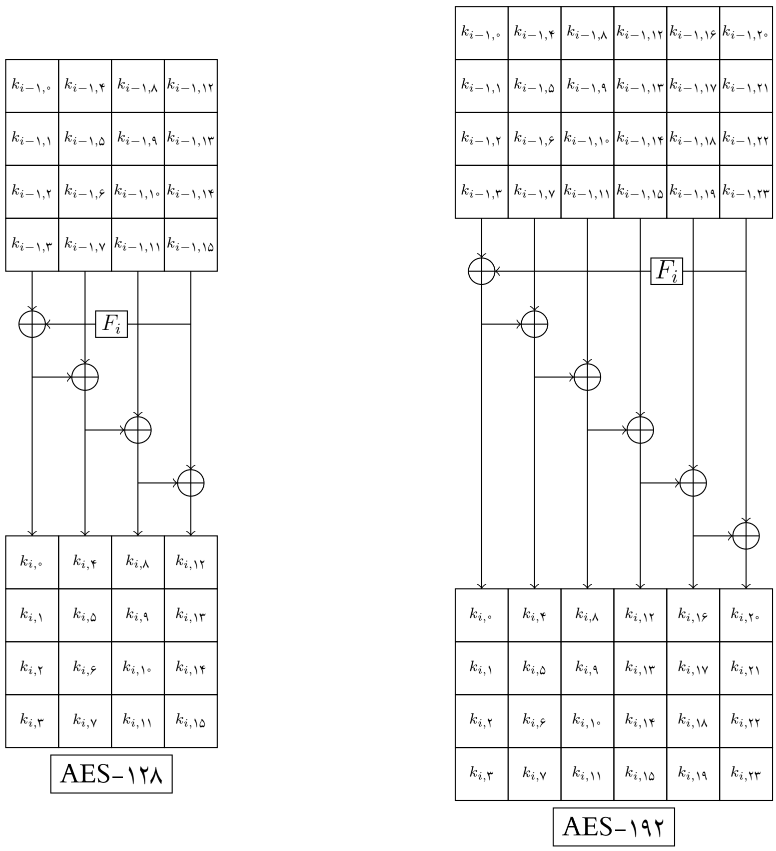
`MixColumns` این تابع هر ستون از ماتریس حالت را به‌طور مستقل از دیگر ستون‌ها تلفیق و درهم‌سازی می‌کند. فرآیند تلفیق بدین ترتیب است که هر ستون از ماتریس حالت، در ماتریس

ثابت

$$\begin{pmatrix} \theta & \theta+۱ & ۱ & ۱ \\ ۱ & \theta & \theta+۱ & ۱ \\ ۱ & ۱ & \theta & \theta+۱ \\ \theta+۱ & ۱ & ۱ & \theta \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \in \mathcal{M}_۴(\mathbb{F}_{۲^۸})$$

ضرب می‌شود تا ستون جدید متناظر، در ماتریس حالت خروجی به‌دست آید.

الگوریتم توسیع یا استخراج کلید وابسته به طول کلید، متفاوت است. در این میان الگوریتم توسیع کلید AES-256 دارای یک تفاوت جزیی نسبت به دو نسخه دیگر است. ما برای سادگی، فقط الگوریتم توسیع کلید نسخه‌های AES-128 و AES-192 را شرح می‌دهیم. کلید هر دور بر اساس کلید دور قبل و به‌صورت نمایش داده شده در شکل ۸.۳ به‌دست می‌آید. تابع F_i در الگوریتم



شکل ۸.۳: یک دور از الگوریتم توسیع کلید AES-128 و AES-192

توسیع کلید شامل یک واحد انتقال چرخشی کلمه‌ها به سمت چپ، عبور از جعبه جانشینی و سپس افزودن یک مقدار ثابت است. برای مثال نحوه عمل‌کرد این عمل‌گر برای AES-128 در

دور $i \in \{۱, \dots, ۱۰\}$ به‌صورت زیر است.

$$T_۰T_۱T_۲T_۳ = F_i(k_{i-۱,۱۲}k_{i-۱,۱۳}k_{i-۱,۱۴}k_{i-۱,۱۵})$$
$$T_۰ = S_{RD}(k_{i-۱,۱۳}) + \theta^{i-۱}, T_۱ = S_{RD}(k_{i-۱,۱۴}), T_۲ = S_{RD}(k_{i-۱,۱۵}), T_۳ = S_{RD}(k_{i-۱,۱۲}).$$

معرفی SR

SR یک رمزقالبی است که توسط مَتیو رابشاو [۹]، سیِن مورفی [۱۰] و کارلس سید [۱۱] در [۱۴]، و به عنوان یک نسخه کوچک‌مقیاس از AES ارائه شد. SR دارای دو نوع است که نوع اول را با $\mathrm{SR}(n,r,c,e)$ و نوع دوم را با $\mathrm{SR}^\star(n,r,c,e)$ نمایش می‌دهیم که تنها تفاوت آن‌ها در دور آخر است. متغیرهای داخل پرانتز پارمترهای SR نام‌دارند و به صورت زیر تعریف می‌شوند.

- n تعداد دورهای رمزنگاری را نشان می‌دهد.
- r تعداد سطرهای ماتریس حالت ورودی است.
- c تعداد ستون‌های ماتریس حالت ورودی است.
- e اندازه (بر حسب بیت) هر یک از کلمه‌ها و یا درایه‌های ماتریس حالت را نشان می‌دهد.

$\mathrm{SR}(n,r,c,e)$ و $\mathrm{SR}^\star(n,r,c,e)$، هر دو دارای $n \in \{۱, \dots, n\}$ دور و طول قالب rce بیت هستند که قالب داده در هر دو آن‌ها به صورت یک آرایه $r \times c$ از کلمه‌های e بیتی در نظر گرفته می‌شود. r و c از مجموعه $\{۱, ۲, ۴\}$ انتخاب می‌شوند که در نتیجه ۹ حالت برای ماتریس حالت وجود دارد. در هر یک از حالات، ترتیب قرار گرفتن داده‌ها در ماتریس حالت را مانند AES، با اولویت پر شدن ستون‌ها در نظر می‌گیریم. برای مثال چند نمونه از ماتریس‌های به ازای r و c های مختلف نشان داده شده است. در ادامه خواهیم دید که $\mathrm{SR}^\star(۱۰, ۴, ۴, ۸)$، همان AES-128 است.

۰

۰
۱

۰	۲
۱	۳

۰	۴
۱	۵
۲	۶
۳	۷

۰	۲	۴	۶
۱	۳	۵	۷

۰	۴	۸	۱۲
۱	۵	۹	۱۳
۲	۶	۱۰	۱۴
۳	۷	۱۱	۱۵

طول کلمه‌ها (بر حسب بیت) یعنی پارامتر e در SR یا $\mathrm{SR}^\star$، از مجموعه $\{۴, ۸\}$ انتخاب می‌شود. همان‌طور که در معرفی AES اشاره کردیم، می‌توانیم هر کلمه e بیتی را به عنوان عضوی از میدان $\mathbb{GF}(۲^e)$ در نظر بگیریم. اگر $e = ۴$ باشد از چندجمله‌ای تحویل ناپذیر $X^۴ + X +$

[9]Matthew J. B. Robshaw

[10]Sean Murphy

[11]Carlos Cid

$\mathbb{GF}(۲)[X] \in ۱$ برای تعریف میدان $\mathbb{GF}(۲^۴)$ استفاده می‌کنیم. فرض کنید ρ ریشه چندجمله‌ای تحویل‌ناپذیر مذکور باشد، در این صورت

$$\mathbb{GF}(۲^۴) = \frac{\mathbb{GF}(۲)[X]}{\langle X^۴ + X + ۱\rangle} = \mathbb{GF}(۲)(\rho).$$

اگر $e = ۸$ باشد از میدان $\mathbb{GF}(۲^۸)$ در $\mathrm{SR}(n, r, c, ۸)$ و $\mathrm{SR}^\star(n, r, c, ۸)$ استفاده می‌کنیم. این میدان با همان چندجمله‌ای تحویل‌ناپذیر راین‌دال که در AES معرفی شد، تعریف می‌شود. به این ترتیب اگر فرض کنیم θ ریشه چندجمله‌ای تحویل‌ناپذیر راین‌دال باشد داریم

$$\mathbb{GF}(۲^۸) = \frac{\mathbb{GF}(۲)[X]}{\langle X^۴ + X^۴ + X^۳ + X + ۱\rangle} = \mathbb{GF}(۲)(\theta).$$

هر دور از SR به مانند دورهای میانی AES از چهار عملیات `SubByte`، `ShiftRows`، `MixColumns` و `AddRoundKey` تشکیل شده است. تنها تفاوت $\mathrm{SR}^\star$ با SR در این است که دور آخر $\mathrm{SR}^\star$ مانند AES فاقد عملگر `MixColumns` است و در نتیجه $\mathrm{AES} = \mathrm{SR}^\star(۱۰, ۴, ۴, ۸)$.

با توجه به این‌که متن رمزی حاصل از رمزکردن یک متن اصلی دلخواه با استفاده از یک کلید دلخواه، تحت $\mathrm{SR}(n, r, c, e)$ و $\mathrm{SR}^\star(n, r, c, e)$ با یک نگاشت خطی به هم مرتبط هستند و یک جواب دستگاه معادلات به‌دست آمده از $\mathrm{SR}^\star$ به‌راحتی قابل تبدیل به جوابی برای دستگاه به‌دست آمده از SR است لذا بدون کاستن از کلیت، در ادامه فقط به شرح جزئیات $\mathrm{SR}(n, r, c, e)$ می‌پردازیم.

`SubByte` عملیات `SubByte` در SR مانند AES از عبور کلمه‌های ماتریس حالت از جعبه جانشینی تشکیل شده است. جعبه جانشینی در $\mathrm{SR}(n, r, c, ۸)$ هماج جعبه جانشینی AES است. بنابراین در این قسمت فقط به معرفی جعبه جانشینی $\mathrm{SR}(n, r, c, ۴)$ می‌پردازیم. جعبه جانشینی $\mathrm{SR}(n, r, c, ۴)$ مانند جعبه جانشینی AES از سه قسمت به شرح زیر تشکیل شده است.

- اولین عملیاتی که در S-Box صورت می‌گیرد، معکوس‌گیری در میدان $\mathbb{GF}(۲^۴)$ است. در ضمن اگر ورودی صفر بود، آن‌را به صفر می‌نگاریم.

- در گام دوم هر یک خروجی‌های مرحله معکوس‌گیری وارد نگاشت $\mathbb{GF}(۲)$ ـ خطی می‌شوند

که با استفاده از ماتریس زیر تعریف می‌شود.

$$x = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \mapsto \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

این نگاشت $\mathbb{GF}(2)$ ـ خطی را می‌توان با ۲ ـ چندجمله‌ای $A(x) = \lambda_0 x^{2^0} + \lambda_1 x^{2^1} + \lambda_2 x^{2^2} + \lambda_3 x^{2^3}$ از $\mathbb{GF}(2^8)[x]$ که $(\lambda_0, \lambda_1, \lambda_2, \lambda_3) = (5, 1, C, 5)$ نمایش داد.

- مرحله آخر S-Box عبارت است از افزودن مقدار ثابت. در این مرحله مقدار ثابت 6 (یا به‌طور معادل $\rho^2 + \rho$) به خروجی مرحله نگاشت $\mathbb{GF}(2)$ ـ خطی افزوده می‌شود، که نتیجه آن خروجی کل S-Box خواهد بود.

خلاصه اطلاعات جعبه جانشینی	$\mathbb{GF}(2^4)$	$\mathbb{GF}(2^8)$
چندجمله‌ای تحویل‌ناپذیر	$X^4 + X + 1$	$X^8 + X^4 + X^3 + X + 1$
نگاشت خطی	$L_4 = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}$	$L_8 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$
مقدار ثابت	$\texttt{0x6} = \rho^2 + \rho$	$\texttt{0x63} = \theta^6 + \theta^5 + \theta + 1$

جدول ۳.۲: نگاشت‌های جعبه جانشینی در SR

ShiftRows همان‌طور که در معرفی AES ذکر شد، در این مرحله کلمه‌های سطر iام از آرایه داده به اندازه $0 \le i \le r-1$ به سمت چپ انتقال چرخشی پیدا می‌کند. همان‌طور که مشخص است این مرحله مستقل از تعداد سطرها است و سطر اول در این مرحله بدون تغییر باقی می‌ماند.

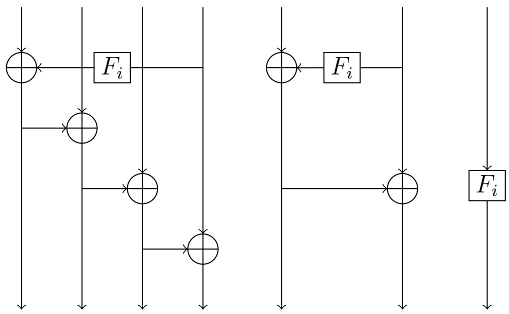
MixColumns در این مرحله هر یک از ستون‌های ماتریس حالت در یک ماتریس معکوس‌پذیر ضرب می‌شود. این ماتریس به تعداد سطرهای ماتریس حالت وابسته و به‌صورت نشان داده شده در جدول ۳.۳ است.

تعداد سطرها	$\mathbb{GF}(2^4)$	$\mathbb{GF}(2^8)$
$r=1$	(1)	(1)
$r=2$	$\begin{pmatrix} \rho & \rho \\ \rho & \rho+1 \end{pmatrix}$	$\begin{pmatrix} \theta+1 & \theta \\ \theta & \theta+1 \end{pmatrix}$
$r=4$	$\begin{pmatrix} \rho & \rho+1 & 1 & 1 \\ 1 & \rho & \rho+1 & 1 \\ 1 & 1 & \rho & \rho+1 \\ \rho+1 & 1 & 1 & \rho \end{pmatrix}$	$\begin{pmatrix} \theta & \theta+1 & 1 & 1 \\ 1 & \theta & \theta+1 & 1 \\ 1 & 1 & \theta & \theta+1 \\ \theta+1 & 1 & 1 & \theta \end{pmatrix}$

جدول ۳.۳: ماتریس `MixColumns` در SR

`AddRoundKey` الگوریتم توسیع یا استخراج کلید در $\mathrm{SR}(n,r,c,e)$، با دریافت کلید اصلی، $n+1$ زیرکلید برای هر یک از دورها تولید می‌کند که در مرحله `AddRoundKey`، هر کلمه از این زیرکلیدها با کلمه متناظر در ماتریس حالت (به عنوان عضوی از $\mathbb{GF}(2^e)$) جمع می‌شود. الگوریتم رمزنگاری در SR مانند AES با `AddRoundKey` آغاز می‌شود.

الگوریتم توسیع کلید در SR با الهام از الگوریتم توسیع کلید در AES طراحی شده است. کلید



شکل ۹.۳: یک دور از الگوریتم توسیع کلید SR

اصلی $\mathrm{SR}(n,r,c,e)$ یک کلید rce بیتی است که به‌صورت یک آرایه $r\times c$ از کلمه‌های e بیتی در نظر گرفته می‌شود. کلید هر دور بر اساس کلید دور قبل و به‌صورت نشان داده شده در شکل ۹.۳ به‌دست می‌آید. تابع F_i در شکل ۹.۳ یک تابع غیر خطی و در واقع نسخه کوچک‌مقیاس از تابع F_i در الگوریتم توسیع کلید AES است. این تابع شامل یک انتقال چرخشی کلمات به سمت چپ، اعمال جعبه جانشینی و افزودن یک مقدار ثابت است که در الگوریتم توسیع کلید AES شرح داده شد، تنها تفاوت آن با الگوریتم توسیع کلید AES در ثابت‌ها و نگاشت‌های تشکیل‌دهنده جعبه جانشینی است که به‌طور خلاصه در جدول ۴.۳ آمده است. جزئیات بیشتر راجع به الگوریتم توسیع کلید در بخش استخراج معادلات الگوریتم توسیع کلید آمده است. اکنون آماده‌ایم تا به استخراج معادلات AES و SR بپردازیم. در استخراج معادلات دو رویکرد وجود دارد. در رویکرد اول معادلات روی $\mathbb{GF}(2)$ و در رویکرد دوم معادلات روی $\mathbb{GF}(2^e)$ استخراج می‌شوند که ما در ادامه نحوه استخراج معادلات بر اساس رویکرد اول را شرح می‌دهیم.

		$\mathbb{GF}(۲^۴)$	$\mathbb{GF}(۲^۸)$
ثابت دور iام که $۱ \le i \le n$.	κ_i	$\rho^{i-۱}$	$\theta^{i-۱}$
ثابت جعبه جانشینی	d	0x6	0x63
معکوس‌گیری در جعبه جانشینی	$z \mapsto z^{-۱}$	معکوس‌گیری در $\mathbb{GF}(۲^۴)$	معکوسگیری در $\mathbb{GF}(۲^۸)$
نگاشت خطی در جعبه جانشینی	$z \mapsto L(z)$	نگاشت خطی $\mathbb{GF}(۲^۴)$	نگاشت خطی $\mathbb{GF}(۲^۸)$

جدول ۴.۳: ثابت‌ها و توابع مورد استفاده در الگوریتم توسیع کلید $\mathrm{SR}(n, r, c, e)$

استخراج معادلات SR و AES روی $\mathbb{GF}(۲)$

می‌دانیم SR یک خانواده پارامتری است که AES-128 حالت خاصی از آن است، لذا بدون کاستن از کلیت، بحث را با $\mathrm{SR}(n, r, c, e)$ ادامه می‌دهیم. تا کنون برای توضیح AES و SR، قالب داده و زیرکلیدها را به‌صورت آرایه‌های دو بعدی در نظر گرفتیم، ولی در این بخش برای استخراج معادلات حاکم بر $\mathrm{SR}(n, r, c, e)$ قالب داده و زیرکلیدها را به‌صورت بردارهای ستونی rce بیتی در نظر می‌گیریم.

همان‌طور که در بخش قبل مشاهده شد، الگوریتم رمز قالبی SR دارای دو بخش است که عبارتند از بخش توسیع کلید و بخش رمزنگاری، که معادلات هر یک از این بخش‌ها را جداگانه استخراج خواهیم کرد. در استخراج معادلات بخش رمزنگاری فرض می‌کنیم متن اصلی و رمزشده متناظر با آن معلوم هستند و لذا آن‌ها را به عنوان مقادیر ثابت معادلات در نظر می‌گیریم. مجهولات یا متغیرهای معادله‌های به‌دست آمده از مرحله رمزنگاری، عبارتند از بیت‌های کلید به‌علاوه بیت‌های داده در مراحل میانی الگوریتم رمزنگاری. به‌این ترتیب با استفاده از هر زوج متن اصلی‐ رمزشده یک دستگاه متفاوت برای مرحله رمزنگاری به‌دست می‌آید. دستگاه معادلات استخراج شده برای الگوریتم توسیع کلید فقط بر حسب بیت‌های کلید و متغیرهای جاری وابسته به بیت‌های کلید خواهد بود و لذا این دستگاه معادلات، مستقل از زوج متن‌های اصلی‐ رمزشده که در اختیار داریم برای تمام رمزنگاری‌ها تحت یک کلید، مشترک خواهد بود.

در دستگاه معادلات بخش رمزنگاری، متغیرهای متناظر با بیت‌های ورودی و خروجی مرحله معکوس‌گیری در جعبه جانشینی را به‌ترتیب با w_{ijl} و x_{ijl} و بیت‌های کلید را با k_{ijl} نمایش می‌دهیم که i شماره دور، j شماره کلمه و l شماره بیت در آن کلمه است. در ادامه برای سهولت گاهی از دو و گاهی فقط از یک اندیس استفاده می‌کنیم. برای مثال x_{ij} نشان‌دهنده کلمه j ام از خروجی مرحله معکوس‌گیری دور i ام است، به همین ترتیب x_i نشان‌دهنده بردار خروجی مرحله معکوس‌گیری دور iام است.

تنها مرحله غیر آفین در تابع دور SR مرحله معکوس‌گیری در جعبه جانشینی است که ابتدا نحوه استخراج معادلات در این قسمت را شرح می‌دهیم. قالب داده در هر دور $\mathrm{SR}(n, r, c, e)$

به صورت بسته‌های e بیتی وارد جعبه جانشینی می‌شود. بنابراین کافی است که نحوه استخراج روابط چندجمله‌ای بین ورودی و خروجی مرحله معکوس‌گیری، به ازای یک ورودی دلخواه e بیتی را بدانیم. فرض کنید بردار ورودی و خروجی مرحله معکوس‌گیری را به ترتیب با $w = (w_0, \dots, w_{e-1})^T$ و $x = (x_0, \dots, x_{e-1})^T$ نمایش دهیم. همچنین فرض کنید چندجمله‌ای‌های متناظر با w و x در $\mathbb{GF}(2^e)$ را با $x = \sum_{i=0}^{e-1} x_i t^i$ و $y = \sum_{j=0}^{e-1} y_j t^j$ نمایش دهیم که t ریشه‌ی چندجمله‌ای تحویل‌ناپذیر مورد نظر در $\mathbb{GF}(2)[x]$ برای تعریف $\mathbb{GF}(2^e)$ است. در این‌صورت به ازای هر $w \neq (0, \dots, 0)$ داریم

$$x = I(w) \Rightarrow (\sum_{i=0}^{e-1} x_i t^i) \cdot (\sum_{j=0}^{e-1} y_j t^j) = \sum_{k=1}^{e-1} 0 \cdot t^k + 1.$$

با توجه به این‌که ضرایب چندجمله‌ای طرف راست و چپ رابطه فوق باید با هم برابر باشند، e رابطه‌ی چندجمله‌ای بین متغیرهای w_i و x_j به‌دست می‌آید. برای مثال در AES که $e = 8$ این معادلات را با استفاده از نرم‌افزار سیج به‌دست آورده‌ایم که در زیر مشاهده می‌کنید.

$$\begin{aligned}
1 =& w_0x_0 + w_1x_7 + w_2x_6 + w_3x_5 + w_4x_4\\
&+ w_5x_3 + w_5x_7 + w_6x_2 + w_6x_6 + w_6x_7\\
&+ w_7x_1 + w_7x_5 + w_7x_6\\
0 =& w_0x_1 + w_1x_0 + w_1x_7 + w_2x_6 + w_2x_7\\
&+ w_3x_5 + w_3x_6 + w_4x_4 + w_4x_5 + w_5x_3\\
&+ w_5x_4 + w_5x_7 + w_6x_2 + w_6x_3 + w_6x_6\\
&+ w_7x_1 + w_7x_2 + w_7x_5 + w_7x_7\\
0 =& w_0x_2 + w_1x_1 + w_2x_0 + w_2x_7 + w_3x_6\\
&+ w_3x_7 + w_4x_5 + w_4x_6 + w_5x_4 + w_5x_5\\
&+ w_6x_3 + w_6x_4 + w_6x_7 + w_7x_2 + w_7x_3\\
&+ w_7x_6\\
0 =& w_0x_3 + w_1x_2 + w_1x_7 + w_2x_1 + w_2x_6\\
&+ w_3x_0 + w_3x_5 + w_3x_7 + w_4x_4 + w_4x_6\\
&+ w_4x_7 + w_5x_3 + w_5x_5 + w_5x_6 + w_5x_7\\
&+ w_6x_2 + w_6x_4 + w_6x_5 + w_6x_6 + w_6x_7\\
&w_7x_1 + w_7x_3 + w_7x_4 + w_7x_5 + w_7x_6\\
&+ w_7x_7
\end{aligned}$$

$$\begin{aligned}
0 =& w_0x_4 + w_1x_3 + w_1x_7 + w_2x_2 + w_2x_6\\
&+ w_2x_7 + w_3x_1 + w_3x_5 + w_3x_6 + w_4x_0\\
&+ w_4x_4 + w_4x_5 + w_4x_7 + w_5x_3 + w_5x_4\\
&+ w_5x_6 + w_6x_2 + w_6x_3 + w_6x_5 + w_7x_1\\
&+ w_7x_2 + w_7x_4 + w_7x_7\\
0 =& w_0x_5 + w_1x_4 + w_2x_3 + w_2x_7 + w_3x_2\\
&+ w_3x_6 + w_3x_7 + w_4x_1 + w_4x_5 + w_4x_6\\
&+ w_5x_0 + w_5x_4 + w_5x_5 + w_5x_7 + w_6x_3\\
&+ w_6x_4 + w_6x_6 + w_7x_2 + w_7x_3 + w_7x_5\\
0 =& w_0x_6 + w_1x_5 + w_2x_4 + w_3x_3 + w_3x_7\\
&+ w_4x_2 + w_4x_6 + w_4x_7 + w_5x_1 + w_5x_5\\
&+ w_5x_6 + w_6x_0 + w_6x_4 + w_6x_5 + w_6x_7\\
&+ w_7x_3 + w_7x_4 + w_7x_6\\
0 =& w_0x_7 + w_1x_6 + w_2x_5 + w_3x_4 + w_4x_3\\
&+ w_4x_7 + w_5x_2 + w_5x_6 + w_5x_7 + w_6x_1\\
&+ w_6x_5 + w_6x_6 + w_7x_0 + w_7x_4 + w_7x_5\\
&+ w_7x_7
\end{aligned}$$

تمام روابط فوق به جز معادله اول که مقدار ثابت سمت راست آن ۱ است، به ازای همه مقادیر $w, x \in \mathbb{GF}(2)^e$ برقرار و لذا با احتمال ۱ درست هستند. معادله اول که مقدار ثابت سمت چپ آن

۱ است زمانی برقرار است که $w \neq (۰, \ldots, ۰)^T$، بنابراین این رابطه با احتمال $\frac{۲۵۵}{۲۵۶}$ درست است. اگر ورودی و خروجی نگاشت معکوس‌گیری را به‌صورت چندجمله‌اهای $w, x \in \mathbb{GF}(۲^e)$ در نظر بگیریم داریم $xw = ۱$. حال اگر طرفین رابطه $xw = ۱$ را در w (یا x) ضرب کنیم، به معادله $w = xw^۲$ (یا $x = x^۲w$) می‌رسیم که به ازای هر $w \in \mathbb{GF}(۲^e)$ برقرار است. بنابراین با ضرب پی‌درپی طرفین $xw = ۱$ در x (و یا y) روابط زیر به‌دست می‌آید.

$$\forall w \in \mathbb{GF}(۲^e) : \begin{cases} x = x^۲w \\ x^۲ = x^۴w^۲ \\ \vdots \\ x^{۲^{e-۱}} = x^{۲^e}w^{۲^{e-۱}} = xw^{۲^{e-۱}} \end{cases} \qquad \forall w \in \mathbb{GF}(۲^e) : \begin{cases} w = xw^۲ \\ w^۲ = x^۲w^۴ \\ \vdots \\ w^{۲^{e-۱}} = x^{۲^{e-۱}}w^{۲^e} = x^{۲^{e-۱}}w \end{cases}$$

با برابر قرار دادن ضرایب چندجمله‌ای‌های طرفین هر یک رابطه‌های $w^{۲^{e-۱}} = x^{۲^{e-۱}}w$ و $x^{۲^{e-۱}} = xw^{۲^{e-۱}}$ در مجموع $۲e$ معادله چندجمله‌ای دیگر به‌دست می‌آید. بنابراین در مجموع $۳e$ معادله به‌دست می‌آید که یکی از این معادلات که ثابت سمت چپ آن ۱ است زمانی درست است که $w \neq ۰$، و در نتیجه با احتمال $۱ - \frac{۱}{۲^e}$ و مابقی با احتمال ۱ درست هستند. در AES که $e = ۸$، با برابر قرار

دادن ضرایب چندجمله‌ای‌های طرفین رابطه $w^{128} = x^{128}w$ معادلات زیر به‌دست می‌آید.

$$0 = w_0x_0 + w_0x_3 + w_0x_5 + w_0x_7 + w_0 + w_1x_1 + w_1x_3 + w_1x_5 + w_1x_7 + w_2x_1 + w_2x_3 + w_2x_5 + w_3x_1 + w_3x_3 + w_3 + w_4x_1 + w_4x_7 + w_5x_5 + w_5x_6 + w_5x_7 + w_5 + w_6x_3 + w_6x_4 + w_6x_5 + w_6x_7 + w_7x_1 + w_7x_2 + w_7x_3 + w_7x_5 + w_7$$

$$0 = w_0x_1 + w_0x_2 + w_0x_3 + w_1x_0 + w_1x_1 + w_1 + w_2x_7 + w_2 + w_3x_5 + w_3 + w_4x_3 + w_4x_7 + w_5x_1 + w_5x_5 + w_5x_6 + w_6x_3 + w_6x_4 + w_6x_6 + w_7x_1 + w_7x_2 + w_7x_4 + w_7x_7$$

$$0 = w_0x_3 + w_0x_4 + w_0x_5 + w_1x_1 + w_1x_2 + w_1x_3 + w_2x_0 + w_2x_1 + w_3x_7 + w_3 + w_4x_5 + w_4 + w_5x_3 + w_5x_7 + w_5 + w_6x_1 + w_6x_5 + w_6x_6 + w_7x_3 + w_7x_4 + w_7x_6$$

$$0 = w_0x_1 + w_0x_3 + w_0x_6 + w_1x_1 + w_1x_4 + w_1x_7 + w_1 + w_2x_2 + w_2x_5 + w_3x_0 + w_3x_3 + w_3 + w_4x_1 + w_5x_6 + w_5x_7 + w_6x_4 + w_6x_5 + w_6 + w_7x_2 + w_7x_3 + w_7x_6$$

$$0 = w_0x_1 + w_0x_7 + w_1x_5 + w_1x_6 + w_1x_7 + w_1 + w_2x_3 + w_2x_4 + w_2x_5 + w_2x_7 + w_3x_1 + w_3x_2 + w_3x_3 + w_3x_5 + w_4x_0 + w_4x_1 + w_4x_3 + w_4x_7 + w_5x_1 + w_5x_5 + w_5x_6 + w_5x_7 + w_6x_3 + w_6x_4 + w_6x_5 + w_6x_6 + w_7x_1 + w_7x_2 + w_7x_3 + w_7x_4 + w_7$$

$$0 = w_0x_1 + w_0x_3 + w_1x_1 + w_1x_7 + w_1 + w_2x_5 + w_2x_6 + w_2x_7 + w_3x_3 + w_3x_4 + w_3x_5 + w_3x_7 + w_3 + w_4x_1 + w_4x_2 + w_4x_3 + w_4x_5 + w_5x_0 + w_5x_1 + w_5x_3 + w_5x_7 + w_6x_1 + w_6x_5 + w_6x_6 + w_6x_7 + w_7x_3 + w_7x_4 + w_7x_5 + w_7x_6$$

$$0 = w_0x_1 + w_0x_3 + w_0x_5 + w_1x_1 + w_1x_3 + w_1 + w_2x_1 + w_2x_7 + w_3x_5 + w_3x_6 + w_3x_7 + w_3 + w_4x_3 + w_4x_4 + w_4x_5 + w_4x_7 + w_5x_1 + w_5x_2 + w_5x_3 + w_5x_5 + w_5 + w_6x_0 + w_6x_1 + w_6x_3 + w_6x_7 + w_7x_1 + w_7x_5 + w_7x_6 + w_7x_7$$

$$0 = w_0x_1 + w_0x_3 + w_0x_5 + w_0x_7 + w_1x_1 + w_1x_3 + w_1x_5 + w_1 + w_2x_1 + w_2x_3 + w_3x_1 + w_3x_7 + w_3 + w_4x_5 + w_4x_6 + w_4x_7 + w_5x_3 + w_5x_4 + w_5x_5 + w_5x_7 + w_5 + w_6x_1 + w_6x_2 + w_6x_3 + w_6x_5 + w_7x_0 + w_7x_1 + w_7x_3 + w_7x_7 + w_7$$

همچنین با برابر قرار دادن ضرایب چندجمله‌ای‌های طرفین رابطه $x^{128} = xw^{128}$ به معادله‌های زیر

دست می‌یابیم.

$$
\begin{aligned}
0 =& w_0x_0 + w_1x_1 + w_1x_2 + w_1x_3 + w_1x_4 +\\
& w_1x_7 + w_2x_7 + w_3x_0 + w_3x_1 + w_3x_2 +\\
& w_3x_3 + w_3x_6 + w_3x_7 + w_4x_6 + w_5x_0 +\\
& w_5x_1 + w_5x_2 + w_5x_5 + w_5x_6 + w_5x_7 +\\
& w_6x_5 + w_7x_0 + w_7x_1 + w_7x_4 + w_7x_5 +\\
& w_7x_6 + x_0 + x_3 + x_5 + x_7\\
0 =& w_0x_1 + w_1x_0 + w_1x_1 + w_1x_5 + w_1x_7 +\\
& w_2x_0 + w_2x_7 + w_3x_0 + w_3x_4 + w_3x_6 +\\
& w_4x_6 + w_4x_7 + w_5x_3 + w_5x_5 + w_6x_5 +\\
& w_6x_6 + w_7x_2 + w_7x_4 + w_7x_7 + x_1 + x_2 +\\
& x_3\\
0 =& w_0x_2 + w_1x_1 + w_1x_2 + w_1x_6 + w_2x_1 +\\
& w_3x_0 + w_3x_1 + w_3x_5 + w_3x_7 + w_4x_0 +\\
& w_4x_7 + w_5x_0 + w_5x_4 + w_5x_6 + w_6x_6 +\\
& w_6x_7 + w_7x_3 + w_7x_5 + x_3 + x_4 + x_5\\
0 =& w_0x_3 + w_1x_0 + w_1x_1 + w_1x_4 + w_2x_2 +\\
& w_2x_7 + w_3x_0 + w_3x_3 + w_3x_7 + w_4x_1 +\\
& w_4x_6 + w_5x_2 + w_5x_6 + w_6x_0 + w_6x_5 +\\
& w_6x_7 + w_7x_1 + w_7x_5 + x_1 + x_3 + x_6
\end{aligned}
$$

$$
\begin{aligned}
0 =& w_0x_4 + w_1x_0 + w_1x_3 + w_1x_4 + w_1x_5 +\\
& w_1x_7 + w_2x_3 + w_2x_7 + w_3x_2 + w_3x_3 +\\
& w_3x_4 + w_3x_6 + w_3x_7 + w_4x_2 + w_4x_6 +\\
& w_4x_7 + w_5x_1 + w_5x_2 + w_5x_3 + w_5x_5 +\\
& w_5x_6 + w_6x_1 + w_6x_5 + w_6x_6 + w_7x_0 +\\
& w_7x_1 + w_7x_2 + w_7x_4 + w_7x_5 + x_1 + x_7\\
0 =& w_0x_5 + w_1x_0 + w_1x_1 + w_1x_4 + w_1x_5 +\\
& w_1x_6 + w_2x_4 + w_3x_0 + w_3x_3 + w_3x_4 +\\
& w_3x_5 + w_3x_7 + w_4x_3 + w_4x_7 + w_5x_2 +\\
& w_5x_3 + w_5x_4 + w_5x_6 + w_5x_7 + w_6x_2 +\\
& w_6x_6 + w_6x_7 + w_7x_1 + w_7x_2 + w_7x_3 +\\
& w_7x_5 + w_7x_6 + x_1 + x_3\\
0 =& w_0x_6 + w_1x_0 + w_1x_1 + w_1x_2 + w_1x_5 +\\
& w_1x_6 + w_1x_7 + w_2x_5 + w_3x_0 + w_3x_1 +\\
& w_3x_4 + w_3x_5 + w_3x_6 + w_4x_4 + w_5x_0 +\\
& w_5x_3 + w_5x_4 + w_5x_5 + w_5x_7 + w_6x_3 +\\
& w_6x_7 + w_7x_2 + w_7x_3 + w_7x_4 + w_7x_6 +\\
& w_7x_7 + x_1 + x_3 + x_5\\
0 =& w_0x_7 + w_1x_0 + w_1x_1 + w_1x_2 + w_1x_3 +\\
& w_1x_6 + w_1x_7 + w_2x_6 + w_3x_0 + w_3x_1 +\\
& w_3x_2 + w_3x_5 + w_3x_6 + w_3x_7 + w_4x_5 +\\
& w_5x_0 + w_5x_1 + w_5x_4 + w_5x_5 + w_5x_6 +\\
& w_6x_4 + w_7x_0 + w_7x_3 + w_7x_4 + w_7x_5 +\\
& w_7x_7 + x_1 + x_3 + x_5 + x_7
\end{aligned}
$$

به این‌ترتیب در مجموع ۲۴ معادله مربعی برای مرحله معکوس‌گیری توسیع‌یافته در AES به‌دست می‌آید، که ۲۳تای آن‌ها با احتمال ۱ و آن‌معادله‌ای که مقدار ثابت سمت چپ آن ۱ است با احتمال $\frac{255}{256}$ درست است. کورتوا و پیپشیک در [۲۲] نشان دادند که ۲۳ معادله‌ای که با احتمال ۱ درست هستند، مستقل خطی‌هستند یا به عبارت دیگر ماتریس ضرایب آن‌ها تحت یک ترتیب یکجمله‌ای دلخواه معکوس‌پذیر است.

در ادامه نشان می‌دهیم که همه‌ی مراحل بعد از مرحله معکوس‌گیری در تابع دور SR را می‌توان با یک نگاشت آفین بیان کرد و به این ترتیب کار استخراج معادلات به سهولت صورت می‌گیرد. مرحله بعد از معکوس‌گیری عبارت‌است از نگاشت خطی جعبه جانشینی. در بخش‌های قبل

ماتریس متناظر با این نگاشت خطی که در کلمه‌های e بیتی ضرب می‌شد را در جدول ۲.۳ با L_e نمایش دادیم، به دلیل این که قالب داده را در این قسمت به‌صورت یک بردار ستونی rce بیتی در نظر گرفته‌ایم می‌توانیم ماتریس متناظر با نگاشت خطی جعبه جانشینی را با یک ماتریس بلوکی قطری به‌صورت زیر نمایش دهیم.

$$L = \begin{pmatrix} [L_e]_{e\times e} & [0]_{e\times e} & \cdots & [0]_{e\times e} \\ [0]_{e\times e} & [L_e]_{e\times e} & \cdots & [0]_{e\times e} \\ \vdots & \vdots & \vdots & \vdots \\ [0]_{e\times e} & [0]_{e\times e} & \cdots & [L_e]_{e\times e} \end{pmatrix}_{rce\times rce}$$

آخرین مرحله در جعبه جانشینی افزودن مقدار ثابت $d \in \mathbb{GF}(2)^8$ است که با توجه به قالب در نظر گرفته شده در این قسمت، آن‌را با بردار ستونی $\mathbf{d}$ که حاصل rc بار تکرار معدل باینری ثابت d در یک ستون است تعویض می‌کنیم.

مرحله بعد از جعبه جانشینی، انتقال چرخشی است. می‌دانیم که برداهای سطری آرایه دوبعدی داده در $\mathrm{SR}(n, r, c, e)$ که در معرفی SR شرح داده شد، اعضای فضای بردای c بعدی روی $\mathbb{GF}(2^e)$ هستند، به این ترتیب انتقال چرخشی یک واحدی کلمه‌های e بیتی یک سطر از آرایه دو بعدی قالب داده به سمت چپ، معادل است با ضرب شدن آن سطر در ماتریس $R \in \mathrm{Mat}_2(\mathbb{GF}(2^e))$ که نسبت به پایه استاندارد به‌صورت زیر تعریف می‌شود.

$$\hat{R} = \begin{pmatrix} 0 & 1_{\mathbb{GF}(2^e)} & 0 & 0 \\ 0 & 0 & 1_{\mathbb{GF}(2^e)} & 0 \\ 0 & 0 & 0 & 1_{\mathbb{GF}(2^e)} \\ 1_{\mathbb{GF}(2^e)} & 0 & 0 & 0 \end{pmatrix}$$

حال فرض کنید قالب داده در $\mathrm{SR}(n, r, c, e)$ را به‌صورت بردارهای rc تایی از اعضای $\mathbb{GF}(2^e)$ در نظر بگیریم و نسبت به پایه استاندارد به‌صورت زیر نمایش دهیم.

$$\hat{S} = (S_0, S_1, \ldots, S_{rc-1})^T \in \mathbb{GF}(2^e)^{rc}$$

اگر پایه فضای برداری حاوی بردارهای داده در SR را طوری تغییر دهیم که بردار فوق نسبت به آن پایه به‌صورت زیر نمایش داده شود

$$S = (S_0, S_r, \ldots, S_{(c-1)\cdot r}, S_1, S_{r+1}, \ldots, S_{(c-1)\cdot r+1}, \ldots, S_{r-1}, S_{2\cdot(r-1)+1}, \ldots, S_{rc-1})$$

آن‌گاه انتقال چرخشی معادل است با ضرب ماتریس قطری بلوکی زیر در بردار S.

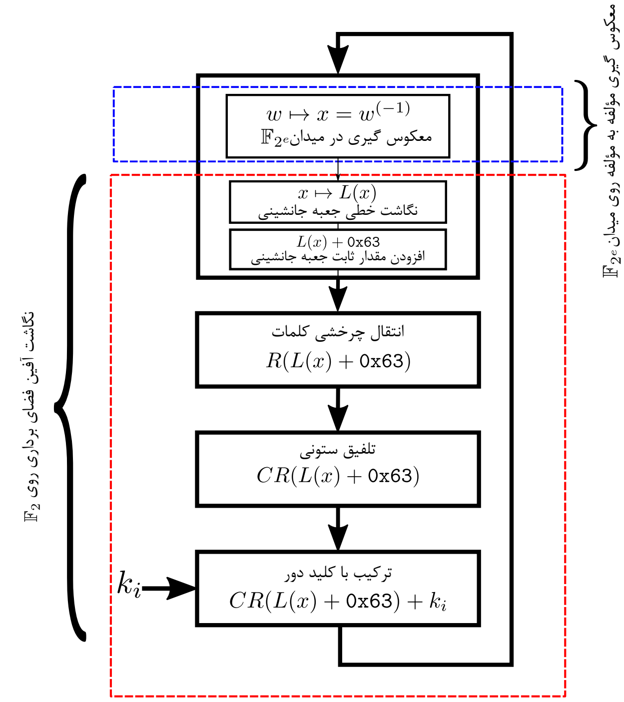
$$\begin{pmatrix} I_c & [\circ]_c & \cdots & [\circ]_c \\ [\circ]_c & \hat{R} & \cdots & [\circ]_c \\ \vdots & \vdots & \vdots & \vdots \\ [\circ]_c & [\circ]_c & \cdots & \hat{R}^{r-۱} \end{pmatrix}_{rc\times rc} \in \mathrm{Mat}_{rc}(\mathbb{GF}(۲^e))$$

با یک جایگشت مناسب روی سطرها و ستون‌های ماتریس فوق می‌توان ماتریسی یافت که عملیات `ShfitRows` را نسبت به پایه و ترتیب استاندارد نمایش می‌دهد، این ماتریس را با $\bar{R}$ نمایش می‌دهیم. درایه‌های ماتریس $\bar{R}$ صفر و یک‌های میدان $\mathbb{GF}(۲^e)$ هستند که اگر صفرها را با ماتریس $[\circ]_{e\times e} \in \mathrm{Mat}_e(\mathbb{F}_۲)$ و یک‌ها را با $I_e \in \mathrm{Mat}_e(\mathbb{F}_۲)$ جایگزین کنیم به یک ماتریس از $\mathrm{Mat}_{rce\times rce}(\mathbb{F}_۲)$ می‌رسیم که با ضرب آن در بردار داده rce بیتی عملیات انتقال چرخشی انجام می‌شود. این ماتریس را که در استخراج معادلات روی $\mathbb{GF}(۲)$ از آن استفاده می‌کنیم را با $\mathbf{R}$ نمایش می‌دهیم.

در مرحله `MixColumns` هر ستون از آرایه داده در ماتریسی که در جدول ۳.۳ آمده ضرب می‌شود، درایه‌های این ماترس اعضای میدان $\mathbb{GF}(۲^e)$ هستند. اگر به $\mathbb{GF}(۲^e)$ به عنوان یک فضای برداری با بعد e روی $\mathbb{GF}(۲)$ بنگریم در این‌صورت، عمل ضرب کردن در یک مقدار ثابت در میدان $\mathbb{GF}(۲^e)$ یک نگاشت $\mathbb{GF}(۲)$ ـ‌خطی است که و در نتیجه می‌توان آن‌را با یک ماتریس مربعی از $\mathrm{Mat}_e(\mathbb{GF}(۲))$ نمایش داد. برای مثال فرض کنید θ ریشه چندجمله‌ای تحویل‌ناپذیر $x^۸+x^۴+x^۳+x+۱ \in \mathbb{GF}(۲)[x]$ (چندجمله‌ای راین‌دال) باشد، در این صورت $\mathbb{GF}(۲)$ ـ ماتریس متناظر با عمل ضرب در مقدار ثابت $\theta \in \mathbb{GF}(۲)(\theta)$ نسبت به پایه $\{\theta^۷,\ldots,\theta^۲,\theta,۱\}$ موسوم به پایه چندجمله‌ای، عبارت است از

$$T_\theta = \begin{pmatrix} \circ & ۱ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & ۱ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & ۱ & \circ & \circ & \circ & \circ \\ ۱ & \circ & \circ & \circ & ۱ & \circ & \circ & \circ \\ ۱ & \circ & \circ & \circ & \circ & ۱ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & ۱ & \circ \\ ۱ & \circ & \circ & \circ & \circ & \circ & \circ & ۱ \\ ۱ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \end{pmatrix}$$

بنابراین اگر به‌جای هر یک از مقادیر ثابت ماتریس `MixColumns` در جدول ۳.۳، $\mathbb{GF}(۲)$ ـ ماتریس متناظر با آن‌را جایگزین کنیم به یک ماتریس از $\mathrm{Mat}_{rce}(\mathbb{F}_۲)$ می‌رسیم که ضرب آن در بردار داده



شکل ۱۰.۳: تقسیم تابع دور AES به دو قسمت آفین و غیرآفین

معادل با عمل MixColumns است، این ماتریس را با $\mathbf{C}$ نمایش می‌دهیم.

بنابراین از بین مجموعه عملیات‌هایی که در یک دور از بخش رمزنگاری SR یا AES روی داده صورت می‌گیرد تنها مرحله معکوس گیری توسیع‌یافته در SubByte است که یک عملیات غیر آفین است و همان‌طور که در شکل ۱۰.۳ نیز ملاحظه می‌شود مابقی عملیات‌ها را می‌توان در یک نگاشت آفین به‌صورت زیر خلاصه کرد. فرض کنید بردار ورودی و خروجی مرحله معکوس گیری در دور i را به‌ترتیب با w_i و x_i و کلید این دور را با k_i نمایش دهیم در این صورت تابع دور بخش رمزنگاری در $\mathrm{SR}(n, r, c, e)$ عبارت است از

$$x_{i-۱} \mapsto w_i = \mathbf{CR}(\mathbf{L}(x_{i-۱}) + \mathbf{d}) + k_i$$

که نحوه به‌دست آوردن ماتریس‌های $\mathbf{C}, \mathbf{R}$ و $\mathbf{L}$ را که در فضای $\mathrm{Mat}_{rce}(\mathbb{F}_۲)$ هستند، در بندهای قبل توضیح دادیم. به‌راحتی می‌توان دید که $\mathbf{C} \cdot \mathbf{d} = \mathbf{R} \cdot \mathbf{d} = \mathbf{d}$ که $\mathbf{d}$ همان بردار ثابت S-Box است. در نتیجه اگر فرض کنیم $\mathbf{M} = \mathbf{CRL}$ می‌توانیم تابع دور را به‌صورت زیر بیان کنیم

$$x_{i-۱} \mapsto w_i = \mathbf{M}x_{i-۱} + k_i.$$

معکوس گیری کلمه‌به‌کلمه از بردار w_i را با $w_i^{(-۱)}$ نمایش می‌دهیم که به‌صورت زیر تعریف می‌شود.

$$x_i = w_i^{(-۱)} = (w_{i,۰}^{-۱}, \ldots, w_{i,rc-۱}^{-۱}) \in \mathbb{F}_{۲^۸}^{rc}.$$

به این ترتیب اگر بردار متن اصلی و رمزشده را به‌ترتیب با p و c نمایش دهیم، دستگاه معادلات به‌دست آمده برای بخش رمزنگاری $\mathrm{SR}(n,r,c,e)$، عبارت‌است از

$$\begin{aligned} w_{۰} &= p + k_{۰} \\ x_i &= w_i^{(-۱)};\ i = ۰, \dots, n-۱ \\ w_i &= \mathbf{M}x_{i-۱} + k_i + \mathbf{d};\ i = ۱, ..., n-۱ \\ c &= \mathbf{M}x_{n-۱} + k_n + \mathbf{d}. \end{aligned} \tag{۱.۳}$$

تنها تفاوت $\mathrm{SR}^{\star}(n,r,c,e)$ این است که در دور آخر عملیات `MixColumns` انجام نمی‌شود ولذا به‌جای ماتریس $\mathbf{M}$ در دور آخر از ماتریس $\mathbf{M}^{\star} = \mathbf{RL}$ استفاده می‌کنیم.

در الگوریتم توسیع کلید از جعبه جانشینی، انتقال چرخشی و ترکیب کردن با استفاده از عمل جمع استفاده می‌شود که ما نحوه استخراج معادلات در هر یک از این فرآیندها را در بخش رمزنگاری تجربه کردیم. فرض کنید بیت‌های ورودی و خروجی مرحله معکوس‌گیری جعبه جانشینی در دور iام الگوریتم توسیع کلید را به‌ترتیب با k_{ijl} و s_{ijl} نمایش دهیم، که j شماره کلمه و l شماره بیت را نشان می‌دهد. می‌دانیم که در هر دور از الگوریتم توسیع کلید، فقط کلمه‌های ستون آخر آرایه کلید از جعبه جانشینی عبور می‌کنند. به همین دلیل، تعداد متغیرهای s_{ijl} از تعداد متغیرهای k_{ijl} کمتر است. برای توضیح و همچنین استخراج معادلات الگوریتم توسیع کلید، همان‌طور که در زیر مشاهده می‌شود، هر زیرکلید یا کلید دور را به صورت یک $\mathbb{GF}(۲^e)$ ـ بردار به‌طول rc در نظر می‌گیریم.

کلید اولیه یا اصلی : $(k_{۰,۰}, \dots, k_{۰,r-۱}, \dots, k_{۰,r(c-۱)}, \dots, k_{۰,rc-۱})^T$

کلید دور اول : $(k_{۱,۰}, \dots, k_{۱,r-۱}, \dots, k_{۱,r(c-۱)}, \dots, k_{۱,rc-۱})^T$

$\vdots$

کلید دور nام : $(k_{n,۰}, \dots, k_{n,r-۱}, \dots, k_{n,r(c-۱)}, \dots, k_{n,rc-۱})^T$

همان‌طور که مشاهده می‌شود کلید هر دور به پارامترهای r و c وابسته است و بر اساس کلید دور قبل به‌دست می‌آید که در ادامه چگونگی این کار را به همراه استخراج معادلات الگوریتم توسیع کلیدشرح می‌دهیم. در ضمن هر یک از درایه‌های بردار $k_{i,j}$ یک کلمه e بیتی است، بنابراین می‌توانیم هر یک از بردارهای فوق را به‌صورت یک $\mathbb{GF}(۲)$ ـ بردار به طول rce نیز در نظر بگیریم.

توسیع کلید وقتی $r = ۱$

$$s_{i-۱,۰} = k_{i-۱,c-۱}^{-۱}.$$

- یک ستون $(r = 1, c = 1)$.

$$(k_{i,0}) = (L(s_{i-1,0})) + (d) + (\kappa_i).$$

- بیش‌از یک ستون $(r = 1, c > 1)$.

$$(k_{i,q}) = (L(s_{i-1,0})) + (d) + (\kappa_i) + \sum_{t=0}^{q} (k_{i-1,t}).$$

که $q \in \{0, ..., c-1\}$.

توسیع کلید وقتی $r = 2$

$$s_{i-1,0} = k^{-1}_{i-1,2c-1}, \ s_{i-1,1} = k^{-1}_{i-1,2c-2}.$$

- یک‌ستون $(r = 2, c = 1)$.

$$\begin{pmatrix} k_{i,0} \\ k_{i,1} \end{pmatrix} = \begin{pmatrix} L(s_{i-1,0}) \\ L(s_{i-1,1}) \end{pmatrix} + \begin{pmatrix} d \\ d \end{pmatrix} + \begin{pmatrix} \kappa_i \\ 0 \end{pmatrix}.$$

- بیش از یک ستون $(r = 2, c > 1)$.

$$\begin{pmatrix} k_{i,rq} \\ k_{i,rq+1} \end{pmatrix} = \begin{pmatrix} L(s_{i-1,0}) \\ L(s_{i-1,1}) \end{pmatrix} + \begin{pmatrix} d \\ d \end{pmatrix} + \begin{pmatrix} \kappa_i \\ 0 \end{pmatrix} + \sum_{t=0}^{q} \begin{pmatrix} k_{i-1,rt} \\ k_{i-1,rt+1} \end{pmatrix}$$

که $q \in \{0, ..., c-1\}$.

توسیع کلید وقتی $r = 4$

$$s_{i-1,0} = k^{-1}_{i-1,4c-1}, \ s_{i-1,1} = k^{-1}_{i-1,4c-2}, \ s_{i-1,2} = k^{-1}_{i-1,4c-3}, s_{i-1,3} = k^{-1}_{i-1,4c-4}.$$

- یک ستون ($r = 4, c = 1$).

$$\begin{pmatrix} k_{i,0} \\ k_{i,1} \\ k_{i,2} \\ k_{i,3} \end{pmatrix} = \begin{pmatrix} L(s_{i-1,0}) \\ L(s_{i-1,1}) \\ L(s_{i-1,2}) \\ L(s_{i-1,3}) \end{pmatrix} + \begin{pmatrix} d \\ d \\ d \\ d \end{pmatrix} + \begin{pmatrix} \kappa_i \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

- بیش از یک ستون ($r = 4, c > 1$).

$$\begin{pmatrix} k_{i,rq} \\ k_{i,rq+1} \\ k_{i,rq+2} \\ k_{i,rqq+3} \end{pmatrix} = \begin{pmatrix} L(s_{i-1,0}) \\ L(s_{i-1,1}) \\ L(s_{i-1,2}) \\ L(s_{i-1,3}) \end{pmatrix} + \begin{pmatrix} d \\ d \\ d \\ d \end{pmatrix} + \begin{pmatrix} \kappa_i \\ 0 \\ 0 \\ 0 \end{pmatrix} + \sum_{t=0}^{q} \begin{pmatrix} k_{i-1,rt} \\ k_{i-1,rt+1} \\ k_{i-1,rt+2} \\ k_{i-1,rt+3} \end{pmatrix}$$

که $q \in \{0, ..., c-1\}$.

در بندهای قبل ضمن توضیح جزئیات الگوریتم توسیع کلید معادله‌های آفین حاکم بر این الگوریتم را نمایش دادیم. تنها معادلات غیر آفین الگوریتم توسیع کلید، از مرحله معکوس‌گیری جعبه جانشینی منشأ می‌گیرند که روش به‌دست آوردن آن‌ها را در بخش استخراج معادله‌های الگوریتم رمزنگاری شرح دادیم.

اکنون به شمارش تعداد معادلات به‌دست آمده برای $\mathrm{SR}(n, r, c, e)$ و تعداد مجهولات دستگاه شامل این معادله‌ها می‌پردازیم. در بخش‌های قبل دیدیم که به ازای هر معکوس‌گیری $3e$ معادله به‌دست می‌آید که $3e-1$ تای آن‌ها با احتمال ۱ و یکی از آن‌ها زمانی درست است که معکوس‌گیری از صفر رخ نداده باشد و لذا با احتمال $1 - \frac{1}{2^e}$ درست است. اگر رخداد ظاهر نشدن صفر در ورودی جعبه جانشینی در دورهای مختلف توسیع کلید و رمزنگاری را رخدادهایی مستقل فرض کنیم در این‌صورت احتمال عدم رخداد معکوس‌گیری از صفر در الگوریتم رمزنگاری برابر $(1 - \frac{1}{2^e})^{nrc}$ و در الگوریتم توسیع کلید برابر $(1 - \frac{1}{2^e})^{nc}$ است. ما در شمارش تعداد معادلات، همه $3e$ معادله به‌دست آمده از مرحله معکوس‌گیری را در نظر می‌گیریم.

طبق دستگاه ۳.۱، از هر دور الگوریتم رمزنگاری $\mathrm{SR}(n, r, c, e)$، rce معادله خطی به‌دست می‌آید. از طرفی چون هر دور الگوریتم رمزنگاری شامل rc عملیات معکوس‌گیری است، $rc \cdot 3e$ معادله مربعی نیز خواهیم داشت. الگوریتم رمزنگاری شامل n دور است و لذا $n \cdot (rce + 3rce) = 4n \cdot rce$ معادله خواهیم داشت. الگوریتم رمزنگاری شامل یک مرحله آغازین نیز است که فقط شامل افزودن کلید اصلی به متن اصلی است، بواسطه این عملیات rce معادله دیگر روی $\mathbb{F}_2$ خواهیم داشت. در نتیجه در مجموع $(4n + 1) \cdot rce$ معادله از الگوریتم رمزنگاری به‌دست می‌آید.

	تعداد معادلات	تعداد متغیرهای حالت	تعداد متغیرهای کلید
الگوریتم رمزنگاری	$(۴n+۱)rce$	$۲nrce$	$(n+۱)rce$
الگوریتم توسیع کلید	$nrce+۳nre$	$-$	nre
مجموع	$(۵n+۱)rce+۳nre$	$۲nrce$	$(n+۱)rce+nre$

جدول ۵.۳: تعداد معادلات استخراج شده از $\mathrm{SR}(n,r,c,e)$

هر دور از الگوریتم توسیع کلید شامل r عملیات معکوس‌گیری است که $۳re$ معادله مربعی برحسب بیت‌های کلید و متغیرهای s_{ijl} تولید می‌کند. از طرفی به ازای هر دور از الگوریتم توسیع کلید rce معادله خطی خواهیم داشت. چون الگوریتم توسیع کلید شامل n دور است، در نتیجه در مجموع $n\cdot(rce+۳re)$ معادله از الگوریتم توسیع کلید به‌دست می‌آید.

اگر متغیرهای متناظر با بیت‌های داده یعنی w_{ijl} و x_{ijl} را ***متغیرهای حالت*** و متغیرهای متناظر با بیت‌های کلید یعنی k_{ijl} را ***متغیرهای کلید*** بنامیم، به ازای هر دور از الگوریتم رمزنگاری $\mathrm{SR}(n,r,c,e)$، به تعداد $۲rce$ متغیر حالت برای نمایش ورودی و خروجی مرحله معکوس‌گیری در جعبه جانشینی، و rce متغیر کلید برای نمایش بیت‌های کلید آن دور نیاز است. بنابراین دستگاهی که برای بخش رمزنگاری به‌دست می‌آید دارای $(n+۱)rce+۲nrce$ متغیر خواهد بود.

در هر دور الگوریتم توسیع کلید، r بار عمل معکوس‌گیری صورت می‌گیرد که به ازای هر معکوس‌گیری باید e متغیر جدید (s_{ijl}) که نشان‌دهنده خروجی این مرحله هستند معرفی کنیم. به این ترتیب در مجموع nre متغیر جدید بواسطه الگوریتم توسیع کلید تولید می‌شود. در نتیجه در مجموع در کل این معادلات، $۲nrce+(n+۱)rce+nre$ مجهول خواهیم داشت. به‌عنوان مثال برای AES-128 که همان $\mathrm{SR}^*(۱۰,۴,۴,۸)$ است، ۷۴۸۸ معادله و ۴۲۸۸ مجهول روی $\mathbb{F}_۲$ به‌دست می‌آید. ص

نرم‌افزار سیج دارای ماژول‌های آماده‌ای برای کار با SR به‌خصوص استخراج معادلات این سامانه است، برای مثال با استفاده از دستور زیر می‌توانیم معادلات AES-128 را استخراج کنیم کنیم.

```
sr = mq.SR(10,4,4,8,gf2 = True,polybori=True,
                    allow_zero_inversions = False,star = True)
plaintext = sr.random_element();  key = sr.random_element()
ciphertext = sr(plaintext,key)
time poly_sys, solution = sr.polynomial_system(P = plaintext, C = ciphertext)
print poly_sys
Time: CPU 0.92 s, Wall: 0.92 s
Polynomial Sequence with 7488 Polynomials in 4288 Variables
```

الگوریتم‌های پیاده‌سازی شده در سیج دقیقا مطابق با روش توضیح‌داده شده در بندهای قبل

معادلات SR را استخراج می‌کند. برای آشنایی با روش استخراج معادلات SR و AES روی $\mathbb{GF}(۲^۸)$ می‌توانید به [۵۵] و [۱۳] رجوع کنید.

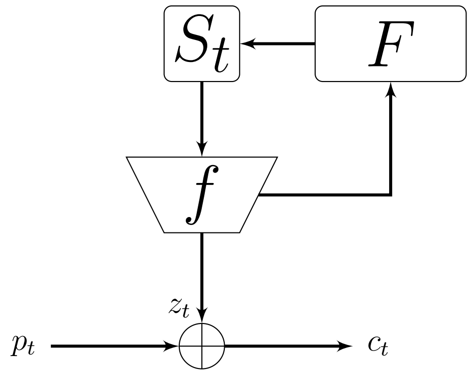
۵.۳ جبری‌سازی رمزهای دنباله‌ای

همان‌طور که در فصل اوّل هم آمد، رمزهای دنباله‌ای از دو الگوریتم قطعی به نام‌های الگوریتم آغازسازی و الگوریتم تولید کلید اجرایی تشکیل شده‌اند. الگوریتم تولید کلید اجرایی در واقع، یک ماشین حالت محدود است که حالت اولیه‌ی آن توسط الگوریتم آغازسازی و وابسته به کلید و یک مقدار اولیه، تعیین می‌شود و به عنوان مدل واقعی یک مولد شبه تصادفی در نظر گرفته می‌شود. الگوریتم آغازسازی تنها در آغاز رمزنگاری، با دریافت کلید و یک مقدار اولیه (معمولاً وابسته به متن اصلی) حالت اولیه‌ی الگوریتم تولید کلید اجرایی را محاسبه می‌کند، بنابراین بخش اصلی رمز دنباله‌ای را الگوریتم تولید کلید اجرایی تشکیل می‌دهد.

یک حمله‌ی مرسوم به رمزهای دنباله‌ای حمله‌ی متن اصلی معلوم است. فرض کنید مهاجم، تعداد کافی از بیت‌های متن اصلی و رمزشده را دراختیار داشته باشد، به این ترتیب با جمع نظیر به نظیر بیت‌های متن اصلی و متن رمزشده، بیت‌های کلید اجرایی به‌دست می‌آیند. بنابراین می‌توان گفت که مهاجم، تعداد کافی از بیت‌های کلید اجرایی را در اختیار دارد. بر اساس هدف مهاجم، حمله به رمزهای دنباله‌ای را می‌توان به دو نوع تقسیم کرد. در نوع اوّل، هدف به‌دست آوردن مقادیر ورودی الگوریتم آغازسازی، یعنی کلید اصلی و بردار حالت اولیه است و در نوع دوم هدف فقط به‌دست آوردن بیت‌های حالت اولیه‌ی مولد شبه تصادفی یا به عبارت دیگر خروجی الگوریتم آغازسازی است. بنابراین در حمله‌ی نوع دوم از الگوریتم آغازسازی صرف نظر شده و در واقع این مولد شبه تصادفی است که مورد حمله واقع می‌شود. ما نیز در مثال‌هایی که در ادامه آمده از الگوریتم آغازسازی صرف نظر می‌کنیم و هدفمان به‌دست آوردن بیت‌های حالت اولیه‌ی الگوریتم تولید کلید اجرایی است.

الگوریتم تولید کلید اجرایی، همان طور که در شکل ۱۱.۳ نمایش داده شده از دو تابع اصلی و چند ثبات یا خانه‌ی حافظه تشکیل شده است. یکی از توابع که تابع حالت نام‌دارد،با دریافت حالت فعلی حالت بعدی ماشین، و دیگری با دریافت حالت فعلی خروجی را مشخص می‌کند. فرض کنید الگوریتم تولید کلیداجرایی (یا مولد شبه تصادفی) G دارای l ثبات برای نگهداری l بیت در حافظه‌ی خود باشد. اگر بردار حالت در لحظه‌ی t را با $S_t = (s_{t,۰}, ..., s_{t,l-۱})$ و تابع حالت را با F نمایش دهیم در این‌صورت تغییر حالت ماشین توسط F به‌صورت زیر انجام می‌شود:

$$F : \mathbb{F}_۲^l \to \mathbb{F}_۲^l$$
$$S_t \mapsto S_{t+۱} = F(S_t) \; ; \; S_{t+۱} = (s_{t+۱,۰}, s_{t+۱,۱}, ..., s_{t+۱,l-۱}).$$



شکل ۱۱.۳: مولد شبه تصادفی در رمزهای دنباله‌ای

بیت t ام از کلید اجرایی نیز توسط تابع f به‌صورت زیر تعین می‌شود:

$$\begin{aligned} f &: \mathbb{F}_2^l \to \{0,1\} \\ S_t &\mapsto z_t = f(S_t). \end{aligned}$$

تابع حالت F و تابع f که به آن فیلتر هم می‌گوییم، توابعی چندجمله‌ای بر حسب متغیرهای حالت هستند، بنابراین اگر حالت اولیه را با بردار $(s_0, ..., s_{l-1})$ نمایش دهیم، بین بیت‌های کلید اجرایی و بردار حالت اولیه روابط چندجمله‌ای زیر برقرار است.

$$\begin{cases} z_0 = f(s_0, s_1, ..., s_{l-1}) \\ z_1 = f(F(s_0, s_1, ..., s_{l-1})) \\ \vdots \\ z_{l-1} = f(F^{l-1}(s_0, s_1, ..., s_{l-1})) \end{cases}$$

اکنون اگر طبق فرض، مهاجم تعدادی از بیت‌های خروجی z_t را در اختیار داشته باشد با جایگذاری آن‌ها در روابط فوق به یک دستگاه معادلات چندجمله‌ای می‌رسد که مجهولات آن بیت‌های حالت اولیه است. بنابراین مهاجم با حل دستگاه به‌دست آمده، قادر خواهد بود حالت اولیه الگوریتم تولید کلید اجرایی را به‌دست آورد.

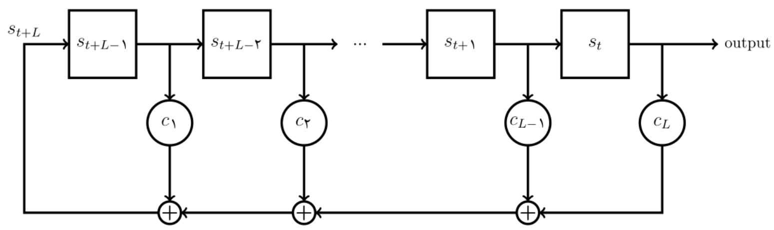
۱.۵.۳ ثبات انتقال با بازخورد خطی

ثبات انتقال با بازخورد خطّی نمونه‌ای ساده از الگوریتم‌های تولید کلید اجرایی است. در این نوع مولد، تابع حالت یک تابع خطّی است و بیت‌های کلید اجرایی بیت‌های موجود در یک رجیستر ثابت از این مولّد هستند.

به‌صورت دقیق‌تر یک ثبات با بازخورد خطّی با استفاده از چند جمله‌ای بازخورد $C(x) = ۱ - \sum_{i=۱}^{L} x^i$ از $\mathbb{F}_۲[x]$ تعریف می‌شود که L طول ثبّات نامیده می‌شود. این ثبات رشته‌ی حالت اوّلیه $(s_۰, s_۱, ..., s_{L-۱})$ را با استفاده از رابطه‌ی بازگشتی زیر به یک دنباله با طول نامتناهی تبدیل می‌کند:

$$\forall\, t \geq ۰ \ : \ s_{t+L} = \sum_{i=۱}^{L-۱} c_i s_{t+L-i}$$

و دنباله‌ی خروجی آن عبارت است از: $(s_t)_{t\geq ۰}$. شکل ۱۲.۳ یک ثبّات با بازخورد خطّی به طول L را نشان می‌دهد. دنباله‌ی خروجی یک ثبات با بازخورد خطی، به‌صورت یکتایی توسط ضرایب چندجمله‌ای بازخورد و حالت اولیه تعیین می‌شود.



شکل ۱۲.۳: ثبّات با بازخورد خطّی به طول L

لم ۱۴.۳. برای هر LFSR با طول L و بردار حالت اوّلیه‌ی $(s_۰, s_۱, ..., s_{L-۱})$ ، به ازای هر $t \geq ۰$ ضرایب $\{a_i^t\}_{i=۰}^{L-۱}$ وجود دارند به طوری که

$$s_t = \sum_{i=۰}^{L-۱} a_i^t s_i$$

یعنی هر بیت حالت ترکیبی خطّی از بیت‌های حالت اوّلیه است.

برهان. فرض کنید $C(x) = ۱ - \sum_{i=۰}^{L-۱} c_i x^i$ چند جمله‌ای بازخورد LFSR و بردار $(s_۰, ..., s_{L-۱})$ بردار حالت اوّلیه باشد در این صورت تابع انتقال حالت یک تابع خطّی است که با ماتریس زیر قابل نمایش است

$$C = \begin{pmatrix} ۰ & ۰ & ۰ & \dots & ۰ & c_L \\ ۱ & ۰ & ۰ & \dots & ۰ & c_{L-۱} \\ ۰ & ۱ & ۰ & \dots & ۰ & c_{L-۲} \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ ۰ & ۰ & ۰ & \dots & ۱ & c_۱ \end{pmatrix}$$

$S_t = (s_t, s_{t+۱}, ..., s_{t+L-۱})\ t \geq ۰$

$$F : \{۰, ۱\}^L \to \{۰, ۱\}^L$$

$$S_t \mapsto S_{t+1} = CS_t$$

s_t

$$s_t = S_0 C^t \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \Longrightarrow \begin{pmatrix} a_0^t \\ a_1^t \\ a_2^t \\ \vdots \\ a_{L-1}^t \end{pmatrix} = C^t \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

□

L $s_{t_1}, ..., s_{t_L}$ LFSR L را بدانیم و ماتریس ضرایب A که در زیر نمایش داده شده معکوس پذیر باشد،

$$A = \begin{pmatrix} a_0^{t_1} & a_0^{t_2} & ... & a_0^{t_L} \\ a_1^{t_1} & a_1^{t_2} & ... & a_1^{t_L} \\ \vdots & \vdots & ... & \vdots \\ a_{L-1}^{t_1} & a_{L-1}^{t_2} & ... & a_{L-1}^{t_L} \end{pmatrix}$$

به‌راحتی می‌توانیم بردار حالت اوّلیه را با محاسبه‌ی ساده‌ی زیر به‌دست آوریم.

$$S_0 = (s_0, ..., s_{L-1}) = (s_{t_1}, ..., s_{t_L}) \begin{pmatrix} a_0^{t_1} & a_0^{t_2} & ... & a_0^{t_L} \\ a_1^{t_1} & a_1^{t_2} & ... & a_1^{t_L} \\ \vdots & \vdots & ... & \vdots \\ a_{L-1}^{t_1} & a_{L-1}^{t_2} & ... & a_{L-1}^{t_L} \end{pmatrix}^{-1}$$

طبق لم زیر احتمال معکوس پذیر بودن ماتریس ضرایب نیز قابل توجه است.

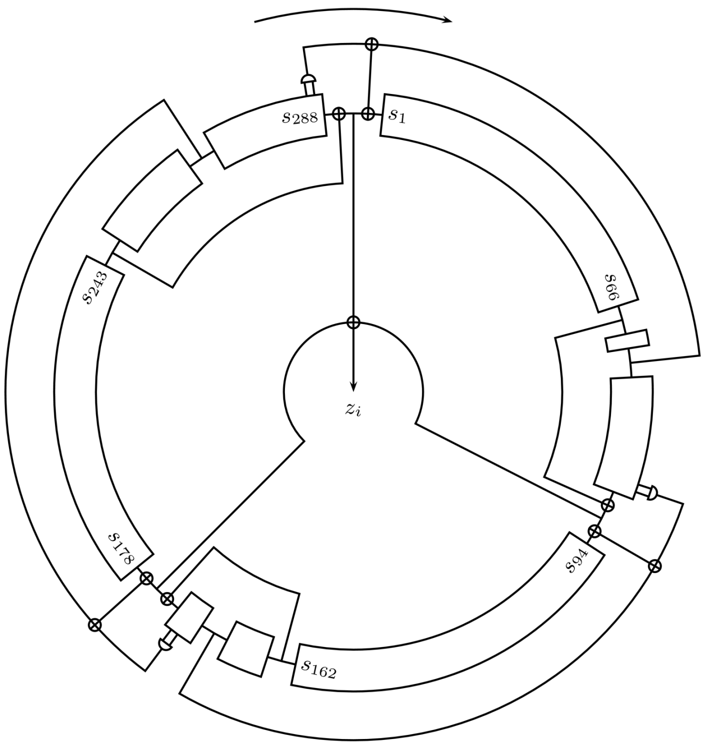
لم ۱۵.۳. احتمال این که یک ماتریس تصادفی روی $\mathcal{M}_n(\mathbb{F}_q)$ معکوس پذیر باشد برابر است با:

$$\gamma_q(n) = Pr\{M \leftarrow \mathrm{Mat}_n(\mathbb{F}_q)\ ;\ M : \text{معکوس پذیر باشد}\} = \prod_{i=1}^{n}(1 - \frac{1}{q^i})$$

$$\lim_{n\to\infty} \gamma_q(n) = 1 - \frac{1}{q} + (\frac{1}{q^2})$$

که با قرار دادن $q = 2$ داریم:

$$\gamma_q(\infty) = \prod_{i=1}^{\infty}(1 - \frac{1}{2^i}) \approx 0/288788095۱$$



شکل ۱۳.۳: رمز دنباله‌ای Trivium

درسال‌های آغازین قرن بیست‌ویکم، تعدادی از رمزهای دنباله‌ای مورد حملات جدّی قرار گرفتند که از مهم‌ترین آن‌ها می‌توان به رمز دنباله‌ای A5/1 وحمله‌ی اعمال شده به آن در [۱۰]، و A5/2 و تحلیل صورت گرفته روی آن در [۸]، اشاره کرد. چنین حمله‌های مؤفقی، زمانی صورت گرفت که این رمزهای دنباله‌ای برای برقراری امنیت سامانه‌ی جهانی ارتباطات همراه، که میلیون‌ها شهروند اروپایی از آن استفاده می‌کردند، به‌کار گرفته شده بود. این تهدیدها سبب شد تا سازمان رمزنگاری اروپایی ECRYPT اقدام به برگزاری مسابقه‌ای برای طراحی رمزهای دنباله‌ای کند. این مسابقه که به پروژه‌ی eStream معروف است در سال ۲۰۰۴ آغاز شد و در مجموع ۳۴ سامانه‌ی رمز دنباله‌ای به این مسابقه ارسال شدند. سامانه‌های پیشنهادی طی سه مرحله مورد آزمون قرار گرفتند تا این‌که در سال ۲۰۰۸ که پایان مسابقه بود تنها هفت الگوریتم از مرحله‌ی نهایی عبور کرده و به عنوان الگوریتم‌های برتر شناخته شدند. رمزهای دنباله‌ی پیشنهادی در این مسابقه در دو دسته‌ی نرم‌افزارمبنا و سخت‌افزارمبنا قرار داشتند. یکی از الگوریتم‌های منتخب در این مسابقه رمزدنباله‌ای Trivium است که در ادامه فرآیند جبری‌سازی آن را بررسی می‌کنیم.

۲.۵.۳ جبری‌سازی Trivium و Bivium

معرفی Trivium

Trivium، یک رمز دنباله‌ای است که توسط کانییری [12] و پرینیِل [13]، [۱۲] ارائه شد. این رمز دنباله‌ای از یک ثبات انتقال همزمان بیتی به طول ۲۸۸ بیت، برای تولید کلید اجرایی دودویی با حداکثر طول $۲^{۶۴}$ از روی یک کلید ۸۰ بیتی به همراه یک بردار حالت اولیه ۸۰ بیتی به‌کار می‌رود. مانند اغلب رمزهای دنباله‌ای این فرآیند شامل دو مرحله اصلی است. در مرحله اول که آغازسازی نام دارد، بیت‌های ثبات انتقال با استفاده از کلید و بردار حالت اولیه مقدار دهی می‌شوند. در مرحله دوم بیت‌های ثبات انتقال یا همان بیت‌های حالت داخلی به طور مرتب و پی‌درپی، بر اساس یک تابع مشخص از حالت قبلی به‌روز می‌شوند. این فرآیند تکراری تا زمانی ادامه می‌یابد که به اندازه کافی بیت کلید اجرایی تولید شود.

ابتدا مرحله دوم یعنی فرآیند به‌روز رسانی بیت‌های حالت داخلی را شرح می‌دهیم. فرض کنید بیت‌های ثبات انتقال را با $(s_۰, ..., s_{۲۸۷})$، نمایش دهیم و z_t، نشان‌دهنده بیت tام کلید اجرایی باشد، در این‌صورت اگر بخواهیم N بیت از کلید اجرایی را (که $N \leq ۲^{۶۴}$) تولید کنیم، فرآیند به‌روزرسانی مطابق الگوریتم ۱۴، صورت می‌گیرد. نمایش تصویری فرآیند به‌روزرسانی بیت‌های حالت داخلی در شکل ۱۳.۳، نمایش داده شده است. مرحله آغازسازی نیز مطابق الگوریتم ۱۵، صورت می‌گیرد.

معرفی Bivium

Bivium، نسخه تقلیل‌یافته‌ای از الگوریتم رمزنگاری Trivium است که توسط رادوم [14] [۵۷]، و در دو نوع A و B ارائه شد. در این سامانه از یک ثبات انتقال همزمان ۱۷۷ بیتی برای تولید کلید اجرایی از روی کلید ۸۰ بیتی و بردار حالت اولیه ۸۰ بیتی استفاده می‌شود. اگر ۱۷۷ بیت حالت داخلی را با $(s_۰, ..., s_{۱۷۶})$، و بیت tام کلید اجرایی را با z_t، نمایش دهیم آن‌گاه، فرآیند به‌روزرسانی و تولید بیت‌های کلید اجرایی مطابق الگوریتم ۱۶ است.

فرآیند آغازسازی در Bivium، شبیه Trivium است که در الگوریتم ۱۵ بیان شد، با این تفاوت که حلقه به‌روزرسانی به‌جای $۱۱۵۲ = ۲۸۸ \times ۴$ مرتبه به تعداد $۷۰۸ = ۱۷۷ \times ۴$، مرتبه تکرار می‌شود.

[12]Canniere
[13]Preneel
[14]Raddum

استخراج معادلات Trivium و Bivium

چون در هر دو الگوریتم مورد بحث، فرآیند آغازسازی هیچ تفاوتی با فرآیند تولید رشته کلید اجرایی ندارد لذا در هر دو روش از فرآیند آغازسازی صرف نظر می‌کنیم و هدف ما به‌دست آوردن مقادیر بیت‌های ثبات انتقال بعد از فرآیند آغازسازی و درست در لحظه شروع تولید رشته کلید اجرایی است. در ضمن فرض بر این است که به تعداد کافی از بیت‌های کلید اجرایی دسترسی داریم. برای استخراج معادلات این سامانه‌ها دو روش معرفی می‌کنیم. در روش اول معادلات را فقط بر حسب بیت‌های حالت در لحظه شروع تولید کلید اجرایی به‌دست می‌آوریم و متغیر جدیدی معرفی نمی‌کنیم. در روش دوم در هر دور تعدادی متغیر جدید معرفی می‌کنیم به طوری که درجه معادلات استخراج شده همواره حداکثر برابر با ۲ باشد.

۱. فرض کنید مقادیر بیت‌های ثبات انتقال در آغاز تولید کلید اجرایی در Trivium را با $(s_۰, ..., s_{۲۸۷})$، و در Bivium را با $(s_۰, ..., s_{۱۷۶})$، نمایش دهیم. در روش اول هیچ متغیر جدیدی معرفی نمی‌کنیم و همه معادلات را بر حسب s_iها که $i \in \{۰, ..., ۲۸۷\}$ (یا $i \in \{۰, ..., ۱۷۶\}$) به‌دست می‌آوریم. در این روش تعداد متغیرها ثابت و برابر با طول ثبات انتقال باقی می‌ماند، ولی درجه معادلات و تعداد یکجمله‌ای‌های معادلات استخراج شده رفته رفته افزایش می‌یابد. برای مثال الگوریتم Trivium، را در نظر بگیرید. در این روش در هر دور z_i را که کلید اجرایی تولید شده در دور iام الگوریتم ۱۴ است، بر حسب بیت‌های ثبات انتقال در لحظه شروع تولید کلید اجرایی به‌دست می‌آوریم. به این ترتیب در ۶۶ دور اول معادلات خطی هستند، سپس به ازای دور $i \in \{۶۷, ..., ۱۴۸\}$ معادله‌ها از درجه‌ی ۲ و به ازای $i \in \{۱۴۹, ..., ۲۱۴\}$ معادله‌ها از درجه ۳ و به ازای $i \in \{۲۱۵, ..., ۲۳۹\}$ از درجه ۴ هستند و این روند افزایشی ادامه می‌یابد. برای مثال تعدادی از معادلات به‌دست آمده با این

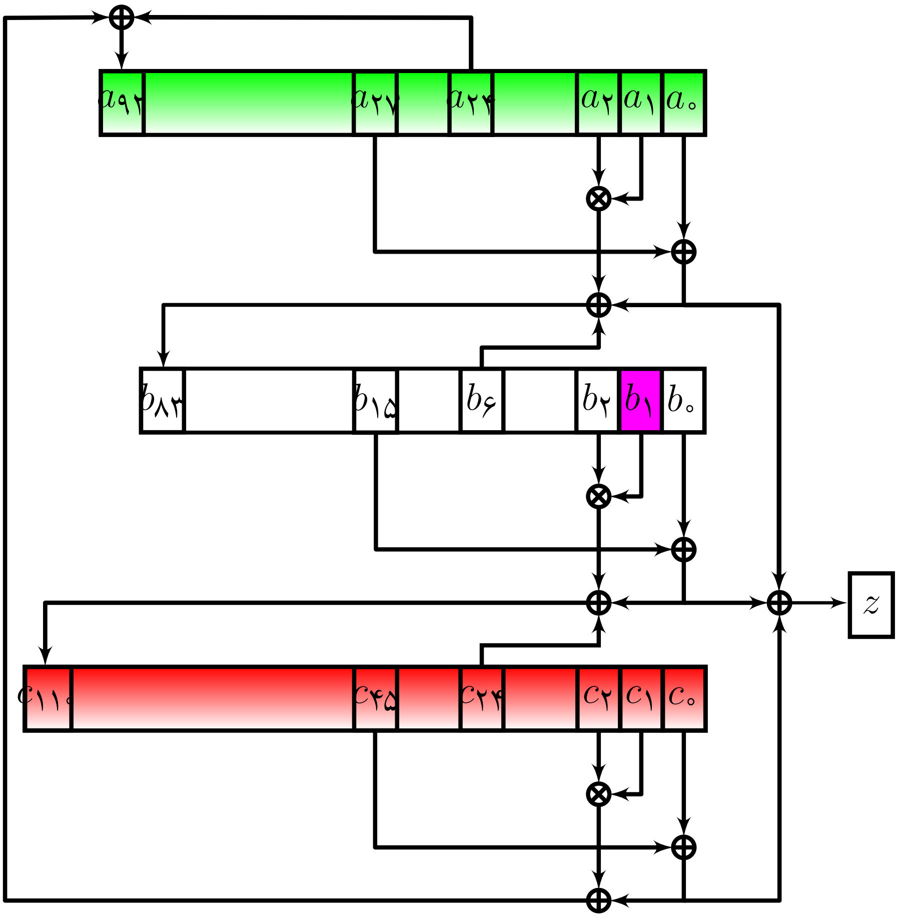
روش در زیر نمایش داده شده است.

$$
\begin{aligned}
z_{1} &= s_{65} + s_{92} + s_{161} + s_{176} + s_{242} + s_{287}\\
z_{2} &= s_{64} + s_{91} + s_{160} + s_{175} + s_{241} + s_{286}\\
&\vdots\\
z_{66} &= s_{0} + s_{27} + s_{96} + s_{111} + s_{177} + s_{222}\\
z_{67} &= s_{26} + s_{68} + s_{95} + s_{110} + s_{161} + s_{174}s_{175} + s_{176} + s_{221} + s_{242} + s_{263} + s_{285}s_{286} + s_{287}\\
&\vdots\\
z_{148} &= s_{2} + s_{12}s_{13} + s_{27}s_{28} + s_{29} + s_{53} + s_{65} + s_{78}s_{79} + s_{80} + s_{90}s_{91} + s_{92} + s_{93}s_{94} + s_{95}\\
&\quad + s_{107} + s_{125} + s_{138}s_{139} + s_{140} + s_{146} + s_{158} + s_{159}s_{160} + s_{161} + s_{170} + s_{182} + s_{188}\\
&\quad + s_{204}s_{205} + s_{206} + s_{227} + s_{231}s_{232} + s_{233} + s_{248}\\
z_{149} &= s_{1} + s_{11}s_{12} + s_{26}s_{27} + s_{28} + s_{52} + s_{64} + s_{65}s_{93} + s_{77}s_{78} + s_{79} + s_{89}s_{90} + s_{90}s_{91}s_{93}\\
&\quad + s_{91} + s_{92}s_{93} + s_{93}s_{170} + s_{94} + s_{106} + s_{124} + s_{137}s_{138} + s_{139} + s_{145} + s_{157} + s_{158}s_{159}\\
&\quad + s_{160} + s_{169} + s_{181} + s_{187} + s_{203}s_{204} + s_{205} + s_{226} + s_{230}s_{231} + s_{232} + s_{247}\\
&\vdots
\end{aligned}
$$

همان‌طور که مشاهده می‌شود درجه و تعداد یکجمله‌ای‌های معادلات استخراج شده با این روش، در دورهای بالاتر افزایش می‌یابد. پس از پیاده‌سازی این روش استخراج، با استفاده از نرم‌افزار سیج، متوجه شدیم که معادلات استخراج شده به ازای دورهای بالاتر آن‌قدر بزرگ هستند که سبب پر شدن حافظه رم کامپیوتر می‌شوند. اگر بتوانیم متغیرهایی را که به دفعات در یکجمله‌ای‌های غیر خطی ظاهر می‌شوند، شناسایی کنیم و بجای آن‌ها مقادیر عددی حدسی قرار دهیم، می‌توانیم تا حدی از حجم یکجمله‌ای‌ها ظاهر شده در معادلات کم کنیم که این کار سبب ساده‌تر شدن حل دستگاه به‌دست آمده خواهد شد.

۲. در روش دوم به ازای هر کلاک (یا هر انتقال ثبات انتقال)، تعدادی متغیر جدید اضافه می‌کنیم. برای فهم بهتر و سادگی پیاده‌سازی این روش، ثبات انتقال Trivium را به سه ثبات با طول‌های ۹۳,۸۴ و ۱۱۱، مطابق شکل ۱۴.۳، تقسیم می‌کنیم. اگر بیت‌های ثبات انتقال در لحظه t را با $(a_t, \ldots, a_{t+92}, b_t, \ldots, b_{t+83}, c_t, \ldots, c_{t+110})$ و بیت tام کلید اجرایی را با z_t نمایش دهیم، فرآیند آغازسازی، به‌روزرسانی بیت‌های حالت و تولید کلید اجرایی طبق الگوریتم ۱۷ است.

فرض کنید بیت‌های ثبات انتقال در پایان فرآیند آغازسازی را با $(a_0, \ldots, a_{92}, b_0, \ldots, b_{83}, c_0, \ldots, c_{110})$،



شکل ۱۴.۳: نمایش طبقاتی الگوریتم رمزنگاری Trivium

نمایش دهیم. فرض کنید به تعداد کافی از بیت‌های کلید اجرایی را داشته باشیم، هدف ما به‌دست آوردن مقادیر حالت اولیه و یا مقادیر بیت‌های ثبات انتقال در لحظه آغاز تولید کلید اجرایی است.

برای تولید معادلات از درجه حداکثر ۲ در این روش از الگوریتم ۱۸ استفاده می‌کنیم.

همان طور که مشاهده می‌شود در هر کلاک ۳ متغیر و ۴ معادله جدید تولید می‌شود و بدین ترتیب با در دست داشتن n بیت از کلید اجرایی دستگاهی شامل $۴n$ معادله و $۳n + ۲۸۸$ مجهول خواهیم داشت. از آن‌جایی که هر یک از معادلات استخراج شده در این روش دارای یکجمله‌ای‌های کم‌تری هستند، این روش به حافظه‌ی رم کم‌تری نیاز دارد و نسبت به روش اول سریع‌تر است. به‌راحتی می‌توان این روش را برای Bivium نیز به‌کار برد، که در این‌صورت در هر دور دو متغیر و ۳ معادله جدید تولید می‌شود و بدین ترتیب با داشتن n بیت از کلید اجرایی، دستگاهی شامل $۳n$ معادله و $۲n + ۱۷۷$ مجهول به‌دست می آید.

برای این که بتوانیم از بین روش‌های معرفی شده در فوق، بهترین روش برای استخراج معادلات را انتخاب کنیم، لازم است تا روش‌های حل دستگاه معادلات چندجمله‌ای را بهتر بشناسیم. لذا بحث راجع به انتخاب روش بهینه را به فصل آخر واگذار می‌کنیم.

تا کنون مطالعات زیادی روی تحلیل جبری رمزهای دنباله‌ای به‌خصوص آن‌دسته از رمزهای دنباله‌ای که تابع انتقال حالت آن‌ها خطی است، صورت گرفته است. در مورد تحلیل جبری این نوع از رمزهای دنباله‌ای خواننده را به [۱۹، ۳۶، ۲۱] ارجاع می‌دهیم.

۶.۳ جبری‌سازی سامانه‌های کلیدهمگانی

برای سادگی و فهم بهتر مطلب، یک نگاشت رمزنگاری کلید همگانی مانند

$$\begin{aligned} \mathsf{Enc}_{pk} :& \mathbb{F}_۲^n \to \mathbb{F}_۲^m \\ & x \mapsto y := \mathsf{Enc}_{pk}(x). \end{aligned}$$

را در نظر بگیرید. می‌دانیم که مهاجم کلید همگانی را دارد. با این پیش‌فرض دو سناریو را مورد بررسی قرار می‌دهیم. در سناریو اول هدف به‌دست آوردن متن اصلی و در سناریو دوم هدف به‌دست آوردن کلید خصوصی است.

۱.۶.۳ رمزگشایی بدون استفاده از کلید خصوصی

در این سناریو مهاجم، نگاشت چندجمله‌ای متناظر با نگاشت رمزنگاری را طوری به‌دست می‌آورد که چندجمله‌ای‌ها مستقل از بیت‌های کلید خصوصی و فقط بر حسب بیت‌های متن اصلی باشند. بنابراین اگر بیت‌های متن اصلی را با $x_1, \dots, x_n$ و بیت‌های متن رمزشده را با $y_1, \dots, y_n$ نمایش دهیم، آن‌گاه نگاشت چندجمله‌ای متناظر با $\mathsf{Enc}_{pk}(\cdot)$ عبارت است از:

$$(y_1, \dots, y_m) = \mathsf{Enc}_{pk}(x_1, \dots, x_n) = (f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$$

فرض کنید مهاجم قصد دارد بدون استفاده از کلید خصوصی متن رمزشده‌ی $c = (c_1, \dots, c_m)$ را رمزگشایی کند در این‌صورت با جایگذاری بیت‌های معلوم c_i در نگاشت فوق به دستگاهی به صورت زیر دست می‌یابد.

$$\mathcal{S} = \begin{cases} f_1(x_1, \dots, x_n) = c_1 \\ \qquad\qquad \vdots \\ f_m(x_1, \dots, x_n) = c_m \end{cases}$$

و با حل دستگاه فوق متن اصلی را می‌یابد.

مثال ۱۶.۳. سامانه‌ی رمزنگاری کلیدهمگانی Plain RSA یا RSA ساده را با پارامترهای زیر در نظر بگیرید.

$$n = 3 \times 5\ ,\ p = 3\ ,\ q = 5\ ;\ \phi(15) = 8\ ,\ e = d = 3\ ,\ ed \overset{8}{\equiv} 18$$

$$pk = (n, e) = (15, 3)\ ,\ sk = (n, d) = (15, 3)$$

نگاشت رمزنگاری فوق را می‌توان به صورت یک نگاشت چندجمله‌ای نمایش داد. برای این کار ابتدا توجّه نمایید که چون فضای متن اصلی و فضای متن رمز شده مجموعه‌ی $\mathbb{Z}_{15}$ است هر عضو این مجموعه را می‌توان با یک چهار بیتی به صورت زیر نمایش داد.

$$x \in \mathbb{Z}_{15} \Longrightarrow x = x_0 + 2x_1 + 4x_2 + 8x_3\ ;\ (x_0, x_1, x_2, x_3) \in \mathbb{F}_2^4$$

بنابراین داریم

$$\forall (x_3, x_2, x_1, x_0) \in \mathbb{F}_2^4 \quad \mathsf{Enc}_{pk}(x_3, x_2, x_1, x_0) = (x_0 + 2x_1 + 4x_2 + 8x_3)^3\ mod\ 15$$

فرض کنیم به عنوان مهاجم نگاشت رمزنگاری را می‌دانیم و کلید عمومی را در اختیار داریم در این صورت می‌توانیم معادلات چند جمله‌ای نگاشت رمزنگاری را با یک محاسبه‌ی ساده به صورت

زیر به‌دست آوریم.

$$(c_۳, c_۲, c_۱, c_۰) = \mathsf{Enc}_{pk}(x_۳, x_۲, x_۱, x_۰) = (f_۳, f_۲, f_۱, f_۰)\ ;\ f_i \in \mathbb{F}_۲[x_۰, x_۱, x_۲, x_۳]$$

با قطعه کد زیر در نرم‌افزار سیج توابع $f_۰, f_۱, f_۲, f_۳$ را استخراج می‌کنیم.

```
b = matrix(16, 4)
for i in range(16):
    b[i] = list(bin(mod(i^3, 15))[2:].zfill(4))
B = b.T;
from sage.crypto.boolean_function import BooleanFunction
c0 = BooleanFunction(list(B[3])); c1 = BooleanFunction(list(B[2]))
c2 = BooleanFunction(list(B[1])); c3 = BooleanFunction(list(B[0]))
f0 = c0.algebraic_normal_form(); f1 = c1.algebraic_normal_form()
f2 = c2.algebraic_normal_form(); f3 = c3.algebraic_normal_form()
```

$$\begin{aligned}
f_۰ =& x_۰x_۱x_۲x_۳ + x_۰x_۱x_۲ + x_۰x_۱x_۳ + x_۰x_۱ + x_۰x_۲x_۳ + x_۰ + x_۱x_۲x_۳ + x_۲x_۳\\
f_۱ =& x_۰x_۱x_۲x_۳ + x_۰x_۱x_۲ + x_۰x_۱x_۳ + x_۰x_۲x_۳ + x_۰x_۳ + x_۱x_۲x_۳ + x_۱x_۲ + x_۳\\
f_۲ =& x_۰x_۱x_۲x_۳ + x_۰x_۱x_۲ + x_۰x_۱x_۳ + x_۰x_۱ + x_۰x_۲x_۳ + x_۱x_۲x_۳ + x_۲x_۳ + x_۲\\
f_۳ =& x_۰x_۱x_۲x_۳ + x_۰x_۱x_۲ + x_۰x_۱x_۳ + x_۰x_۲x_۳ + x_۰x_۳ + x_۱x_۲x_۳ + x_۱x_۲ + x_۱
\end{aligned}$$

اکنون فرض کنید متن رمز شده‌ی $(۰, ۰, ۱, ۰) = ۲$ را دریافت کنیم، در این صورت بدون نیاز به داشتن کلید خصوصی می‌توانیم متن اصلی را بیابیم، برای این کار ابتدا دستگاه معادلات $f_۳ = f_۲ = f_۱ + ۱ = f_۰ = ۰$ تشکیل داده، سپس آن را روی میدان $\mathbb{F}_۲$ حل می‌کنیم.

یکی از روش‌های مناسب برای حلّ دستگاه فوق روش پایه‌ی گروبنر است که آن را در بخش‌های قبل معرّفی کردیم، برای حلّ به روش مذکور باید پایه‌ی گروبنر ایده‌ال تولید شده توسّط معادلات فوق را بیابیم. ممکن است دستگاه فوق روی توسیع‌های میدان $\mathbb{F}_۲$ نیز جواب داشته باشد ولی ما فقط به دنبال جواب‌های روی $\mathbb{F}_۲$ هستیم، از این‌رو معادلات میدان $\mathbb{F}_۲$ را به دستگاه فوق ضمیمه می‌کنیم تا مطمئن باشیم جواب‌های به‌دست آمده جواب‌هایی روی میدان $\mathbb{F}_۲$ هستند و نه روی توسیع‌های آن.

$$\{x_۰{}^۲ - x_۰ = ۰, x_۱{}^۲ - x_۱ = ۰, x_۲{}^۲ - x_۲ = ۰, x_۳{}^۲ - x_۳ = ۰\}$$

بنابراین کافی است پایه‌ی گروبنر ایده‌ال زیر را بیابیم.

$$J = \langle \{f_i - c_i | i = ۰, ..., ۳\} \cup \{x_i^۲ - x_i | i = ۰, ..., ۳\} \rangle$$

$$J = \langle f_۰, f_۱ + ۱, f_۲, f_۳, {x_۰}^۲ - x_۰, {x_۱}^۲ - x_۱, {x_۲}^۲ - x_۲, {x_۳}^۲ - x_۳ \rangle$$

این قسمت را با نرم‌افزار سِیج انجام می‌دهیم.

```
P.<x0,x1,x2,x3> = PolynomialRing(GF(2), order = 'lex')
I = P.ideal([f0, f1 + 1, f2, f3])
J = I + sage.rings.ideal.FieldIdeal(P)
J.groebner_basis()
                        [x0, x1, x2, x3 + 1]
J.variety()
                  [{x2: 0, x1: 0, x0: 0, x3: 1}]
```

همان‌طور که مشاهده می‌کنید واریته آفین ایده‌ال J، که همان مجموعه‌ی جواب دستگاه است، فقط شامل یک نقطه است. این نقطه که در مبنای ده برابر عدد ۸ است متن اصلی متناظر با متن رمزشده ۲ است.

۲.۶.۳ به‌دست آوردن کلید خصوصی

با توجه به این‌که مهاجم کلید عمومی را دارد، می‌تواند هر تعداد متن اصلی و رمزشده‌ی متناظر با آن را به‌دست آورد. اکنون فرض کنید مهاجم نگاشت رمزگشایی را به صورت یک نگاشت چندجمله‌ای برحسب بیت‌های کلید خصوصی و متن اصلی و رمزشده مدل کند. در این صورت با جایگذاری زوج متن‌های اصلی و رمزشده‌ی معلوم، در این نگاشت، دستگاهی به‌دست می‌آید که مجهولات آن بیت‌های کلید خصوصی هستند. با حل این دستگاه مهاجم قادر است کلید خصوصی را بیابد.

در این بخش با نحوه استخراج معادلات انواع سامانه‌های رمزنگاری آشنا شدیم. اما باید توجه کرد که صرف به‌دست آوردن معادلات یک سامانه‌ی رمزنگاری نمی‌تواند دلیل بر شکسته شدن آن سامانه باشد، بلکه دستگاه به‌دست آمده باید با الگوریتم‌های موجود قابل حل باشد. در بخش بعدی مروری داریم بر الگوریتم‌های حل دستگاه‌های به‌دست آمده از سامانه‌های رمزنگاری.

الگوریتم ۱۰ الگوریتم بوخبرگر– مولر۲

ورودی $\mathbb{Y} = \{(p_1, k_1), ..., (p_s, k_s)\} \subseteq K^n \times K^l$ و $q_1, ..., q_m$.
خروجی $\mathcal{G}$ و $(f_1, ..., f_m) \in Q^m$ (با شرایط ذکر شده در صورت قضیه.)

۱: قرار می‌دهیم $\mathcal{G} = \emptyset, O = \emptyset, \mathcal{S} = \emptyset, L = \{1\}$ و $\mathcal{M} = (m_{ij}) \in \mathrm{Mat}_{0,s}(K)$ را ماتریسی در نظر می‌گیریم که در ابتدا دارای s ستون و ۰ ردیف است. یک ترتیب یکجمله‌ای σ انتخاب کرده و به گام ۲ می‌رویم.

۲: اگر $L = \emptyset$ به گام ۶ می‌رویم. در غیر این‌صورت $t = \min_\sigma(L)$ را انتخاب کرده و سپس آن را از L حذف می‌کنیم.

۳: بردار مقادیر $(t(p_1, k_1), ..., t(p_s, k_s)) \in K^s$ را محاسبه کرده و آن را نسبت به سطرهای ماتریس $\mathcal{M}$ تحویل می‌کنیم تا بردار $(v_1, ..., v_s)$ به‌صورت

$$(v_1, ..., v_s) = (t(p_1), ..., t(p_s)) - \sum_i a_i(m_{i1}, ..., m_{is})$$

به‌دست آید بطوری که $a_i \in K$.

۴: اگر $(v_1, ..., v_s) = (0, ..., 0)$ آن‌گاه چندجمله‌ای $t - \sum_i a_i s_i$ را که s_i ، عضو i ام $\mathcal{S}$ است، به مجموعه‌ی $\mathcal{G}$ اضافه می‌کنیم. سپس همه‌ی مضارب t را از L حذف کرده و به گام ۲ می‌رویم.

۵: در غیر این‌صورت اگر $(v_1, ..., v_s) \neq 0$ ، بردار $(v_1, ..., v_s)$ را به عنوان یک سطر جدید به $\mathcal{M}$ و چندجمله‌ای $t - \sum_i a_i s_i$ را به عنوان یک عضو جدید به $\mathcal{S}$ ، اضافه می‌کنیم. یکجمله‌ای t را به O اضافه می‌کنیم و سپس اعضای از $\{x_1 t, ..., x_n t, y_1 t, ..., y_l t\}$ را که مضرب هیچ عضوی از $L \cup \mathrm{LM}_\sigma(\mathcal{G})$ نیستند، به L اضافه می‌کنیم و به گام ۲ می‌رویم.

۶: ماتریس $\mathcal{M}$، را به یک ماترس قطری تحویل کرده و همان عملیاتی که در طول تحویل کردن $\mathcal{M}$ انجام داده‌ایم، روی اعضای چندتایی مرتب $\mathcal{S}$ نیز اعمال می‌کنیم. سپس با در نظر گرفتن $\mathcal{S}$ به عنوان یک بردار ستونی، $\mathcal{M}^{-1}\mathcal{S}$ را محاسبه کرده و آن‌را جایگزین $\mathcal{S}$ می‌کنیم.

۷: به ازای $i = 1, ..., s$ ، q_i را به‌صورت $q_i = (q_{i1}, ..., q_{im})$ که $q_{ij} \in K$ می‌نویسیم. سپس به ازای هر $j = 1, ..., m$ ، چندجمله‌ای‌های $f_j = \sum_{i=1}^s q_{ij} s_i$ را که s_i عضو i ام از چندتایی مرتب $\mathcal{S}$ است محاسبه می‌کنیم. با بازگرداندن $(f_1, ..., f_m)$ و $\mathcal{G}$ الگوریتم خاتمه می‌یابد.

الگوریتم ۱۱ الگوریتم رمزنگاری در KeeLoq

Input: متن اصلی : $P = P_{31}, ..., P_0$, کلید : $K = k_{63}, ..., k_0$
Output: متن رمز شده : $C = C_{31}, ..., C_0$
$X \leftarrow P$ بارگذاری متن اصلی در شیفت رجیستر غیرخطی
for $i = 0, ..., 527$ **do**
$OUT \leftarrow NLF(x_{31}, x_{26}, x_{20}, x_9, x_1)$ بیت خروجی تابع غیر خطی
$XOR \leftarrow k_{i \mod 64} \oplus x_{16} \oplus x_0 \oplus OUT$
بروزرسانی بیت‌های حالت :
1. $X \leftarrow X \ggg 1$ شیفت به راست رجیستر
2. $x_{31} \leftarrow XOR$ پر شدن بیت پر ارزش شیفت رجیستر غیر خطی
end for
return X

الگوریتم ۱۲ الگوریتم رمزگشایی در KeeLoq

Input: متن رمزشده : $C = C_{31}, ..., C_0$, کلید $K = k_{63}, ..., k_0$
Output: متن اصلی : $P = P_{31}, ..., P_0$
$Y \leftarrow C$ بارگذاری متن رمزشده در شیفت رجیستر غیرخطی
for $i = 0, ..., 527$ **do**
$OUT \leftarrow NLF(y_{30}, x_{25}, x_{19}, x_8, x_0)$ بیت خروجی تابع غیر خطی
$XOR \leftarrow k_{(15-i) \mod 64} \oplus y_{31} \oplus x_{15} \oplus OUT$
بروزرسانی بیت‌های حالت
1. $Y \leftarrow Y \lll 1$ شیفت به چپ رجیستر
2. $y_0 \leftarrow XOR$ پر شدن بیت کم ارزش شیفت رجیستر غیر خطی
end for
return X

الگوریتم ۱۳ الگوریتم رمزنگاری AES

Input: $P \in \mathrm{Mat}_4(\mathbb{F}_{2^8})$ متن اصلی
Input: $K_0, ..., K_n \in \mathrm{Mat}_4(\mathbb{F}_{2^8})$ کلیدهای هر یک از دورها که توسط الگوریتم استخراج کلید تولید شده‌اند
Output: $C \in \mathrm{Mat}_4(\mathbb{F}_{2^8})$
  $S_0 \leftarrow \texttt{AddRoundKey}(P, K_0)$ سفیدسازی
  for $r = 1, ..., n$ **do**
    $S_r \leftarrow \texttt{SubByte}(S_{r-1})$
    $S_r \leftarrow \texttt{ShiftRows}(S_r)$
    if $r \neq N_r$ **then**
      $S_r \leftarrow \texttt{MixColumns}(S_r)$
    end if
    $S_r \leftarrow \texttt{AddRoundKey}(S_r, K_r)$
  end for
  $C \leftarrow S_n$
  return C

الگوریتم ۱۴ الگوریتم به‌روزرسانی و تولید کلید اجرایی در رمز دنباله‌ای Trivium

  for $t = 0, ..., N-1$ **do**
    $t_1 \leftarrow s_{65} + s_{92}$
    $t_2 \leftarrow s_{161} + s_{176}$
    $t_3 \leftarrow s_{242} + s_{287}$
    $z_t \leftarrow t_1 + t_2 + t_3$
    $t_1 \leftarrow t_1 + s_{90} \cdot s_{91} + s_{170}$
    $t_2 \leftarrow t_2 + s_{174} \cdot s_{175} + s_{263}$
    $t_3 \leftarrow t_3 + s_{285} \cdot s_{286} + s_{68}$
    $(s_0, ..., s_{92}) \leftarrow (t_3, s_0, ..., s_{91})$
    $(s_{93}, ..., s_{176}) \leftarrow (t_1, s_{93}, ..., s_{175})$
    $(s_{177}, ..., s_{287}) \leftarrow (t_2, s_{177}, ..., s_{286})$
  end for

الگوریتم ۱۵ مرحله آغازسازی در رمز دنباله‌ای Trivium

  $(s_0, ..., s_{92}) \leftarrow (K_0, ..., K_{79}, 0, ..., 0)$
  $(s_{93}, ..., s_{176}) \leftarrow (IV_0, ..., IV_{79}, 0, ..., 0)$
  $(s_{177}, ..., s_{287}) \leftarrow (0, ..., 0, 1, 1, 1)$
  for $0 = 1, ..., 1151$ **do**
    $t_1 \leftarrow s_{65} + s_{90} \cdot s_{91} + s_{93} + s_{170}$
    $t_2 \leftarrow s_{161} + s_{174} \cdot s_{175} + s_{176} + s_{263}$
    $t_3 \leftarrow s_{242} + s_{285} \cdot s_{286} + s_{287} + s_{68}$
    $(s_0, ..., s_{92}) \leftarrow (t_3, s_0, ..., s_{91})$
    $(s_{93}, ..., s_{176}) \leftarrow (t_1, s_{93}, ..., s_{175})$
    $(s_{177}, ..., s_{287}) \leftarrow (t_2, s_{177}, ..., s_{286})$
  end for

الگوریتم ۱۶ الگوریتم به‌روزرسانی و تولید کلید اجرایی در رمز دنباله‌ای Bivium

for $t = 0, ..., N-1$ **do**
$t_1 \leftarrow s_{65} + s_{92}$
$t_2 \leftarrow s_{161} + s_{176}$
$z_t \leftarrow t_2(\text{Variant A})/t_1 + t_2(\text{Variant B})$
$t_1 \leftarrow t_1 + s_{90} \cdot s_{91} + s_{170}$
$t_2 \leftarrow t_2 + s_{174} \cdot s_{175} + s_{69}$
$(s_0, ..., s_{92}) \leftarrow (t_2, s_0, ..., s_{91})$
$(s_{93}, ..., s_{176}) \leftarrow (t_1, s_{93}, ..., s_{175})$
end for

الگوریتم ۱۷ الگوریتم رمزنگاری Trivium

Input: $K = (k_0, ..., k_{79}), IV = (v_0, ..., v_{79}), N$
Output: $Z = (z_0, ..., z_{N-1})$
$(a_0, ..., a_{92}, b_0, ..., b_{83}, c_0, ..., c_{110}) \leftarrow (\underbrace{0, ..., 0, k_{79}, ..., k_0}_{\text{A}}, \underbrace{0, 0, 0, 0, v_{79}, ..., v_0}_{B}, \underbrace{1, 1, 1, 0, ..., 0}_{C})$
for $t = 0, ..., 1151$ **do**
$a_{t+93} \leftarrow a_{t+24} + c_{t+45} + c_t + c_{t+1} \cdot c_{t+2}$
$b_{t+84} \leftarrow b_{t+6} + a_{t+27} + a_t + a_{t+1} \cdot a_{t+2}$
$c_{t+111} \leftarrow c_{t+24} + b_{t+15} + b_t + b_{t+1} \cdot b_{t+2}$
end for
for $t = 0, ..., N-1$ **do**
$z_t \leftarrow a_t + a_{t+27} + b_t + b_{t+15} + c_t + c_{t+45}$
$a_{t+93} \leftarrow a_{t+24} + c_{t+45} + c_t + c_{t+1} \cdot c_{t+2}$
$b_{t+84} \leftarrow b_{t+6} + a_{t+27} + a_t + a_{t+1} \cdot a_{t+2}$
$c_{t+111} \leftarrow c_{t+24} + b_{t+15} + b_t + b_{t+1} \cdot b_{t+2}$
end for

الگوریتم ۱۸ استخراج معادلات از درجه حداکثر ۲ حاکم بر Trivium

Input: $Z = (z_0, ..., z_{n-1})$
Output: معادلات حاکم بر Trivium از درجه حداکثر ۲
$P \leftarrow \emptyset$
for $i = 0, ..., n-1$ **do**
$P \leftarrow P \cup \{z_i + a_i + a_{i+27} + b_i + b_{i+15} + c_i + c_{i+45}\}$
$P \leftarrow P \cup \{a_{i+93} + a_{i+24} + c_{i+45} + c_i + c_{i+1} \cdot c_{i+2}\}$
$P \leftarrow P \cup \{b_{i+84} + b_{i+6} + a_{i+27} + a_i + a_{i+1} \cdot a_{i+2}\}$
$P \leftarrow P \cup \{c_{i+111} + c_{i+24} + b_{i+15} + b_i + b_{i+1} \cdot b_{i+2}\}$
end for
return P

فصل ۴

حمله‌های جبری و روش‌های حل دستگاه معادلات چندجمله‌ای

در بخش قبل نشان دادیم که مسئله‌ی شکستن سامانه‌ی رمزنگاری را می‌توان به مسئله‌ی حل دستگاه معادلات چندجمله‌ای تبدیل کرد و معادلات حاکم بر چند سامانه‌ی واقعی را نیز استخراج کردیم. بنابراین اگر بتوان الگوریتم مناسبی برای حل دستگاه معادلات چندجمله‌ای یافت آن‌گاه امنیت تمام سامانه‌های رمزنگاری با خطر مواجه می‌شود. به همین خاطر در این فصل به معرفی الگوریتم‌هایی می‌پردازیم که رمزنگارها غالباً از آن‌ها برای حل دستگاه‌های به‌دست آمده از سامانه‌های رمزنگاری استفاده می‌کنند. بخش عمده‌ای از مطالب این فصل بر پایه‌ی مراجع [۷، ۱۶، ۴۵] و [۲، ۲۲، ۶۶] است.

۱.۴ روش‌های مبنی بر خطی سازی

۱.۱.۴ خطی سازی

در روش خطی‌سازی به‌جای هر یکجمله‌ای یک متغیر جدید جایگزین می‌کنیم، با این کار دستگاه معادلات چندجمله‌ای به یک دستگاه خطی بر حسب متغیرهای جدید تبدیل می‌شود. دستگاه خطی با روش حذفی گاوس به سرعت قابل حل است. روش خطی سازی گرچه خیلی ساده است، ولی یکی از مراحل اصلی در حمله‌هایی است که در ادامه معرفی می‌کنیم.

مثال ۱.۴. دستگاه معادلات زیر روی میدان $\mathbb{F}_۷$ را در نظر بگیرید.

$$\begin{cases} x_۱ + x_۲ + x_۱x_۲ = ۱ \\ x_۲ + x_۱x_۲ = ۱ \\ x_۱ + x_۱x_۲ = ۰ \end{cases}$$

با جایگزینی‌های $y_1 = x_1$ و $y_2 = x_2, y_{12} = x_1x_2$ به دستگاه خطی زیر می‌رسیم.

$$\begin{cases} y_1 + y_2 + y_{12} = 1 \\ y_2 + y_{12} = 1 \\ y_1 + y_{12} = 0 \end{cases}$$

با حل دستگاه فوق داریم:

$$\{y_1 = 0, y_{12} = 0, y_2 = 1\} \Longrightarrow \{x_1 = 0, x_2 = 1\}.$$

بدیهی است که هر جواب از دستگاه اصلی در دستگاه خطی به‌دست آمده نیز صدق می‌کند ولی عکس آن لزوماً برقرار نیست. به عبارت دیگر ممکن است یک جواب از دستگاه خطی با هیچ جوابی از دستگاه اصلی متناظر نباشد. برای مثال اگر معادله‌ی آخر دستگاه مثال قبل را با معادله‌ی $x_1 + x_1x_2 = 1$ جایگزین کنیم، با وجود این‌که دستگاه خطی جواب دارد، دستگاه اصلی فاقد جواب است، زیرا:

$$\{y_1 = 0, y_{12} = 1, y_2 = 0\} \Longrightarrow (x_1 = 0, x_2 = 0) \wedge (x_1x_2 = 1) \Longrightarrow \text{تناقض}.$$

فایده‌ی روش خطی‌سازی چیست؟

می‌دانیم که یک دستگاه خطی روی میدان‌های نامتناهی نظیر $\mathbb{Q}, \mathbb{R}$ یا $\mathbb{C}$ ، یا فاقد جواب است، یا جوابی یکتا و یا نامتناهی جواب دارد. ولی اگر میدان یک میدان متناهی نظیر $\mathbb{F}_2$ باشد، آن‌گاه با فرض این‌که رتبه‌ی دستگاه خطی r و تعداد متغیرهای خطی n باشد، دستگاه یا جواب ندارد یا به تعداد 2^{n-r} جواب خواهد داشت.

اگر دستگاه معادلات از یک سامانه‌ی رمزنگاری به‌دست آمده باشد، حتماً دارای جواب است (زیرا کلید سامانه و متن اصلی و رمزشده با آن کلید باید در دستگاه صدق کنند).علاوه بر این در رمزنگاری همواره با میدان‌های متناهی به خصوص $\mathbb{F}_2$ کار می‌کنیم. از طرفی می‌دانیم که در خطی سازی جواب‌های دستگاه اصلی حذف نمی‌شوند و فقط ممکن است جواب‌های خارجی بوجود آید، بنابراین تنها یک حالت برای دستگاه‌های خطی به‌دست آمده در رمزنگاری وجود دارد و آن متناهی بودن فضای جواب دستگاه است.

فرض کنید میدان متناهی مورد نظر $\mathbb{F}_q$ باشد و پس از خطی‌سازی به دستگاهی با m معادله و n مجهول برسیم. اگر رتبه‌ی دستگاه، r باشد، تعداد کل جواب‌های دستگاه برابر است با q^{n-r}. توجه کنید که همواره $r \leq n$ ، حال اگر $n - r$ کوچک باشد، فضای جواب کوچک است و می‌توانیم هر یک از جواب‌های دستگاه خطی را در دستگاه اصلی قرار داده و درستی آن را امتحان

کنیم ولی وقتی $n-r$ عدد بزرگی باشد، این‌کار از جست‌وجوی فراگیر فضای کلید سامانه سخت‌تر است. در ادامه روش‌هایی را معرفی می‌کنیم که قبل از خطی سازی یک دستگاه داده شده، تعداد معادلات مستقل را افزایش می‌دهد، تا با این‌کار $n-r$ در دستگاه خطی نهایی کاهش یافته و جست‌وجوی جواب‌های اصلی آسان‌تر شود.

۲.۱.۴ بازخطی سازی

روش *بازخطی‌سازی یا خطی‌سازی* مکرر اولین‌بار توسط کیپنیس [1] و شَمیر [2] در [۴۲] ، ارائه شد. این روش را با یک مثال ساده شرح می‌دهیم.

مثال ۲.۴. دستگاه مربعی و همگن زیر در $\mathbb{F}_۷[x_۱, x_۲, x_۳]$ را در نظر بگیرید.

$$\begin{cases} ۳x_۱{}^۲+۵x_۱x_۲+۵x_۱x_۳+۲x_۲{}^۲+۶x_۲x_۳+۴x_۳{}^۲=۵\\ ۶x_۱{}^۲+x_۱x_۲+۴x_۱x_۳+۴x_۲{}^۲+۵x_۲x_۳+x_۳{}^۲=۶\\ ۵x_۱{}^۲+۲x_۱x_۲+۶x_۱x_۳+۲x_۲{}^۲+۳x_۲x_۳+۲x_۳{}^۲=۵\\ ۲x_۱{}^۲+x_۱x_۳+۶x_۲{}^۲+۵x_۲x_۳+۵x_۳{}^۲=۰\\ ۴x_۱{}^۲+۶x_۱x_۲+۲x_۱x_۳+۵x_۲{}^۲+x_۲x_۳+۴x_۳{}^۲=۰ \end{cases}$$

با تغییر متغیّر $x_ix_j \mapsto y_{ij}$ دستگاه خطی زیر به‌دست می‌آید.

$$\begin{cases} ۳y_{۱۱}+۵y_{۱۲}+۵y_{۱۳}+۲y_{۲۲}+۶y_{۲۳}+۴y_{۳۳}=۵\\ ۶y_{۱۱}+y_{۱۲}+۴y_{۱۳}+۴y_{۲۲}+۵y_{۲۳}+y_{۳۳}=۶\\ ۵y_{۱۱}+۲y_{۱۲}+۶y_{۱۳}+۲y_{۲۲}+۳y_{۲۳}+۲y_{۳۳}=۵\\ ۲y_{۱۱}+y_{۱۳}+۶y_{۲۲}+۵y_{۲۳}+۵y_{۳۳}=۰\\ ۴y_{۱۱}+۶y_{۱۲}+۲y_{۱۳}+۵y_{۲۲}+y_{۲۳}+۴y_{۳۳}=۰ \end{cases}$$

همان طور که مشاهده می‌شود تعداد معادلات مستقل برابر ۵ و تعداد مجهولات برابر ۶ است، بنابراین در جوابی که در زیر مشاهده می‌شود یک متغیر آزاد وجود دارد.

$$y_{۱۱}=۲+۵z\ ,\ y_{۱۲}=z\ ,\ y_{۱۳}=۳+۲z\ ,\ y_{۲۲}=۶+۴z \tag{۱.۴}$$

$$y_{۲۳}=۶+z\ ,\ y_{۳۳}=۵+۳z\ ;\ z\in\mathbb{F}_۷ \text{متغیّر آزاد}$$

همان‌طور که مشاهده می‌شود این‌بار دستگاه خطی دارای جواب یکتا نیست و یک متغیر آزاد وجود دارد. البته چون $z\in\mathbb{F}_۷$، تنها ۷ حالت برای z وجود دارد و می‌توان با آزمودن هر یک از این

[1] kipnis
[2] shamir

حالات جواب دستگاه اصلی را به‌دست آورد. این‌راه‌کار شاید برای یک مثال ساده نظیر دستگاه فوق کارساز باشد ولی وقتی اندازه‌ی میدان زمینه بزرگ و تعداد متغیرهای آزاد هم زیاد باشد، آزمودن تمام حالات کار آسانی نیست، پس چه باید کرد؟ راهکاری که روش بازخطی‌سازی ارائه می‌دهد این است که، با اضافه کردن قیدهای به‌دست آمده از روابط ضربی بین یکجمله‌ای‌ها تعداد جواب‌های خارجی ناشی از خطی سازی را کاهش دهیم.

با توجه به روابط ضربی بین یکجمله‌ای‌ها و نحوه‌ی تغییر متغیر، روابط زیر بین متغیرهای جدید برقرار است.

$$y_{۱۱}y_{۲۳} = y_{۱۲}y_{۱۳}\ ,\ y_{۱۲}y_{۲۳} = y_{۱۳}y_{۲۲}\ ,\ y_{۱۲}y_{۳۳} = y_{۱۳}y_{۲۳} \qquad (۲.۴)$$

با جایگذاری مقادیر بر حسب z به‌دست آمده در ۱.۴ در معادلات ۲.۴، روابط زیر به‌دست می‌آید.

$$۳z^۲ + z + ۵ = ۰\ ,\ ۰z^۲ + ۴z + ۴ = ۰\ ,\ z^۲ + ۴z + ۳ = ۰$$

گام بازخطی سازی در این مرحله با تغییر متغیرهای $z_۱ = z, z_۲ = z^۲$ ، سبب می‌شود باز هم یک دستگاه با سه معادله‌ی خطی به‌دست آید، که دارای جواب یکتای ۱ $z_۱ = ۶, z_۲ =$ ، است. اکنون به عقب بازمی‌گردیم و جواب دستگاه اصلی را می‌یابیم. با جایگذاری $z = ۶$ در روابط ۴٫۱ داریم، $y_{۱۱} = ۴, y_{۲۲} = y_{۳۳} = ۲$. برای یافتن $x_۱, x_۲, x_۳$ ریشه‌ی دوم آن‌ها را نسبت ۷ به‌صورت زیر محاسبه می‌کنیم.

$$x_۱^۲ = ۲ \ mod\ ۷ \iff x_۱ \in \{۳,۴\}\ ;\ x_۲^۲ = x_۳^۲ = ۴\ mod\ ۷ \iff x_۲, x_۳ \in \{۲,۵\}$$

بنابراین داریم، $x_۱ = \pm ۲\ ,\ x_۲ = \pm ۳\ ,\ x_۳ = \pm ۳$. اگر قیدهای $y_{۱۲} = ۶$ و $y_{۲۳} = ۵$ از ۲.۴ را نیز در نظر بگیریم، تنها جواب‌های قابل قبول عبارتند از، $(x_۱, x_۲, x_۳) = (۲,۳,۴)$ و $(x_۱, x_۲, x_۳) = (۵,۴,۳)$. که همان جواب‌های دستگاه اصلی هستند.

دو d تایی مرتب از اندیس‌ها مثل $o_۱$ و $o_۲$ را هم‌ارز گوییم و با

$$o_۱ = (a, b, c, d, ..., e, f)\ \sim\ o_۲ = (a', b', c', d', ..., e', f').$$

نمایش می‌دهیم هر گاه $o_۲$ جایگشتی از $o_۱$ باشد. در روش بازخطی سازی از درجه‌ی d با استفاده از d تایی هم‌ارز از اندیس‌ها و روابط ضربی حاکم بر یکجمله‌ای‌ها، معادلات جدیدی برای متغیرهای دستگاه خطی به‌صورت زیر به‌دست می آید.

$$(x_a x_b)(x_c x_d)\cdots(x_e x_f) = (x_{a'} x_{b'})(x_{c'} x_{d'})\cdots(x_{e'} x_{f'}) \Rightarrow y_{ab}y_{cd}\cdots y_{ef} = y_{a'b'}y_{c'd'}\cdots y_{e'f'}.$$

در نهایت با در نظر گرفتن همه‌ی d تایی‌های هم‌ارز به یک دستگاه جدید بر حسب متغیرهای آزاد دستگاه خطی می‌رسیم. این فرآیند با یک خطی‌سازی مجدد یا بازخطی‌سازی، تا رسیدن به یک دستگاه که جوابی یکتا داشته باشد ادامه می یابد.

بازخطی‌سازی از درجه‌ی ۴

دستگاهی با $m = \varepsilon n^2$ معادله‌ی مربعی همگن، بر حسب n متغیر به‌صورت

$$\sum_{1 \leq j \leq n} a_{ijk} x_i x_j = b_k \ , \ 1 \leq k \leq m$$

را در نظر بگیرید. سؤال این است که اگر n عدد بزرگی باشد، حداقل مقدار ε برای این‌که دستگاه به‌دست آمده بعد از بازخطی سازی از درجه‌ی ۴ دارای جواب یکتا باشد چقدر است؟ ما با یک تحلیل مجانبی به ازای n های بزرگ، کران پایینی برای ε ارائه می‌دهیم. بعد از خطی سازی دستگاه اصلی به εn^2 معادله‌ی خطی، بر حسب (تقریباً) $\frac{n^2}{2}$ متغیر جدید به‌صورت $y_{ij} = x_i x_j$ که $i \leq j$ می‌رسیم. اگر فرض کنیم تمام معادلات خطی به‌دست آمده مستقل هستند آن‌گاه بُعد فضای جواب دستگاه خطی برابر است با $(\frac{1}{2} - \varepsilon)n^2$، بنابراین به همین تعداد متغیر آزاد داریم و می‌توانیم تمام جواب‌ها را به‌صورت ترکیبی خطی از متغیرهای آزد که آن‌ها را با متغیر جدید z_k نمایش می‌دهیم، به‌دست آوریم. می‌دانیم که چنین نمایش پارامتری (بر حسب z_k ها) برای مجموعه‌ی جواب دستگاه، به سرعت با روش حذفی گاوس به‌دست می‌آید.

بسیاری از جواب‌هایی که برای مجهولات y_{ij} دستگاه خطی به‌دست می‌آید، متناظر با هیچ جوابی از دستگاه اصلی نبوده و فقط ناشی از عمل خطی سازی‌اند. بنابراین برای حذف این جواب‌های زائد، معادلات دیگری که از تعریف $y_{ij} = x_i x_j$ و روابط ضربی بین یکجمله‌ای‌ها نتیجه می‌شوند را نیز در نظر می‌گیریم. این معادلات قیدهای بیشتری روی y_{ij} ها اعمال کرده و تعداد زیادی از جواب‌های خارجی را حذف می‌کنند. برای به‌دست آوردن تعداد این معادلات، یک ۴ تایی از اندیس‌ها مثل $1 \leq a \leq b \leq c \leq d \leq n$ را در نظر بگیرید. می‌توانیم $x_a x_b x_c x_d$ را به سه صورت زیر با هم ترتیب کنیم،

$$(x_a x_b)(x_c x_d) = (x_a x_c)(x_b x_d) = (x_a x_d)(x_b x_c) \implies y_{ab} y_{cd} = y_{ac} y_{bd} = y_{ad} y_{bc}$$

همان‌طور که مشاهده می‌شود یک انتخاب ۴ تایی از اندیس‌ها به دو معادله برای y_{ij} ها منتج شد. از آن‌جایی که با تقریباً $\frac{n^4}{4!}$ روش مختلف می‌توانیم این ۴ تایی‌ها از اندیس‌ها را انتخاب کنیم و هر انتخاب منجر به دو معادله‌ی مربعی برای y_{ij} ها می‌شود، لذا $\frac{n^2}{12}$ معادله‌ی مربعی برای $\frac{n^2}{2}$ متغیر y_{ij} به‌دست می‌آید که اثبات مستقل بودن آن‌ها کار دشواری نیست. پس از آن‌که y_{ij} ها در دستگاه

خطی را بر حسب متغیرهای آزاد z_k بنویسیم، تعداد متغیرهای دستگاه به $(\frac{1}{2}-\varepsilon)n^2$ متغیر کاهش می‌یابد.

بر اساس روش بازخطی سازی ، $\frac{n^4}{12}$ معادله‌ی مربعی جدید بر حسب $(\frac{1}{2}-\varepsilon)n^2$ متغیر z_i را، با تغییر متغیر $w_{ij}=z_iz_j$ که $i\leq j$ دوباره خطی سازی می‌کنیم. دستگاه جدبد دارای $\frac{n^4}{12}$ معادله‌ی خطی بر حسب $\frac{((\frac{1}{2}-\varepsilon)n^2)^2}{2}$ متغیر جدید w_{ij} است. اکنون انتظار می‌رود که اگر تعداد معادلات خطی از تعداد متغیرها بیشتر باشد، یعنی داشته باشیم،

$$\frac{n^4}{12}\geq\frac{((\frac{1}{2}-\varepsilon)n^2)^2}{2}$$

آن‌گاه، دستگاه خطی نهایی دارای جواب یکتا خواهد بود. در نتیجه کران پایین ε ، برای این‌که با یک‌بار بازخطی‌سازی به دستگاهی با جواب یکتا برسیم عبارت است از، $\varepsilon\geq\frac{1}{2}-\frac{1}{\sqrt{6}}\approx 0/1$. همان‌طور که در فوق نشان دادیم، بازخطی‌سازی از درجه ۴ به ازای n های بزرگ و وقتی تعداد معادلات بزرگتر یا مساوی $0/1n^2$ باشد خوب عمل می‌کند. ولی در [۱۶] نشان‌داده شده که بیشتر معادلات اضافه شده به دستگاه در بازخطی‌سازی از درجه‌ی بزرگتر از ۴ وابستگی خطی دارند، و این‌روش کارایی خوبی ندارد. از طرفی همان‌طور که در مثال ساده‌ی ۲.۴ نیز مشاده می‌شود، تعداد متغیرها در روش بازخطی سازی به سرعت افزایش می‌یابد. در ادامه روشی ساده‌تر و در عین حال قدرتمندتر از بازخطی‌سازی را معرفی می‌کنیم.

۳.۱.۴ روش XL

روش XL [3] یک روش مبتنی بر خطی‌سازی است که در سال ۲۰۰۰، بطور مشترک توسط، کورتوا [4] پاتارین [5] کلیمُو [6] و شَمیر در [۱۶]، و به عنوان جایگزینی برای روش بازخطی‌سازی ارائه شد. همان‌طور که در بخش قبل هم نشان دادیم، روش‌های خطی‌سازی و بازخطی‌سازی زمانی خوب عمل می‌کنند که تعداد معادلات از تعداد یکجمله‌ای‌ها بیشتر باشد. الگوریتم XL طوری طراحی شده که در صورت کافی نبودن تعداد معادلات، تعداد آن‌ها را قبل از خطی‌سازی افزایش دهد، تا از این طریق تعداد معادلات مستقل خطی به تعداد یکجمله‌ای‌ها (متغیرهای جدید در دستگاه خطی) نزدیک‌تر شده و جواب دستگاه به سمت یکتایی پیش‌رود.

[3] eXtended linearization
[4] Courtois
[5] Patarin
[6] Klimov

فرض کنید K یک میدان متناهی باشد، دستگاه چندجمله‌ای زیر را در نظر بگیرید.

$$\mathcal{S} := \begin{cases} f_1(x_1, ..., x_n) = 0 \\ f_2(x_1, ..., x_n) = 0 \\ \vdots \\ f_m(x_1, ..., x_n) = 0 \end{cases}$$

درجه‌ی دستگاه $\mathcal{S}$ را به‌صورت $d := \max\{\deg(f_i)|\ f_i \in \mathcal{S}\}$ تعریف می‌کنیم. روش XL برای یافتن جواب‌هایی از $\mathcal{S}$ که در K باشند به‌صورت نشان داده شده در الگوریتم ۱۹ عمل می‌کند.

الگوریتم ۱۹ الگوریتم‌ـ XL

ورودی دستگاه $\mathcal{S} := \{f_1, ..., f_m\} \in K[x_1, ..., x_n]$

خروجی جواب‌هایی از دستگاه که در میدان K قرار دارند.

۱: درجه‌ی دستگاه =: d

۲: پارامتر $D > d$ را طوری انتخاب‌می کنیم که $D \geq \frac{n}{\sqrt{m}}$.

۳: $L \leftarrow \{x^\alpha \in \mathbb{T}^n |\deg(x^\alpha) \leq D - d\}$

۴: **(مرحله‌ی ضرب)** $\mathcal{S}' \leftarrow \{x^\alpha f_i|\ x^\alpha \in L,\ f_i \in L\}$ (چون دستگاه اصلی $\mathcal{S}$ دارای m معادله بود دستگاه جدید $\mathcal{S}'$ دارای $|L| \cdot m$ معادله است. در ضمن درجه‌ی دستگاه به D افزایش یافته. همچنین باید توجه کرد که ضرب یکجمله‌ای‌ها در معادلات دستگاه جواب‌های دستگاه را حذف نمی‌کند و فقط ممکن است سبب ایجاد جواب‌های خارجی شود.)

۵: **(مرحله‌ی خطی سازی)** با جایگزینی هر یکجمله‌ای x^α با متغیر جدید y_α به یک دستگاه خطی بر حسب متغیرهای جدید y_α می‌رسیم. عملیات حذفی گاوس را روی دستگاه خطی به‌دست آمده اجرا می‌کنیم. ترتیب روی متغیرهای جدید در دستگاه خطی را طوری در نظر می‌گیریم که متغیرهای خطی y_{α_j} که متناظر با متغیرهای $\{1, x_k, ..., x_k^d\}$ به ازای یک $1 \leq k \leq n$ هستند در آخرین مرحله حذف شوند.

۶: **(مرحله‌ی حل)** فرض کنید دستگاه به‌دست آمده از مرحله‌ی قبل حداقل شامل یک معادله‌ی یک‌متغیره به‌صورت $c_0 + c_1 x_k + \cdots + c_D x^D = 0$ باشد، در این‌صورت این معادله را (با الگوریتم‌هایی نظیر الگوریتم Berlkamp) حل می‌کنیم و x_k را به‌دست می‌آوریم.

۷: **(تکرار)** مقدار x_k به‌دست آمده از گام ۶ (در صورت وجود) را در دستگاه اصلی جایگذاری کرده و آن را ساده می‌کنیم و دوباره الگوریتم را برای یافتن سایر مجهولات از سرمی‌گیریم.

مثال ۳.۴. دستگاه معادلات مربعی زیر روی $\mathbb{F}_2$ را در نظر بگیرید:

$$\begin{cases} 1 + x + y + z + wz + yz = 0 \\ x + z + wx + wy + wz + xy + xz + yz = 1 \\ w + y + wx + xz + yz = 0 \\ x + wx + wy + wz + yz = 1 \end{cases}$$

دستگاه فوق دارای ۴ معادله و ۴ مجهول است. با ۴ متغیر مجهول می‌توان $۱۰ = \binom{۴}{۱} + \binom{۴}{۲}$ یکجمله‌ای از درجه‌ی کمتر یا مساوی ۲ ساخت از این رو اگر از خطّی سازی استفاده کنیم یک دستگاه خطّی متشکل از ۴ معادله و ۱۰ مجهول خواهیم داشت! در این صورت تعداد معادلات از تعداد مجهولات خیلی کمتر است و لذا روش خطّی سازی به تنهایی چاره ساز نخواهد بود.

حال فرض کنید با یک دستگاه درجّه‌ی ۳ با n مجهول روبرو باشیم. تعداد یکجمله‌ای‌ها از درجه‌ی حداکثر ۳ برابر است با $\binom{n}{۱} + \binom{n}{۲} + \binom{n}{۳}$، اگر معادلات دستگاه فوق را در یک یکجمله‌ای از درجه‌ی ۱ ضرب کنیم به یک دستگاه مکعبی می‌رسیم، در این صورت برای این که دستگاه خطی متناظر با این دستگاه جواب یکتا داشته باشد به ۱۴ معادله نیاز داریم. روش XL قبل از خطّی سازی کمبود معادلات را با ساختن معادلات جدید به طوری که جواب دستگاه قبلی در آن‌ها صدق کند جبران می‌کند. درجه‌ی دستگاه برابر است با $d = ۲$، اگر پارامتر D الگوریتم XL را برابر ۳ در نظر بگیریم، باید همه‌ی معادلات را در یکجمله‌ای‌های $\{w, x, y, z, ۱\}$ ضرب کنیم. از آن‌جا که تعداد معادلات دستگاه اصلی ۴ و تعداد یکجمله‌ای‌ها از درجه‌ی حداکثر ۱ برابر ۵ است در نتیجه به یک دستگاه معادلات با $۲۰ = ۵ \times ۴$ معادله به‌صورت زیر می‌رسیم. (توجه کنید که با در نظر گرفتن معادلات میدان یعنی $X^۲ = X$ به ازای $X \in \{x, y, z, w\}$ همه‌ی معادلات فاقد جمله مربعی است.)

$$
\begin{array}{ll}
۱ + x + y + z + wz + yz = ۰ & w + y + wx + xz + yz = ۰ \\
w + wx + wy + wyz = ۰ & w + wy + wx + wxz + wyz = ۰ \\
xy + xz + wxz + xyz = ۰ & xy + xz + xyz = ۰ \\
xy + wyz = ۰ & wy + y + wxy + xyz + yz = ۰ \\
xz + wz = ۰ & wz + wxz + xz = ۰ \\
x + z + wx + wy + wz + xy + xz + yz = ۱ & x + wx + wy + wz + yz = ۱ \\
wy + wxy + wxz + wyz = w & wy + wz + wyz = w \\
wx + wxy + wxz + xy + xyz = ۰ & wx + wxy + wxz + xyz = ۰ \\
wxy + wy + wyz + xyz = y & xy + wxy + wy + wyz + yz = y \\
xz + wxz + wyz + wz + xyz + yz = ۰ & xz + wxz + wyz + wz + yz = z
\end{array}
$$

اگر هر یکجمله‌ای از دستگاه فوق را با یک متغیّر جدید جایگزین کنیم یک دستگاه خطّی ۲۰ معادله و ۱۴ مجهول به‌دست می‌آید. دستگاه خطی به‌دست آمده دارای جواب یکتای زیر است.

$$\{w = ۰, x = ۱, y = ۰, z = ۱, wx = ۰, wy = ۰, wy = ۰, wz = ۰, xy = ۰$$

$$, xz = ۱, yz = ۰, wxy = ۰, wyz = ۰, wxz = ۰, xyz = ۰\}$$

این مجموعه جواب از دستگاه خطّی یک جواب دستگاه معادلات چند جمله‌ای اصلی نیز است. از طرفی می‌دانیم که هر جواب از دستگاه اصلی یک جواب از دستگاه خطّی نیز هست بنابراین ما تمام جواب‌های دستگاه اصلی را یافته‌ایم.

در ادامه مثال دیگری از اجرای الگوریتم XL آوردیم که از [۱] اخذ شده است.

مثال ۴.۴. دستگاه معادلات مربعی زیر از حلقه‌ی $\mathbb{F}_{127}[x_1, x_2]$ را در نظر بگیرید:

$$\mathcal{S} := \begin{cases} f_1 = x_1^2 + 80x_1x_2 + 114 = 0 \\ f_2 = x_2^2 + 107x_1x_2 + 29 = 0 \end{cases}$$

الگوریتم XL را به صورت زیر روی دستگاه فوق اجرا می‌کنیم:

۱. فرض می‌کنیم $D = 4$

۲. چون درجه‌ی دستگاه برابر است با $d = 2$ لذا همه‌ی یکجمله‌ای‌ها از درجه‌ی حداکثر $D-d = 2$ را در تمام معادلات دستگاه $\mathcal{S}$ ضرب می‌کنیم و دستگاه زیر به‌دست می‌آید.

$$\begin{array}{ll}
x_1{}^4 + 80\, x_1{}^3x_2 + 114x_1{}^2 = 0 & 107x_1{}^2x_2{}^2 + x_1\, x_2{}^3 + 29\, x_1\, x_2 = 0 \\
x_1{}^3x_2 + 80\, x_1{}^2x_2{}^2 + 114x_1\, x_2 = 0 & 107x_1\, x_2{}^3 + x_2{}^4 + 29\, x_2{}^2 = 0 \\
x_1{}^2x_2{}^2 + 80\, x_1\, x_2{}^3 + 114x_2{}^2 = 0 & 107x_1{}^2x_2 + x_1\, x_2{}^2 + 29\, x_1 = 0 \\
x_1{}^3 + 80\, x_1{}^2x_2 + 114x_1 = 0 & 107x_1\, x_2{}^2 + x_2{}^3 + 29\, x_2 = 0 \\
x_1{}^2x_2 + 80\, x_1\, x_2{}^2 + 114x_2 = 0 & x_1{}^2 + 80\, x_1\, x_2 + 114 = 0 \\
107x_1{}^3x_2 + x_1{}^2x_2{}^2 + 29\, x_1{}^2 = 0 & 107x_1\, x_2 + x_2{}^2 + 29 = 0
\end{array}$$

۳. با اعمال تغییر متغیر زیر

$$x_ix_jx_kx_l \mapsto y_{ijkl},$$

و جایگزینی هر یکجمله‌ای با یک متغیّر جدید معادلات خطی زیر به‌دست می‌آید.

$$\begin{cases} y_{111} + 80\, y_{112} + 114 y_1 = 0 \\ y_{1111} + 80\, y_{1112} + 114 y_{11} = 0 \\ y_{1112} + 80\, y_{1122} + 114 y_{12} = 0 \\ y_{112} + 80\, y_{122} + 114 y_2 = 0 \\ y_{1122} + 80\, y_{122} + 114 y_{22} = 0 \\ 107 y_{1112} + y_{1122} + 29\, y_{11} = 0 \\ 107 y_{112} + y_{122} + 29\, y_1 = 0 \\ 107 y_{1122} + y_{1222} + 29\, y_{12} = 0 \\ 107 y_{122} + y_{222} + 29\, y_2 = 0 \\ 107 y_{1222} + y_{2222} + 29\, y_{22} = 0 \\ y_{11} + 80\, y_{12} + 114 = 0 \\ y_{22} + 107 y_{12} + 29 = 0 \end{cases} \xrightarrow{\text{حل به‌روش حذفی گاوس}} \begin{cases} y_1 = 16\, z_1 + 48 + 10\, z_2 \\ y_{11} = 110 + 79\, z_2 \\ y_{111} = 3 z_1 + 121 + 94 z_2 \\ y_{1111} = 5 + 125\, z_2 \\ y_{1112} = 97 z_2 + 121 \\ y_{112} = 83 z_2 + 119 + 74 z_1 \\ y_{1122} = 30\, z_2 + 119 \\ y_{12} = 48 + 26\, z_2 \\ y_{122} = 100\, z_2 + 99 \\ y_{1222} = 100\, z_2 + 99 \\ y_2 = 35\, z_1 + 42 + 104\, z_2 \\ y_{22} = 42 + 12\, z_2 \\ y_{222} = z_1 \\ y_{2222} = z_2 \end{cases}$$

در دستگاه فوق z_1, z_2 متغیرهای آزاد هستند.

۴. یک معادله یک متغیّره که در بین معالات به‌دست آمده مشاهده می‌شود عبارت‌است از

$$y_{22} = 42 + 12 y_{2222} \Longrightarrow x_2^2 = 42 + 12 x_2^4 \Longrightarrow x_2 = \pm 36 = 36, 91$$

۵. با جایگذاری مقدار $x_2 = \pm 36$ در معادله مقدار x_1 نیز به‌دست می‌آید که عبارت است از $x_1 = \pm 38$.

زمان اجرای الگوریتم XL تا حد زیادی وابسته به عملیات حذفی گاوس است که در هر بار اجرای الگوریتم باید انجام شود. می‌دانیم که عملیات حذفی گاوس به اندازه‌ی دستگاه خطی ورودی وابسته است، از طرفی اندازه‌ی دستگاه خطی به‌دست آمده به پارامتر D وابسته است. در نتیجه زمان اجرای الگوریتم XL بر حسب پارامتر D نمایی است و این پارامتر باید به دقت و به‌صورت بهینه انتخاب شود.

انتخاب پارامتر D الگوریتم XL

یک دستگاه معادلات مربعی با m معادله و n مجهول را در نظر بگیرید. در الگوریتم XL به ازای پارامتر D مشخص، ابتدا همه‌ی معادلات در یکجمله‌ای‌های از درجه‌ی حداکثر $D-۲$ یعنی همه‌ی یکجمله‌ای‌های $\mathbb{T}^{n}_{\leq D-۲}$ ضرب می‌شوند. بنابراین اگر تعداد معادلات جدید را با R و تعداد یکجمله‌ای‌های ظاهر شده در معادلات جدید را با T نمایش دهیم داریم،

$$R = m \cdot (\sum_{i=\circ}^{D-d} \binom{n}{i}) \approx \binom{n}{D-۲} \cdot m \overset{D \lll n}{\Longrightarrow} R \approx \frac{n^{D-۲}}{(D-۲)!} m$$

$$T = \sum_{i=\circ}^{D} \binom{n}{i} \approx \binom{n}{D} \overset{D \lll n}{\Longrightarrow} T \approx \frac{n^{D}}{D!}.$$

پس از خطی سازی همه‌ی یکجمله‌ایی‌ها با یک متغیر جدید جایگزین می‌شوند و یک دستگاه خطی به‌دست می‌آید. مشکل اصلی این‌جاست که همه‌ی معادلات خطی به‌دست آمده مستقل خطی نیستند. اگر تعداد معادلات مستقل خطی ظاهر شده را با Free نمایش دهیم، در این‌صورت بدیهی است که، $\mathrm{Free} \leq T$ و $\mathrm{Free} \leq R$. بنابراین اگر فرض کنیم تعداد زیادی از معادلات به‌دست آمده پس از ضرب یکجمله‌ای‌ها مستقل خطی هستند، آن‌گاه با انتخاب D به اندازه‌ی کافی بزرگ به طوری که $R \geq T$، می‌توانیم به تقریب $\mathrm{Free} \approx T$ دست پیدا کنیم و جواب دستگاه معادلات خطی به سمت یکتایی پیش می‌رود و حل ساده‌تر خواهد بود. در نتیجه کران پایین پارامتر D برای این‌که الگوریتم XL مؤفق عمل کند، به‌صورت زیر محاسبه می‌شود.

$$R \geq T \;\Rightarrow\; m \geq \frac{\binom{n}{D}}{\binom{n}{D-۲}} \approx \frac{n^{۲}}{D^{۲}} \Longrightarrow D \gtrsim \frac{n}{\sqrt{m}}$$

نکته ۵.۴. پیچیدگی محاسباتی روش حذفی گاوس برای حلّ یک دستگاه خطّی با T متغیّر برابر است با، $\mathcal{O}(T^{\omega})$. در عمل معمولاً فرض می‌شود که $\omega = ۳$ ولی بهترین نتایج به‌دست آمده نشان می‌دهد که، $\omega \leq ۲/۳۷۶$ و ما نیز از فرض $\omega = ۲/۳۷۶$ استفاده می‌کنیم.

بنابراین پیچیدگی الگوریتم XL برابر است با:

$$\mathcal{O}(\binom{n}{D}^{\omega}) \approx \mathcal{O}((\frac{n^{D}}{D!})^{\omega}); \quad D = \mathcal{O}(\frac{n}{\sqrt{m}}) \;,\; \omega = ۲/۳۷۶.$$

تعریف ۶.۴ (دستگاه معادلات بیش‌تعریف). فرض کنید در یک دستگاه چندجمله‌ای با m معادله و n مجهول نسبت تعداد معادلات به تعداد مجهولات را با $\gamma := \frac{m}{n}$ نمایش دهیم. اگر $\gamma > ۱$ باشد دستگاه را بیش‌تعریف و اگر $r < ۱$ باشد دستگاه را کم‌تعریف می گوییم.

تعریف ۷.۴ (دستگاه معادلات چندجمله‌ای تنک). فرض کنید S یک دستگاه چندجمله‌ای با m معادله و n مجهول و از درجه‌ی d باشد. فرض کنید تعداد کل یکجمله‌ای‌ها با n متغیر و از درجه‌ی d را با M و تعداد یکجمله‌ای‌های ظاهر شده در معادله را با t نمایش دهیم، در این‌صورت نسبت $\beta := \frac{t}{M}$ بیان‌گر میزان تنک بودن دستگاه است و اگر $\beta \leq \frac{1}{۲}$ ، دستگاه را تُنُک می‌گوییم.

مبدعان روش XL در [۱۶]، امیدوار بودند که الگوریتم XL قادر باشد دستگاه معادلات چندجمله‌ای مربعی بیش تعریف با ابعاد بزرگ روی یک میدان متناهی را در زمان زیرنمایی حل نماید اما تحلیل‌های دقیق‌تر روی این الگوریتم در [۱۶، ۲۷]، حاکی از بعید بودن این ادعا است. برای مثال یکی از بهترین جبری سازی‌ها برای الگوریتم رمزنگاری AES-128 ، دستگاه معادلاتی است که در [۲۲]، برای AES-128 به‌دست آمده، این دستگاه دارای $m = ۸۰۰۰$ معادله و $n = ۱۶۰۰$ مجهول است. بنابراین $D \approx \frac{n}{\sqrt{m}} \approx ۱۸$ و لذا مرتبه‌ی پیچیدگی حمله‌ی XL تقریباً برابر است با $\binom{n}{D}^{\omega} \approx ۲^{۳۳۰}$ ، که خیلی فراتر از حمله‌ی جست‌وجوی فراگیر از مرتبه‌ی پیچیدگی $۲^{۱۲۸}$ است!

در برخی از مقالات نظیر [۶۴]، ارتباطهایی که بین روش XL و روش پایه‌گروبنر وجود دارد مورد بحث قرار گرفته و برای مثال در [۵] نشان داده شده است که الگوریتم XL حالت خاصی از الگوریتم F4، برای یافتن پایه‌ی گروبنر است. الگوریتم XL در نرم‌افزار ApCoCoA[۴۴] پیاده‌سازی شده است و با استفاده از دستور `CharP.XLSolve(.)`، فراخوانی می‌شود. الگوریتم XL پیاده‌سازی شده در این نرم‌افزار قادر به یافتن یک جواب است. این الگوریتم را برای حل دستگاه‌های به‌دست آمده از رمزهای قالبی CTC و SR آزمودیم و طبق نتایج به‌دست آمده آن‌را در مقایسه با روش‌های دیگر که در ادامه معرفی می‌کنیم بسیار ناکارآمد یافتیم، برای مثال دستگاه به‌دست‌آمده از $CTC(B = ۱, N = ۱)$ به ازای یک زوج متن اصلی و رمزشده که دارای جواب یکتا بود، با استفاده از دستور `CharP.XLSolve(.)` در رایانه با پردازنده $۱/۶GHz$ و حافظه رم $۶GB$ حل گردید، و مشاهده شد که زمان خالصی که پردازنده صرف حل کرده است برابر است با ۱۲/۹۶ ثانیه، که خواهیم دید در مقایسه با سرعت حل‌کننده‌های دیگر بسیار کند عمل کرده است.

۴.۱.۴ روش XSL

بدلیل ناکارآمدی حمله‌ی XL، تلاش‌های زیادی برای بهبود این الگوریتم صورت گرفته و راه‌حل‌هایی نیز ارائه شده است. یکی از اولین پیشنهادها الگوریتمی تحت عنوان XSL [۷] بود که به‌طور مشترک توسط کورتوا و پیپشیک در سال ۲۰۰۲ در [۲۲] معرفی شد. یکی از مشکلات الگوریتم XL این است که ضمن افزایش تعداد معادلات، متغیرهای دستگاه را نیز افزایش می‌دهد، چرا که در مرحله‌ی ضرب الگوریتم XL، معادلات در تمام یکجمله‌ای ضرب می‌شوند و این سبب می‌شود یکجمله‌ای‌های جدیدی ظاهر شوند. الگوریتم XSL از این لحاظ متفاوت از الگوریم XL عمل

[7] e**X**tended **S**parse **L**inearization or multiply(**X**) by **S**pecial monomial and **L**inearization

می‌کند، در حالی که در الگوریتم XL معادلات در تمام یکجمله‌ای‌ها از درجه‌ی $D-d$ ضرب می‌شوند، در الگوریتم XSL معادلات تنها در یک سری یکجمله‌ای‌ها که به دقّت از بین همه‌ی یکجمله‌ای‌ها انتخاب شده‌اند ضرب می‌شوند و هدف از این کار تولید کمتر یکجمله‌ای‌ها به هنگام ساختن معادلات جدید است. در ضمن در این الگوریتم مرحله‌ای تحت عنوان « روش T'» وجود دارد که در آن سعی می‌کنیم بدون تولید هیچ یکجمله‌ای جدید معادلات مستقل خطّی جدید تولید کنیم که در نوع خود جالب است.

این حمله با بهره‌گیری از ویژگی تنک و یا کم پشت بودن معادلات چند جمله‌ای که در رمزنگاری با آن‌ها سروکار داریم توانسته روش XL را بهبود بخشد و درخور دسته‌ای از سامانه‌های رمزنگاری متقارن تحت عنوان سامانه‌های رمزنگاری XSL است.

تعریف ۸.۴ (رمزهای قالبی جانشینی-خطی). یک تعمیم از رمزهای قالبی با معماری شبکه‌ی جانشینی-جایگشتی رمزهای قالبی با معماری جانشینی-خطّی است، که در آن‌ها از لایه‌های جانشینی و لایه‌های خطّی (نه لزوما جایگشت) در شبکه استفاده می‌شود. به این سامانه‌های رمز اصطلاحاً رمزهای جانشینی- خطّی گفته می‌شود.

تعریف ۹.۴ (سامانه‌های رمزنگاری XSL). یک دسته‌ی خاص از رمزهای جانشینی-خطی رمزهای XSL متشکل از N_r دور مشابه، به صورت زیر است:

X **ترکیب کلید با متن اصلی**: دور آغازین ($i=۱$)، یا با ترکیب متن اصلی و کلید ($k_{i-۱}$) آغاز می‌شود این ترکیب می‌تواند با عمل xor با استفاده از یک عملیات ریاضی دلخواه دیگر صورت گیرد.

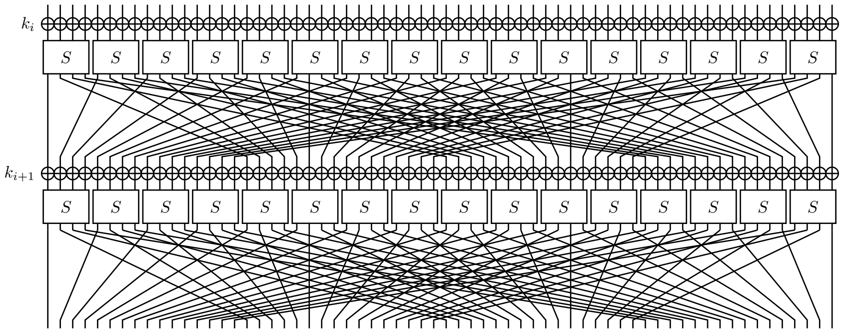
S **لایه‌های غیر خطی**: یک لایه‌ی غیر خطّی شامل B ، s-box موازی s بیتی روی بیت‌های خروجی از لایه‌ی قبل اعمال می‌شود.

L **لایه‌ی خطی**: یک لایه‌ی خطی جهت انتشار و توزیع یکنواخت بیت‌ها روی بیت‌های خروجی مرحله‌ی قبل، اعمال می‌شود. این لایه‌ی خطی می‌تواند یک جایگشت باشد.

X بیت‌های خروجی از لابه‌ی خطّی با کلید دور k_i (با استفاده از عمل‌های مختلف ریاضی مثل xor یا جمع پیمانه‌ای و ...) ترکیب می‌شوند.

- اگر $i=N_r$ الگوریتم خاتمه می‌یابد و در غیر این صورت پس از افزایش ۱ واحدی i به گام S می‌رویم.

شکل، ۱.۴ دور i ام از یک رمز XSL، که لایه‌ی خطّی آن تنها شامل یک جایگشت است را نمایش می‌دهد.



شکل ۱.۴: دور i ام از یک XSL-Cipher که لایه‌ی خطّی آن فقط شامل یک جایگشت است

به طور خلاصه در یک رمز قالبی XSL، لایه‌های s-box به‌وسیله‌ی لایه‌های خطی وابسته به کلید به هم متصل می‌شوند و یک شبکه‌ی N_r دوری که در هر دور b عدد s-box موازی شده‌اند را تشکیل می‌دهند. به عنوان نمونه‌ای از این نوع رمزها می‌توان به الگوریتم رایندال [۸] اشاره کرد.

با وجود این‌که حمله‌ی XSL برای سامانه‌های رمزنگاری XSL شرح داده شده امّا این حملات به سایر رمزهای جانشینی_خطی حتی در حالت کلی برای سایر رمزهای قالبی نظیر DES که نگاشت‌های جانشینی آن‌ها از یک ساختار خاص تبعیّت کند قابل تعمیم است.

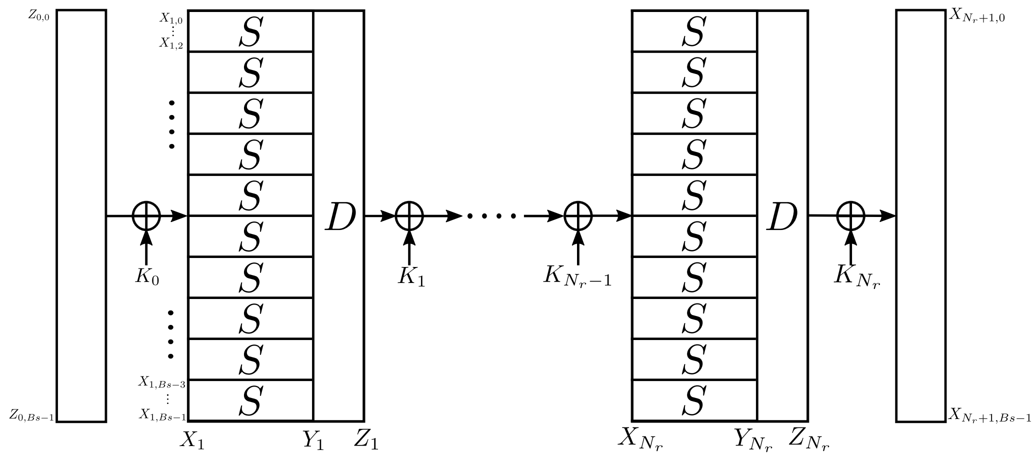
نمادگذاری‌ها در این‌بخش به‌صورت زیر است.

بیت‌های کلید را با k_{ij} که $i = \circ ... N_r$ و $j = \circ ... (B \cdot s - ۱)$ نمایش می‌دهیم. در کل $N_r + ۱$ کلید دور، خواهیم داشت که توسط الگوریتم استخراج کلید استخراج شده و $k_\circ$ و k_{N_r} به ترتیب اوّلین و آخرین آن‌ها می‌باشند. بیت j ام از ورودی دور i ام (بعد از اعمال کلید دور قبل) از الگوریتم را با x_{ij} نمایش می‌دهیم. بیت‌های ورودی لایه‌ی خطّی دور i ام را با y_{ij} نمایش می‌دهیم. به همین ترتیب بیت‌های خروجی دور i ام (پس از اعمال کلید دور) را با z_{ij} نمایش می‌دهیم. بنابراین $Z_\circ = (z_{\circ j})_{j=\circ}^{Bs-۱}$ نشان دهنده‌ی متن اصلی و $X_{N_r+۱} = (x_{(N_r+۱)j})_{j=\circ}^{Bs-۱}$ نشان دهنده‌ی متن رمز شده است . در حمله‌ای XSL هدف پیدا کردن کلید مخفی است و فرض را بر این می‌گذاریم که مهاجم یک یا چند زوج متن اصلی و رمز شده را دارد و در واقع، حمله از نوع متن اصلی معلوم است. لذا $X_{N_r+۱}$ و $Z_\circ$ مجهول نبوده و ثابت هستند. با توجّه به نمادگذاری‌های فوق داریم:

$$z_{ij} \oplus k_{ij} = x_{i+۱\ j}\ ,\ i = \circ ... N_r$$

این نماد گذاری‌ها برای سامانه‌ی رمزنگاری CTC که یک نوع رمز XSL است در شکل ۲.۴ نمایش داده شده است.

[8]Rijndael



شکل ۲.۴: الگوریتم رمزنگاری CTC، به ازای $B = ۱۰$

سناریوی حمله در روش XSL

در روش XSL دو نوع سناریو حمله با نام‌های حمله‌ی عام (یا نوع اول) و حمله‌ی خاص و(یا نوع دوم) وجود دارد، که در زیر به شرح آن‌ها می‌پردازیم.

تعریف ۱۰.۴ (حمله‌ی XSL عام‌ـ صرف نظر از الگوریتم تولید کلید). در این حمله، هدف یافتن کلید اصلی نیست، بلکه هدف، یافتن کلید هر یک از دورها است و معادلات بر حسب بیت‌های زیرکلیدها (کلید هر یک از دورها) و سایر متغیرهای میانی سامانه رمزنگاری نوشته می‌شوند. بنابراین مهم نیست که الگوریتم تولید کلید چگونه کار می‌کند. از آن‌جایی‌که $N_r + ۱$ زیر کلید هم اندازه با متن اصلی وجود دارد به تعداد کافی قید یا معادله نیاز داریم تا این مجهولات را بیابیم از این رو به $N_r + ۱$ زوج متن اصلی و رمزشده آن‌ها نیاز داریم. در حالتی که حمله از نوع متن اصلی انتخابی باشد، می‌توان متن‌های اصلی را با این خاصیّت که تنها در تعداد کمی از بیت‌هاي متناظر با یک s-box، متمایز باشند انتخاب کرد تا معادلات به‌دست آمده برای زوج متن‌های اصلی و رمز شده دارای مجهولات مشترک بیشتری باشند و تعداد مجهولات کل دستگاه نسبت به حالت عادی کمتر باشد.

تعریف ۱۱.۴ (حمله‌ی XSL خاص با در نظر گرفتن الگوریتم تولید زیر کلید). در این سناریو، هدف یافتن کلید مخفی اصلی است. این روش بر این واقعیّت استوار است که الگوریتم‌های تولید کلید در برخی از رمزهای XSL نظیر رایندال و سرپنت [9] ، به الگوریتم رمزنگاری متناظرشان شباهت زیادی دارند. لذا در این روش از الگوریتم تولید زیر کلید در حمله‌ی جبری و نوشتن معادلات استفاده می‌شود. بدیهی است که شرط وجود شباهت بین الگوریتم‌های تولید زیرکلید و الگوریتم رمزنگاری از عمومیّت این حمله نسبت به حمله‌ی عام می‌کاهد ولی مزیّت آن این است که تنها داشتن یک یا دو زوج متن اصلی ـ رمزشده برای حمله کفایت می کند.

[9] serpent

هسته‌ی حمله‌ی XSL عام

یکی از مراحل اصلی در روش XSL مرحله موسوم به هسته‌ی XSL است. هدف این مرحله تولید معادلات مستقل خطی جدید است. این‌کار با ضرب معادلات هر یک از $s-box$ ها در یکجمله‌ای‌هایی که به روشی خاص انتخاب شده‌اند صورت می‌گیرد. فرض کنید **A** یک s-box از سامانه‌ی رمز XSL مورد نظر باشد، به s-box ، A که در این وضعیت آن را انتخاب کرده‌ایم s-box *فعال* و به سایر s-box ها s-box های *غیر فعال* می‌گوییم. فرض کنید، دستگاه متناظر با A شامل r معادله‌ی ضمنی درجه‌ی ۲ به‌صورت زیر باشد.

$$\circ = \sum \alpha_{ijk} x_{ij} Y_{ik} + \sum \beta_{ij} x_{ij} + \sum \gamma_{ij} y_{ij} + \sigma \qquad (۳.۴)$$

فرض کنید کنید تعداد یکجمله‌ای‌های موجود در معادله‌ی فوق t باشد.

فرض کنید S تعداد کلّ s-box های شرکت کننده در حمله باشد. از آن‌جایی که در حمله‌ی XSL عام باید $N_r + ۱$ زوج متن اصلی_رمزشده متمایز را در اختیار داشته باشیم لذا الگوریتم رمزنگاری باید $N_r + ۱$ بار اجرا شده باشد. از طرفی در ساختار رمزنگاری از $B \times N_r$ ، s-box استفاده شده است در نتیجه به تعداد $B \times N_r \times (N_r + ۱)$ دستگاه معادله‌ی مختلف شبیه دستگاه معادله‌ی ۳.۴ خواهیم داشت و در نتیجه رابطه‌ی $S = B \times N_r \times (N_r + ۱)$ برقرار است.

بر خلاف روش XL که همه‌ی تک جمله‌ای‌ها از درجه‌ی $D - ۲$ را در معادلات دستگاه ضرب می‌کردیم این بار این کار را به صورت گزینشی و نه برای همه‌ی یکجمله‌ای‌ها انجام می‌دهیم. برای این گزینش از پارامتری تحت عنوان *پارامتر بحرانی* استفاده می‌کنیم و آن را با P نمایش می‌دهیم.

تعریف ۱۲.۴ (پارامتر بحرانی). پارامتر بحرانی ، که آن را با P نمایش می‌دهیم، نشان دهنده این است که، برای به‌دست آوردن معادلات جدید برای یک s-box فعال، باید تمام معادلات آن‌را در تمام یکجمله‌ای‌های حاصل از ضرب $p-۱$ یکجمله‌ای انتخاب شده از s-box های غیرفعال، ضرب کنیم. برای مثال اگر $P = ۲$ ، آن‌گاه به منظور اضافه کردن معادلات جدید به دستگاه معادلات متناظر با یک s-box فعّال، معادلات آن‌را در تمام تک جمله‌ای‌های به‌دست آمده از s-box های غیر فعّال ضرب می‌کنیم و اگر $P = ۳$ باید معادلات s-box فعال را در تمام یکجمله‌ای‌های حاصل از ضرب دو یکجمله‌ای به‌دست آمده از دو s-box غیرفعال ضرب کنیم.

تعداد کلّ معادلات به‌دست آمده پس از افزودن معادلات جدید برابر است با:

$$R \approx r \times S \times t^{P-۱} \times \binom{S-۱}{P-۱}$$

تعداد همه‌ی یکجمله‌ای‌های ظاهر شده در معادلات جدید که با T، نمایش می‌دهیم در زیر محاسبه

شده.

$$T \approx \frac{R}{r} \cdot t = t^p \times S \times \binom{S-1}{P-1} = t^p \times P \times \binom{S}{P} \approx t^p \times \binom{S}{P}$$

معادلات جدید که در گام هسته‌ی XSL و با ضرب یکجمله‌ای‌ها در معادلات دستگاه به‌دست می‌آیند دارای وابستگی خطی بوده و اعمال روش خطی‌سازی روی آن‌ها کارایی ندارد. به همین دلیل قبل از عملیات خطی سازی با استفاده از روشی موسوم به روش T' تعداد معادلات مستقل خطی را افزایش می‌دهیم.

پس از حذف معادلات وابسته‌ی خطی، تعداد کلّ معادلات و یکجمله‌ای‌ها به صورت زیر خواهد بود:

$$R \approx \binom{S}{P}(t^P - (t-r)^P)$$

$$T \approx \binom{S}{P}(t-r)^P$$

روش T'

در حمله‌ی XSL ، مرحله‌ای وجود دارد که بدون تولید یکجمله‌ای‌های جدید، تعداد معادلات مستقل را افزایش می‌دهد. این مرحله به روش T' معروف است. روش T'، آخرین مرحله قبل از خطی‌سازی در حمله‌ی XSL را تشکیل می‌دهد. به‌یاد داریم که عملیات خطی‌سازی زمانی در حل دستگاه مؤفق عمل می‌کند که تعداد معادلات مستقل خطی، تقریباً برابر با تعداد یکجمله‌ای‌های دستگاه باشد. معمولاً دستگاه به‌دست آمده از مرحله هسته XSL، به اندازه‌ی کافی معادله‌ی مستقل ندارد. هدف روش T' این است که کمبود معادلات مستقل را بدون این‌که یکجمله‌ای جدیدی به دستگاه اضافه شود جبران کند. به این ترتیب دستگاه‌ی که از مرحله‌ی T' عبور کرده و به مرحله‌ی نهایی خطی‌سازی می‌رسد، به اندازه‌ی کافی معادله‌ی مستقل خطی خواهد داشت. روش T' در [۲۲] برای معدلات روی میدان $\mathbb{F}_2$ معرفی شده ولی می‌توان آن را به نحوی مناسب برای میدان‌های دیگر نیز طراحی کرد.

تعریف ۱۳.۴ ($\mathcal{T}'_i$). اگر $\mathcal{T}$ مجموعه‌ی همه‌ی یکجمله‌ای‌های دستگاه باشد، مجموعه‌ی $\mathcal{T}'_i$ مجموعه‌ی همه‌ی یکجمله‌ای‌هایی از دستگاه است که به ازای آن‌ها داریم، $x_i\mathcal{T}'_i \subseteq \mathcal{T}$. به عبارت دیگر با ضرب متغیر x_i در یکجمله‌ای‌های موجود در $\mathcal{T}'_i$ ، یکجمله‌ای جدیدی به دستگاه اضافه نمی‌شود. در ادامه اندازه‌ی $\mathcal{T}'$ را با T' نمایش می‌دهیم.

فرض کنید $S \subseteq \mathbb{F}_2[x_1, ..., x_n]$ یک دستگاه معادلات چندجمله‌ای باشد که دارای E معادله‌ی مستقل خطی است. در ضمن فرض کنید به ازای هر $1 \leq i \leq n$ داشته باشیم، $E \geq T - T'_i + C_i$ که $C_i \geq 1$ یک ثابت است. در این‌صورت الگوریتم روش T' به‌صورت زیر تعداد معادلات مستقل خطی دستگاه S را افزایش می‌دهد.

الگوریتم ۲۰ T'-Method

ورودی دستگاه $\mathcal{S} := \{f_1, ..., f_m\} \in K[x_1, ..., x_n]$ با E معادله‌ی مستقل خطی، بطوری که $E \geq T - T'_i + C_i$ و $C_i \geq 1$.

خروجی دستگاه $\mathcal{S}'$ با همان یکجمله‌ای‌های موجود در $\mathcal{S}$ ، و با E معادله‌ی مستقل خطی بطوری که $E = T - 1$.

۱: دو متغیر x_i و x_j را انتخاب کرده و $\mathcal{T}'_i$ و $\mathcal{T}'_j$ را به‌دست می‌آوریم.

۲: با استفاده از عملیات حذفی گاوس یکجمله‌ای‌های موجود در $\mathcal{T}\backslash\mathcal{T}'_i$ را بر حسب یکجمله‌ای‌های $\mathcal{T}'_i$ می‌نویسیم. همین کار را برای $\mathcal{T}'_j$ نیز انجام می‌دهیم. بعد از انجام این‌کار با توجه به شرط $E \geq T - T'_i + C_i$ ، تقریباً C_i معادله خواهیم داشت که تمام یکجمله‌ای‌های آن متعلق به $\mathcal{T}'_i$ است. (همین وضعیت برای دستگاه متناظر با $\mathcal{T}'_j$ نیز برقرار است.) مجموعه‌ی معادلاتی از دستگاه اول را که فقط بر حسب یکجمله‌ای‌های $\mathcal{T}'_i$ هستند با $\mathcal{C}_i$ و مجموعه‌ی معادلاتی از دستگاه دوم، که فقط بر حسب جملات $\mathcal{T}'_j$ است را با $\mathcal{C}_j$ نمایش می‌دهیم.

۳: معادلات $\mathcal{C}_i$ را در x_i و معادلات $\mathcal{C}_j$ را در x_j ضرب می‌کنیم. بدیهی است که با این‌کار هیچ یکجمله‌ای جدیدی بوجود نمی‌آید.

۴: یکجمله‌ای‌های به‌دست آمده پس از ضرب x_i در $\mathcal{C}_i$ را بر حسب یکجمله‌ای‌های $\mathcal{T}'_j$ و یکجمله‌ای‌های به‌دست آمده پس از ضرب x_j در $\mathcal{C}_j$ را بر حسب یکجمله‌ای‌های $\mathcal{T}'_i$ بازنویسی می‌کنیم. معادلات مستقل خطی به‌دست آمده را به دستگاه $\mathcal{S}$ اضافه می‌کنیم.

۵: گام‌های ۱ تا ۴ را تا رسیدن به شرط $E = T - 1$ تکرار می‌کنیم. اگر الگوریتم به ازای $\mathcal{T}'_i$ و $\mathcal{T}'_j$ انتخاب شده به شکست انجامید، با دو زوج دیگر مثل $\mathcal{T}'_k$ و $\mathcal{T}'_l$ الگوریتم را از سر می‌گیریم.

نویسندگان در [۲۲] ادعا کرده‌اند که اگر دستگاه اولیه دارای جواب یکتا باشد، تعداد معادلات مستقل در الگوریتم T' به‌صورت نمایی افزایش یافته، بطوری‌که الگوریتم پس از متناهی مرحله اجرا با رسیدن تعداد معادلات مستقل به عدد $T - 1$ پایان می‌یابد. الگوریتم فوق را با مثالی از ارائه دهندگان این روش در [۲۲]، به‌صورت دقیق‌تر شرح می‌دهیم.

مثال ۱۴.۴ (T'-Methode)**.** فرض کنید تعداد متغیّرها $n = 5$ باشد در این صورت $T = 16$ و $T' = 10$. ما با یک معادله‌ی تصادفی با جواب یکتا، که در آن $E = T - T' + 2$ (یعنی دو معادله‌ی اضافی دارد)، آغاز می‌کنیم. $\mathcal{T}'_1$ را متناظر با x_1 و $\mathcal{T}'_2$ را متناظر با متغیّر x_2 در نظر می‌گیریم.

دستگاه مورد نظر پس از این‌که تمام یکجمله‌ای‌های آن بر حسب یکجمله‌ای‌های $\mathcal{T}' = \mathcal{T}'_1$

نوشته شده‌اند به‌صورت زیر است.

$$\begin{cases} x_3x_2 = x_1x_3 + x_2 \\ x_3x_4 = x_1x_4 + x_1x_5 + x_5 \\ x_3x_5 = x_1x_5 + x_4 + 1 \\ x_2x_4 = x_1x_3 + x_1x_5 + 1 \\ x_2x_5 = x_1x_3 + x_1x_2 + x_3 + x_4 \\ x_4x_5 = x_1x_2 + x_1x_5 + x_2 + 1 \\ 0 = x_1x_3 + x_1x_4 + x_1 + x_5 \\ 1 = x_1x_4 + x_1x_5 + x_1 + x_5 \end{cases} \tag{۴.۴}$$

به طور مشابه با نوشتن تمام یکجمله‌ای‌های دستگاه اصلی بر حسب یکجمله‌ای‌های $T' = T'_2$ دستگاه به‌صورت زیر درمی‌آید.

$$\begin{cases} x_1x_3 = x_3x_2 + x_2 \\ x_1x_4 = x_3x_2 + x_2 + x_1 + x_5 \\ x_1x_5 = x_2x_4 + x_3x_2 + x_2 + 1 \\ x_3x_5 = x_2x_4 + x_3x_2 + x_2 + 1 + x_4 + 1 \\ x_3x_4 = x_2x_4 + x_1 + 1 \\ x_4x_5 = x_1x_2 + x_2x_4 + x_3x_2 \\ 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \end{cases} \tag{۵.۴}$$

توجه کنید که هر دو دستگاه فوق، فقط نمایش‌های متفاوتی از یک دستگاه هستند. در این مرحله، رتبه دستگاه خطّی متناظر، برابر ۸ است. دو معادله‌ی آخر دستگاه ۴.۴ فقط بر حسب یکجمله‌ای‌های موجود در T'_1 هستند، بنابراین با ضرب این معادلات در x_1 ، بدون این که هیچ یکجمله‌ای جدیدی بوجود آید معادلات زیر را به‌دست می‌آوریم.

$$\begin{cases} 0 = x_1x_3 + x_1x_4 + x_1 + x_1x_5 \\ 0 = x_1x_4 \end{cases} \tag{۶.۴}$$

چون معادلات فوق نسبت به معادلات دستگاه اول، مستقل خطی هستند، با اضافه کردن آن‌ها به دستگاه اصلی، رتبه‌ی دستگاه به ۱۰ افزایش پیدا می‌کند.

اکنون معادلات به‌دست آمده در ۶.۴ را با استفاده از معادلات دستگاه ۵.۴ بازنویسی می‌کنیم تا معادلات به‌دست آمده فقط بر حسب یکجمله‌ای‌های موجود در T'_2 باشند. به این ترتیب چهار معادله‌ی اضافی (۲ معادله که از قبل وجود داشت به همراه دومعادله که در این گام به‌دست

آوردیم) برای دستگاه ۵.۴ به صورت زیر داریم:

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_3x_2 + x_2 + x_3 + x_4 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5 + x_2 + 1 \\ 0 = x_3x_2 + x_2 + x_1 + x_5 \end{cases} \tag{7.4}$$

بدون این که هیچ یکجمله‌ای جدیدی بوجود آید معادلات دستگاه ۷.۴ را در x_2 ضرب می‌کنیم و به معادلات زیر می‌رسیم:

$$\begin{cases} 0 = x_1x_2 + x_2x_5 + x_2x_4 + x_2 \\ 0 = x_2x_4 \\ 0 = x_2x_4 + x_3x_2 + x_5x_2 \\ 0 = x_3x_2 + x_2 + x_1x_2 + x_2x_5 \end{cases} \tag{8.4}$$

در مورد معادله‌ی دوّم خوش‌شانس نبودیم چرا که تحت ضرب در x_2 ناورداست و شکل آن تغییری نمی‌کند با این حال تا این مرحله مؤفق شده‌ابم که سه معادله‌ی مستقل خطی جدید دیگر به‌دست آوریم. به این ترتیب رتبه دستگاه به عدد ۱۳ می‌رسد. معادلات مستقل جدید را به دستگاه اضافه می‌کنیم.

سه معادله‌ی جدید به‌دست آمده از مرحله‌ی قبل را با استفاده از دستگاه ۴.۴ بازنویسی می‌کنیم تا تمام جملات معادلات بر حسب یکجمله‌ای‌های موجود در $\mathcal{T}'_1$ باشد. به این ترتیب به معادلات زیر می‌رسیم:

$$\begin{cases} 1 = x_1x_5 + x_2 + x_3 + x_4 \\ 1 = x_1x_2 + x_1x_3 + x_1x_5 + x_2 + x_3 + x_4 \\ 0 = x_3 + x_4 \end{cases} \tag{9.4}$$

معادلات فوق نسبت به سایر معادلات موجود در دستگاه مستقل خطی نیستند و رتبه‌ی دستگاه همان ۱۳ باقی می‌ماند.
بدون این که هیچ یکجمله‌ای جدیدی بوجود آید معادلات فوق را در x_1 ضرب می‌کنیم و به معادلات زیر می‌رسیم:

$$\begin{cases} 1 = x_1x_5 + x_1x_2 + x_1x_3 + x_1x_4 \\ 1 = x_1x_5 + x_1x_4 \\ 0 = x_3 + x_4 \end{cases} \tag{10.4}$$

عملیات فوق با به‌دست آوردن یک معادله مستقل خطّی جدید همراه است و به این ترتیب رتبه‌ی دستگاه به عدد ۱۴ می‌رسد. با استفاده از دستگاه ۵.۴ معادله‌ی مستقل به‌دست آمده از مرحله‌ی قبل را بر حسب یکجمله‌ای‌های T_2' می‌نویسم:

$$0 = x_1x_2 + x_2x_4 + x_3x_2 + x_1 + x_2 + x_5$$

معادله‌ی فوق نسبت به معادلات دستگاه، معادله‌ی مستقلی نیست، با ضرب آن در x_2 (بدون این که هیچ یکجمله‌ای جدیدی بوجود آید) معادله‌ی مستقل خطی زیر را به‌دست می‌آوریم:

$$0 = x_2x_4 + x_3x_2 + x_2x_5 + x_2$$

در نتیجه با اضافه کردن این معادله به دستگاه اصلی، رتبه‌ی دستگاه به عدد $\text{rank} = 15$ می‌رسد! این بیشنه‌ی رتبه‌ای است که می‌توان برای دستگاه داشت چرا که تعداد یکجمله‌ای‌های غیر صفر موجود در دستگاه (و در نتیجه تعداد متغیّرهای دستگاه خطّی) برابر ۱۵ است.

روش T' گرچه برای اولین بار در حمله‌ی XSL در [۲۲] ، معرفی شد، ولی روشن است که می‌توان از آن در روش XL نیز استفاده کرد. در ادامه بخش دیگری از الگوریتم XSL ، که به هسته‌ی الگوریتم معروف است را معرفی می‌کنیم.

الگوریتم XSL

الگوریتم XSL از کنار هم قرار دادن مراحل قبلی، به صورت زیر به‌دست می‌آید.

الگوریتم ۲۱ الگوریتم XSL

ورودی دستگاه معادلات چندجمله‌ای مربعی به‌دست آمده از سامانه‌ی رمزنگاری XSL
خروجی جواب دستگاه

۱: فرض کنید هر s-box دارای r معادله و t یکجمله‌ای باشد. یک پایه‌ی خطی از بین یکجمله‌ای‌ها شامل $t-r$ یکجمله‌ای انتخاب کرده (به انضمام ۱ و بجز متغیرها) و r یکجمله‌ای دیگر را بر حسب آن‌ها می‌نویسیم.
۲: پارامتر بحرانی P را انتخاب می‌کنیم.(راجع به نحوه‌ی انتخاب بهینه‌ی P در [۲۲] بحث شده است.) معادلات به‌دست آمده از لایه‌های خطی را در یکجمله‌ای‌های پایه‌ی $P-1$، s-box مختلف ضرب می‌کنیم.
۳: تا جایی که امکان دارد، همه‌ی یکجمله‌ای‌ها را به‌صورت ترکیب خطی از یکجمله‌ای‌های پایه می‌نویسیم.
۴: الگوریتم T' را روی معادلات به‌دست آمده از مرحله‌ی قبل اعمال می‌کنیم.
۵: خروجی الگوریتم T' با روش خطی‌سازی به‌راحتی قابل حل است، لذا آن‌را خطی‌سازی کرده و با روش حذفی گاوس حل می‌کنیم.

الگوریتم XSL یک الگوریتم عمومی برای حل دستگاه‌های چندجمله‌ای نیست و فقط برای حل دستگاه‌های خاص متناظر با رمزهای قالبی XSL نظیر AES و Serpent طراحی شده است. از آنجایی‌که تحلیل‌های صورت گرفته توسط ارائه دهندگان حمله‌ی XSL در [۲۲] مورد پذیرش عموم رمزنگارها نیست و بحث‌های زیادی بر سر کارایی این حمله وجود دارد،(برای نمونه بنگرید به [۶۸، ۵۴]) بررسی بیشتر این روش را رها کرده و به مطالعه روش‌های جدیدتری که برای بهبود الگوریتم XL ارائه شده است، می‌پردازیم.

۵.۱.۴ روش MutatntXL

روش MutantXL که به اختصار آن را با نماد MXL نمایش می‌دهیم، اولین بار در سال ۲۰۰۸، توسط دینگ [10] و همکاران در [۲۹] به عنوان یکی دیگر از نسخه‌های بهبود یافته‌ی روش XL، معرفی شد. به‌طور کلی بسیاری از روش‌ها نظیر XL و F4 برای حل دستگاه چندجمله‌ای

$$S := \begin{cases} f_1(x_1, ..., x_n) = 0 \\ f_2(x_1, ..., x_n) = 0 \\ \vdots \\ f_m(x_1, ..., x_n) = 0 \end{cases}$$

[10] Ding

ابتدا با ضرب یکجمله‌ای‌ها در معادلات دستگاه، اعضای بیشتری از ایده‌ال تولیده شده توسط چندجمله‌ای‌های دستگاه یعنی $I = \langle f_۱, ..., f_m \rangle$، را می‌یابند. سپس دستگاه جدید را خطی‌سازی، و در انتها با عملیات حذفی گاوس آن را حل می‌کنند. ارائه دهندگان MXL با آزمایش‌هایی روی روش‌های مذکور، مشاهده کردند که طی مرحله‌ی حذفی گاوس، چندجمله‌ای‌هایی که درجه‌ی آن‌ها کمتر از آن چیزی است که در ابتدا به‌نظر می‌رسد، یا به عبارت دیگر چندجمله‌ای‌هایی که درجه‌ی آن‌ها در مرحله‌ی حذفی گاوس کاهش می‌یابد، می‌توانند سبب تسریع در فرآیند الگوریتم شوند. آن‌ها این چندجمله‌ای‌ها را *تغییرپذیر* نامیده و از آن‌ها برای بهبود الگوریتم XL استفاده کردند. در ادامه این چندجمله‌ای‌ها را به‌صورت دقیق تعریف کرده و الگوریتم MXL را معرفی می‌کنیم.

فرض کنید $\mathbb{F}_q$ یک میدان متناهی از مرتبه‌ی q و $f_۱, ..., f_m \in \mathbb{F}_q[x_۱, ..., x_n]$ چندجمله‌ای‌های دستگاه باشند. چون هدف ما پیدا کردن جواب‌هایی از دستگاه است که در $\mathbb{F}_q$ هستند، لذا روی حلقه‌ی خارج قسمتی

$$R := \frac{\mathbb{F}_q[x_۱, ..., x_n]}{\langle x_۱^q - x_۱, ..., x_n^q - x_n \rangle}$$

کار می‌کنیم، یا به‌طور معادل هر چندجمله‌ای از R را به پیمانه‌ی ایده‌ال $\langle x_۱^q - x_۱, ..., x_n^q - x_n \rangle$ تحویل می‌کنیم. فرضیات فوق تا پایان بحث روش MXL ، برقرار است.

تعریف ۱۵.۴ (سطح یک چندجمله‌ای). فرض کنید $I = \langle f_۱, ..., f_m \rangle$، و $g \in I$ ، دارای نمایشی به‌صورت $g = h_۱ f_۱ + \cdots + h_m f_m$ باشد. *سطح* این نمایش از g، را به‌صورت زیر تعریف می‌کنیم.

$$l = \max\{\deg(h_i f_i) |\ i \in \{۱, ..., m\}, h_i \neq ۰\}.$$

در این‌صورت کوچکترین سطح در بین تمام نمایش‌های $g = h_۱ f_۱ + \cdots + h_m f_m$ را سطح g نسبت به $F = (f_۱, ..., f_m)$ می‌گوییم و با نماد $\mathrm{level}_F(g)$ ، نمایش می‌دهیم.

تعریف ۱۶.۴ (چندجمله‌ای تغییر پذیر). فرض کنید $P = \mathbb{F}_q[x_۱, ..., x_n]$ و $F = (f_۱, ..., f_m) \subseteq P^m$. چندجمله‌ای $g \in P$ را نسبت به F، *تغییرپذیر* گوییم، هرگاه $\deg(g) < \mathrm{level}_F(g)$.

به‌طور شهودی وقتی $g \in P$ نسبت به $F = (f_۱, ..., f_m)$ تغییر پذیر است، یعنی در هر نمایش g، به‌صورت $g = h_۱ f_۱ + \cdots + h_m f_m$، یک $i \in \{۱, ..., m\}$ وجود دارد به‌طوری که $\deg(g) < \deg(h_i f_i)$. به عبارت دیگر g را نمی‌توان به‌صورت ترکیب خطی از tf_i ها نوشت بطوری که t یک یکجمله‌ای باشد و به ازای هر i، داشته باشیم $\deg(tf_i) \leq \deg(g)$.

الگوریتم MXL

فرض کنید $P = \mathbb{F}_q[x_1, ..., x_n]$ و $R = \frac{P}{\langle x_1^q - x_1, ..., x_n^q - x_n\rangle}$ در این‌صورت روش MXL برای حل دستگاه معادلات $f_1 = \cdots = f_m = 0$ در الگوریتم ۲۲ بیان شده است.

الگوریتم ۲۲ الگوریتم MXL

ورودی $F = \{f_1, ..., f_m\} \in R$

خروجی جواب دستگاه معادلات $f_1 = \cdots = f_m = 0$

۱: $d = e \leftarrow \min\{\deg(f_i)|\ f_i \in F\},\quad G \leftarrow F$

۲: G را با عملیات حذفی گاوس به‌شکل سطری پلکانی تحویل یافته تبدیل می‌کنیم.

۳: اگر G دارای چندجمله‌ای یک‌متغیره بود، معادله یک‌متغیره را حل کرده و مقدار متناظر با آن متغیر را به‌دست می‌آوریم، اگر دستگاه با این کار حل شود، جواب به‌دست آمده و الگوریتم متوقف می‌شود و در غیر این‌صورت مقدار به‌دست آمده را در چندجمله‌ای‌های دستگاه(یا F) جایگذاری کرده و به گام ۱ بازمی‌گردیم. اگر دستگاه فاقد چندجمله‌ای یک متغیره باشد به گام ۴ می‌رویم.

۴: $M \leftarrow \{f \in G|\ \deg(f) < e\}$. (در واقع چندجمله‌ای‌های M نسبت به F تغییرپذیر هستند.)

۵: اگر $M \neq \emptyset$، همه‌ی اعضای $g \in M$ را در یکجمله‌ای‌های با درجه $d - \deg(g)$ ضرب کرده و نتایج حاصل از ضرب یکجمله‌ای‌ها را جایگزین چندجمله‌ای‌های قبلی در G می‌کنیم. قرار می‌دهیم $e = \min\{\deg(g)|\ g \in M\} + 1$ و به گام ۲ بازمی‌گردیم. در غیر این‌صورت اگر $M = \emptyset$، به گام ۶ می‌رویم.

۶: همه‌ی چندجمله‌ای‌های $g \in G$ را با چندجمله‌ای‌های $x_i g$ که $i \in \{1, ..., n\}$، جایگزین می‌کنیم، سپس d را یک واحد افزایش داده، قرار می‌دهیم $e = d$ و به گام ۲ بازمی‌گردیم.

اگر گام‌های ۴ و ۵ را حذف کنیم همان الگوریتم XL به‌دست می‌آید. در ضمن در گام ۵ وقتی چندجمله‌ای‌های تغییرپذیر g را در یکجمله‌ای‌های از درجه‌ی $d - \deg(g)$ ضرب می‌کنیم، درجه‌ی دستگاه افزایش نمی‌یابد.

مثال ۱۷.۴. دستگاه معادلات زیر از حلقه‌ی $R = \frac{\mathbb{F}_2[x_1, ..., x_4]}{\langle x_1^2 - x_1, ..., x_4^2 - x_4\rangle}$ را در نظر بگیرید.

$$\begin{cases} f_1 = x_1x_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_3 + 1 = 0 \\ f_2 = x_1x_2 + x_1x_3 + x_1x_4 + x_3x_4 + x_2 + x_3 + 1 = 0 \\ f_3 = x_1x_2 + x_1x_3 + x_2x_3 + x_3x_4 + x_1 + x_4 + 1 = 0 \\ f_4 = x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + 1 = 0 \end{cases}$$

طی گام‌های زیر، با استفاده از الگوریتم MXL و در نظر گرفتن ترتیب الفبایی مدرج، دستگاه را حل می‌کنیم.

۱. قرار می‌دهیم $d = e = 2$ و $G = F = \{f_1, ..., f_4\}$

۲. پس از عملیات حذفی گاوس دستگاه به‌صورت زیر درمی‌آید.

$$\begin{cases} g_1 = x_1x_2 + x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_3 + 1 = 0 \\ g_2 = x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_1x_2 = 0 \\ g_3 = x_1x_4 + x_2x_3 + x_1 + x_2 + x_3 + x_4 = 0 \\ g_4 = x_1 + x_2 + 1 = 0 \end{cases}$$

۳. قرار می‌دهیم $M = \{g_4\}$. همان طور که مشاهده می‌شود درجه‌ی g_4 پس از عملیات حذفی گاوس کاهش یافته و نسبت به F تغییرپذیر است.

۴. قرار می‌دهیم $G = \{g_1, g_2, g_3, g_5, g_6, g_7\}$ به‌طوری که $g_5 = x_1x_2, g_6 = x_1x_3 + x_2x_3 + x_3$ و $g_7 = x_1x_4 + x_2x_4 + x_4$. در ضمن قرار می‌دهیم $e = 2$.

۵. پس از عملیات حذفی گاوس روی G، چندجمله‌ای‌های g_5, g_6, g_7 به‌صورت زیر تغییر می‌کنند.

$$\tilde{g}_5 = x_2x_3+x_2x_4+x_3x_4+x_1+x_3+1\ ,\ \tilde{g}_6 = x_3x_4+x_1+x_3+x_4+1\ ,\ \tilde{g}_7 = x_3+x_4+1.$$

۶. قرار می‌دهیم $M = \{\tilde{g}_7\}$.

۷. چون $\deg(\tilde{g}_7) = 1$ و $d = 2$ ، لذا همه یکجمله‌ای‌ها از درجه‌ی ۱ را در $\tilde{g}_7$ ضرب کرده و نتایج به‌دست آمده که عبارتند از

$$g_8 = x_1x_3 + x_1x_4\ ,\ g_9 = x_2x_3 + x_2x_4 + x_2\ ,\ g_{10} = x_3x_4,$$

را جایگزین $\tilde{g}_7$ در G می‌کنیم. در ضمن قرار می‌دهیم $e = 2$.

۸. پس از عملیات حذفی گاوس چندجمله‌ای‌های g_8 و g_9 به‌ترتیب به‌صورت $\tilde{g}_8 = x_2 + x_4$ و $\tilde{g}_9 = x_4 + 1$ در می‌آیند.

۹. با حل معادله‌ی تک‌متغیره $\tilde{g}_9 = 0$ مقدار x_4 برابر با ۱ خواهد بود. با جایگذاری $x_4 = 1$ در دستگاه گسترش پیدا کرده تا کنون، جواب برابر با $(x_1, x_2, x_3, x_4) = (0, 1, 0, 1)$، به‌دست می‌آید.

ارئه دهندگان MXL ، اندکی پس از انتشار نسخه‌ی اولیه‌ی این الگورریتم، نسخه‌ی بهبود یافته آن را نیز در [۵۰] ، منتشر کردند و آن‌را MXL2 نامیدند. آن‌ها با استفاده از شبیه‌سازی‌ها به مقایسه الگوریتم MXL2 با الگوریتم F4 که یک روش مبنی بر پایه‌ی گروبنر است پرداختند و نشان دادند که ابعاد ماتریس‌های به‌دست آمده در روش MXL2 نسبت به F4 کوچکتر است. آن‌ها همچنین

	m	n	XL	MXL	تعداد جواب‌ها
دستگاه مثال ۱۷.۴	۴	۴	۰٫۲۰	۰٫۲۳	۱
$CTC(B=۱, N=۱)$	۲۶	۱۵	۱۰٫۸۴	۱۰٫۸۱	۱

جدول ۱.۴: مقایسه الگوریتم‌های XL و MXL پیاده‌سازی شده در نرم‌افزار ApCoCoA

یکسال بعد از انتشار نسخه اولیه، در [۴۹] روشی جدید بر پایه‌ی ایده‌ی چندجمله‌ای‌های تغییرپذیر و الگوریتم XL، برای یافتن پایه گروبنر ایده‌ال‌های صفربعدی، ارائه دادند و آن‌را MXL3 نامیدند. آن‌ها با انجام آزمایش‌هایی نشان دادند که MXL3 نه تنها حافظه کمتری در مقایسه با F4 مصرف می‌کند، بلکه سریع‌تر نیز است.

الگوریتم MXL در نرم‌افزار ApCoCoA [۴۴] پیاده‌سازی شده، و با دستور `CharP.MXLSolve(.)` قابل فراخوانی است. پس از آزمودن این حل‌کننده برای حل دستگاه‌های به‌دست آمده از رمزهای قالبی CTC و SR و رمز دنباله‌ای Bivium، مشاهده شد که این الگوریتم از نظر سرعت تفاوت چندانی با الگوریتم XL ندارد. برای مثال سرعت این دو حل‌کننده در جدول ۱.۴، به‌ازای دو دستگاه متفاوت مقایسه شده است. محاسبات در رایانه با پردازنده $۱٫۶GHz$ و حافظه رم $۶GB$ و با استفاده از نرم‌افزار ApCoCoA انجام شده، زمان‌ها بر حسب ثانیه است و m و n به‌ترتیب نشان‌دهنده تعداد معادلات و تعداد مجهولات هستند.

۲.۴ حمله پایه‌ی گروبنر

پایه‌های گروبنر کاربردهای زیادی دارد که یکی از مهمترین آن‌ها مسئله‌ی عضویت در ایده‌ال است. می‌دانیم که اگر I یک ایده‌ال در $P = K[x_۱, ..., x_n]$ و G یک پایه گروبنر آن باشد، چندجمله‌ای f در I قرار دارد اگر و تنها اگر $NF_G(f) = ۰$. بنابراین الگوریتمی برای بررسی عضویت یک چندجمله‌ای در یک ایده‌ال وجود دارد.

کاربرد دیگری که می‌توان به آن اشاره کرد، مسئله‌ی مقایسه‌ی دو ایده‌ال است. اگر I و J دو ایده‌ال در P باشند، با محاسبه پایه گروبنر تحویل‌یافته آن‌ها که منحصر بفرد است، می‌توان شمول هر کدام در دیگری و یا تساوی آن‌ها را فقط با مقایسه‌ی پایه‌های آن‌ها دریافت. امّا کاربردی دیگر که برای ما بیشتر اهمیت دارد حل دستگاه معادلات چندجمله‌ای با استفاده از پایه گروبنر است که در این بخش به آن می‌پردازیم.

تعریف ۱۸.۴ (فضای آفین). فرض کنید $\mathbb{F}$ یک میدان و n یک عدد طبیعی باشد. *فضای آفین* n بعدی روی $\mathbb{F}$ عبارت است از مجموعه‌ی زیر،

$$\mathbb{F}^n := \{(a_۱, ..., a_n) \ : \ a_۱, ..., a_n \in \mathbb{F}\}.$$

فرض کنید K یک توسیع جبری میدان $\mathbb{F}$ باشد. هر چندجمله‌ای مثل $f \in \mathbb{F}[x_1, ..., x_n]$ را می‌توانیم به عنوان تابعی به‌صورت

$$f : K^n \to K$$

در نظر بگیریم که مقدار این تابع در هر نقطه‌ی $(a_1, ..., a_n) \in K^n$ با جایگذاری a_i ها به‌جای x_i ها در f محاسبه می‌شود. چون K یک توسیع $\mathbb{F}$ و ضرایب چندجمله‌ای f در $\mathbb{F} \subseteq K$ قرار دارند مقدار تابع در K خواهد بود. منظور از صفرهای f در K مقادیری است که به ازای آن‌ها تابع $f : K^n \to K$ صفر می‌شود. بنابراین یک چندجمله‌ای مثل $f \in \mathbb{F}[x_1, ..., x_n]$ رفتاری دوگانه دارد، هم می‌تواند به عنوان یک چندجمله‌ای صوری از حلقه‌ی P باشد، هم می‌توان آن را به‌صورت یک تابع در نظر گرفت. اما وقتی می‌گوییم $f = 0$ است، باید روشن کنیم که f به عنوان یک چندجمله‌ای صفر است یا یک تابع صفر است، چرا که این دو می‌توانند متفاوت باشند. برای مثال چندجمله‌ای ناصفر $f = x^2 - x \in \mathbb{F}_2[x]$ چون به ازای هر مقدار $a \in \mathbb{F}_2$ صفر می‌شود، یک تابع صفر است. بنابراین f چندجمله‌ای ناصفری است که متناظر با یک تابع صفر روی فضای آفین $\mathbb{F}_2^1$ است. چنین تفاوتی در تعبیر $f = 0$ در رمزنگاری که با میدان‌های متناهی کار می‌کنیم امری طبیعی است، ولی وقتی میدان ضرایب چندجمله‌ای نامتناهی باشد، ثابت می‌شود [۲۳، ص.۳] که f به عنوان یک چندجمله‌ای صفر است اگر و تنها اگر به عنوان یک تابع، صفر باشد. با این حال می‌دانیم هیچ میدان متناهی نمی‌تواند بسته‌ی جبری باشد.

تعریف ۱۹.۴ (واریته یا چندگونای آفین). فرض کنید K یک توسیع از میدان $\mathbb{F}$ و $f_1, ..., f_m$ چندجمله‌ای‌هایی از حلقه‌ی $\mathbb{F}[x_1, ..., x_n]$ باشند. مجموعه‌ی

$$\mathcal{Z}(f_1, ..., f_m) := \{(a_1, ..., a_n) \in K^n \mid \forall 0 \leq i \leq m \ f_i(a_1, ..., a_n) = 0\}$$

را $\mathbb{F}$- *واریته‌ی آفین* تعریف شده به‌وسیله $f_1, ..., f_m$ می‌گوییم.

توجه شود که $\mathbb{F}$ در عبارت «$\mathbb{F}$- واریته‌ی آفین» مجموعه‌ی ضرایب چندجمله‌ای را مشخص می‌کند، در حالی که صفرها یا جواب‌ها در K^n قرار دارند. در ادامه برای اختصار $\mathbb{F}$ ـ واریته‌ی آفین را واریته‌ی آفین می‌گوییم.

تعریف ۲۰.۴ (مسئله‌ی حل دستگاه معادلات چندجمله‌ای). به ازای یک مجموعه‌ی متناهی داده شده مثل $F = \{f_1, ..., f_m\}$ از چندجمله‌ای‌های حلقه‌ی $\mathbb{F}[x_1, ..., x_n]$، مسئله‌ی *حل دستگاه معادلات چندجمله‌ای* عبارت است از یافتن واریته‌ی آفین F یعنی $\mathcal{Z}(f_1, ..., f_m)$.

لم ۲۱.۴. فرض کنید $F = \{f_1, ..., f_s\}$ و $G = \{g_1, ..., g_t\}$ پایه‌های مشترک یک ایده‌ال از حلقه‌ی $\mathbb{F}[x_1, ..., x_n]$ باشند. در این‌صورت،

$$\mathcal{Z}(f_1, ..., f_s) = \mathcal{Z}(g_1, ..., g_t).$$

برهان. می‌دانیم که $\langle f_1, ..., f_s\rangle = \langle g_1, ..., g_t\rangle$ بنابراین اگر $f \in \langle f_1, ..., f_s\rangle$ آن‌گاه $f \in \langle g_1, ..., g_t\rangle$ و در نتیجه چندجمله‌ای‌های $h_1, ..., h_s$ از $P = \mathbb{F}[x_1, ..., x_n]$ موجودند بطوری که:

$$f = h_1 g_1 + \cdots + h_t g_t.$$

بنابراین به ازای هر $a \in \mathcal{Z}(g_1, ..., g_t)$ داریم، $f(a) = 0$. چون f را دلخواه انتخاب کردیم، نتیجه می‌شود $a \in \mathcal{Z}(f_1, ..., f_s)$ و $\mathcal{Z}(g_1, ..., g_t) \subseteq \mathcal{Z}(f_1, ..., f_s)$. شمول در جهت عکس را می‌توان بطور مشابه ثابت کرد. $\square$

تعریف ۲۲.۴. فرض کنید I یک ایده‌ال از حلقه $\mathbb{F}[x_1, ..., x_n]$ باشد، واریته‌ی آفین تعریف شده به‌وسیله I عبارت است از

$$\mathcal{Z}(I) := \{(a_1, ..., a_n) \in K^n \mid \forall\, f \in I: \ f(a_1, ..., a_n) = 0\}.$$

که K یک توسیع جبری $\mathbb{F}$ است.

بر اساس قضیه پایه هیلبرت ۲۲.۲، واریته آفین متناظر با یک ایده‌ال همان واریته‌ی آفین متناظر با مجموعه مولد آن است.

قضیه ۲۳.۴. اگر I یک ایده‌ال حلقه $\mathbb{F}[x_1, ..., x_n]$ و $F = \langle f_1, ..., f_m\rangle$ یک مولد داده شده برای آن باشد، آن‌گاه، $\mathcal{Z}(I) = \mathcal{Z}(f_1, ..., f_m)$.

برهان. نتیجه‌ی مستقیم قضیه‌ی پایه‌ی هیلبرت ۲۲.۲. $\square$

اکنون به دستگاه

$$\mathcal{S} = \begin{cases} f_1(x_1, ..., x_n) = 0 \\ \quad\vdots \\ f_m(x_1, ..., x_n) = 0 \end{cases}$$

باز می‌گردیم. نخستین سؤالی که در رابطه با حل دستگاه معادلات به ذهن می‌رسد، مسئله وجود جواب است. پاسخ این سؤال در حالت کلّی داده نشده ولی در حالتی که میدان K به طور جبری بسته باشد، قضیه صفرهای هیلبرت جواب کاملی ارائه می‌دهد. فرض کنید $I = \langle f_1, ..., f_m\rangle$ در این‌صورت مجموعه جواب دستگاه $\mathcal{S}$ عبارت است از $\mathcal{Z}(I)$.

قضیه ۲۴.۴ (صورت ضعیف قضیه‌ی صفرهای هیلبرت). فرض کنید K یک میدان بطور جبری بسته و I یک ایده‌ال حلقه‌ی $K[x_1, ..., x_n]$ باشد. در این‌صورت $\mathcal{Z}(I)$ تهی است اگر و تنها اگر

$K \subsetneq I$. یا بطور معادل:

$$\mathcal{Z}(I) = \emptyset \iff 1 \in I \iff I = K[x_1, ..., x_n]$$

برهان. رجوع کنید به [۲۳، ص.۱۷۷]. $\square$

بنابراین، اگر دستگاه S داده شده باشد، می‌توان با محاسبه پایه گروبنر تحویل‌یافته ایده‌ال تولید شده توسط چندجمله‌ای‌های دستگاه و بررسی این که این پایه برابر $\{1\}$ است یا نه، به وجود یا عدم وجود جواب برای دستگاه پی‌برد. لازم به ذکر است که وقتی معادلات به‌صورت قطعی از یک سامانه رمزنگاری به‌دست آمده باشند دستگاه حتماً جواب خواهد داشت، چون حداقل کلید سامانه و متن اصلی و رمزشده با آن کلید در دستگاه صدق می‌کنند، مگر این‌که در استخراج معادلات سامانه اشتباهی صورت گرفته باشد.

فرض کنید وجود جواب دستگاه S برای ما محرز شده باشد، اکنون این سؤال پیش می‌آید که آیا مجموعه‌ی جواب‌های دستگاه یا به عبارتی $\mathcal{Z}(I)$ که I ایده‌ال تولید شده توسط معادلات دستگاه است، متناهی است یا نامتناهی؟ با استفاده از لم تناهی که در ادامه می‌آید، می‌توانیم الگوریتمی برای حل این سؤال ارائه دهیم.

لم ۲۵.۴ (محک تناهی). فرض کنید S دستگاه معادلاتی از چندجمله‌ای‌های $f_1, ..., f_m$ در حلقه‌ی $\mathbb{F}[x_1, ..., x_n]$ و $I = \langle f_1, ..., f_m \rangle$ ایده‌ال تولید شده توسط چندجمله‌ای‌های این دستگاه باشد. در ضمن فرض کنید $>$ یک ترتیب یکجمله‌ای روی $\mathbb{T}^n$ باشد. در این‌صورت گزاره‌های زیر معادل‌اند:

۱. دستگاه S دارای تعداد متناهی جواب در $\overline{\mathbb{F}}$ (بستار جبری $\mathbb{F}$) است.

۲. به ازای هر $i = 1, ..., n$، داریم $I \cap \mathbb{F}[x_i] \neq \{0\}$.

۳. مجموعه‌ی $\mathbb{T}^n \setminus \mathrm{LM}_>(I)$ متناهی است.

۴. $\mathbb{F}$ ـ فضای برداری $\dfrac{\mathbb{F}[x_1, ..., x_n]}{I}$ متناهی بعد است.

۵. به ازای هر $i \in \{1, ..., n\}$ ، عدد $\alpha_i \in \mathbb{Z}_{\geq 0}$ وجود دارد بطوری که $x_i^{\alpha_i} \in \langle \mathrm{LT}_>(I) \rangle$.

برهان. رجوع کنید به [۴۶، ص.۲۴۳]. $\square$

از بین گزاره‌های معادل در لم ۲۵.۴ برقراری گزاره‌های (۳) و (۵) را می‌توان با به‌دست آوردن پایه گروبنر، برای ایده‌ال I، بررسی کرد، در نتیجه مسئله تشخیص متناهی بودن مجموعه‌ی جواب دستگاه S نیز با استفاده از الگوریتم‌های محاسبه پایه‌ی گروبنر، قابل حل است.

تعریف ۲۶.۴ (ایده‌ال صفربُعدی). [۱۱] ایده‌ال $I = \langle f_1, ..., f_s \rangle$ از حلقه‌ی $P = \mathbb{F}[x_1, ..., x_n]$ را صفر *بُعدی* گوییم، اگر در یکی از شرط‌های معادل لم ۲۵.۴ که یکی از آن‌ها متناهی بودن $\mathcal{Z}(I)$ است صدق کند.

مثال ۲۷.۴. فرض کنید $P = \mathbb{R}[x, y, z]$ و ترتیب یکجمله‌ای که به‌کار می‌بریم ترتیب الفبایی است. ایده‌ال زیر را در نظر بگیرید، $I = \langle x + y + z, xy + xz + yz, xyz - 1 \rangle$ با استفاده از نرم‌افزار سِیج پایه گروبنر آن را در زیر محاسبه می‌کنیم:

```
P.<x,y,z> = PolynomialRing(RR, order = 'lex')
I = sage.rings.ideal.Cyclic(P)
I.groebner_basis()
            [x + y + z, y^2 + y*z + z^2, z^3 - 1.0]
```

با توجه به پایه گروبنر به‌دست آمده مشاهده می‌شود که $\mathrm{LM}(I) = \langle x, y^2, z^3 \rangle$ و طبق گزاره‌ی (۵) لم ۲۵.۴ ایده‌ال I صفربعدی و مجموعه جواب‌های دستگاه متناهی خواهد بود. اگر فرض کنیم $P' = \mathbb{R}[w, x, y, z]$ آن‌گاه برای ایده‌ال

$$J = \langle x + y + z, xy + xz + yz, xyz - 1 \rangle \subseteq P'$$

به ازای هر $i \in \mathbb{Z}_{\geq 0}$ داریم $w^i \notin \langle \mathrm{LM}(J) \rangle$ و طبق گزاره (۳) لم ۲۵.۴، J صفر بعدی نخواهد بود.

در حالتی که مجموعه‌ی جواب دستگاه چندجمله‌ای S متناهی است علاقه‌مندیم تا تعداد جواب‌ها را بدانیم، قضیه‌ی زیر کرانی برای تعداد جواب‌ها ارائه می‌دهد.

قضیه ۲۸.۴. فرض کنید $f_1, ..., f_m \in P = \mathbb{F}[x_1, ..., x_n]$ چندجمله‌ای‌های دستگاه معادلات S باشند، اگر $I = \langle f_1, ..., f_m \rangle$ صفربعدی باشد آن‌گاه تعداد جواب‌های دستگاه S در $\bar{F}^n$ حداکثر برابر است با بعد فضای برداری $\frac{P}{I}$. به عبارت دیگر داریم

$$|\mathcal{Z}(I)| \leq \dim_{\mathbb{F}}(\frac{P}{I}).$$

برهان. رجوع کنید به [۴۶، ص.۲۴۵] □

بنابراین می‌توان در حالت کلّی، کران بالایی برای تعداد جواب‌های یک دستگاه به‌دست آورد.

تعریف ۲۹.۴. رادیکال ایده‌ال I از حلقه‌ی P به‌صورت زیر تعریف می‌شود

$$\sqrt{I} := \{f \in P \mid \exists m \in \mathbb{N} : f^m \in I\}.$$

[۱۱] وجه تسمیه‌ی صفر بعدی نامیدن ایده‌ال‌هایی که در شرایط لم ۲۵.۴ صدق می‌کنند، این است که در این ایده‌ال‌ها بعد فضای برداری $\frac{\mathbb{F}[x_1,...,x_n]}{I}$ متناهی است که نتیجه می‌دهد بعد کرول حلقه‌ی $\frac{\mathbb{F}[x_1,...,x_n]}{I}$ باید صفر باشد.

به‌راحتی می‌توان نشان داد که $\sqrt{I}$ خود یک ایده‌ال است. به ایده‌الی که برابر با رادیکال خودش باشد *ایده‌ال رادیکال* می‌گویند.

تعریف ۳۰.۴ (میدان کامل). میدان $\mathbb{F}$ یک *میدان کامل* است هرگاه، یا مشخصه‌اش صفر باشد و یا اگر دارای مشخصه‌ی ناصفر $p > 0$ است، هر عضو $\mathbb{F}$ دارای ریشه‌ی p ام باشد.

برای مثال به ازای هر عدد اوّل p میدان $\mathbb{F}_p$ یک میدان کامل است چرا که هر عضو آن ریشه‌ی p ام خودش است. بطور کلی هر میدان متناهی مثل $\mathbb{F}_q$ که $q = p^e$ و p عددی اوّل است یک میدان کامل است زیرا هر $x \in \mathbb{F}_q$ دارای ریشه‌ی p ام $x^{p^{e-1}}$ است. در نتیجه همه‌ی میدان‌های متناهی و همچنین همه‌ی میدان‌های نامتناهی که مشخصه صفر دارند میدان‌های کامل هستند.
قضیه‌ی زیر تعداد دقیق جواب‌های دستگاه را وقتی از میدان کامل استفاده می‌کنیم، مشخص می‌کند.

قضیه ۳۱.۴. فرض کنید I یک ایده‌ال صفر بعدی از حلقه‌ی $P = \mathbb{F}[x_1, ..., x_n]$ و $\overline{\mathbb{F}}$ بستار جبری $\mathbb{F}$ باشد. در ضمن فرض کنید $\bar{P} := \bar{\mathbb{F}}[x_1, ..., x_n]$. اگر $\mathbb{F}$ یک میدان کامل باشد آن‌گاه تعداد جواب‌های هر دستگاه معادلات چندجمله‌ای روی $\mathbb{F}$ مثل S برابر است با تعداد ایده‌ال‌های ماکزیمال $\overline{P}$ که شامل ایده‌ال $I\overline{P}$ هستند. در ضمن این عدد دقیقاً برابر است با $\dim_{\mathbb{F}}(\frac{P}{I})$.

برهان. رجوع کنید به [۲۳، ص.۲۵۳] $\square$

در رمزنگاری با میدان‌های متناهی کار می‌کنیم که بسته‌ی جبری نیستند، به همین خاطر به جواب‌هایی که خارج از میدان مورد نظر و در یک توسیع آن به‌دست می‌آیند علاقه‌ای نداریم. بنابراین اگر قیدهایی را به دستگاه چندجمله‌ای S روی میدان $\mathbb{F}$ اضافه کنیم، به‌طوری‌که به‌واسطه آن‌ها جواب‌های دستگاه در $\mathbb{F}$ محصور شوند، آن‌گاه به هدف خود می‌رسیم. تعریف زیر قیدهای مناسب را به ما معرفی می‌کند.

تعریف ۳۲.۴ (چندجمله‌ای‌های میدان). میدان متناهی $\mathbb{F}_q[x_1, ..., x_n]$ از مرتبه‌ی $q = p^n$ که p یک عدد اوّل و n یک عدد طبیعی است را در نظر بگیرید. *چندجمله‌ای‌های میدان* حلقه‌ی $\mathbb{F}_q[x_1, ..., x_n]$ عبارتند از

$$\{x_1^q - x_1, ..., x_n^q - x_n\}.$$

در ضمن به ایده‌ال تولید شده توسط مجموعه‌ی فوق یعنی،

$$\langle x_1^q - x_1, ..., x_n^q - x_n\rangle$$

ایده‌ال میدان $\mathbb{F}_q[x_1, ..., x_n]$ می‌گوییم.

نتیجه ۳۳.۴. فرض کنید S یک دستگاه معادلات چندجمله‌ای از حلقه‌ی $\mathbb{F}_q[x_1, ..., x_n]$ و $I =$ $\langle f_1, ..., f_m\rangle$ ایده‌ال تولید شده توسط چندجمله‌ای‌های دستگاه باشد. در این صورت بعد از اضافه کردن چندجمله‌ای‌های میدان به ایده‌ال I ایده‌ال

$$I' = \langle f_1, ..., f_m, x_1^q - x_1, ..., x_n^q - x_n\rangle$$

به‌دست می‌آید که طبق گزاره‌ی (۵) لم ۲۵.۴ یک ایده‌ال صفر بعدی است و لذا مجموعه‌ی جواب متناهی خواهد بود.

نتیجه ۳۴.۴. فرض کنید $I = \langle f_1, ..., f_m\rangle$ یک ایده‌ال از حلقه‌ی $\mathbb{F}_q[x_1, ..., x_n]$ باشد. اگر J ایده‌ال تولید توسط

$$\{f_1, ..., f_m\} \cup \{x_1^q - x_1, ..., x_n^q - x_n\}$$

باشد، آن‌گاه تنها تفاوت واریته‌ی $\mathcal{Z}(J)$ با واریته‌ی $\mathcal{Z}(I)$ در این است که شامل هیچ‌یک از نقاط $\bar{\mathbb{F}}^n \backslash \mathbb{F}^n$ نیست.

برهان. در هر میدان متناهی ار مرتبه‌ی q نظیر $\mathbb{F}$ معادله‌ی $x^q = x$ به ازای هر $x \in \mathbb{F}$ صادق است. بنابراین تمام نقاط $\mathbb{F}^n$ در معادلات $x_i^q - x_i = 0$ به‌ازای $i = 1, ..., n$، صدق می‌کنند. علاوه بر این هر یک از چندجمله‌ای‌های $x_i^q - x_i$ به طور کامل روی $\mathbb{F}$ تجزیه می‌شوند و در نتیجه هیچ‌یک از نقاط $\bar{\mathbb{F}}^n \backslash \mathbb{F}$ در معادلات $x_i^q - x_i$ صدق نمی‌کنند. □

تعریف ۳۵.۴. فرض کنید $\mathcal{Z} \subseteq \mathbb{F}^n$ یک $\mathbb{F}$– واریته‌ی آفین باشد. در این‌صورت $I(\mathcal{Z})$ به‌صورت زیر تعریف می‌شود:

$$I(\mathcal{Z}) := \{f \in \mathbb{F}[x_1, ..., x_n] \mid \forall (a_1, ..., a_n) \in \mathcal{Z} : \; f(a_1, ..., a_n) = 0\}.$$

لم ۳۶.۴. اگر $\mathcal{Z} \subseteq \mathbb{F}^n$ یک واریته‌ی آفین باشد آن‌گاه $I(\mathcal{Z})$ یک ایده‌ال حلقه‌ی $\mathbb{F}[x_1, ..., x_n]$ است.

برهان. به [۲۳، ص.۳۲] رجوع کنید. □

قضیه ۳۷.۴ (صورت قوی قضیه‌ی صفرهای هیلبرت). فرض کنید $\mathbb{F} = \bar{\mathbb{F}}$ و $f_1, ..., f_s \in$ $\mathbb{F}[x_1, ..., x_n]$. در این‌صورت،

$$f \in I(\mathcal{Z}(\langle f_1, ..., f_s\rangle)) \iff \exists m \in \mathbb{N} : \; f^m \in \langle f_1, ..., f_s\rangle.$$

یا به بیان دیگر برای هر ایده‌ال $I \subseteq \mathbb{F}[x_1, ..., x_n]$ داریم، $I(\mathcal{Z}(I)) = \sqrt{I}$.

برهان. به [۲۳، ص.۱۷۹] رجوع کنید. □

تعریف ۳۸.۴ (ایده‌ال حذف). ایده‌ال $I = \langle f_1, ..., f_s\rangle \subseteq \mathbb{F}[x_1, ..., x_n]$ را در نظر بگیرید. فرض کنید $l \in \{0, ..., n-1\}$ مجموعه‌ی همه‌ی چندجمله‌ای‌هایی از I که فقط از متغیرهای $x_{l+1}, ..., x_n$ تشکیل شده‌اند عبارت است از

$$I_l = I \cap \mathbb{F}[x_{l+1}, ..., x_n]$$

که به آن l امین ایده‌ال حذف I می‌گوییم.

با توجه به تعریف ایده‌ال حذف نتیجه می‌شود که I_l ایده‌الی از حلقه‌ی $\mathbb{F}[x_{l+1}, ..., x_n]$ است که به آن ایده‌ال حذف I نسبت به متغیرهای $\{x_1, ..., x_l\}$ هم می‌گویند زیرا این ایده‌ال از حذف تمام چندجمله‌ای‌های I که شامل متغیرهای $\{x_1, ..., x_l\}$ هستند، به‌دست می‌آید.

گزاره ۳۹.۴ (لم سایدنبرگ). فرض کنید $\mathbb{F}$ یک میدان و $P = \mathbb{F}[x_1, ..., x_n]$ و I یک ایده‌ال صفر بعدی P باشد. فرض کنید به ازای هر $i \in \{1, ..., n\}$ چندجمله‌ای ناصفر $g_i \in I \cap \mathbb{F}[x_i]$ وجود داشته باشد بطوری که $\gcd(g_i, g_i') = 1$ باشد. (g_i' مشتق g_i نسبت به x_i است) در این‌صورت I یک ایده‌ال رادیکال است.

برهان. به [۲۳، ص.۲۵۱] مراجعه کنید. □

فرض کنید $\mathbb{F}_q$ یک میدان متناهی باشد که $q = p^n$ و p یک عدد اوّل است. فرض کنید $I = \langle f_1, ..., f_m \in\rangle \subseteq \mathbb{F}_q[x_1, ..., x_n]$، در این‌صورت دستگاه

$$\mathcal{S} = \begin{cases} f_1(x_1, ..., x_n) = 0 \\ \vdots \\ f_m(x_1, ..., x_n) = 0 \end{cases}$$

ممکن است جواب‌هایی خارج از $\mathbb{F}^n$ یعنی جواب‌هایی در $\bar{\mathbb{F}}^n \backslash \mathbb{F}$ داشته باشد که $\bar{\mathbb{F}}$ بستار جبری $\mathbb{F}$ است. طبق لم سایدنبرگ ۳۹.۴ با ضمیمه کردن معادلات

$$\{x_i^q - x_i : 1 \leq i \leq n\}$$

به دستگاه $\mathcal{S}$ ، ایده‌ال تولید شده توسط معادلات جدید که آنرا با J نمایش می‌دهیم، یک ایده‌ال رادیکال با واریته‌ی $\mathcal{Z}(J) = \mathcal{Z}(I) \cap \mathbb{F}^n$ است.

اگر I ایده‌ال تولید شده توسط چندجمله‌ای‌های دستگاه $\mathcal{S}$ باشد، می‌دانیم که این ایده‌ال پایه‌های متفاوتی دارد، سؤال این است که آیا این ایده‌ال پایه‌ای دارد که با استفاده از آن بتوان به‌راحتی جواب‌های دستگاه معادلات چندجمله‌ای $\mathcal{S}$ را به‌دست آورد؟ طبق قضیه‌ی زیر پاسخ این سؤال مثبت است.

قضیه ۴۰.۴ (**قضیه‌ی حذف**). فرض کنید $I \subseteq \mathbb{F}[x_1, ..., x_n]$ یک ایده‌ال و G پایه‌ی گروبنر آن نسبت به ترتیب الفبایی و با فرض $x_1 > x_2 > \cdot > x_n$ باشد. در این‌صورت به ازای هر $0 \leq l < n$ ، مجموعه‌ی

$$G_l = G \cap \mathbb{F}[x_{l+1}, ..., x_n]$$

یک پایه‌ی گروبنر برای ایده‌ال I_l است.

برهان. به [۲۳، ص.۱۲۳] رجوع کنید. □

نتیجه ۴۱.۴. طبق قضیه‌ی حذف پایه‌ی گروبنر یک شکل مثلثی دارد، به این معنی اگر در یک‌جهت مناسب اعضای پایه را انتخاب کنیم هر بار یکی از تعداد متغیرهای ظاهر شده کم می‌شود. برای روشن شدن موضوع به‌یاد بیاورید که برای حل دستگاه معادلات خطّی به روش حذفی گاوس، ابتدا آن را به‌صورت مثلثی تبدیل می‌کردیم و سپس از آخرین معادله که فقط شامل یک مجهول بود شروع به حل می‌کردیم و با به‌دست آوردن جواب مجهول مورد نظر و جایگذاری آن در معادله‌ی قبلی مجهولات دیگر را پیدا می‌کردیم. مثال بعدی این مطلب را بهتر نشان می‌دهد.

نکته ۴۲.۴. همان طور که قبلاً ذکر شد، با استفاده از ترتیب الفبایی است که پایه گروبنر ایده‌ال، شکل مثلثی به خود می‌گیرد و امکان استخراج جواب‌های دستگاه فراهم می‌شود. از طرفی می‌دانیم که محاسبه پایه گروبنر با استفاده از ترتیب الفبایی مدرج معکوس سریع‌تر صورت می‌گیرد. بنابراین روش بهینه این است که ابتدا پایه گروبنر را با ترتیب الفبایی مدرج معکوس محاسبه کنیم، سپس با استفاده از الگوریتم‌های تبدیل پایه‌ی گروبنر تحت یک ترتیب به پایه‌ی گروبنر تحت ترتیب دیگر نظیر [۳۳] FGLM، پایه گروبنر تحت ترتیب الفبایی را به‌دست آوریم.

مثال ۴۳.۴ (**مثلثی سازی دستگاه چندجمله‌ای با پایه‌ی گروبنر**). فرض کنید $P = \mathbb{F}_{127}[x, y, z]$ و ترتیب یکجمله‌ای الفبایی را بکار ببریم. ایده‌ال زیر را در نظر بگیرید:

$$I = \langle x + y + z, xy + xz + yz, xyz - 1 \rangle.$$

ابتدا معادلات میدان $\mathbb{F}_{127}$ را به آن اضافه کرده، سپس پایه‌ی گروبنر تحویل‌یافته‌ی آن را محاسبه می‌کنیم.

```
P.<x,y,z> = PolynomialRing(GF(127), order = 'lex')
I = sage.rings.ideal.Cyclic(P); print I
 Ideal (x + y + z, x*y + x*z + y*z, x*y*z - 1) of Multivariate Polynomial
Ring in x, y, z over Finite Field of size 127
J = I + sage.rings.ideal.FieldIdeal(P)
g1, g2, g3 = J.groebner_basis()
g1, g2, g3
              (x + y + z, y^2 + y*z + z^2, z^3 - 1)
```

همان طور که بر اساس قضیه حذف ۴۰.۴ پیش‌بینی می‌کردیم، و در زیر مشاهده می‌شود، پایه گروبنر نسبت به ترتیب الفبایی شکل مثلثی دارد.

$$\left.\begin{array}{l}\left.\begin{array}{l}\left.\{z^{3}+126,\qquad\right\}\ \text{فقط بر حسب } z\\ y^{2}+yz+z^{2},\end{array}\right\}\ \text{فقط بر حسب } y,z\\ \quad x+y+z\}.\end{array}\right\}\ \text{همه‌ی متغیرها}$$

اکنون به استخراج جواب‌ها می‌پردازیم.

```
factor(g3)
                    (z - 19) * (z - 1) * (z + 20)
factor(g2(x,y,-108))
                         (y - 1) * (y + 20)
factor(g1(x,-126,-108))
                               x + 20
all(f(-20, 1, 19) == 0 for f in I.gens())
```

مجموعه‌ی جواب‌ها یا واریته‌ی آفین $\mathcal{Z}(J)$ با استفاده از تابع `variety` از کلاس `ideal` قابل محاسبه است.

```
J.variety()
[{y: 19, z: 1, x: 107}, {y: 107, z: 1, x: 19}, {y: 1, z: 19, x: 107},
{y: 107, z: 19, x: 1}, {y: 1, z: 107, x: 19}, {y: 19, z: 107, x: 1}]
```

همان‌طور که مشاهده می‌شود، پایه گروبنر ابزار خوبی برای حل دستگاه معادلات چندجمله‌ای است. در رمزنگاری از الگوریتم‌های بهینه‌شده محاسبه پایه‌ی گروبنر نظیر F4 و F5

الگوریتم‌های محاسبه پایه گروبنر به‌خصوص الگوریتم F4 در نرم‌افزارهای جبری مانند Maple [۴۷]، Singular [۲۶] و ApCoCoA [۴۴] پیاده‌سازی شده است. در جدول ۲.۴، مقایسه‌ای بین الگوریتم‌های پیاده‌سازی شده در این‌نرم‌افزارها صورت گرفته است. زمان‌های گزارش شده بر حسب ثانیه و m و n به‌ترتیب نشان‌دهنده تعداد معادلات و تعداد مجهولات هستند. دستگاه‌های معادلات به‌دست آمده مربوط به الگوریتم رمزنگاری قالبی CTC هستند که به‌ازای یک زوج متن اصلی و رمزشده معلوم و با استفاده از نرم‌افزار Sage [۶۲] استخراج گشته‌اند. در ضمن محاسبات روی یک رایانه با پردازنده $۱/۶GHz$ و حافظه رم $۶GB$ انجام شده است.

در جدول ۲.۴ برای محاسبه جواب دستگاه با استفاده از پایه‌گروبنر در نرم‌افزار Maple، از کتاب‌خانه FGb [۳۴] که آن‌را مبدع الگوریتم‌های F4 و F5 یعنی فوجِر پیاده‌سازی نموده، استفاده شده است. محاسبه پایه گروبنر در ApCoCoA نیز با استفاده از دستور `GBasis5(.)` صورت گرفته که بعد از هر بار فراخوانی این‌دستور ApCoCoA از کتاب‌خانه `CoCoALib-9945.0` برای محاسبه پایه گروبنر استفاده کرده است. در نرم‌افزار Sage نیز برای محاسبه پایه گروبنر از Singular

$CTC(B,N)$	m	n	Sage(Singular)	Maple(FGb)	ApCoCoA(CoColib)	تعداد جوابها
$CTC(۱,۱)$	۲۶	۱۵	۰٫۰۲۰	۰٫۰۳۱	۰٫۰۱۵	۱
$CTC(۲,۲)$	۹۸	۵۴	۰٫۰۶۴	۰٫۰۷۸	۰٫۰۷۰	۱
$CTC(۲,۳)$	۱۴۴	۷۸	۰٫۱۲۸	۰٫۲۶۶	۰٫۳۷۵	۱
$CTC(۳,۲)$	۱۴۷	۸۱	۰٫۲۲۴	۰٫۳۴۳	۰٫۵۳۹	۱
$CTC(۴,۲)$	۱۹۶	۱۰۸	۰٫۳۵۲	۱٫۶۹	۳٫۵۶۲	۱
$CTC(۳,۳)$	۲۱۶	۱۱۷	۰٫۹۰۴	۴٫۶۴	۹٫۲۰۸	۱
$CTC(۳,۴)$	۲۸۵	۱۵۳	۴٫۲۸	۴۹٫۷۳	۵۹۸٫۸۲۸	۱
$CTC(۴,۳)$	۲۸۸	۱۵۶	۲٫۶۱	۴۷٫۸۹	۴۲۰٫۵۹۵	۱

جدول ۲.۴: مقایسه الگوریتم‌های محاسبه پایه گروبنر در نرم‌افزارهای Maple، Sage و ApCoCoA

استفاده شده است. همان‌طور که مشاهده می‌شود سرعت محاسبه پایه گروبنر در Singular بیشتر از Maple و در Maple بیشتر از ApCoCoA است. نتیجه دیگری که می‌توان گرفت این است که زمان حل دستگاه به‌دست آمده از CTC از بین دو پارامتر B و N به تغییرات N یعنی تعداد دورها حساسیت بیشتری نشان داده است، به‌طوری که زمان حل با استفاده از پایه گروبنر وقتی تعداد دورها را افزایش می‌دهیم، بیشتر از زمانی است که با ثابت نگاه داشتن تعداد دورها، فقط تعداد جعبه‌های جانشینی را افزایش می‌دهیم.

۳.۴ حمله پایه مرزی

طبق قضیه ۶۵.۲، از فصل ۲، می‌دانیم که اگر ایده‌ال ترتیبی، از یک ترتیب یکجمله‌ای به‌دست آمده باشد، آن‌گاه پایه مرزی محاسبه شده نسبت به آن ایده‌ال ترتیبی، شامل پایه گروبنر تحویل‌یافته نیز خواهد بود. بنابراین با محاسبه پایه مرزی نیز می‌توانیم دستگاه چندجمله‌ای را حل کنیم. در این بخش الگوریتم بهبودیافته پایه مرزی را شرح می‌دهیم که گزینه مناسب‌تری نسبت به نسخه عادی آن برای حمله به سامانه‌های رمزنگاری است.

قضیه ۴۴.۴. فرض کنید $I = \langle f_1, ..., f_m \rangle$ یک ایده‌ال صفربعدی از حلقه‌ی $P = K[x_1, ..., x_n]$ باشد. در این‌صورت الگوریتم ۲۳، یک ایده‌ال ترتیبی $\mathcal{O}$ مجاز برای ایده‌ال I یافته، و $\mathcal{O}$- پایه‌ی مرزی I را محاسبه می‌کند.

برهان. رجوع کنید به [۴۵]. □

مثال ۴۵.۴. ایده‌ال $I = \langle f_1, ..., f_6 \rangle$ از حلقه‌ی $\mathbb{F}_2[x, y, z]$ را که

$$f_1 = xy+xz+1, f_2 = xz+yz+z, f_3 = xy+xz+y+1, f_4 = x^2+x, f_5 = y^2+y, f_6 = z^2+z$$

الگوریتم ۲۳ الگوریتم بهبودیافته پایه مرزی

ورودی ایده‌ال صفر بعدی $I = \langle f_1, ..., f_m\rangle$ از حلقه‌ی $P = K[x_1, ..., x_n]$.
خروجی ایده‌ال ترتیبی پذیرفتنی $\mathcal{O}$ و یک $\mathcal{O}$ ـ پایه‌ی مرزی برای ایده‌ال I.

۱: فرض کنید U ایده‌ال ترتیبی تولید شده به‌وسیله‌ی $\bigcup_{i=1}^{m} \mathrm{Supp}(f_i)$ باشد. یک ترتیب یکجمله‌ای سازگار با درجه مثل σ انتخاب می‌کنیم.
۲: مجموعه‌ی $\{f_1, ..., f_m\}$ را به یک پایه برای فضای برداری $\langle f_1, ..., f_m\rangle_K$ تقلیل می‌دهیم بطوری که همه یکجمله‌ای‌های پیشرو اعضای آن متمایز باشند و آن را با V نشان می‌دهیم.
۳: پایه V برای $\langle F\rangle_K$ را به پایه $V \cup W'$ برای فضای برداری F^+ گسترش می‌دهیم بطوری که اعضای $V \cup W'$ دارای یکجمله‌ای‌های پیشرو متمایز باشند.
۴: $W = \{w \in W' |\ \mathrm{LM}_\sigma(w) \in U\}$.
۵: اگر $\bigcup_{w\in W} \mathrm{Supp}(w) \nsubseteq U$، U را با یکجمله‌ای‌ها موجود در ایده‌ال ترتیبی تولید شده توسط $\bigcup_{w\in W} \mathrm{Supp}(w)$ گسترش داده و به گام ۴ می‌رویم.
۶: اگر $W \neq \emptyset$ ، آن‌گاه W را به V ضمیمه کرده و F^+ را جایگزین F می‌کنیم. با گام ۳ ادامه می‌دهیم.
۷: قرار می‌دهیم $\mathcal{O} = U \setminus \mathrm{LM}_\sigma(V)$. اگر $\partial\mathcal{O} \nsubseteq U$ ، آن‌گاه U^+ را جایگزین U کرده و با گام ۳ ادامه می‌دهیم.
۸: الگوریتم تحویل نهایی ۷، فراخوانی می‌شود که خروجی آن را با $G = \{g_1, ..., g_\nu\}$ نشان می‌دهیم. $(G, \mathcal{O})$ خروجی نهایی هستند و الگوریتم متوقف می‌شود.

را در نظر بگیرید. با استفاده از الگوریتم پایه مرزی بهبود یافته ۲۳ صفرهای ایده‌ال I را می‌یابیم (شماره‌ها به ترتیب نیستند و نشان‌دهنده گامی از الگوریتم هستند که در آن مرحله اعمال می‌شود.)

(۱) $U = \mathbb{T}^3_{\leq 2}$ و $\sigma = degrevlex$.

(۲) $V = \{f_1, f_2, \tilde{f}_3, f_4, f_5, f_6\}$ که $\tilde{f}_3 = y$ پایه‌ای برای فضای برداری $\langle F\rangle_K$ است.

(۳) $V \cup W' = V \cup \{x+1, z+1, yz, x^3+1, x^2y, xy^2, y^3, x^2z+1, xyz, y^2z, xz^2+1, yz^2, z^3+1\}$ پایه‌ای برای $V \cup xV \cup yV \cup zV$ است.

(۴) $W = \{x+1, z+1, yz\}$.

(۶) قرار می‌دهیم $V = \{f_1, f_2, \tilde{f}_3, f_4, f_5, f_6, f_7, f_8, f_9\}$ که $f_7 = x+1, f_8 = z+1$ و $f_9 = yz$.

(۳) $V \cup W' = V \cup \{x^3+1, x^2y, xy^2, y^3, x^2z+1, xyz, y^2z, xz^2+1, yz^2, z^3+1\}$ پایه‌ای برای $V \cup xV \cup yV \cup zV$ است.

(۴) داریم $W = \emptyset$

(۷) قرار می‌دهیم $\mathcal{O} = U \setminus \mathrm{LM}_\sigma(V) = \{1\}$ که در نتیجه $\partial\mathcal{O} = \{x, y, z\} \subseteq U$.

	m	n	BB.BBasis(.)	GBasis5(.)	تعداد جواب
$CTC(۱,۱)$	۲۶	۱۵	۰٫۰۳۱	۰٫۰۱۵	۱
$CTC(۲,۲)$	۹۸	۵۴	۲۱٫۹۵۳	۰٫۰۷۰	۱
$SR(۱,۱,۱,۴)$	۳۶	۲۰	۲٫۷۸	۰٫۰۱۵	۲

جدول ۳.۴: مقایسه زمان حل دستگاه‌های چندجمله‌ای با استفاده از روش پایه گروبنر و روش پایه مرزی در نرم‌افزار ApCoCoA

(۸) با فراخوانی الگوریتم تحویل نهایی ۷، $\mathcal{O}$- پایه‌ی مرزی $G = \{x+۱, y, z+۱\}$ برای ایده‌ال I به‌دست می‌آید. با توجه به پایه‌ی مرزی به‌دست آمده $(۱,۰,۱)$ تنها صفر ایده‌ال و در واقع تنها جواب معادله‌ی $f_۱ = \cdots = f_۶ = ۰$ است.

یکی از مزیت‌های الگوریتم پایه‌ی مرزی نسبت به الگوریتم‌های پایه‌ی گروبنر این است که فضای محاسباتی الگوریتم (U در الگوریتم ۲۳) تا حد ممکن کوچک در نظر گرفته می‌شود و تا وقتی نیاز نباشد اندازه‌ی آن افزایش نمی‌یابد، این موضوع سبب می‌شود الگوریتم کنترل خوبی روی حافظه و زمان مورد نیاز برای اجرا داشته باشد. این در حالی است که الگوریتم‌های محاسبه پایه گروبنر کنترلی روی پایه در حال گسترش ندارند و این پایه می‌تواند به سرعت رشد کند که سبب مصرف بیشتر حافظه و افزایش زمان اجرا خواهد بود. البته باید اقرار کرد که تا کنون هیچ حمله‌ای با استفاده از پایه‌های مرزی گزارش نشده و رمزنگارها بیشتر از پایه‌های گروبنر برای حل دستگاه‌ها استفاده کرده‌اند و به همین جهت مطالعات بسیار کمتری در زمینه پایه‌های مرزی صورت گرفته است. اما دلایل فوق نشان می‌دهد پایه‌های مرزی شایسته تحقیقات بیشتری است و می‌تواند گزینه دیگری برای حمله به سامانه‌های رمزنگاری باشد. الگوریتم پایه مرزی تا کنون فقط در نرم‌افزار CoCoA [۹] پیاده سازی شده است. البته نسخه پیاده‌سازی شده در این نرم‌افزار هم نسخه‌ای عمومی برای محاسبه پایه مرزی ایده‌ال‌های صفربعدی است و برای حمله‌های جبری بهینه‌سازی نشده است.

پس از آزمایش الگوریتم پیاده‌سازی‌شده در ApCoCoA، مشاهده شد که سرعت آن از الگوریتم‌های پایه گروبنر کم‌تر است. در جدول ۳.۴، زمان اجرای الگوریتم‌های پایه‌مرزی و پایه گروبنر برای حل دستگاه‌ها به‌دست آمده از CTC و SR، به‌ازای یک زوج متن اصلی و رمزشده معلوم، آمده است. برای محاسبه پاسخ به روش پایه گروبنر از دستور `GBasis5(.)` و برای محاسبه پایه مرزی از دستور `BB.BBasis(.)` در نرم‌افزار ApCoCoA استفاده شده است. زمان‌ها بر حسب ثانیه و مربوط به اجرای الگوریتم روی رایانه با پردازنده $۱٫۶GHz$ و حافظه رم $۶GB$ است. مانند قبل m و n به‌ترتیب نشان‌دهنده تعداد معادلات و تعداد متغیرها هستند.

۴.۴ حمله برنامه‌ریزی عدد صحیح

در این بخش توجه خود را به حمله جبری مبتنی بر دستگاه‌های معادلات روی $\mathbb{F}_2$ معطوف می‌کنیم. گرچه می‌توان نتایج به‌دست آمده را به میدان‌های متناهی دیگر هم تعمیم داد، ولی هدف ما تمرکز روی مفاهیم اصلی و کلیدی است. ما در این بخش قصد نداریم الگوریتم‌های حل مسئله *برنامه ریزی خطی عدد صحیح* را بررسی کنیم، بلکه هدف اصلی ما معرفی الگوریتم‌هایی برای تبدیل مسئله حل دستگاه معادلات چندجمله‌ای روی میدان $\mathbb{F}_2$ به مسئله برنامه‌ریزی خطی عدد صحیح است تا به این ترتیب، از حل کننده‌های مسائل برنامه‌ریزی خطی برای حل دستگاه معادلات استفاده کنیم.

یک مسئله برنامه‌ریزی خطی عدد صحیح آمیخته (MILP) مسئله‌ای به‌صورت

$$\begin{aligned} \min\ & z = cx + dy \\ s.t\ & \mathcal{A}x + \mathcal{B}y \leq b \\ & x \geq 0,\ x \in \mathbb{Z}^n_{\geq 0} \\ & y \geq 0,\ y \in \mathbb{R}^p_{\geq 0} \end{aligned}$$

است که در آن $c \in \mathbb{Q}^n, d \in \mathbb{Q}^p$ و $\mathcal{A} \in \mathbb{Q}^{m\times n}, \mathcal{B} \in \mathbb{Q}^{m\times p}$ و $b \in \mathbb{Q}^m$. مجموعه

$$S := \{(x, y) \in \mathbb{Z}^n_{\geq} \times \mathbb{R}^p_{\geq 0} \mid \mathcal{A}x + \mathcal{B}y \leq b\}$$

را *فضای شدنی* یا موجه و به هر یک از نقاط آن نقطه شدنی یا موجه می‌گویند. مسئله MILP شدنی است هرگاه $S \neq \emptyset$ و ناشدنی است هرگاه $S = \emptyset$. تابع $z = cx + dy$ را *تابع هدف* گوییم و هدف مسئله کمینه و یا بیشینه کردن مقدار تابع هدف است.

یک مسئله MILP یا دارای *جواب بهینه* است، یا بی‌کران و یا ناشدنی است. اگر به ازای هر $w \in \mathbb{R}$ یک $(x', y') \in S$ وجود داشته باشد بطوری که $cx' + dy' < w$ ، آن‌گاه مسئله MILP را بی‌کران گوییم. همچنین مسئله دارای جواب بهینه $(x'', y'') \in S$ است هرگاه (x'', y'') تابع هدف را بهینه (بنا به نوع مسئله، کمینه یا بیشینه) کند. برای مثال در حالت کمینه سازی، $(x'', y'') \in S$ ، یک جواب بهینه است هرگاه به ازای هر $(x, y) \in S$ ، داشته باشیم، $cx'' + dy'' \leq cx + dy$.

حالت‌های خاصی از مسئله‌ی MILP، *مسئله‌ی برنامه‌ریزی خطی* (LP) و مسئله‌ی برنامه‌ریزی عدد صحیح محض (یا به اختصار *برنامه ریزی صحیح*) هستند. اگر در مسئله MILP، $c = 0$، یعنی همه متغیرها پیوسته باشند، مسئله LP و اگر $d = 0$، یعنی همه متغیرها صحیح باشند مسئله IP به‌دست می‌آید.

روش حل مسئله برنامه‌ریزی عدد صحیح

روش‌های مختلفی برای حل مسئله IP وجود دارد ولی از آنجایی‌که هدف ما پرداختن به روش‌های حل این مسائل نیست تنها یک روش را به اختصار بیان می‌کنیم. این روش *انشعاب و تحدید* نام دارد که الگوریتم آن در زیر آمده است.

الگوریتم ۲۴ الگوریتم انشعاب و تحدید برای حل مسئله‌ی IP

ورودی مسئله‌ی IP استاندارد.
خروجی پاسخ مسئله IP در صورت وجود.

۱: با برداشتن شرط صحیح بودن متغیرهای مسئله IP، مسئله LP متناظر با آن را به‌دست آورده و با P_0 نمایش می‌دهیم.

۲: مسئله برنامه ریزی خطی P_0 را با حل کننده‌های LP حل می‌کنیم و اگر همه متغیرها مقدار صحیح گرفتند توقف می‌کنیم. در غیر این‌صورت به گام ۳ می‌رویم.

۳: قرار می‌دهیم $Z_L = +\infty$.

۴: **انشعاب**: یکی از متغیرها مثل $x_j = x_j^*$، که مقدار به‌دست آمده برای آن عدد صحیح نیست را انتخاب کرده، و دو مسئله فرعی P_1 و P_2 را به‌صورت زیر ایجاد می‌کنیم:

1. مسئله P_1: همان مسئله P_0 است، با این تفاوت که قید $x_j \leq [x_j^*]$ به آن اضافه شده است.
2. مسئله‌ی P_2: همان مسئله P_0 است، با این تفاوت که قید $x_j \geq [x_j^*] + 1$ به آن اضافه شده است.

۵: **تحدید**: مسائل به‌دست آمده P_1 و P_2 را حل کرده و بهترین مقداری که برای تابع هدف به ازای یک جواب موجه عدد صحیح برای ۲ مسئله P_1 و P_2 به‌دست آمده را جایگزین Z_L می‌کنیم.

۶: شاخه‌های فرعی در یکی از ۳ وضعیت زیر متوقف می‌شوند:

1. تمام متغیرها مقداری صحیح بگیرند.
2. مسئله‌ی فرعی ناشی از انشعاب نشدنی باشد.
3. مقدار تابع هدف از مقدار Z_L بزرگتر شود.

۷: اگر تمام انشعاب‌ها به انتها برسند الگوریتم متوقف شده و مسئله‌ای که تابع هدف آن مساوی Z_L است انتخاب می‌کنیم، جواب این مسئله همان جواب بهینه است. در غیر این‌صورت به گام ۴ می‌رویم.

همان‌طور که مشاهده می‌شود، این الگوریتم مسئله برنامه‌ریزی عدد صحیح را با استفاده از حل کننده‌های برنامه‌ریزی خطی حل می‌کند. به این ترتیب اگر بتوانیم به نحوی مسئله حل دستگاه معادلات چندجمله‌ای را به مسئله برنامه‌ریزی عدد صحیح تبدیل کنیم، می‌توانیم از حل‌کننده‌های قدرتمند برنامه‌ریزی خطی نیز در حمله‌های جبری استفاده کنیم.

۱.۴.۴ تبدیل دستگاه معادلات چندجمله‌ای به مسئله برنامه‌ریزی خطی

فرض کنید $P = \mathbb{F}_2[x_1, ..., x_n]$ و $f_1, ..., f_m \in P$ چندجمله‌ای‌های ناصفر باشند. هدف ما حل دستگاه معادلات

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_m(x_1, \ldots, x_n) &= 0. \end{aligned}$$

است. برای رسیدن به این هدف الگوریتمی ارائه می‌دهیم که این مسئله را به مسئله‌ی برنامه‌ریزی عدد صحیح تبدیل کرده و سپس آن را با حل‌کننده‌های مسئله برنامه‌ریزی عدد صحیح حل می‌کند. برای بیان این الگوریتم ابتدا چند مفهوم مقدماتی را تعریف می‌کنیم.

تعریف ۴۶.۴. فرض کنید $n \geq 1$.

۱. یکجمله‌ای $t \in \mathbb{T}^n$ به‌صورت $t = x_{i_1} \cdots x_{i_s}$ ، که $1 \leq i_1 < i_2 < \cdots < i_s \leq n$، را *مربع آزاد* گوییم. به همین ترتیب هر چندجمله‌ای که همه یکجمله‌ای‌هایش مربع آزاد باشند را مربع آزاد گوییم.

۲. برای یک یکجمله‌ای مربع آزاد مثل $t_j = x_{i_1} \cdots x_{i_s} \in \mathbb{T}^n$، که $1 \leq i_1 < i_2 < \cdots < i_s \leq n$، مجموعه N_j را به‌صورت $N_j = \{i_1, ..., i_s\}$ تعریف می‌کنیم.

در ادامه این بخش متغیرهای حقیقی (یا صحیح) را با $X_1, ..., X_n$ و متغیرهای میدان $\mathbb{F}_2$ را با $x_1, ..., x_n$ نمایش می‌دهیم.

تعریف ۴۷.۴ (تبدیل استاندارد). نگاشت $\varphi : \mathbb{F}_2 = \{\bar{0}, \bar{1}\} \to \{0, 1\} \subseteq \mathbb{R}$ را که $\varphi(\bar{0}) = 0$ و $\varphi(\bar{1}) = 1$ را تبدیل استاندارد گوییم. تبدیل φ را می‌توانیم به‌صورت زیر توسیع دهیم.

$$\begin{aligned} \Phi : \mathbb{F}_2[x_1, ..., x_n] &\to \mathbb{R}[X_1, ..., X_n] \\ c &\mapsto \varphi(c) \\ x_i &\mapsto X_i, \end{aligned}$$

به‌طوری‌که $c \in \mathbb{F}_2$. با توجه به تعریف فوق، به ازای هر $f \in \mathbb{F}_2[x_1, ..., x_n]$ ، $\Phi(f)$ را *نمایش استاندارد* f گوییم.

فرض کنید $f_1, ..., f_m \in \mathbb{F}_2[x_1, ..., x_n]$ به ترتیب دارای نمایش‌های استاندارد $F_1, ..., F_m \in \mathbb{Z}[X_1, ..., X_n]$ باشند. در این‌صورت می‌توانیم مسئله حل دستگاه $f_1 = \cdots = f_m = 0$ را به مسئله

یافتن نقاط $(a_1, \ldots, a_n) \in \{0,1\}^n$ که در شرایط

$$\begin{aligned} F_1(a_1, \ldots, a_n) &\overset{2}{\equiv} 0 \\ &\vdots \\ F_m(a_1, \ldots, a_n) &\overset{2}{\equiv} 0 \end{aligned} \tag{۱۱.۴}$$

صدق می‌کنند، تبدیل کنیم. بنابراین باید به‌دنبال جواب‌های $(a_1, \ldots, a_n) \in \mathbb{Z}^n_{\geq 0}$ از معادله‌ی ۱۱.۴، باشیم به‌طوری‌که به‌ازای هر $i \in \{1, \ldots, n\}$ داشته باشیم $0 \leq a_i \leq 1$. اکنون اگر بتوانیم به نحوی معادلات همنهشتی ۱۱.۴، را به‌صورت مجموعه‌ای از برابری‌ها و نابرابری‌ها بیان کنیم، آن‌گاه حل دستگاه همنهشتی با قیدهای مورد نظر به مسئله IP تبدیل می‌شود و می‌توانیم از حل‌کننده‌های IP برای حل مسئله استفاده کنیم. الگوریتم ۲۵ که در ادامه می‌آید، قادر است تمام مراحل فوق را انجام داده و جواب مسئله را بیابد.

گزاره ۴۸.۴. فرض کنید $P = \mathbb{F}_2[x_1, \ldots, x_n]$ و $f_1, \ldots, f_m \in P$. در این‌صورت الگوریتم ۲۵ همه نقاط $(a_1, \ldots, a_n) \in \{0,1\}^n$ را که کلاس مانده آن‌ها در $\mathbb{F}_2^n$، صفرهای ایده‌ال رادیکال صفربعدی

$$I = \langle f_1, \ldots, f_m, x_1^2 + x_1, \ldots, x_n^2 + x_n \rangle$$

هستند را محاسبه می‌کند.

برهان. چون به ازای هر $\alpha = 1, \ldots, n$، عدد صحیح نامنفی a_α باید در نابرابری I'_α صدق کند، لذا داریم $a_\alpha \in \{0,1\}$. به‌طور مشابه، به‌ازای هر j، نامساوی $I_{j\alpha}$ نتیجه می‌دهد که $a_{n+j} \leq a_\alpha$ ولذا $a_{n+j} \in \{0,1\}$. علاوه بر این، اگر $t_j = \prod_{\alpha \in N_j} x_\alpha \in S$ و به‌ازای یک $\alpha \in N_j$، عدد a_α برابر صفر باشد، آن‌گاه نابرابری $I_{j\alpha}$، نتیجه می‌دهد که $a_{n+j} = 0$. از طرف دیگر، اگر به ازای هر $\alpha \in N_j$ داشته باشیم $a_\alpha = 1$، آن‌گاه نامساوی I_{n+j} نتیجه می‌دهد که $a_{n+j} \geq 1$. بنابراین a_{n+j} برابر است با مقدار t_j در نقطه‌ی $(a_1, \ldots, a_n)$، یعنی داریم $a_{n+j} = \prod_{\alpha \in N_j} a_\alpha$.

برقراری شرط $F_i : \sum_j X_{n+j} + L_i - 2K_i = 0$ باعث می‌شود مقدار $F_i = \Phi(f)$ در نقطه‌ی $(a_1, \ldots, a_n)$ زوج باشد، به عبارت دیگر داریم $F_i(a_1, \ldots, a_n) = 2K_i$. در ضمن شرط I_i تأثیری در مقدار F_i نداشته و فقط کران بالایی، برابر با تعداد یکجمله‌ای‌های f_i، برای K_i تعیین می‌کند. بنابراین جواب مسئله‌ی IP به‌دست آمده در الگوریتم ۲۵، به طور یکتا متناظر با جواب $(a_1, \ldots, a_n) \in \{0,1\}^n$ از دستگاه معادلات $F_1 = \cdots = F_m = 0$ در $\mathbb{Z}[x_1, \ldots, x_n]$ است. □

نکته ۴۹.۴. الگوریتم ۲۵ را در نظر بگیرید. اگر به ازای یک تابع هدف دلخواه، بتوانیم یک جواب شدنی (نه لزوماً بهینه) برای مسئله‌ی IP متناظر با دستگاه معادلات چندجمله‌ای پیدا کنیم، آن‌گاه این‌جواب، در تمام قیدها صدق می‌کند و در نتیجه جوابی برای دستگاه اصلی است. بنابراین مهم نیست جوابی که برای مسئله‌ی IP پیدا می‌کنیم یک جواب بهینه باشد، بلکه تنها کافی است

جوابی شدنی یا موجه باشد. لازم به ذکر است که نحوه‌ی انتخاب تابع هدف و جهت بهینه‌سازی(یعنی این‌که مسئله‌ی IP از نوع کمینه‌سازی باشد یا بیشینه‌ی سازی)، می‌توانند تأثیرات زیادی روی زمان اجرای الگوریتم ۲۵ داشته باشند. در [۶۶]، طی آزمایش‌هایی روی دستگاه‌های معادلات به‌دست آمده از سامانه‌ی رمز CTC، نشان داده شده که انتخاب تابع هدف و جهت بهینه‌سازی، به پارامترهایی از جمله ویژگی‌های دستگاه اصلی و نوع حل‌کننده بستگی دارد و نمی‌توان یک فرمول کلی برای آن‌ها بیان کرد. عامل دیگری که در سرعت اجرای الگوریتم ۲۵، تأثیر زیادی دارد، نوع محدودیتی است که برای متغیرهای مسئله IP در نظر می‌گیریم. از آن‌جایی که پیچیدگی حل مسئله‌ی MILP بیشتر تحت تأثیر تعداد متغیرهای صحیح است، معمولاً بهتر است تا آن‌جایی که می‌توانیم قیدهای صحیح بودن متغیرها را کم کنیم و آن‌ها را متغیرهای حقیقی در نظر بگیریم. با این وجود در [۶۶]، با انجام آزمایش‌هایی نشان داده شده که گاهی اوقات در یک نوع حل‌کننده IP، دودویی در نظر گرفتن متغیرها می‌تواند بیشتر از حقیقی در نظر گرفتن متغیرها سبب تسریع اجرای الگوریتم ۲۵ شود، در نتیجه این نکته که حل‌کننده از چه روشی برای حل مسئله IP استفاده می‌کند نیز در زمان اجرا و انتخاب پارامترهای مناسب تأثیر گذار است.

مثال ۵۰.۴. فرض کنید $f_1, f_2, f_3 \in \mathbb{F}_2[x_1, x_2, x_3]$، به‌طوری‌که

$$f_1 = x_1x_2 + x_1x_3 + 1,\ f_2 = x_1x_3 + x_2x_3 + x_1 + x_3 + 1,\ f_3 = x_1x_2 + x_1x_3 + x_2 + 1.$$

با استفاده از الگوریتم ۲۵ به‌صورت گام‌به‌گام به حل دستگاه معادلات $f_1 = f_2 = f_3 = 0$ می‌پردازیم.

۱. قرار می‌دهیم $S_1 = \{x_1x_2, x_1x_3\}, S_2 = \{x_1x_3, x_2x_3\}$ و $S_3 = \{x_1x_2, x_1x_3\}$. که در این‌صورت داریم

$$s_1 = 3, s_2 = 5, s_3 = 4;\ S = \{x_1x_2, x_1x_3, x_2x_3\}$$

۲. متغیرهای صحیح جدید K_1, K_2, K_3 و سه شرط $I_1 : K_1 \leq 1,\ I_2 : K_2 \leq 2$ و $I_3 : K_3 \leq 2$، را در نظر می‌گیریم.

۳. متغیرهای صحیح جدید X_4, X_5, X_6 را معرفی کرده و معادلات زیر را به قیدهای مسئله اضافه می‌کنیم.

$$\begin{aligned} F_1 &: X_4 + X_5 + 1 - 2K_1 = 0 \\ F_2 &: X_5 + X_6 + X_1 + X_3 + 1 - 2K_2 = 0 \\ F_3 &: X_4 + X_5 + X_2 + 1 - 2K_3 = 0 \end{aligned}$$

$CTC(B,N)$	m	n	t	زمان GBasis5	زمان GLPK	تعداد جواب
$CTC(۲,۲)$	۹۸	۵۴	۶۰	۰/۰۷۰	۰/۰۱۰	۱
$CTC(۲,۳)$	۱۴۴	۷۸	۹۰	۰/۳۷۵	۰/۲۰۰	۱
$CTC(۳,۲)$	۱۴۷	۸۱	۹۰	۰/۵۳۹	۰/۷۰۰	۱
$CTC(۳,۳)$	۲۱۶	۱۱۷	۱۳۵	۹/۲۰۸	۳/۳۰۰	۱
$CTC(۳,۴)$	۲۸۵	۱۵۳	۱۸۰	۵۹۸/۸۲۸	۲۰/۷۰۰	۱

جدول ۴.۴: مقایسه حمله‌های جبری پایه گروبنر و برنامه‌ریزی عدد صحیح روی CTC

۴. نابرابری‌های خطی زیر را به قیدها اضافه می‌کنیم.

$$\begin{aligned} I_۴ &: X_۱ + X_۲ - X_۴ \le ۱, & I_{۱۱} &: X_۱ \ge X_۴, & I_{۱۲} &: X_۲ \ge X_۴,\\ I_۵ &: X_۱ + X_۳ - X_۵ \le ۱, & I_{۲۱} &: X_۱ \ge X_۵, & I_{۲۳} &: X_۳ \ge X_۵,\\ I_۶ &: X_۲ + X_۳ - X_۶ \le ۱, & I_{۳۲} &: X_۲ \ge X_۶, & I_{۳۳} &: X_۳ \ge X_۶. \end{aligned}$$

۵. قیدهای جدید $I'_۱ : X_۱ \le ۱, I'_۲ : X_۲ \le ۱$ و $I'_۳ : X_۳ \le ۱$ را معرفی می‌کنیم.

۶. تابع هدف را برابر با $C = X_۱+X_۲+X_۳$ انتخاب می‌کنیم. اکنون از یک حل‌کننده‌ی IP برای کمینه‌کردن C تحت قیدهای $\{I_۱, ..., I_۶, F_۱, F_۲, F_۳, I_{۱۱}, I_{۱۲}, I_{۲۱}, I_{۲۳}, I_{۳۲}, I_{۳۳}, I'_۱, I'_۲, I'_۳\}$، استفاده می‌کنیم.

۷. حل کننده‌ی مسئله‌ی IP جواب $(X_۱, X_۲, X_۳) = (۱, ۰, ۱)$ را به‌دست می‌دهد.

مثال ۵۱.۴. الگوریتم رمزنگاری CTC به ازای یک زوج متن اصلی و رمزشده معلوم را در نظر بگیرید. در جدول ۴.۴، مقایسه‌ای بین حمله جبری مبتنی بر پایه گروبنر و حمله جبری با استفاده از روش برنامه‌ریزی با عدد صحیح، صورت گرفته است. برای محاسبه پایه گروبنر از دستور `GBasis5(.)` در نرم‌افزار ApCoCoA، و برای حل مسئله برنامه‌ریزی با عدد صحیح متناظر با دستگاه معادلات استخراج شده از CTC، که توسط الگوریتم ۲۵ به‌دست آمده، از بسته‌ی نرم‌افزاری GLPK [۶۵] استفاده شده است. در ضمن زمان‌های گزارش شده بر حسب ثانیه، و محاسبات با استفاده از یک رایانه با پردازنده $۱/۶GHz$ و حافظه رَم $۶GB$ انجام شده است. m و n در جدول فوق به‌ترتیب تعداد معادلات و تعداد مجهولات متناظر با $CTC(B,N)$ را نشان می‌دهد و پارامتر t بیان‌گر تعداد یکجمله‌ای‌های غیر خطی ظاهر شده در معادلات است.

در دستگاه‌های به‌دست آمده در جدول ۴.۴، تعداد یکجمله‌ای‌های غیرخطی از تعداد معادلات کمتر است و همان‌طور که مشاهده می‌شود در همه این حالات روش برنامه‌ریزی با عدد صحیح سریع‌تر از روش پایه گروبنر عمل می‌کند. اما این نتیجه‌گیری برای دستگاه‌هایی که تعداد یکجمله‌ای‌های غیرخطی آن‌ها بیشتر از تعداد معادلات باشد صحیح نیست و همان‌طور که در [۴۳] نیز اشاره شده است، در چنین مواردی روش پایه گروبنر بهتر از روش برنامه‌ریزی با عدد صحیح عمل خواهد

کرد.

۵.۴ حمله جبری مبتنی بر مسئله صدق‌پذیری

در این بخش دستگاه معادلات به‌دست آمده از سامانه رمزنگاری را به مسئله صدق‌پذیری تبدیل می‌کنیم و سپس با استفاده از حل‌کننده‌های مسئله صدق‌پذیری جواب مسئله صدق‌پذیری و در نتیجه جواب دستگاه معادلات را به‌دست می‌آوریم. فرض کنید مجموعه متغیرهای بولی (منطقی) را با $X = \{X_۱, ..., X_n\}$ و مجموعه شامل همه گزاره‌های منطقی مرکب از متغیرهای X_i و عمل‌های $\wedge$، $\vee$ و $\neg$ (یا نقیض) را با $\widehat{X}$ نمایش دهیم.

تعریف ۵۲.۴ (مسئله صدق‌پذیری). فرض کنید $\widehat{X}$ مجموعه شامل همه گزاره‌های منطقی مرکب از متغیرهای منطقی $X = \{X_۱, ..., X_n\}$ و عمل‌گرهای $\wedge, \vee$ و $\neg$ باشد. در مسئله *صدق‌پذیری* که یک مسئله تصمیم است، سؤال این است که، آیا می‌توان به متغیرهای منطقی یک گزاره‌ی منطقی مثل $P \in \widehat{X}$ طوری مقادیر منطقی «true» و «false» نسبت داد تا گزاره‌ی منطقی مرکب P، «true» باشد؟ به این ترتیب یک فرمول یا گزاره منطقی $P \in \widehat{X}$ صدق‌پذیر است، اگر بتوان متغیرهای منطقی آن‌را طوری مقداردهی کرد که ارزش منطقی آن گزاره درست باشد، در غیر این‌صورت آن‌را *صدق‌ناپذیر* گوییم. برای مثال گزاره منطقی $P = (X_۱ \vee \neg X_۲ \vee \neg X_۳) \wedge (\neg X_۱ \vee X_۲ \vee X_۴)$، صدق‌پذیر است زیرا اگر قرار دهیم $X_۱ = \texttt{true}, X_۲ = \texttt{false}$ و $X_۳ = \texttt{false}, X_۴ = \texttt{true}$، آن‌گاه $P = \texttt{true}$.

تعریف ۵۳.۴ (صورت متعارف عطفی). یک گزاره منطقی مرکب $P \in \widehat{X}$ زمانی دارای *صورت متعارف عطفی* است که، از ترکیب عطفی چند گزاره مرکب فصلی تشکیل شده باشد. به عبارت دیگر صورت متعارف عطفی یک گزاره منطقی مثل $P \in \widehat{X}$ به‌صورت زیر است.

$$P = \bigwedge_{i=۱}^{m} (\bigvee_{j=۱}^{k_i} l_{ij}); \quad m, k_i \in \mathbb{N},\ l_{ij} \in \{X_t, \neg X_t :\ t = ۱, \ldots, n\}.$$

به هر یک از l_{ij}ها یک لیترال و به هر یک از گزاره‌های داخل پرانتز که ترکیب فصلی چند لیترال هستند یک بند می‌گوییم.

تعریف ۵۴.۴ (صورت متعارف فصلی). گزاره منطقی $P \in \widehat{X}$ زمانی دارای صورت متعارف فصلی است که از ترکیب فصلی چند گزاره مرکب عطفی تشکیل شده باشد. به عبارت دیگر صورت متعارف فصلی گزاره منطقی $P \in \widehat{X}$ به‌صورت زیر تعریف می‌شود

$$P = \bigvee_{i=۱}^{m} (\bigwedge_{j=۱}^{k_i} l_{ij}); \quad m, k_i \in \mathbb{N},\ l_{ij} \in \{X_t, \neg X_t :\ t = ۱, \ldots, n\}.$$

به روش‌های مختلفی نظیر استفاده از قوانین دمورگان یا جدول ارزش می‌توان صورت متعارف عطفی یا فصلی هر گزاره منطقی را به‌دست آورد. یکی از این روش‌ها استفاده از قاعده‌ای موسوم به *بسط شانون* است. فرض کنید $F(X_1, \dots, X_n)$ یک گزاره منطقی متشکل از متغیرهای X_i باشد، در این‌صورت به ازای هر $i \in \{1, ..., n\}$ می‌توان F را به‌صورت

$$F(X_1, \dots, X_n) =(\neg X_i \vee f(X_1, \dots, X_{i-1}, 1, X_{i+1}, \dots, X_n))\wedge$$
$$(X_i \vee f(X_1, \dots, X_{i-1}, 0, X_{i+1}, \dots, X_n)),$$

یا به‌صورت

$$F(X_1, \dots, X_n) =(X_i \wedge f(X_1, \dots, X_{i-1}, 1, X_{i+1}, \dots, X_n))\vee$$
$$(\neg X_i \wedge f(X_1, \dots, X_{i-1}, 0, X_{i+1}, \dots, X_n)),$$

که $\texttt{true} = 1$ و $\texttt{false} = 0$ نمایش داد. این روش بسط دادن گزاره‌های منطقی، به بسط شانون معروف است. با تکرار اولین قاعده می‌توان صورت متعارف عطفی گزاره F را به‌صورت زیر به‌دست آورد.

$$\begin{aligned} F(X_1, ..., X_n) =&(F(0, 0, \dots, 0) \vee X_1 \vee X_2 \vee \dots \vee X_n)\wedge \\ &(F(1, 0, \dots, 0) \vee \neg X_1 \vee X_2 \vee \dots \vee X_n)\wedge \\ &\dots \\ &(F(1, 1, \dots, 1) \vee \neg X_1 \vee \neg X_2 \vee \dots \neg X_n). \end{aligned}$$

به‌طور مشابه و با تکرار قاعده دوم می‌توانیم صورت متعارف فصلی هر گزاره منطقی را نیز به‌دست آوریم.

۱.۵.۴ الگوریتم‌های حل مسئله صدق‌پذیری

در این بخش به طور مختصر به بررسی دو دسته از الگوریتم‌های حل کننده مسئله صدق‌پذیری می‌پردازیم که با وجود قدمت و سادگی هنوز هم پایه و اساس بسیاری از الگوریتم‌های نوین هستند. الگوریتم‌های حل‌کننده مسئله صدق‌پذیری به دو دسته کامل و ناتمام یا ناکامل تقسیم می‌شوند. الگوریتم‌های کامل الگوریتم‌هایی هستند که پس از متناهی مرحله خاتمه می‌یابند در حالی که صدق‌پذیر یا صدق‌ناپذیر بودن مسئله صدق‌پذیری داده شده را مشخص می‌کنند، اما الگوریتم‌های ناتمام یا ناکامل الگوریتم‌هایی هستند که پس از متناهی مرحله خاتمه می‌یابند ولی تضمینی نیست

صدق‌پذیری یا صدق‌ناپذیری مسئله داده شده را مشخص کنند، به‌عبارت دیگر خروجی چنین الگوریتم‌هایی یا اعلام صدق‌پذیری مسئله داده شده است یا این‌که بدون هیچ نتیجه‌ای به پایان می‌رسند. در ابتدا دو الگوریتم ناتمام یا ناکامل و سپس سه الگوریتم کامل، که از اهمیت بیشتری برخوردار بوده و در الگوریتم‌های نوین نیز مورد استفاده قرار می‌گیرند را بررسی می‌کنیم.

الگوریتم‌های GSAT و WalkSAT

یک گزاره منطقی به‌صورت متعارف عطفی مثل F را در نظر بگیرید، فرض کنید ابتدا همه متغیرها را به‌صورت تصادفی و با احتمال یکسان با مقادیر ۰ و ۱ مقداردهی کنیم، به‌ازای این انتساب، مقدار برخی از بندها ۱ و مقدار برخی دیگر ۰ خواهد بود، بندهایی که مقدار آن‌ها ۱ است را بند صادق و بندهایی که مقدار ۰ می‌گیرند را بند ناصادق می‌گوییم. طبیعی است که برای تبدیل یک گزاره ناصادق به گزاره صادق باید حداقل مقدار یکی از متغیرهای به‌کار رفته در آن بند را تغییر دهیم. به همین دلیل فرض کنید در گام بعد یکی از بندهای ناصادق را با احتمال یکنواخت انتخاب کرده و سپس به‌تصادف (باز هم با احتمال یکنواخت) مقدار یکی از متغیرهای به‌کار رفته در آن بند ناصادق را تغییر دهیم. به این ترتیب بند ناصادق انتخاب شده به بند صادق تبدیل می‌شد. این کار را تا جایی ادامه می‌دهیم که همه بندها(در صورت امکان) به بند صادق تبدیل شوند. گرچه این فرآیند برای تعیین صدق‌پذیری یک گزاره داده شده بسیار ساده است و ممکن است در صورت صدق‌ناپذیری گزاره داده شده اصلاً به‌پایان نرسد، ولی پاپادیمیتریو [۱۲] و استایگلیتز [۱۳] در سال ۱۹۸۲ در [۵۶]، نشان‌دادند که، اگر یک گزاره منطقی داده‌شده به صورت متعارف عطفی با حداکثر دو متغیر در هر بند ($2CNF$) و صدق‌پذیر باشد، ای روش قادر است در زمان $\mathcal{O}(n^2)$ یک انتساب مقادیر مناسب برای گزاره داده شده بیابد که n تعداد کل متغیرهای به‌کار رفته در گزاره مورد نظر است. ۱۰ سال بعد، الگوریتم GSAT[14] معرفی شد که در آن به‌جای انتخاب تصادفی، یک روش حریصانه برای انتخاب متغیری که مقدارش باید تغییر کند در نظر گرفته شد.

الگوریتم GSAT که در سال ۱۹۹۲ توسط سِلمَن و همکاران در [۵۸] معرفی شد، در زمره الگوریتم‌های ناتمام یا ناکامل قرار می‌گیرید. الگوریتم GSAT چنان‌چه جزئیات آن در الگوریتم ۲۶ نشان داده شده است، با یک تخصیص مقدار تصادفی برای متغیرهای گزاره داده شده F، آغاز می‌شود. با جایگذاری هر یک از مقادیر تخصیص داده شده به جای متغیرهای به‌کار رفته در بندهای F برخی از بندها صادق و برخی ناصادق خواهند بود. طبیعی است که F زمانی صدق‌پذیر است که به ازای تخصیص مقادیر انتخاب شده، همه بندها صادق باشند. الگوریتم GSAT بعد از تخصیص اولیه که به‌صورت تصادفی به هر یک از متغیرها یکی از مقادیر ۰ یا ۱ را با احتمال یکسان نسبت می‌دهد، مقادیر تخصیص داده شده به هر یک از متغیرهای منطقی را طوری تغییر می‌دهد

[12] Papadimitriou

[13] Steiglitz

[14] Greedy SAT

که این تغییرات سبب بزرگترین تغییر در جهت کاهش تعداد بندهای ناصادق شود. از آن‌جایی که هر بار فقط مقدار یکی از متغیرها تغییر می‌کند لذا تخصیص جدید فقط در مقدار یک متغیر با تخصیص قبلی متمایز است و فاصله همینگ دو تخصیص پی‌درپی ۱ خواهد بود، به همین دلیل این الگوریتم نمونه‌ای از الگوریتم‌های جست‌وجوی محلی محسوب می‌شود چرا که جست‌وجو برای یافتن یک تخصیص مناسب در همسایگی با فاصله‌ی همینگ ۱ از تخصیص قبلی صورت می‌گیرد. این روند تا جایی ادامه می یابد که شرط توقف حاصل شود. الگوریتم GSAT زمانی خاتمه می‌یابد که σ، یک انتساب رضایت‌بخش باشد و به‌ازای آن $F|_\sigma = ۱$، یا این‌که شمارنده i به $n_{restart}$ برسد که در این‌صورت الگوریتم به شکست انجامیده است. بنابراین تضمینی وجود ندارد که GSAT بتواند صدق‌پذیری یا صدق‌ناپذیری گزاره داده شده را تعیین کند. به‌عبارت دیگر اگر GSAT، یک انتساب رضایت‌بخش بیابد مطمئن خواهیم شد که گزاره داده شده صدق‌پذیر است ولی وقتی الگوریتم به شکست می‌انجامد نمی‌توان با قطعیت گفت گزاره داده شده صدق‌ناپذیر بوده است.

k_v که در خط ۱۰ الگوریتم ۲۶ محاسبه می‌شود، می‌تواند منفی هم باشد. هر چه k_v متناظر با یک متغیر کوچک‌تر باشد، نشان‌دهنده این است که تغییر مقدار آن متغیر سبب کاهش تعداد بیشتری از بندهای ناصادق می‌شود. پارامتر $n_{restart}$ برای اطمینان یافتن از خاتمه الگوریتم بعد از تعدادی متناهی مرحله است و تعداد از سرگیری‌های الگوریتم را نشان می‌دهد. پارامتر n_{flip} نشان‌دهنده تعداد دفعاتی است که در هربار از سرگیری الگوریتم، باید یک متغیر را انتخاب و مقدار آن را تغییر دهیم. سِلمَن و همکاران در [۵۸] نشان‌دادند که الگوریتم GSAT، در دسته‌ای از مسائل صدق‌پذیری، نسبت به الگوریتم‌های کامل زمان خود (DP[۲۸]) عملکرد بهتری داشته است.

حالتی را در نظر بگیرید که طی چند گام متوالی در GSAT، تعداد بندهای ناصادق دست‌نخورده باقی بماند و پیشرفتی در روند حل حاصل نشود، این سؤال پیش می‌آید که در عمل، سرانجام چنین حالاتی چیست؟ آیا ممکن است بعد از متناهی مرحله به حالتی رسید که باز هم تعداد بندهای ناصادق کاهش یابد؟ فرانک [15] و همکاران در [۳۵]، با انجام آزمایش‌هایی نشان دادند که، تقریبا در همه مواردی که با چنین وضعیتی روبرو می‌شویم، بعد از متناهی مرحله به حالتی مشابه ولی با تعداد بندهای ناصادق کم‌تر می‌رسیم، یعنی در وضعیت جدید نیز با دنباله‌ای از گام‌های متوالی مواجه می‌شویم که تعداد بندهای ناصادق در آ نها یکسان، ولی تعداد بندهای ناصادق در این وضعیت نسبت به وضعیت قبلی کم‌تر است. آن‌ها نشان دادند در موارد بسیار نادری نیز ممکن است الگوریتم GSAT در وضعیتی قرار بگیرد که دنباله‌ی گام‌ها با تعداد بندهای ناصادق یکسان، به هیچ حالتی با تعداد بند ناصادق کم‌تر ختم نشود، چنین حالاتی را کمینه موضعی می‌گوییم و وقتی الگوریتم در چنین وضعیتی قرار بگیرد بعد از متناهی مرحله به شکست خواهد انجامید.

[15] Frank

آزمایش‌ها همچنین نشان می‌داد که در ابتدای اجرای GSAT، روند کاهش تعداد بندهای ناصادق سریع است ولی بعد از مدتی الگوریتم بیشتر زمان خود را در دنباله گام‌ها با تعداد بندهای ناصادق یکسان می‌گذراند. به‌دنبال نتایج به‌دست آمده تلاش‌های زیادی برای فرار از وضعیت‌های کمینه موضعی و بهبود زمان اجرای الگوریتم GSAT صورت گرفت که یکی از بهترین پیشنهادات الگوریتم WalkSAT بود.

الگوریتم WalkSAT نیز در دسته الگوریتم‌های ناتمام یا ناکامل قرار می‌گیرد که توسط سِلمَن و همکاران در سال ۱۹۹۶ در [۵۹]، معرفی شد. در این الگوریتم روش انتخاب متغیری که قرار است مقدار آن تغییر کند با الگوریتم GSAT متفاوت است. فرض کنید $breakcount_x$ نشان‌دهنده تعداد بندهایی باشد که با تغییر مقدار فعلی متغیر x، از حالت صادق به حالت ناصادق درمی‌آیند، در این صورت جزئیات الگوریتم WalkSAT به‌صورت نشان داده شده در الگوریتم ۲۷ است. همان‌طور که مشاهده می‌شود، در الگوریتم ۲۷، متغیری که قرار است مقدارش تغییر کند از بین متغیرهای یک بند ناصادق و به دو صورت مختلف انتخاب می‌شود، به‌طوری که با احتمال p_{noise} این کار به‌صورت تصادفی، و با احتمال $1 - p_{noise}$ با روشی حریصانه و مشابه روش الگوریتم GSAT، صورت می‌گیرد. در این میان متغیرهایی که با تغییر مقدار آن هیچ بند صادقی به بند ناصادق تبدیل نمی‌شود، در اولویت قرار دارند و در صورت وجود چنین متغیری فقط مقدار آن را تغییر می‌دهیم و گام‌های ۱۰ و ۱۱ الگوریتم ۲۷ اجرا نخواهند شد. این الگوریتم نیز مانند GSAT ناتمام است و در صورت صدق‌ناپذیر بودن گزاره داده شده، حتماً به شکست می‌انجامد ولی عکس این موضوع درست نیست. به عبارت دیگر اگر WalkSAT یک انتساب رضایت‌بخش بازگرداند، مطمئن می‌شویم که گزاره داده شده صدق‌پذیر است ولی در حالتی که الگوریتم به شکست می‌انجامد، راجع به صدق‌پذیری گزاره داده شده نمی‌توانیم با اطمینان قضاوت کنیم. در ادامه به بررسی الگوریتم‌های کامل می‌پردازیم که در الگوریتم‌های نوین بیشتر مورد توجه قرار گرفته‌اند.

الگوریتم DP

الگوریتم DP که در دسته الگوریتم‌های کامل قرار می‌گیرد، در سال ۱۹۶۰ توسط دِیویس [16] و پاتنِم [17]، در [۲۵] معرفی شد. برای شرح این الگوریتم و سایر الگوریتم‌های کاملی که در این بخش معرفی می‌شود یک شیوه نمایش مجموعه‌ای، برای صورت متعارف عطفی معرفی می‌کنیم. در این نحوه نمایش هر بند مثل $l_1 \vee \ldots \vee l_m$ را با مجموعه $\{l_1, \ldots, l_m\}$ و هر گزاره به‌صورت متعارف عطفی مثل $C_1 \wedge \ldots \wedge C_n$ را با مجموعه $\{C_1, \ldots, C_n\}$ نمایش می‌دهیم برای مثال صورت

[16] Davis

[17] Putnam

متعارف عطفی زیر

$$P = (X_1 \vee X_2 \vee \neg X_3) \wedge (\neg X_1 \vee X_4) \wedge (X_2 \vee X_3 \vee X_4)$$

به‌صورت مجموعه

$$P = \{\{X_1, X_2, \neg X_3\}, \{\neg X_1, X_4\}, \{X_2, X_3, X_4\}\}$$

نیز نمایش داده می‌شود. دو عملگر مهمی که جهت ساده‌سازی و حذف متغیرهای گزاره‌های منطقی در الگوریتم DP مورد استفاده قرار می‌گیرند عبارت‌اند از رفع و استنتاج، که در ادامه آن‌ها را تعریف می‌کنیم.

تعریف ۵۵.۴ (عمل‌گر رفع). فرض کنید $C_1 = \bigvee_{j=1}^{m} l_{1,j}$ و $C_2 = \bigvee_{j=1}^{n} l_{2,j}$ دو گزاره به‌صورت ترکیب فصلی باشند که $l_{1,j}$ و $l_{2,j}$ ها لیترال‌های منطقی هستند. همچنین فرض کنید که دقیقاً به‌ازای یک زوج (p, q) که $1 \leq p \leq m$ و $1 \leq q \leq n$ داشته باشیم $l_{1,p} = \neg l_{2,q}$ در این‌صورت حاصل **رفع** دو گزاره را که با $C_1 \odot C_2$ نمایش می‌دهیم به‌صورت زیر تعریف می‌شود.

$$C_1 \odot C_2 = l_{1,1} \vee \ldots \vee l_{1,p-1} \vee l_{1,p+1} \ldots \vee l_{1,m} \vee l_{2,1} \vee \ldots \vee l_{2,q-1} \vee l_{2,q+1} \vee \ldots \vee l_{2,n}.$$

در واقع رفع دو گزاره C_1 و C_2 شامل همه لیترال‌های C_1 و C_2 به جز دو لیترال نقیض هم است، که در دو گزاره وجود دارند. به متغیر متناظر با لیترال حذف شده در رفع دو گزاره **متغیر محوری** می‌گوییم.

با توجه به تعریف عمل‌گر رفع[18] روشن است که $C_1 \wedge C_2$ به لحاظ منطقی $C_1 \odot C_2$ را نتیجه می‌دهد، به عبارت دیگر گزاره $C_1 \wedge C_2 \Rightarrow C_1 \odot C_2$ در صورتی که رفع C_1 و C_2 قابل تعریف باشد، برقرار است. در نتیجه اگر F یک گزاره منطقی به‌صورت متعارف عطفی و C' گزاره‌ای باشد که از رفع دو گزاره $C_1, C_2 \in F$ به‌دست آمده باشد آن‌گاه F صدق‌پذیر است اگر و تنها اگر $F \cup C'$ صدق‌پذیر باشد. بنابراین افزودن بندهای حاصل از رفع بندهای موجود در یک گزاره متعارف عطفی، صدق‌پذیری مسئله را دچار تغییر نمی‌کند.

در ادامه عمل‌گر استنتاج را تعریف می‌کنیم که اعمال آن پس از عمل‌گر رفع سبب سادگی و کوچک‌تر شدن صورت متعارف عطفی می‌شود.

تعریف ۵۶.۴ (عمل‌گر استنتاج). فرض کنید C_1 و C_2 دو بند از صورت متعارف عطفی $F = \bigwedge_{i=1}^{n} C_i$ باشند به طوری‌که $C_2 \subset C_1$، در این صورت، استنتاج C_1 و C_2 به‌صورت $C_1 \boxdot C_2 := C_1$ تعریف می‌شود.

[18] resolution

روشن است که اگر $C_۲ \subset C_۱$ و $C_۲$ صدق‌پذیر باشد، آن‌گاه $C_۱$ نیز صدق‌پذیر است و بالعکس. بنابراین بدون این که صدق‌پذیری گزاره تغییر کند می‌توانیم بندهای بزرگ‌تر که شامل بندهای دیگر هستند را حذف کنیم.

مثال ۵۷.۴. گزاره $F = C_۱ \wedge C_۲$ را که $C_۱ = X_۱ \vee X_۲ \vee X_۳$ و $C_۲ = \neg X_۱ \vee X_۲ \vee X_۳$، در نظر بگیرید، فرآیند ساده‌سازی با استفاده از دو عمل‌گر رفع و استنتاج در زیر نشان داده شده است.

$$\text{گزاره اولیه} : F = (X_۱ \vee X_۲ \vee X_۳) \wedge (\neg X_۱ \vee X_۲ \vee X_۳)$$
$$\text{رفع} : (X_۱ \vee X_۲ \vee X_۳) \wedge (\neg X_۱ \vee X_۲ \vee X_۳) \wedge (X_۲ \vee X_۳)$$
$$\text{استنتاج} : (X_۲ \vee X_۳)$$

اگرچه گزاره $(X_۲ \vee X_۳)$ از لحاظ منطقی معادل F نیست ولی همان‌طور که قبلاً هم ذکر شد از لحاظ صدق‌پذیری با F معادل است، ضمن این‌که بسیار ساده‌تر و کوچک‌تر از F است.

در نمایش مجموعه‌ای صورت متعارف عطفی، رفع دو گزاره $C_۱$ و $C_۲$ را در صورتی‌که X متغیر محوری رفع باشد، می‌توانیم به‌صورت زیر نمایش دهیم

$$C_۱ \odot C_۲ := \{l \in C_۱ \cup C_۲ | l \neq X,\ l \neq \neg X\}.$$

در الگوریتم DP، صدق‌پذیری یک صورت متعارف عطفی داده شده بر اساس حذف متوالی متغیرها با استفاده از عمل رفع، بررسی می‌شود. صورت متعارف عطفی داده شده به الگوریتم، پس از متناهی مرحله و حذف پی‌درپی متغیرها، سرانجام یا تهی خواهد شد یا شامل یک مجموعه تهی خواهد بود، که در حالت اول مسئله صدق‌پذیر و در حالت دوم مسئله صدق‌ناپذیر است. فرض کنید مجموعه لیترال‌های یک گزاره به‌صورت متعارف عطفی مثل F را با $\mathrm{Lit}(F)$ و مجموعه متغیرهای آن‌را با $\mathrm{Vars}(F)$ و مجموعه بندهای آن‌را با Clauses(F) نمایش دهیم. در ضمن فرض کنید به‌ازای هر $C_۱, C_۲ \in F$ و هر $X \in \mathrm{Vars}(F)$ تابع $\texttt{IsPossibleResol}(C_۱, C_۲, X)$ را به‌صورت زیر تعریف کنیم.

$$\texttt{IsPossibleResol}(C_۱, C_۲, X) = \begin{cases} \texttt{true} & \text{و}C_۱\text{نسبت به متغیر محوری }X\text{ رفع‌پذیر باشند}C_۲ \\ \texttt{false} & \text{در غیر این‌صورت} \end{cases}$$

در این‌صورت، روش کار DP به‌صورت نشان داده شده در الگوریتم ۲۸ است. در این‌جا قصد نداریم درستی الگوریتم ۲۸ را ثابت کنیم، در عوض به‌طور مختصر به دلیل هر یک از مراحل الگوریتم اشاره می کنیم. گزاره منطقی F را در نظر بگیرید. اگر یک بند از گزاره فقط شامل یک لیترال باشد، اصطلاحاً به آن بند، **بند منفرد** می‌گویند. حال اگر یک گزاره شامل یک بند منفرد

باشد، بدیهی است که متغیر متناظر با آن بند باید طوری مقدار دهی شود که ارزش آن بند `true` باشد. برای مثال فرض کنید بند $C = \{l\}$ یک بند از F و $l \in \mathrm{Lit}(F)$، در این‌صورت باید داشته باشیم $l = \mathtt{true}$ که در نتیجه آن همه بندهای شامل l نیز دارای ارزش منطقی `true` خواهند بود، به این‌ترتیب مسئله صدق‌پذیری F به مسئله صدق پذیری $F|_{l=true}$ تحویل می‌یابد. در ضمن روشن است که می‌توانیم همه لیترال‌های $\neg l$ را از بین تمام بندهای موجود حذف کنیم، به این ترتیب با انجام دو عملیات فوق یک گزاره منطقی ساده‌تر، ولی از لحاظ صدق‌پذیری معادل با F به‌دست می‌آید. این عملیات که در خطوط ۵ و ۶ الگوریتم ۲۸ انجام می‌شود به **قاعده بند منفرد** معروف است.

فرض کنید تنها یک صورت از لیترال l در گزاره منطقی F وجود داشته باشد، یعنی تنها یکی از حالات $l \in \mathrm{Lit}(F)$ یا $\neg l \in \mathrm{Lit}(F)$ رخ دهد، به چنین لیترال‌هایی **لیترال‌های محض** می‌گوییم و روشن است که، یک تخصیص مقادیر برای متغیرهای F که منجر به صدق‌پذیری F شود وجود دارد اگر و تنها اگر، تخصیصی وجود داشته باشد که به ازای آن ضمن این که F صدق‌پذیر است داشته باشیم $l = \mathtt{true}$. در نتیجه با حذف همه لیترال‌های محض F، به مسئله صدق‌پذیری کوچکتری که از لحاظ صدق‌پذیری با F معادل است دست می‌یابیم. عمل حذف لیترال‌های محض F که در گام‌های ۸ و ۹ الگوریتم ۲۸ انجام می‌شود را **قاعده لیترال محض** می‌نامیم.

در گام‌های ۱۲ تا ۱۸ الگوریتم ۲۸ عمل رفع بین گزاره‌هایی که رفع آن‌ها امکان‌پذیر است انجام می‌شود و حاصل رفع آن‌ها نسبت به متغیری که در خط ۱۱ به‌تصادف انتخاب شده است، به مجموعه بندهای موجود اضافه می‌شود، سپس در خطوط ۱۹ تا ۲۱ تمام بندهای شامل متغیر انتخاب شده حذف می‌شوند. با توجه به این‌که حذف بندها طوری انجام می شود که صدق‌پذیری گزاره اولیه تغییر نکند، لذا گزاره‌ای که در پایان الگوریتم به‌دست آید از لحاظ صدق‌پذیری با گزاره اولیه معادل است. به این‌ترتیب اگر مجموعه متناظر با گزاره نهایی تهی باشد، گزاره اصلی صدق‌پذیر، و اگر در فرآیند رفع، بند تهی به‌دست آید، گزاره صدق‌ناپذیر خواهد بود.

همان‌طور که مشاهده می‌شود ، طراحی الگوریتم ۲۸ به‌گونه‌ای است که طی مراحل متوالی متغیرهای موجود، از گزاره به‌دست آمده از مراحل قبل حذف می‌شوند، به‌این ترتیب سرانجام، یا مجموعه متناظر با گزاره به‌دست‌آمده تهی است و یا بند حاصل شده از رفع دو گزاره در خط ۱۳ تهی خواهد بود که در هر دو حالت شرط توقف حاصل شده و الگوریتم با بازگرداندن یکی از حالات SAT یا UNSAT پایان می‌یابد.

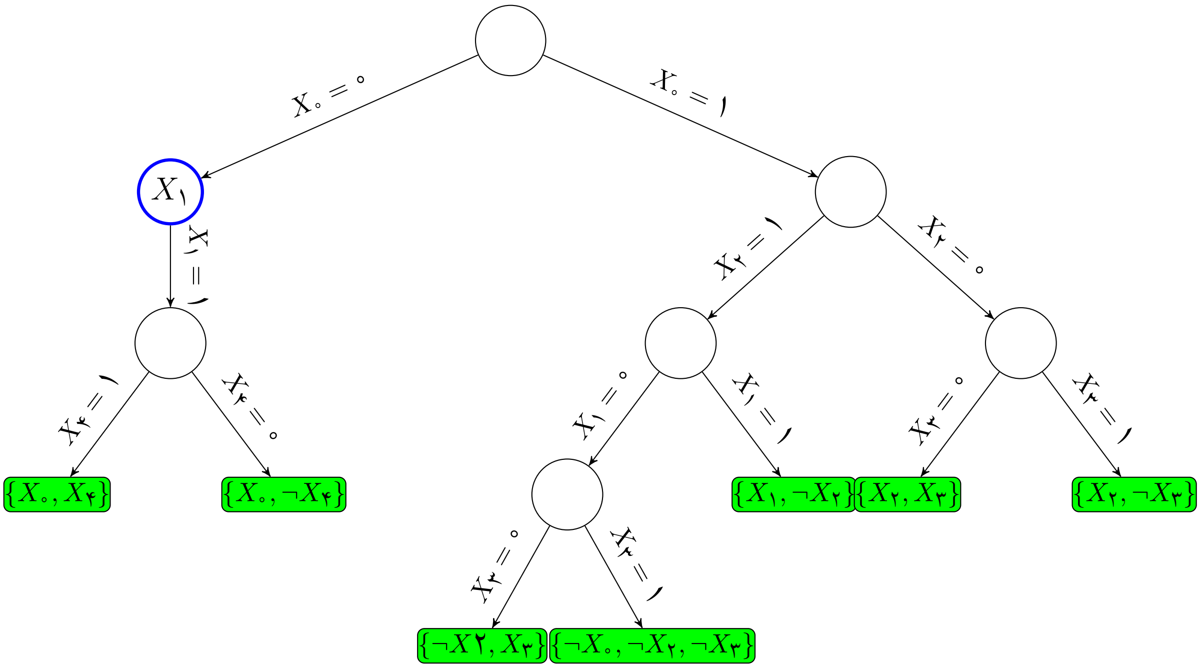
لازم به ذکر است که اگر چه از عمل‌گر استنتاج به‌صورت صریح در الگوریتم ۲۸ استفاده نشده ولی به‌سادگی می‌توان این عملیات را به الگوریتم افزود. در ضمن در صورتی که مسئله داده شده صدق‌پذیر باشد، مقادیری که به‌ازای آن مسئله صدق‌پذیر است توسط الگوریتم بازگردانده نمی‌شوند، که به‌راحتی می‌توان این قابلیت را نیز به الگوریتم ۲۸ داد.

الگوریتم DPLL

دو سال بعد از ارائه الگوریتم DP یعنی در سال ۱۹۶۲، الگوریتم DPLL به‌صورت مشترک توسط دیویس، لاجمن [۱۹]، و لاولند [۲۰] در [۲۴] ارائه شد. این الگوریتم که در دسته الگوریتم‌های کامل قرار می‌گیرد، یک الگوریتم تقسیم و غلبه است. روش کار DPLL در الگوریتم ۲۹، نشان داده شده است. همان‌طور که مشاهده می‌شود، قاعده‌های ساده‌سازی بندهای منفرد و لیترال محض که در الگوریتم DP استفاده شدند، در الگوریتم DPLL نیز استفاده می‌شوند. تفاوت بارز الگوریتم DPLL با الگوریتم DP در این است که در الگوریتم DPLL برای غلبه بر مشکل مصرف زیاد حافظه در DP، به‌جای حذف متغیرها از جست‌وجو به روش پیمایش معکوس استفاده می‌شود. روش کار الگوریتم ۲۹ در هر مرحله به‌این‌صورت است که گزاره داده شده ابتدا در خطوط ۵ تا ۷ و ۸ تا ۱۰ با استفاده از قاعده‌های بند منفرد و لیترال محض ساده‌سازی و سپس یک متغیر مثل X از بین مجموعه متغیرهای گزاره انتخاب می‌شود. خط ۱۲ معادل این است که قرار دهیم $X = ۰$، و همه بندهای شامل $\neg X$ را حذف کنیم و سپس همه لیترال‌های $\{X\}$ را نیز از بین لیترال‌های باقی‌مانده حذف کنیم، حذف لیترال‌ها و بندهای مذکور با توجه به فرض $X = ۰$ کاملاً منطقی است. خط ۱۶ معادل این است که فرض کنیم $X = ۱$ و همه بندهای شامل X را حذف کرده و سپس همه لیترال‌های $\{X\}$ را نیز از میان بندهای باقی‌مانده حذف کنیم. روش کار الگوریتم ۲۹ مانند روش جست‌جوی کل فضای حالت است اما مزیت مهم آن نسبت به جست‌وجوی کامل این است که دو قاعده ساده‌سازی که در ابتدای این الگوریتم به‌کار گرفته شده‌اند سبب می‌شوند برخی از حالات بدون این‌که صریحاً بررسی شوند کنار گذاشته شوند. در واقع در این الگوریتم پس از یک ساده‌سازی اولیه یک متغیر از بین متغیرهای گزاره داده شده را انتخاب کرده و یک مقدار (در این‌جا ۰) را برای آن در نظر می‌گیریم. این مرحله از الگوریتم را **مرحله تصمیم** و متغیری که مقدار دهی می‌شود را **متغیر تصمیم** می‌نامیم. پس از تخصیص مقدار متغیر تصمیم، مقدار متناظر با آن را در عبارت جایگذاری کرده و عبارت را ساده‌سازی می‌کنیم و پس از آن مجدداً از دو قاعده بند منفرد و لیترال محض برای ساده‌سازی گزاره به‌دست آمده استفاده می‌کنیم، و در طی این دو فرآیند ساده‌سازی است که به‌خاطر شرایط به‌وجود آمده، ممکن است یک متغیر فقط یک مقدار منطقی را بتواند بپذیرد، به همین دلیل نیازی به تصمیم‌گیری برای مقدار چنین متغیرهایی نیست و به این‌ترتیب به‌جای جست‌وجوی کل فضای حالت، زیرمجموعه‌ای از حالات را بررسی می‌کنیم.

فرآیند مقدار دهی متغیرهای تصمیم و ساده‌سازی تا جایی ادامه می‌یابد که یا به یک گزاره تهی برسیم یا این‌که دیگر هیچ متغیر تصمیم وجود نداشته باشد و هیچ‌یک از قاعده‌های بند منفرد و لیترال محض قادر به ساده‌سازی عبارت به‌دست آمده نباشند. در حالت اول که به گزاره تهی دست می‌یابیم، مسئله صدق‌پذیر است ولی در حالت دوم با یک تناقض مواجه می‌شویم که در این‌صورت

[19] Logemann
[20] Loveland



شکل ۳.۴: گراف الگوریتم DPLL

باید به وضعیت آخرین متغیر تصمیم باز گشته و نقیض مقدار فعلی را برای متغیر تصمیم مورد نظر اختیار کنیم، اگر به ازای این تخصیص نیز به تناقض برسیم یک گام دیگر به عقب بازگشته و مقدار دیگری را برای متغیر تصمیم ماقبل آخر اتخاذ می‌کنیم این کار را تا رسیدن به گزاره تهی یا بررسی تمام متغیرها ادامه می‌دهیم. فرآیند بازگشت به عقب و تجدید نظر در تخصیص مقدار متغیرهای تصمیم را **پیمایش معکوس** می‌نامیم. روشن است که سرعت الگوریتم تا حد زیادی به چگونگی انتخاب متغیر تصمیم در خط ۱۱ و همچنین روش پیمایش معکوس بستگی دارد.

روش کار الگوریتم DPLL را می‌توان با استفاده از یک گراف جهت‌دار فاقد دور نمایش داد، برای مثال فرض کنید گزاره منطقی F به‌صورت زیر داده شده باشد.

$$\begin{aligned} F =&(X_۰ \vee X_۴) \wedge (X_۰ \vee \neg X_۴) \wedge (X_۰ \vee X_۱) \wedge (X_۱ \vee \neg X_۲) \vee (\neg X_۲ \vee X_۳) \wedge \\ &(\neg X_۰ \vee \neg X_۱ \vee \neg X_۲) \wedge (\neg X_۰ \vee \neg X_۲ \vee \neg X_۳) \wedge (X_۲ \vee \neg X_۳) \vee (X_۲ \vee X_۳) \end{aligned} \qquad (۱۲.۴)$$

نمایش مجموعه‌ای صورت متعارف عطفی فوق به‌صورت زیر است.

$$\begin{aligned} F =&\{\{X_۰, X_۴\}, \{X_۰, \neg X_۴\}, \{X_۰, X_۱\}, \{X_۱, \neg X_۲\}, \{\neg X_۲, X_۳\}, \\ &\{\neg X_۰, \neg X_۱, \neg X_۲\}, \{\neg X_۰, \neg X_۲, \neg X_۳\}, \{X_۲, \neg X_۳\}, \{X_۲, X_۳\}\} \end{aligned}$$

با در نظر گرفتن مجموعه فوق به عنوان ورودی الگوریتم ۲۹، گراف شکل ۳.۴ نحوه عمل‌کرد الگوریتم را نشان می‌دهد. در گراف ۳.۴ آن دسته از رئوس میانی که با دایره‌های توپر نمایش داده

شده‌اند، نشان‌دهنده متغیرهای تصمیم، که در خط ۱۱ از الگوریتم ۲۹ انتخاب شده‌اند هستند. رئوسی که با دایره‌های توخالی نمایش داده شده‌اند نشان‌دهنده‌ی متغیرهایی هستند که در یکی از مراحل ساده‌سازی با استفاده از قواعد بند منفرد و لیترال محض مقدار دهی شده‌اند، و همان‌طور که مشاهده می‌شود تنها یک یال از این رئوس خارج شده که نشان می‌دهد تنها یک مقدار برای این متغیرها قابل قبول است و انتخاب دیگری وجود ندارد. مقادیر روی یال‌ها در هر مسیری که از ریشه آغاز و به رأس انتهایی یا برگ ختم می‌شود، نشان‌دهنده یک تخصیص مقادیر برای بخشی از متغیرهای گزاره داده شده هستند. برگ‌ها با مجموعه‌هایی برچسب‌گذاری شده‌اند که به‌ازای تخصیص متناظر با مسیر واصل بین آن برگ و ریشه، سبب بروز تناقض می‌شوند. با توجه به این که همه برگ‌های گراف ۳.۴ شامل یک تناقض هستند لذا گزاره ۱۲.۴ صدق‌پذیر نیست. در راستای بهبود الگوریتم DPLL تلاش‌های زیادی صورت گرفته و یا در حال انجام است، این تلاش‌ها سبب شکل‌گیری خانواده‌ای از الگوریتم‌ها، موسوم به الگوریتم‌های CDCL گشته است که در بخش بعد به‌صورت مختصر به معرفی آن می‌پردازیم.

الگوریتم CDCL

روش CDCL برای حل مسئله صدق‌پذیری، اولین بار به‌طور مشترک توسط سیلوا [21] و ساکالا [22] در الگوریتم GRASP، [۴۸]، [۶۱] معرفی شد. ایده‌های به‌کار رفته در روش CDCL هنوز هم در الگوریتم‌های نوین امروزی مانند MiniSat مورد استفاده قرار می‌گیرد.

همان‌طور که در بخش قبل دیدیم، در الگوریتم DPLL، از عملیات‌های ساده‌سازی بر اساس قاعده‌های بند منفرد و لیترال محض برای کاهش اندازه فضای جست‌وجو استفاده می‌شود و هر بار که به یک تناقض می‌رسیدیم، به آخرین وضعیت تصمیم بازمی‌گشتیم و مقدار دیگری را برای متغیر تصمیم اتخاذ می‌کردیم به این روش بازگشت که پس از هر بار به تناقض رسیدن به آخرین سطح تصمیم بازمی‌گریدم پیمایش معکوس ترتیبی می‌گویند. دو وجه تمایز الگوریتم CDCL نسبت به الگوریتم DPLL که سبب سریع‌تر بودن آن شده است، در پیمایش معکوس غیرترتیبی و همچنین یادگیری گزاره‌های جدید در هر بار به تناقض رسیدن است. قبل از شرح دقیق الگوریتم CDCL برخی مفاهیم مورد نیاز را تعریف می‌کنیم.

با توجه به این‌که هر یک از متغیرهای منطقی در طول الگوریتم می‌توانند یکی از مقادیر $\texttt{true} = 1$ و $\texttt{false} = 0$ را اتخاذ کنند یا این‌که هنوز به مرحله تصمیم‌گیری نرسیده و مقدار نگرفته باشند، نگاشت تخصیص را به‌صورت زیر تعریف می‌کنیم.

تعریف ۵۸.۴. فرض کنید مجموعه متغیرهای مسئله صدق‌پذیری را با V نمایش دهیم، نگاشت تخصیص $\nu : V \to \{0, 1, u\}$، به هر یک از متغیرها یکی از مقادیر $\{0, 1\}$ یا u به معنای مقدار

[21] Marques Silva

[22] Karem A. Sakallah

نگرفته، را تخصیص می‌دهد. اگر همه متغیرها فقط یکی از مقادیر ۰ یا ۱ را اتخاذ کرده باشند، ν را تخصیص کامل و در غیر این‌صورت آن‌را تخصیص جزیی می‌گوییم.

مقداری که لیترال l، بند C و گزاره‌ی F به‌ازای تخصیص ν اختیار می‌کنند را به‌ترتیب با نمادهای l^ν، C^ν و F^ν نمایش می‌دهیم. ترتیب $0 < u < 1$ را روی $\{0, u, 1\}$ در نظر بگیرید و فرض کنید $u = 1 - u$، در این‌صورت داریم

$$l^\nu = \begin{cases} \nu(X_i) & l = X_i \\ 1 - \nu(X_i) & l = \neg X_i \end{cases} \quad C^\nu = \max\{l^\nu : \ l \in C\} \quad F^\nu = \min\{C^\nu : \ C \in F\}.$$

نگاشت تخصیص ν را می‌توانیم به‌صورت مجموعه‌ای از زوج‌های مرتب (X_i, ν_i) که X_i یک متغیر و $\nu_i \in \{0, 1, \}$ است نیز در نظر بگیریم. به این ترتیب افزودن زوج (X_i, ν_i) به مجموعه متناظر با ν معادل است با تخصیص ν_i به X_i به‌طوری که $\nu(X_i) = \nu_i$. به همین ترتیب حذف زوج (X_i, ν_i) به‌طوری که $\nu(X_i) \neq u$ معادل است با تخصیص u به X_i.

در بخش قبل بند منفرد را بندی در نظر گرفتیم که فقط شامل یک لیترال بود، به‌طور مشابه در این بخش بند منفرد را با توجه به تعریف نگاشت تخصیص در فوق، بندی در نظر می‌گیریم که غیر از یک لیترال آن که هنوز مقدار نگرفته و تصویر آن تحت نگاشت تخصیص برابر u است، مقدار سایر لیترال‌های آن تحت نگاشت تصویر ۰ باشد. به همین ترتیب بندی را که همه لیترال‌های آن مقدار ۰ را اختیار کرده باشند بند ناصادق و بندی را که حداقل یکی از لیترال‌هایش مقدار ۱ را اختیار کرده باشند بند صادق می‌گوییم. بندهایی را که نه صادق باشند و نه ناصادق، بندهای حل‌نشده می‌گوییم.

روشن است که تنها مقدار مجاز برای لیترال مقدار نگرفته در یک بند منفرد مقدار ۱ است. تکرار فرآیند تخصیص مقدار ۱ به لیترال مقدار نگرفته در بندهای منفرد و سپس ساده‌سازی گزاره منطقی با استفاده از قاعده بندهای منفرد تا رسیدن به یکی از حالات عدم وجود بند منفرد یا تناقض را قاعده انتشار واحد می‌گوییم. قاعده انتشار واحد بعد از هر تصمیم‌گیری یا انشعاب و همچنین در مرحله‌ی پیش‌پردازش، در CDCL صورت می‌گیرد و اگر طی این فرآیند به تناقض برسیم، پیمایش معکوس خواهیم داشت.

در الگوریتم CDCL هر متغیر دارای مشخصه‌هایی است که در ادامه آن‌ها را تعریف می‌کنیم. متغیر X_i از مسئله صدق‌پذیری داده شده F را در نظر بگیرد، اولین مشخصه این متغیر مقدار آن است که آن‌را با $\nu(X_i) \in \{0, u, 1\}$ نمایش می‌دهیم. همان‌طور که در الگوریتم DPLL نیز دیدیم برخی از متغیرها بدون نیاز به تصمیم‌گیری راجع به مقدار آن‌ها، به‌طور ضمنی و طی فرآیند انتشار واحد یک مقدار اتخاذ می‌کنند، به چنین متغیرهایی متغیرهای مقدر و به بندی که برای استنباط مقدار این متغیر مورد استفاده قرار گرفته است، بند مقدم متغیر X_i می‌گوییم. مشخصه

دوم متغیر X_i را که نشان‌دهنده بند مقدم X_i است را با $\alpha(X_i) \in F \cup \{NIL\}$ نمایش می‌دهیم. برای متغیرهایی که متغیر مقدر نبوده و جزو متغیرهای تصمیم بوده و یا هنوز مقادیر ۰ یا ۱ اتخاذ نکرده‌اند، قرار می‌دهیم $\alpha(X_i) = NIL$. مشخصه سوم متغیر Xi که سطح تصمیم نام‌دارد کمیتی است که عمق درخت تصمیم‌گیری متناظر با الگوریتم CDCL را در جایی که برای مقدار X_i تصمیم‌گیری می‌شود، نشان می‌دهد. سطح تصمیم متغیر X_i را با $\delta(X_i)$ نمایش می‌دهیم. برای متغیر X_i که هنوز یکی از مقادیر ۰ یا ۱ را اختیار نکرده است قرار می‌دهیم $\delta(X_i) = -۱$ و سطح تصمیم متغیرهای مقدر را به صورت زیر تعریف می‌کنیم.

$$\delta(X_i) := \max(\{۰\} \cup \{\delta(X_j)|\ X_j \in C\ \wedge\ X_j \neq X_i\}) \quad s.t \quad \alpha(X_i) = C.$$

با توجه به تعریف فوق برای هر متغیر تصمیم مثل X_i داریم $\alpha(X_i) = NIL$ و $\delta(X_i) > ۰$. برای ساده‌نویسی و اختصار، از نمادگذاری $X_i = v@d$ برای نمایش مقدار و سطح تصمیم متغیر X_i استفاده می‌کنیم، به‌طوری که $\nu(X_i) = v$ و $\delta(X_i) = d$. گاهی به‌جای سطح تصمیم یک متغیر از سطح تصمیم یک لیترال سخن به‌میان می‌آید که در این‌صورت منظور همان سطح تصمیم متغیر متناظر با آن لیترال خواهد بود.

مثال ۵۹.۴. گزاره منطقی زیر را در نظر بگیرید

$$F = C_۱ \wedge C_۲ \wedge C_۳ = (X_۱ \vee \neg X_۴) \wedge (X_۱ \vee X_۳) \wedge (\neg X_۳ \vee X_۲ \vee X_۴).$$

فرض کنید تصمیم‌گیری را با $X_۴$ آغاز کنیم به طوری که $X_۴ = ۰@۱$. در این‌صورت قاعده انتشار واحد تخصیص جدید را نتیجه نمی‌دهد و ناچاریم یک تصمیم‌گیری دیگر انجام دهیم. فرض کنید در تصمیم‌گیری دوم قرار دهیم $X_۲ = ۰@۲$. این‌بار قاعده انتشار واحد سبب می‌شود داشته باشیم $X_۳ = ۱@۲$ و در نتیجه $X_۲ = ۱@۲$. در ضمن داریم $\alpha(X_۱) = NIL, \alpha(X_۲) = C_۳, \alpha(X_۳) = C_۲$ و $\alpha(X_۴) = NIL$.

متغیرهای مقدار دهی شده و بندهای مقدم متناظر با هر یک از آن‌ها در الگوریتم CDCL، گرافی جهت‌دار و فاقد دور موسوم به گراف التزام را تشکیل می‌دهند، که با $I = (V_I, E_I)$ نمایش می‌دهیم و به‌صورت زیر تعریف می‌شود.

تعریف ۶۰.۴ (گراف التزام)**.** مجموعه رئوس گراف التزام $I = (V_I, E_I)$ عبارت‌است از مجموعه همه متغیرهای مقداردهی شده به‌انضمام گره κ، به‌این ترتیب $V_I \subseteq X \cup \{\kappa\}$. مجموعه یال‌های این گراف نیز از بندهای مقدم هر یک از متغیرهای مقدار دهی‌شده به‌دست می‌آید به‌طوری که اگر $C = \alpha(X_i)$، آن‌گاه به ازای هر متغیر به‌کار رفته در بند C به جز X_i یک یال جهت‌دار از آن متغیر به X_i وجود خواهد داشت. در ضمن اگر قاعده انتشار واحد یک بند ناصادق مثل C_j را نتیجه

دهد، در این‌صورت متناظر با این گزاره ناصادق یک رأس که با κ نشان‌داده می‌شود را به گراف اضافه می‌کنیم و قرار می‌دهیم $\alpha(\kappa) = C_j$.

مجموعه رئوس گرف التزام را به‌صورت دقیق تعریف کردیم اما برای تعریف دقیق مجموعه یال‌ها به چند مفهوم جدید نیاز است که در ادامه با آن‌ها آشنا می‌شویم. اولین تابعی که تعریف می‌کنیم $\lambda(Z,C)$ است که برای آزمودن عضویت لیترال شامل متغیر Z در بند C به‌کار می‌رود. به‌عبارت اگر فرض کنیم F یک گزاره منطقی باشد که به صورت مجموعه‌ای نمایش داده شده، به‌ازای بند $C \in F$ و متغیر $Z \in V_I$ داریم

$$\lambda(Z,C) := \begin{cases} ۱ & Z \in C \ \vee \ \neg Z \in C \\ ۰ & \text{در غیر این‌صورت} \end{cases}$$

تابع دیگری که با $\nu_۰(Z,C)$ نشان می‌دهیم برای آزمودن صفر بودن مقدار متناظر با لیترال شامل Z در بند C به‌کار می‌رود و به‌صورت زیر تعریف می‌شود.

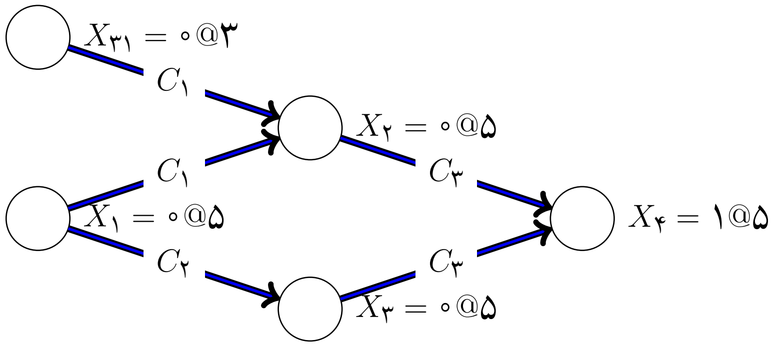
$$\nu_۰(Z,C) := \begin{cases} ۱ & \lambda(Z,C) \ \wedge \ Z \in C \ \wedge \ \nu(Z) = ۰ \\ ۱ & \lambda(Z,C) \ \wedge \ \neg Z \in C \ \wedge \ \nu(Z) = ۱ \\ ۰ & \text{در غیر این‌صورت} \end{cases}$$

تابع $\nu_۱(Z,C)$ نیز برای آزمودن ۱ بودن مقدار متناظر با لیترال شامل Z در بند C به‌کار می‌رود. به‌عبارت دیگر داریم

$$\nu_۱(Z,C) := \begin{cases} ۱ & \lambda(Z,C) \ \wedge \ Z \in C \ \wedge \ \nu(Z) = ۱ \\ ۱ & \lambda(Z,C) \ \wedge \ \neg Z \in C \ \wedge \ \nu(Z) = ۰ \\ ۰ & \text{در غیر این‌صورت} \end{cases}$$

در نهایت به‌ازای $Z_۱, Z_۲ \in V_I$ تابع $\epsilon(Z_۱, Z_۲)$ را به‌صورت زیر تعریف می‌کنیم که نشان‌دهنده وجود یال جهت‌دار از $Z_۱$ به‌سوی $Z_۲$ است.

$$\epsilon(Z_۱, Z_۲) := \begin{cases} ۱ & Z_۲ = \kappa \ \wedge \ \lambda(Z_۱, \alpha(\kappa)) \\ ۱ & Z_۲ \neq \kappa \ \wedge \ \alpha(Z_۲) = C \ \wedge \ \nu_۰(Z_۱, C) \ \wedge \ \nu_۱(Z_۲, C) \\ ۰ & \text{در غیر این‌صورت} \end{cases}$$



شکل ۴.۴: یک نمونه گراف التزام

در نتیجه مجموعه یال‌های گراف التزام I به صورت زیر تعریف می‌شود

$$E_I := \{(Z_۱, Z_۲)|\ \epsilon(Z_۱, Z_۲) = ۱\}.$$

در ضمن هر یال از $Z_۱$ به $Z_۲$ را با $\alpha(Z_۲)$ برچسب‌گذاری می‌کنیم.

مثال ۶۱.۴. گزاره منطقی زیر را در نظر بگیرید

$$F = C_۱ \wedge C_۲ \wedge C_۳ = (X_۱ \vee X_{۳۱} \vee \neg X_۲) \wedge (X_۱ \vee \neg X_۳) \wedge (X_۲ \vee X_۳ \vee X_۴).$$

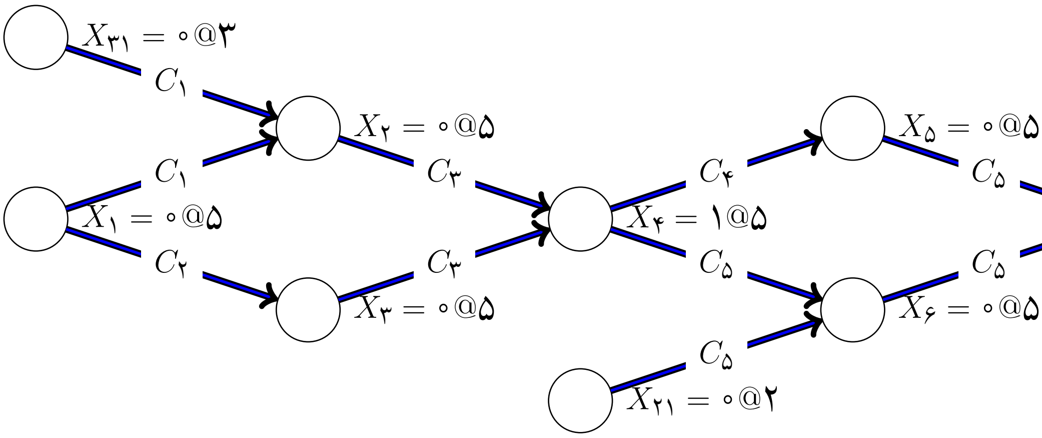
فرض کنید داشته باشیم $X_{۳۱} = ۰@۳$. در ضمن فرض کنید سطح تصمیم فعلی برابر ۵ باشد و در تصمیم‌گیری راجع به مقدار متغیر تصمیم $X_۱$ قرار دهیم $X_۱ = ۰@۵$، در این‌صورت گراف التزام به‌صورت نشان داده شده در شکل ۴.۴ است. در مثال قبل قاعده انتشار واحد سبب به‌وجود آمدن بند ناصادق نشد، در مثال بعد نمونه‌ای را خواهیم دید که نشان می‌دهد نتیجه قاعده انتشار واحد ممکن است یک گزاره ناصادق باشد.

مثال ۶۲.۴. گزاره منطقی زیر که دارای صورت متعارف عطفی است را در نظر بگیرید.

$$\begin{aligned} F =& C_۱ \wedge C_۲ \wedge C_۳ \wedge C_۴ \wedge C_۵ \wedge C_۶ \\ =&(X۱ \vee X_{۳۱} \vee \neg X_۲) \wedge (X_۱ \vee \neg X_۳) \wedge (X_۲ \vee X_۳ \vee X_۴) \wedge \\ &(\neg X_۴ \vee \neg X_۵) \wedge (X_{۲۱} \vee \neg X_۴ \vee \neg X_۶) \wedge (X_۵ \vee X_۶). \end{aligned}$$

فرض کنید داشته باشیم $X_{۲۱} = ۰@۲, X_{۳۱} = ۰@۳$. همچنین فرض کنید در سطح تصمیم فعلی که برابر ۵ است برای متغیر تصمیم $X_۱$ قرار دهیم $X_۱ = ۰@۵$. در این‌صورت گراف التزام متناظر با این تصمیم در شکل ۵.۴ نشان داده شده است و همان‌طور که مشاهده می‌شود این تصمیم سبب می‌شود مقدار بند $X_۵ \vee X_۶$ برابر ۰ باشد و یک بند ناصادق داشته باشیم.

همان‌طور که قبلاً ذکر شد، مهم‌ترین وجه تمایز الگوریتم CDCL نسبت به الگوریتم DPLL



شکل ۵.۴: یک نمونه گراف التزام

وجود فرآیند یادگیری بند جدید در هر بار به تناقض رسیدن و پیمایش معکوس غیرترتیبی بعد از آن است. فرض کنید F یک گزاره منطقی و ν تخصیصی باشد که برای متغیرهای منطقی آن در نظر گرفته‌ایم. در الگوریتم CDCL پس از هربار تصمیم‌گیری راجع به مقدار یک متغیر تصمیم فرآیند انتشار واحد تاجایی که امکان دارد انجام می‌شود (یعنی تا جایی که به بند ناصادق برسیم یا دیگر هیچ بند منفردی وجود نداشته باشد.) اگر این فرآیند یک بند ناصادق را نتیجه دهد و سبب بروز یک تناقض شود، الگوریتم CDCL زیربرنامه‌ای را فراخوانی می‌کند که قادر است با تحلیل تناقض راخ‌داده شده، علت بروز تناقض را در قالب یک یا چند بند جدید بازگرداند. این زیربرنامه را با $\texttt{AnalyzeConflict}(F,\nu)$ نمایش می‌دهیم و به بندهای جدیدی که این زیربرنامه به‌دست می‌آورد بندهای آموخته شده می‌گوییم. علاوه بر این AnalyzeConflict، سطح تصمیم برای آغاز فرآیند پیمایش معکوس غیرترتیبی را مشخص می‌کند، تا بدین وسیله مشخص شود که بعد از رسیدن به تناقض، الگوریتم از کدام سطح تصمیم از سرگیری شود. برای تشریح دقیق الگوریتم AnalyzeConflict لازم است تا چند مفهوم جدید را تعریف کنیم.

فرآیند یادگیری بند جدید با شروع از بند ناصادق به‌دست آمده از قاعده انتشار واحد آغاز می‌شود، یعنی همان بندی که در گراف التزام با κ نمایش داده می‌شود، زیربرنامه AnalyzeConflict همه متغیرهای مقدر که سطح تصمیمشان برابر سطح تصمیم فعلی (یعنی بزرگ‌ترین سطح تصمیم موجود) است را بررسی کرده و بندهای مقدم این متغیرها را شناسایی می‌کند، سپس لیترال‌هایی از این بندها که سطح تصمیم آن‌ها کم‌تر از سطح تصمیم فعلی است را به بندی که قرار است آموخته شود اضافه می‌کند. این فرآیند تا زمانی ادامه می‌یابد که آخرین متغیر تصمیم در گزاره آموخته شده ظاهر شود. با توجه به این که سطح تصمیم متناظر با متغیرهای مقدار دهی شده یک ترتیب جزیی روی مجموعه این متغیرها القا می‌کند، منظور از آخرین متغیر تصمیم، متغیر تصمیم با بزرگ‌ترین سطح تصمیم است. اما توصیف فوق یک توصیف کیفی است، برای بیان ریاضی

$\{l:\ l\in\alpha(\kappa)\}$	$C_L^{۵,۰}=\{X_۵,X_۶\}$
$C_L^{۵,۰}\odot\alpha(X_۵)$	$C_L^{۵,۱}=\{\neg X_۴,X_۶\}$
$C_L^{۵,۱}\odot\alpha(X_۶)$	$C_L^{۵,۲}=\{\neg X_۴,X_{۲۱}\}$
$C_L^{۵,۲}\odot\alpha(X_۴)$	$C_L^{۵,۳}=\{X_۲,X_۳,X_{۲۱}\}$
$C_L^{۵,۳}\odot\alpha(X_۲)$	$C_L^{۵,۴}=\{X_۱,X_{۳۱},X_۳,X۲۱\}$
$C_L^{۵,۴}\odot\alpha(X_۳)$	$C_L^{۵,۵}=\{X_۱,X_{۳۱},X_{۲۱}\}$
توقف	$C_L^{۵,۶}=\{X_۱,X_{۳۱},X_{۲۱}\}$

جدول ۵.۴: عملیات رفع طی فرآیند یادگیری بند جدید در الگوریتم AnalyzeConflict

فرآیند یادگیری تابع ξ را به‌ازای سطح تصمیم d و لیترال l و بند C به‌صورت زیر تعریف می‌کنیم.

$$\xi(C,l,d):=\begin{cases} ۱ & l\in C\ \wedge\ \delta(l)=d\ \wedge\ \alpha(l)\neq NIL \\ ۰ & \text{در غیر این‌صورت} \end{cases}$$

با توجه به تعریف فوق بند $C_L^{d,i}$ که $i=۰,۱,\ldots$ را به‌صورت زیر تعریف می‌کنیم

$$C_L^{d,i}:=\begin{cases} \alpha(\kappa) & i=۰ \\ C_L^{d,i-۱}\odot\alpha(l) & i\neq ۰\ \wedge\ \xi(C_L^{d,i-۱},l,d)=۱ \\ C_L^{d,i-۱} & i\neq ۰\ \wedge\ \forall\, l:\ \xi(C_L^{d,i-۱},l,d)=۰. \end{cases} \qquad (۱۳.۴)$$

اکنون می‌توانیم AnalyzeConflict را به‌صورت الگوریتم ۳۰ بیان کنیم.

مثال ۶۳.۴. گزاره مثال ۶۲.۴ را در نظر بگیرید، بندهای میانی حاصل از عمل رفع که بواسطه اعمال الگوریتم AnalyzeConflict [۳۰] روی بند ناصادق حاصل‌شده در این مثال به‌دست آمده‌اند، در جدول ۵.۴ نمایش داده شده است. در نتیجه بند آموخته شده عبارت است از $\{X_۱,X_{۳۱},X_{۲۱}\}$.

حال فرض کنید نمایش مجموعه‌ای صورت متعارف عطفی F به همراه یک تخصیص اولیه (که می‌تواند تهی باشد) داده شده باشد. فرض کنید فرآیند انتشار واحد توسط تابع UnitPrppagation(F,ν) انجام شود. این تابع تا جایی که ممکن است عملیات انتشار واحد را انجام می‌دهد و در صورتی که به بند ناصادق برسد CONFLICT و در غیر این‌صورت مقدار NoCONFLICT را بازمی‌گرداند. فرض کنید فرآیند انتخاب متغیر تصمیم و مقدار دهی آن توسط تابع PickBrnchingVariable(F,ν) انجام شود که خروجی آن متغیر تصمیم و مقدار متناظر با آن است. علاوه بر این فرض کنید که فرآیند حصول اطمینان از مقدار دهی شدن همه متغیرهای گزاره F توسط تابع AllVariablesAssigned انجام شود به‌طوری که اگر همه متغیرها مقدار دهی شده بودند true و در غیر این‌صورت false را تولید کند. همچنین فرض کنید عملیات پیمایش معکوس غیرترتیبی توسط تابع BacktrackTo انجام شود، این تابع با دریافت (F,ν,β) وضعیت را به حالتی که در سطح تصمیم β وجود داشته

است بازمی‌گرداند با این تفاوت که بند آموخته شده C_L نیز به F افزوده می‌شود. به این‌ترتیب الگوریتم ۳۱ با توجه به فرضیات فوق، ساختار متعارف یک الگوریتم CDCL را نشان می‌دهد.

با توجه به خط ۱۱ الگوریتم ۳۰ روشن می‌شود که الگوریتم دومین سطح تصمیم از لحاظ بزرگی در بین سطوح تصمیم بند آموخته شده را برای بازگشت انتخاب می‌کند. دلیل انتخاب چنین سطح تصمیمی، این است که بازگشتن به این سطح تصمیم، در حالی که گزاره آموخته شده به مجموعه بندهای گزاره اولیه اضافه شده است، سبب می‌شود که با از سرگیری مجدد، قبل از رخ‌دادن مجدد تناقض قبلی، آخرین متغیر تصمیم در مرحله فعلی که تصمیم‌گیری راجع به مقدار آن منجر به به‌وجود آمدن بند ناصادق شده است، طی فرآیند `UnitPrppagation` به یک متغیر مقدر تبدیل شده و مقداری صحیح اتخاذ کند. برای مثال فرض کنید بند جدید $C_L = \{l_۱, l_۲, l_۳, l_۴\}$ بند آموخته شده باشد و داشته باشیم $\delta(l_۱) = ۱۰, \delta(l_۲) = ۲۰, \delta(l_۳) = ۳۰$ و $\delta(l_۴) = ۴۰$ در این‌صورت الگوریتم به سطح تصمیم ۳۰ بازخواهد گشت. با بازگشت به این سطح تصمیم لیترال $l_۴$ که در سطح تصمیم ۴۰ مقدار دهی شده بود اکنون بدون مقدار است و بند C_L در این موقعیت یک بند منفرد محسوب شده و سبب تحریک تابع `UnitPrppagation` می‌شود. به این ترتیب با از سرگیری الگوریتم از سطح تصمیم ۳۰ لیترال $l_۴$ قبل از رسیدن به سطح تصمیم ۴۰ در همان سطح تصمیم ۳۰ به‌واسطه قاعده انتشار واحد مقدار دهی می‌شود، به همین دلیل است که الگوریتم CDCL در فرآیند پیمایش معکوس غیرترتیبی به دومین سطح تصمیم متغیرهای موجود در بند آموخته شده بازمی‌گردد.

یک روش بهبود الگوریتم `AnalyzeConflict` و کاهش اندازه بند آموخته شده استفاده از نقاط التزام واحد است. نقطه التزام واحد نقطه‌ای از گراف التزام است که همه مسیرها از آخرین متغیر تصمیم به سمت κ از آن نقطه عبور کند. برای مثال $X_۴$ در شکل ۵.۴ یک نقطه التزام واحد است. در واقع عملیات رفع در الگوریتم `AnalyzeConflict` برای یادگیری بند جدید را تا جایی ادامه می‌دهیم که بند به‌دست آمده از رفع دو بند قبلی فقط شامل یک متغیر با سطح تصمیم فعلی باشد. به عبارت دیگر کافی است معادله ۱۳.۴ را به‌صورت زیر اصلاح کنیم.

$$C_L^{d,i} := \begin{cases} \alpha(\kappa) & i = ۰ \\ C_L^{d,i-۱} \odot \alpha(l) & i \neq ۰ \ \wedge \ \xi(C_L^{d,i-۱}, l, d) = ۱ \\ C_L^{d,i-۱} & i \neq ۰ \ \wedge \ \sigma(C_L^{d,i-۱}, d) = ۱. \end{cases} \qquad (۱۴.۴)$$

که $\sigma(C, d) = |\{l \in C | \ \delta(l) = d\}|$. به روش یادگیری بند جدید با استفاده از معادله ۱۴.۴، روش اولین نقطه التزام واحد یا 1s-UIP گفته می‌شود.

مثال ۶۴.۴. گزاره منطقی مثال ۶۲.۴ را در نظر بگیرید، فرآیند یادگیری گزاره بر اساس معادله ۱۳.۴ بند جدید $(X_۱ \vee X_{۳۱} \vee X_{۲۱})$ را نتیجه داد. اکنون با توجه به این‌که $X_۴ = ۱@۵$ یک نقطه التزام واحد است، فرآیند یادگیری بر اساس معادله ۱۴.۴ منجر به یادگیری بند کوچک‌تر $(\neg X_۴ \vee X_{۲۱})$

$\{l \mid l \in \alpha(\kappa)\}$	$C_L^{۵,۰} = \{X_۵, X_۶\}$
$C_L^{۵,۰} \odot \alpha(X_۵)$	$C_L^{۵,۱} = \{\neg X_۴, X_۶\}$
توقف	$C_L^{۵,۲} = \{\neg X_۴, X_{۲۱}\}$

جدول ۶.۴: عملیات رفع طی فرآیند یادگیری بند جدید با استفاده از روش نقطه التزام واحد

می‌شود. بندهای میانی به‌دست آمده طی انجام فرآیند یادگیری در الگوریتم AnalyzeConflict در جدول ۶.۴ نمایش داده شده است.

۲.۵.۴ تبدیل مسئله حل دستگاه معادلات چندجمله‌ای به مسئله صدق‌پذیری

فرآیند تبدیل یک دستگاه معادلات چندجمله‌ای به مسئله صدق‌پذیری شامل سه مرحله اصلی به شرح زیر است.

۱. ابتدا به ازای هر یکجمله‌ای غیرخطی یک متغیر جدید معرفی می‌کنیم. هر تغییر متغیر به این صورت، به یک گزاره‌ی منطقی جدید در مسئله صدق‌پذیری منجر خواهد شد که نحوه‌ی به‌دست آوردن آن را در ادامه شرح می‌دهیم.

۲. پس از تغییر متغیر در گام ۱ دستگاه چندجمله‌ای به یک دستگاه خطی تبدیل می‌شود که هر یک از معادلات خطی آن مجموع تعدادی از متغیرهای خطی است. در این مرحله پارامتری تحت عنوان *عدد برشی* که با c نمایش می‌دهیم را مقداردهی می‌کنیم و سپس به ازای هر c متغیرخطی که با هم جمع‌شده‌اند یک متغیر جدید دیگر معرفی می‌کنیم تا حاصل‌جمع‌های طولانی به حاصل‌جمع‌هایی با طول کمتر تبدیل شوند و همان‌طور که در ادامه خواهیم دید این کار سبب تسهیل فرآیند تبدیل دستگاه معادلات به مسئله صدق‌پذیری خواهد شد.

۳. در گام پایانی گزاره منطقی به‌صورت ترکیب عطفی استاندارد معادل با هر یک از حاصل‌جمع‌های کوچکتر به‌دست آمده در گام ۲ را می‌یابیم.

تعریف زیر رابطه بین صفرهای یک چندجمله‌ای از حلقه $\mathbb{F}[x_۱, ..., x_n]$، و مقدار گزاره منطقی معادل با آن را بیان می‌کند.

تعریف ۶۵.۴. چندجمله‌ای $f \in \mathbb{F}[x_۱, ..., x_n]$ را در نظر بگیرید. گزاره منطقی $F \in \widehat{X}$، را *نمایش منطقی* چندجمله‌ای f گوییم هر گاه به ازای هر $a = (a_۱, ..., a_n) \in \mathbb{F}_۲^n$ داشته باشیم $\varphi_a(F) = f(a_۱, ..., f_n) + ۱$ که φ_a نشان‌دهنده‌ی مقدار بولی F در نقطه‌ی a و با در نظر گرفتن $\mathtt{true} = ۱$ و $\mathtt{false} = ۰$ است.

روشن است که اگر F نمایش منطقی چندجمله‌ای f باشد، آن‌گاه هر چندتایی از مقادیر منطقی که گزاره منطقی F را ارضا کند، به طور یکتا متناظر با یک صفر چندجمله‌ای f خواهد بود. اگر نمایش منطقی متناظر با یک چندجمله‌ای را داشته باشیم، لم بعد روشی برای به‌دست آوردن نمایش منطقی عبارت حاصل از افزودن یک متغیر جدید به عبارت مورد نظر را به‌دست می‌دهد.

لم ۴.۶۶. فرض کنید $f \in \mathbb{F}[x_۱, ..., x_n]$ یک چندجمله‌ای باشد و $F \in \widehat{X}$ نمایش منطقی آن باشد. اگر y یک متغیر میدان $\mathbb{F}_۲$ و Y متغیر بولی متناظر با آن باشد، آن‌گاه نمایش منطقی چندجمله‌ای $g = f + y$ عبارت است از $G = (\neg F \iff Y)$.

برهان. فرض کنید $\bar{a} = (a_۱, ..., a_n, b) \in \mathbb{F}_۲^{n+۱}$. به ازای مقادیری که b اتخاذ می‌کند دو حالت به‌صورت زیر داریم.

۱. اگر $b = ۱$ آن‌گاه $g(\bar{a}) = f(a) + ۱ = \phi_a(F)$. از طرفی $\varphi_b(Y) = ۱$ و در نتیجه $\varphi_{\bar{a}}(\neg F \iff Y) = \varphi_a(\neg F) = \varphi_a(F) + ۱$.

۲. اگر $b = ۰$ آن‌گاه $g(\bar{a}) = f(a) = \varphi_a(F) + ۱$ و $\varphi_b(Y) = ۰$ که نتیجه می‌دهد

$$\varphi_{\bar{a}}(\neg F \iff Y) = \varphi_a(F).$$

همان‌طور که مشاهده می‌شود در هر دو حالت فوق داریم $\varphi_{\bar{a}}(\neg F \iff Y) = g(\bar{a}) + ۱$ و حکم ثابت می‌شود. □

لم ۴.۶۷. فرض کنید $f \in \mathbb{F}[x_۱, ..., x_n, y]$ یک چندجمله‌ای به صورت $f = l_۱ \cdots l_s + y$ باشد که $۱ \leq s \leq n$ و به ازای هر $i = ۱, ..., s$ داریم $l_i \in \{x_i, x_{i+۱}\}$. در ضمن اگر $l_i = x_i$، آن‌گاه نمایش منطقی x_i را به‌صورت $L_i = X_i$، در نظر می‌گیریم. به‌همین ترتیب اگر $l_i = x_i + ۱$، قرار می‌دهیم $L_i = \neg X_i$. در این‌صورت

$$F = (\neg Y \vee L_۱) \wedge \cdots \wedge (\neg Y \vee L_s) \wedge (Y \vee \neg L_۱ \vee \cdots \vee \neg L_s)$$

نمایش منطقی f خواهد بود و به این ترتیب F یک گزاره‌ی منطقی به صورت ترکیب عطفی استاندارد و شامل $s + ۱$ بند یا گزاره کوچکتر است که هر یک از آن‌ها ترکیب فصلی از متغیرهای منطقی هستند.

برهان. فرض کنید $a = (a_۱, ..., a_n, b) \in \mathbb{F}_۲^{n+۱}$. با استقرا روی s حکم را ثابت می‌کنیم. در حالتی که $s = ۱$ ، داریم $f = x_۱ + y + c$ که $c \in \{۰, ۱\}$، در این‌صورت $F = (\neg Y \vee L_۱) \wedge (Y \vee \neg L_۱)$ که اگر $c = ۰$ آن‌گاه $L_۱ = X_۱$، و اگر $c = ۱$ داریم $L_۱ = \neg X_۱$. درستی رابطه‌ی $\varphi_a(F) = f(a) + ۱$ به‌راحتی با استفاده از جدول ارزش تابع منطقی F قابل تحقیق است.

اکنون فرض کنید حکم به ازای $s-۱$ برقرار باشد و F' نمایش منطقی $f' = l_۱ \cdots l_{s-۱} + y$ به‌صورت بیان شده در صورت قضیه باشد. فرض کنید $l_s = x_s$، در این‌صورت دو حالت زیر وجود دارد.

۱. اگر $a_s = ۰$، داریم $\varphi_a(F) = \varphi_a((\neg Y \vee L_۱) \wedge \cdots \wedge (\neg Y \vee L_{s-۱}) \wedge \neg Y) = \varphi_b(\neg Y)$ و $f(a) = b$ که نتیجه می‌دهد $\varphi_a(F) = f(a) + ۱$.

۲. اگر $a_s = ۱$، آن‌گاه $f(a) = f'(a)$. با توجه به این‌که $\varphi_a(L_s) = ۱$، به‌دست می‌آوریم

$$\varphi_a(F) = \varphi_a((\neg Y \vee L_۱) \wedge \cdots \wedge (\neg Y \vee L_{s-۱}) \wedge (Y \vee \neg L_۱ \vee \cdots \vee \neg L_{s-۱})) = \varphi_a(F').$$

بنابراین طبق فرض استقرا $\varphi_a(F) = \varphi_a(F') = f'(a) + ۱ = f(a) + ۱$.

در حالتی که $l_s = x_s + ۱$ نیز اثبات به‌طور مشابه صورت می‌گیرد و لذا حکم استقرا به ازای هر $s \in \{۱, ..., n\}$ برقرار است. □

با استفاده از دو لم قبل، سه روش برای تبدیل دستگاه معادلات چندجمله‌ای روی $\mathbb{F}_۲$ به یک دستگاه خطی و سپس تبدیل آن به یک گزاره منطقی به‌صورت ترکیب عطفی استاندارد معرفی می‌کنیم. لازم به ذکر است که روش‌های معرفی شده برای خطی‌سازی دستگاه معادلات چندجمله‌ای، در تبدیل مسئله حل دستگاه چندجمله‌ای به مسئله برنامه‌ریزی با عدد صحیح نیز کارایی دارند.

تعریف ۶۸.۴. فرض کنید $f \in \mathbb{F}[x_۱, ..., x_n]$ یک چندجمله‌ای باشد.

۱. **روش استاندارد:** به ازای هر جمله غیرخطی مثل $t \in \mathrm{Supp}(f)$، متغیر جدید y و متغیر بولی متناظر با آن Y را معرفی می‌کنیم. y را جایگزین t در f کرده و گزاره منطقی متناظر با $t + y$ را که مطابق لم ۶۷.۴ به‌دست می‌آید، به مجموعه گزاره‌های منطقی در ترکیب عطفی استاندارد، اضافه می‌کنیم.

۲. **روش شریک خطی:** فرض کنید $\deg(f) = ۲$، ابتدا عباراتی به صورت $x_i x_j + x_i$ از چندجمله‌ای f را یافته، سپس به ازای هر یک از این عبارات متغیر جدید y و متغیر بولی متناظر با آن یعنی Y را معرفی می‌کنیم. y را جایگزین عبارت $x_i x_j + x_i$، در چندجمله‌ای f کرده و سپس گزاره منطقی متناظر با $x_i(x_j + ۱) + y$ را که طبق لم ۶۷.۴ به‌دست می‌آید، به مجموعه گزاره‌های منطقی در ترکیب عطفی استاندارد اضافه می‌کنیم.

۳. **روش شریک خطی مضاعف:** فرض کنید $\deg(f) = ۲$ ، ابتدا به ازای هر عبارات به‌صورت $x_i x_j + x_i + x_j + ۱$ در f، متغیر جدید y و متغیر بولی متناظر با آن یعنی Y را معرفی می‌کنیم. سپس y را جایگزین عبارت $x_i x_j + x_i + x_j + ۱$ در f کرده و گزاره منطقی متناظر با

روش	روش استاندارد (SS)				روش شریک خطی (LPS)				روش شریک خطی مضاعف (DPS)			
عدد برشی	۳	۴	۵	۶	۳	۴	۵	۶	۳	۴	۵	۶
$\#v$	۱۰	۸	۸	۷	۸	۷	۷	۷	۷	۷	۷	۷
$\#c$	۲۶	۲۶	۳۰	۴۲	۱۸	۱۸	۱۸	۱۸	۱۴	۱۴	۱۴	۱۴

جدول ۷.۴: مقایسه تعداد بندها و متغیرهای صورت متعارف عطفی به‌دست آمده در روش‌های مختلف تبدیل دستگاه چندجمله‌ای به CNF

$(x_i+1)(x_j+1)+y$ که طبق لم ۶۷.۴ به‌دست می‌آید را به مجموعه گزاره‌های منطقی در ترکیب عطفی استاندارد اضافه می‌کنیم.

در مثال بعد، سه روش تبدیل تعریف شده در فوق را با هم مقایسه کرده‌ایم.

مثال ۶۹.۴. چندجمله‌ای $f = x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_2 + 1$ از حلقه $\mathbb{F}_2[x_1, x_2, x_3]$ را در نظر بگیرید. فرض کنید تعداد متغیرهای منطقی در گزاره منطقی CNF متناظر با f را با $\#c$ و تعداد بندها یا ترکیب‌های فصلی ظاهر شده در صورت متعارف عطفی یا CNF به‌دست آمده را با $\#v$ نمایش دهیم. جدول ۷.۴ تعداد بندها و متغیرها در ترکیب عطفی استاندارد متناظر با f را نمایش می‌دهد. برای تبدیل از تابع `SAT.ConvertToCNF(.)` در نرم‌افزار ApCoCoA استفاده شده است.

گزاره ۷۰.۴ (جایگزینی شریک مربعی). فرض کنید $f = x_ix_j + x_ix_k + y \in \mathbb{F}_2[x_1, ..., x_n, y]$ یک چندجمله‌ای باشد به طوری که i, j, k دوبه‌دو متمایز باشند. در این‌صورت گزاره منطقی

$$F = (X_i \vee \neg Y) \wedge (X_j \vee X_k \vee \neg Y) \wedge (\neg X_j \vee \neg X_k \vee \neg Y) \wedge (\neg X_i \vee \neg X_j \vee X_k \vee Y) \wedge (\neg X_i \vee X_j \vee \neg X_k \vee Y)$$

نمایش منطقی f خواهد بود.

برهان. با استفاده از جدول درستی (ارزش) براحتی می‌توان دید که $g = x_ix_j + x_ix_k \in \mathbb{F}_2[x_1, ..., x_n]$ دارای نمایش منطقی $G = (\neg X_i \vee \neg X_j \vee X_k) \wedge (\neg X_i \vee X_j \vee \neg X_k)$ است. در این‌صورت با استفاده از لم ۶۶.۴، نتیجه می‌گیریم که $Y \iff F = \neg G$ نمایش منطقی f است. در نهایت با یک ساده‌سازی می‌توان نشان داد که F فرمول ارائه شده در صورت قضیه برابر است. □

با استفاده از گزاره فوق می‌توانیم یک روش دیگر برای تبدیل چندجمله‌ای‌های مربعی حلقه $\mathbb{F}[x_1, ..., x_n]$ به گزاره‌های منطقی معرفی کنیم. این روش که *روش شریک مربعی* یا QPS [23] نام دارد شامل جایگذاری عبارات $x_ix_j + x_ix_k$ با یک متغیر جدید y و سپس افزودن گزاره منطقی متناظر با این جایگزینی (که طبق گزاره ۷۰.۴ به‌دست می‌آید) به ترکیب عطفی استاندارد، است.

[23] Quadratic partner strategy

مثال ۷۱.۴. چندجمله‌ای $f = x_1x_2 + x_1x_3 + x_2x_3 + x_1 + x_2 + 1 \in \mathbb{F}_2[x_1, x_2, x_3]$ را در نظر بگیرید. اگر با استفاده از روش QPS، نمایش منطقی این چندجمله‌ای را بیابیم، به ازای عدد برشی ۶ به یک گزاره منطقی به صورت ترکیب عطفی استاندارد شامل ۶ متغیر منطقی و ۲۵ بند خواهیم رسید. گرچه تعداد بندهای به‌دست آمده در این روش از تعداد بندهای به‌دست آمده از روش‌های LPS و DPS بیشتر است ولی مزیت کم بودن تعداد متغیرهای منطقی در این روش خیلی مهم‌تر است و به همین دلیل ترجیح می‌دهیم در چنین شرایطی از روش QPS استفاده کنیم، زیرا تعداد متغیرهای منطقی یک گزاره منطقی تأثیر زیادی در پیچیدگی حل مسئله صدق‌پذیری می‌گذارد.

برای تبدیل چندجمله‌ای‌های درجه سوم به مسئله صدق پذیری می‌توانیم از گزاره زیر استفاده کنیم.

گزاره ۷۲.۴ (جایگزینی شریک مکعبی). فرض کنید $f = x_ix_jx_k+x_ix_jx_l+y \in \mathbb{F}_2[x_1, ..., x_n, y]$، به‌طوری‌که i, j, k و l دوبه‌دو متمایز باشند. در این‌صورت گزاره منطقی

$$\begin{aligned} F =&(X_i \vee \neg Y) \wedge (X_j \vee \neg Y) \wedge (X_k \vee X_l \vee \neg Y) \wedge (\neg X_k \vee \neg X_l \vee \neg Y) \\ &\wedge (\neg X_i \vee \neg X_j \vee \neg X_k \vee X_l \vee Y) \wedge (\neg X_i \vee \neg X_j \vee \neg X_k \vee \neg X_l \vee Y) \end{aligned}$$

نمایش منطقی f خواهد بود.

برهان. رجوع کنید به [۳۸]، قضیه شماره ۸. □

با به‌کارگیری گزاره‌های قبلی الگوریتمی ارائه می‌کنیم که ورودی آن معادلات یک دستگاه چندجمله‌ای روی $\mathbb{F}_2$ و خروجی آن گزاره منطقی به‌صورت ترکیب عطفی استاندارد متناظر با آن دستگاه است.

گزاره ۷۳.۴ (الگوریتم تبدیل دستگاه معادلات بولی به مسئله صدق‌پذیری). فرض کنید $f_1, ..., f_m$ چندجمله‌ای‌هایی از حلقه $\mathbb{F}_2[x_1, ..., x_n]$ باشند و $l \geq 3$. در این‌صورت الگوریتم ۳۲ (در زمان چندجمله‌ای) ترکیب عطفی استاندارد K متناظر با دستگاه معادلات $f_1 = \cdots = f_m = 0$ را طوری محاسبه می‌کند که جواب‌های دستگاه معادلات مذکور در تناظر ۱-۱ با مقادیر منطقی هستند که گزاره K را ارضا می‌کنند.

برهان. به [۳۸]، قضیه ۹ رجوع کنید. □

مثال ۷۴.۴. الگوریتم رمزنگاری $CTC(B, N)$ که B تعداد جعبه‌های جانشینی موازی در هر دور و N، تعداد دورها را نشان می‌دهد در نظر بگیرید. دستگاه معادلات متناظر با این سامانه رمز را به ازای پارامترهای N, B مختلف و عدد برشی $l = 4$ با استفاده الگوریتم ۳۲، به ترکیب عطفی نرمال تبدیل کرده‌ایم. جدول ۸.۴ تعداد متغیرها و بندهای فصلی به‌دست آمده در هریک از این روش‌ها را به‌ازای چهار روش متفاوت به‌کار رفته در گام ۳ الگوریتم ۳۲ را نشان می‌دهد. در

سامانه	$(\#v_{SS}, \#c_{SS})$	$(\#v_{LPS}, \#c_{LPS})$	$(\#v_{DPS}, \#c_{DPS})$	$(\#v_{QPS}, \#c_{QPS})$
CTC(۳, ۳)	(۳۴۳, ۲۲۳۰)	(۳۳۴, ۱۸۸۸)	(۳۱۶, ۱۷۴۴)	(۳۴۳, ۲۱۹۴)
CTC(۴, ۴)	(۶۰۵, ۳۹۷۱)	(۵۸۹, ۳۳۶۳)	(۵۵۷, ۳۱۰۷)	(۶۰۵, ۳۹۰۷)
CTC(۵, ۵)	(۹۴۱, ۶۲۰۴)	(۹۱۶, ۵۲۵۴)	(۸۶۶, ۴۸۵۴)	(۹۴۱, ۶۱۰۴)
CTC(۶, ۶)	(۱۳۵۱, ۸۹۲۳)	(۱۳۱۵, ۷۵۵۵)	(۱۲۴۳, ۶۹۷۹)	(۱۳۵۱, ۸۷۷۹)

جدول ۸.۴: مقایسه روش‌های مختلف تبدیل دستگاه معادلات چندجمله‌ای به مسئله صدق‌پذیری

جدول ۸.۴ $\#v.$ و $\#c.$، به ترتیب نشان‌دهنده تعداد بندهای فصلی و تعداد متغیرهای منطقی در ترکیب عطفی به‌دست آمده برای دستگاه مورد نظر با استفاده از الگوریتم ۳۲، هستند. همان‌طور که مشاهده می‌شود روش‌های شریک خطی (LPS) و جانشینی شریک مربعی (QPS) نتوانسته‌اند روش استاندارد (SS) را بهبود چندانی ببخشند، ولی روش شریک خطی مضاعف (DPS) توانسته تا ۸ درصد تعداد متغیرها و تا ۲۲ درصد تعداد بندهای فصلی را نسبت به روش استاندارد کاهش دهد. دستگاه‌هایی که در جدول ۸.۴، به‌ازای پارامترهای مختلف برای CTC و به‌ازای یک زوج متن اصلی و رمزشده معلوم به‌کار رفته‌اند همگی دارای جواب یکتا هستند.

لازم به ذکر است که حل‌کننده‌های مسئله صدق‌پذیری، در صورتی که مسئله داده شده صدق‌پذیر باشد، تنها یک جواب برای مسئله داده شده می‌یابند، برای این‌که با استفاده از این حل‌کننده‌ها بتوانیم تمام جواب‌ها را بیابیم باید هر بار که جواب جدیدی به‌دست آمد، نقیض آن‌را به مسئله داده شده اضافه کنیم و دوباره مسئله جدید را حل کنیم، این‌کار را تا جایی ادامه می‌دهیم که مسئله صدق‌ناپذیر شود، در این‌صورت است که تمام جواب‌ها به‌دست آمده است. اما از آن‌جایی که دستگاه‌های به‌دست آمده برای CTC در جدول ۸.۴، همگی دارای جواب یکتا هستند نیازی به اعمال مجدد حل‌کننده نیست. ابتدا برای مقایسه روش‌های مختلف تبدیل دستگاه چندجمله‌ای به مسئله صدق‌پذیری، با استفاده از حل‌کننده `MiniSat`، صورت‌های متعارف عطفی که با استفاده از روش‌های مختلف برای CTC(۶,۶) به‌دست آمده‌اند را حل می‌کنیم تا مشخص شود کدام روش بهینه است. جدول ۹.۴، زمان حل با استفاده از حل‌کننده `MiniSat`، وقتی از عدد برشی $l =$ ۴ برای تبدیل دستگاه به CNF استفاده شده است را نشان می‌دهد. لازم به ذکر است که تمام محاسبات این بخش به‌وسیله رایانه با پردازنده ۱٫۶GHz و حافظه رم ۶GB و روی سیستم‌عامل لینوکس اوبونتو ۱۶٫۰۴ انجام شداند.

در آزمایش بعدی، عدد برشی را بجای $l =$ ۴ برابر $l =$ ۶ اختیار کرده‌ایم و دستگاه به‌دست آمده از CTC(۶,۶) را با استفاده از روش‌های مختلف حل کرده و زمان حل توسط حل‌کننده `MiniSat` را در جدول ۱۰.۴ آورده‌ایم. همان‌طور که در آزمایش‌های فوق نیز مشاهده می‌شود زمانی که از روش LPS برای تبدیل استفاده می‌کنیم، سرعت حل بیشتر است. در ضمن زمان‌های گزارش شده در جدول ۱۰.۴ نشان می‌دهد که وقتی عدد برشی را بجای $l =$ ۴ برابر با $l =$ ۶ اختیار می‌کنیم، سرعت بیشتر می‌شود. به همین دلیل در ادامه دستگاه‌های به‌دست آمده را با استفاده از روش LPS

عدد برشی $l=۴$				
روش	$\#v$	$\#c$	$t_{Minisat}$	حافظه مصرفی
SS	۱۳۵۱	۸۹۲۳	۱۹٫۵۷s	۲۳٫۰۰MB
LPS	۱۳۱۵	۷۵۵۵	۸٫۰۹۶s	۲۳٫۰۰MB
DPS	۱۲۴۳	۶۹۷۹	۵۵٫۰۴s	۲۳٫۰۰MB
QPS	۱۳۵۱	۸۷۷۹	۲۱٫۵۳۲s	۲۳٫۰۰MB

جدول ۹.۴: مقایسه روش‌های مختلف تبدیل دستگاه چندجمله‌ای $CTC(۶,۶)$ به مسئله صدق‌پذیری

عدد برشی $l=۶$				
روش	$\#v$	$\#c$	$t_{Minisat}$	حافظه مصرفی
SS	۱۰۶۳	۱۳۰۹۹	۱۵٫۱۹۶s	۲۳٫۰۰MB
LPS	۱۰۶۳	۹۴۲۷	۴٫۲۴۸s	۲۳٫۰۰MB
DPS	۱۰۶۳	۸۱۳۱	۵٫۷۴۴s	۲۳٫۰۰MB
QPS	۱۰۶۳	۱۲۰۹۱	۴٫۸۰۸s	۲۳٫۰۰MB

جدول ۱۰.۴: مقایسه روش‌های مختلف تبدیل دستگاه چندجمله‌ای $CTC(۶,۶)$ به مسئله صدق‌پذیری

و عدد برشی $l=۶$، به مسئله صدق‌پذیری تبدیل می‌کنیم.

این بخش را با مقایسه‌ای بین چهار نوع حل‌کننده مسئله صدق‌پذیری که در دوره‌های مختلف مسابقات انتخاب برترین حل‌کننده‌های مسئله صدق‌پذیری حائز رتبه‌های برتر شده‌اند، به پایان می‌بریم. دستگاه‌هایی که برای مقایسه انتخاب کرده‌ایم دستگاه‌های به‌دست آمده از $SR(n,r,c,e)$ به‌ازای مقادیر مختلف برای پارامترهای n,r,c و e و همچنین دستگاه به‌دست آمده از الگوریتم رمزنگاری BiviumA است. برای تبدیل دستگاه چندجمله‌ای به مسئله صدق‌پذیری نیز از روش LPS استفاده کرده‌ایم. زمان حل دستگاه با استفاده از حل‌کننده‌های مختلف بر حسب ثانیه در جدول ۱۱.۴ آمده است. همه دستگاه‌های معادلات متناظر با $SR(n,r,c,e)$ در جدول ۱۱.۴ به‌ازای فقط یک زوج متن اصلی و رمزشده معلوم به‌دست آمده‌اند. در استخراج دستگاه معادلات

	m	n	$\#v$	$\#c$	MiniSat	Cryptominisat5	Riss6	Lingeling	تعداد جواب
$SR(۲,۲,۲,۴)$	۲۲۴	۱۲۸	۵۲۷	۷۵۱۱	۰٫۶۳۶	۰٫۶۹۰	۱٫۵۱۲	۱٫۳۰۰	۱
$SR(۱,۱,۲,۸)$	۱۲۰	۷۲	۶۸۸	۱۴۱۹۷	۱٫۷۷۶	۳٫۳۴۰	۰٫۴۷۶	۸٫۰۰۰	۱
$SR(۳,۱,۱,۸)$	۲۰۰	۱۰۴	۱۳۱۶	۲۷۶۲۷	۲٫۷۴۰	۱۹٫۸۷۰	۱۱٫۱۳۲	۲۷٫۷۰۰	۱
$SR(۵,۲,۲,۴)$	۵۳۶	۲۹۶	۱۳۰۳	۱۸۹۵۱	۳٫۸۷۲	۵٫۱۶۰	۷٫۸۲۸	۲۵٫۰۰۰	۱
$SR(۹,۲,۲,۴)$	۹۵۲	۵۲۰	۲۳۳۲	۳۴۱۰۵	۱۴٫۷۵۶	۱۶٫۶۵۰	۴٫۹۶۸	۷٫۴۰۰	۱
$BiviumA(۱۷۸)$	۵۳۴	۵۳۳	۵۳۳	۸۹۰۰	۱٫۰۸۸	۰٫۳۹۰	۲٫۵۱۶	۰٫۴۰۰	۱

جدول ۱۱.۴: مقایسه حل‌کننده‌های مسئله صدق‌پذیری

BiviumA مربوط به جدول ۱۱.۴ نیز فرض کرده‌ایم که ۱۷۸ بیت از خروجی این مولد کلید اجرایی، معلوم است و دستگاه بر اساس روش معرفی متغیر جدید که در بخش‌های قبل شرح داده شد، استخراج شده است. چنان‌چه در جدول ۱۱.۴ نیز مشاهده می‌شود، رمز دنباله‌ای BiviumA درمدت زمان ۰٫۳۹ ثانیه توسط Cryptominisat5 شکسته شده است، کاری که هرگز با روش‌های دیگر نظیر پایه گروبنر میسر نیست.

۶.۴ نتیجه‌گیری

۱.۶.۴ مقاوم‌سازی الگوریتم‌های رمز نوین در مقابل حمله‌های جبری

امروزه حمله‌های جبری به یک آزمون استاندارد برای طراحی الگوریتم‌های رمز نگاری نوین تبدیل شده است و لازم است تا طراحان الگوریتم‌های رمزنگاری، شناخت کافی از حمله‌های جبری داشته باشند. با این وجود هیچ‌گاه نمی‌توان با اطمینان کامل در مورد مصونیت یک الگوریتم رمزنگاری در مقابل حمله‌های جبری اظهار نظر کرد چرا که اولاً روش‌های جبری‌سازی و استخراج معادلات الگوریتم‌های رمزنگاری یکتا نیست و یک مهاجم ممکن است بتواند معادلاتی بهینه برای الگوریتم بیابد به‌طوری که اعمال حمله جبری را ساده‌تر از آن‌چه طراحان پیش‌بینی می‌کردند، گرداند. ثانیاً، چنان‌چه در دهه‌های اخیر شاهد بوده‌ایم، پیشرفت خوبی در روش‌های حل دستگاه‌های معادلات چندجمله‌ای حاصل شده و این روند ادامه دارد. بنابراین حداقل کاری که می‌توان برای مصونیت یک الگوریتم رمزنگاری در مقابل حمله‌های جبری انجام داد شناخت کامل روش‌های استخراج معادلات اجزای سازنده الگوریتم‌های رمزنگاری نوین، و همچنین شناخت کامل از روش‌های حل دستگاه‌های به‌دست آمده از الگوریتم‌های رمزنگاری و مقاوم‌سازی الگوریتم رمزنگاری مورد نظر در مقابل تهدیدات شناخته شده است.

۲.۶.۴ کدام حل‌کننده بهتر عمل می‌کند؟

می توان گفت که از میان حل‌کننده‌های معرفی شده در این پایان‌نامه، حل‌کننده‌های مسئله صدق‌پذیری از لحاظ سرعت و مصرف حافظه در جایگاه برتر قرار دارند. البته یک محدودیت اساسی در استفاده از حل‌کننده‌های مسئله صدق‌پذیری وجود دارد که عبارت است از این که، چنین حل‌کننده‌هایی فقط برای حل دستگاه‌های معادلات چندجمله‌ای روی $\mathbb{F}_2$ قابل استفاده هستند. با این وجود روش‌هایی وجود دارد که با استفاده از آن‌ها می‌توان دستگاه معادلات چندجمله‌ای روی $\mathbb{F}_{2^n}$ را به دستگاه معادلات چندجمله‌ای روی $\mathbb{F}_2$ تبدیل کرد، برای نمونه به الگوریتمی که در این‌رابطه در [۳۸] معرفی شده است رجوع کنید.

در موارد متعددی که از پایه گروبنر برای حل دستگاه‌های به‌دست آمده استفاده کردیم مشاهده

شد که یک محدودیت اساسی الگوریتم‌های محاسبه پایه گروبنر مصرف زیاد حافظه است، به‌طوری که در بسیاری از موارد این امر سبب متوقف شدن محاسبه و بی‌نتیجه ماندن آن می گردید. در مقابل، الگوریتم‌های برنامه‌ریزی عدد صحیح حافظه کم‌تری مصرف می‌کنند، با این وجود این الگوریتم‌ها همیشه جایگزین مناسبی برای روش پایه گروبنر نیستند چرا که در برخی موارد ممکن است حل دستگاه با استفاده از روش پایه گروبنر سریع‌تر از حل با استفاده از الگوریتم‌های برنامه‌ریزی عد صحیح باشد، به‌خصوص در مواردی که تعداد جملات غیر خطی دستگاه از تعداد معادلات بیشتر باشد.

در مورد روش پایه مرزی باید گفت که این روش بسیار کم‌تر از روش پایه گروبنر مورد توجه قرار گرفته است و بر خلاف پایه گروبنر که الگوریتم‌های آن سال‌ها است که در حال توسعه و بهبود هستند، الگوریتم‌های پایه مرزی در آغاز راه توسعه قرار دارند. با این وجود بنا بر دلایلی که در بخش پایه‌مرزی برشمردیم، به‌نظر می‌رسد پایه مرزی شایسته تحقیقات بیشتری است.

۳.۶.۴ راه‌کاری‌های مناسب‌تر

حمله جبری گزینه مناسبی برای ترکیب با سایر حمله‌های شناخته شده رمزنگاری است و به‌نظر می‌رسد این رویکرد ترکیبی بسیار مؤثرتر از رویکرد جبری محض برای حمله به سامانه‌های رمزنگاری است. برای مثال تحقیقات زیادی در رابطه با ترکیب حمله جبری و حمله تفاضلی صورت گرفته است که برای نمونه می‌توانید به [۳] رجوع کنید. به عنوان مثالی دیگر می‌توان به ترکیب حمله‌های جبری با حمله‌های کانال جانبی اشاره کرد.

الگوریتم ۲۵ الگوریتم تبدیل دستگاه معادلات به مسئله‌ی برنامه‌ریزی عدد صحیح و سپس حل آن

ورودی $f_۱, ..., f_m \in \mathbb{F}_۲[x_۱, ..., x_n]$
خروجی جواب‌های دستگاه معادلات $f_۱ = \cdots = f_m = ۰$ که در $\mathbb{F}_۲$ قرار دارند.

۱: همه چندجمله‌ای‌های $f_۱, ..., f_m$ را به پیمانه‌ی معادلات میدان $\mathbb{F}_۲$ تحویل می‌کنیم. به ازای هر $i \in \{۱, ..., m\}$ ، S_i, s_i و S را به‌صورت زیر تعریف می‌کنیم.

$$S_i := \{t \in \mathrm{Supp}(f_i) |\ \deg(t) \geq ۲\}\ ,\ s_i = |\,\mathrm{Supp}(f_i)|\ ,\ S = \bigcup_{i=۱}^{m} S_i.$$

۲: به ازای هر $i = ۱, ..., m$ ، متغیر صحیح جدید K_i را معرفی کرده و شرط $I_i : K_i \leq [\frac{s_i}{۲}]$ را برای آن در نظر می‌گیریم.

۳: به ازای هر یکجمله‌ای $t_j \in S$، متغیر صحیح جدید X_{n+j} را معرفی می‌کنیم. به ازای هر $i = ۱, ..., m$، f_i را به‌صورت $f_i = \sum_{t_j \in S_i} t_j + l_i$ در نظر می‌گیریم که $l_i \in P_{\leq ۱}$ (یعنی l_i خطی است). سپس معادلات $F_i : \sum_j X_{n+j} + L_i - ۲K_i = ۰$ ، که $L_i \in \mathbb{Z}[X_۱, ..., X_n]_{\leq ۱}$ نمایش استاندارد l_i روی $\mathbb{Z}$ است را تشکیل می‌دهیم.

۴: با در نظر گرفتن هر $t_j \in S$، به‌صورت $t_j = \prod_{\alpha \in N_j} x_\alpha$، نابرابری‌های زیر را به مجموعه قیدهای مسئله اضافه می‌کنم.

$$I_{n+j} :\ \sum_{\alpha \in N_j} X_\alpha - X_{n+j} \leq |N_j| - ۱,$$
$$\forall\ \alpha \in N_j \qquad I_{j\alpha} :\ X_\alpha \geq X_{n+j}.$$

۵: به ازای هر $\alpha \in \{۱, ..., n\}$، قرار می‌دهیم $I'_\alpha :\ X_\alpha \leq ۱$.

۶: یک چندجمله‌ای خطی $C \in \mathbb{Z}[X_\alpha, X_{n+j}, K_i]$ را انتخاب می‌کنیم و با استفاده از حل‌کننده‌های مسئله برنامه‌ریزی عدد صحیح، چندتایی مرتب (a_α, a_{n+j}, c_i) از اعداد صحیح نامنفی را که در معادلات و نامعادلات $\{I_i, F_i, I_{n+j}, I_{j\alpha}, I'_\alpha\}$، صدق کرده، و C را کمینه (یا بیشینه) می‌کنند می‌یابیم.

۷: با بازگرداندن n تایی $(a_۱, ..., a_n)$ الگوریتم خاتمه می‌یابد.

الگوریتم ۲۶ الگوریتم GSAT برای حل مسئله صدق‌پذیری

Input: گزاره منطقی F به‌صورت متعارف عطفی و پارامترهای $n_{restart}$ و n_{flip}
Output: یک انتساب رضایت بخش یا FAIL

for $i = 1 \ldots n_{restart}$ **do**
$\sigma \leftarrow$ انتساب تصادفی مقادیر صفر و یک با احتمال یکسان به هر یک از متغیرها
for $j = 1 \ldots n_{flip}$ **do**
if $F|_\sigma = 1$ **then**
return σ
end if
$a_v \leftarrow$ تعداد بندهایی که با تغییر مقدار متغیر v از وضعیت صادق به وضعیت ناصادق در می‌آیند
$b_v \leftarrow$ تعداد بندهایی که با تغییر مقدار متغیر v از وضعیت ناصادق به وضعیت صادق در می‌آیند
$k_v \leftarrow a_v - b_v$
$v \leftarrow$ متغیری که به ازای آن k_v کمینه شود(در شرایط یکسان یک متغیر به‌تصادف انتخاب می‌شود)
تغییر مقدار v در σ
end for
end for
return FAIL

الگوریتم ۲۷ الگوریتم WalkSAT برای حل مسئله صدق‌پذیری

Input: گزاره منطقی F به‌صورت متعارف عطفی و پارامترهای $n_{restart}, n_{flip}$ و $p_{noise} \in [۰, ۱]$
Output: یک انتساب رضایت بخش یا FAIL

for $i = 1 \ldots n_{restart}$ **do**
$\sigma \leftarrow$ انتساب تصادفی مقادیر صفر و یک با احتمال یکسان به هر یک از متغیرها
for $j = 1 \ldots n_{flip}$ **do**
if $F|_\sigma = 1$ **then**
return σ
end if
$C \leftarrow$ یک بند ناصادق از F که به تصادف و با احتمال یکنواخت انتخاب شده
if $\exists\ x \in C\ s.t\ breakcount_x = 0$ **then**
else
با احتمال p_{noise}:
$v \leftarrow$ یک متغیر که به‌تصادف از C انتخاب شده است
با احتمال $۱ - p_{noise}$:
$v \leftarrow$ متغیر x در C به‌طوری که $breakcount_x$ کمینه شود
end if
end for
تغییر مقدار v در σ
end for
return FAIL

الگوریتم ۲۸ الگوریتم DP برای حل مسئله صدق‌پذیری

Input: F : صورت متعارف عطفی مسئله صدق‌پذیری به‌صورت مجموعه‌ای
Output: SAT یا UNSAT

DP(F)
  if $F = \emptyset$ **then**
    return SAT
  end if
  if $\exists\ l \in \mathrm{Lit}(F)\ s.t\ \{l\} \in \mathrm{Clauses}(F)$ **then**
    return DP$(\{C \backslash \{\neg l\} : C \in F \wedge l \notin C\})$
  end if
  if $\exists\ l \in \mathrm{Lit}(F)\ s.t\ \neg l \notin \mathrm{Lit}(F)$ **then**
    return DP$(\{C \in F :\ l \notin C \wedge \neg l \notin C\})$
  end if
  $X \leftarrow \mathrm{Vars}(F)$
  while $\exists\ C_1, C_2 \in F\ s.t$ `IsPossibleResol`(C_1, C_2, X) **do**
    $F_1 \leftarrow \{C_1 \odot C_2\}$
    if $F_1 = \{\emptyset\}$ **then**
      return UNSAT
    end if
    $F \leftarrow F \cup F_1$
  end while
  while $\exists\ C \in F\ s.t\ X \in C \vee \neg X \in C$ **do**
    $F \leftarrow F \backslash \{C\}$
  end while
  return DP(F)
end

الگوریتم ۲۹ الگوریتم DPLL برای حل مسئله صدق‌پذیری

Input: F : صورت متعارف عطفی مسئله صدق‌پذیری به‌صورت مجموعه‌ای
Output: SAT یا UNSAT

DPLL(F)
  if $F = \emptyset$ **then**
    return SAT
  end if
  if $\exists\, l \in \mathrm{Lit}(F)\ s.t\ \{l\} \in \mathrm{Clauses}(F)$ **then**
    return **DPLL**($\{C \backslash \{\neg l\} : C \in F \wedge l \notin C\}$)
  end if
  if $\exists\, l \in \mathrm{Lit}(F)\ s.t\ \neg l \notin \mathrm{Lit}(F)$ **then**
    return **DPLL**($\{C \in F :\ l \notin C \wedge \neg l \notin C\}$)
  end if
  $X \leftarrow \mathrm{Vars}(F)$ انتخاب متغیر تصمیم
  $F_0 \leftarrow \{C \backslash \{X\} :\ C \in F\ \wedge\ \neg X \notin C\}$
  if **DPLL**(F_0) $=$ SAT **then**
    return SAT
  end if
  $F_1 \leftarrow \{C \backslash \{\neg X\} :\ C \in\ \wedge\ X \notin C\}$
  if **DPLL**(F_1) $=$ SAT **then**
    return SAT
  end if
  return UNSAT
end

الگوریتم ۳۰ الگوریتم AnalyzeConflict به‌کار رفته در الگوریتم CDCL

Input: C : بند ناصادق به‌دست آمده در فرآیند انتشار واحد
Output: (C_L : بند آموخته شده, β : سطح تصمیم در پیمایش معکوس)

AnalyzeConflict(C)
  $d \leftarrow \max\{\delta(l) :\ \alpha(l) \neq NIL\}$
  $i \leftarrow 0$
  repeat
    $C_L \leftarrow C_L^{d,i}$
    $i \leftarrow i + 1$
  until $C_L = C_L^{d,i-1}$
  if $|C_L| = 1$ **then**
    return $(C_L, 0)$
  else
    $\beta \leftarrow \max\{\delta(l) :\ l \in C_L\ \wedge\ \delta(l) \neq d\}$
    return (C_L, β)
  end if
end

الگوریتم ۳۱ الگوریتم CDCL

Input: (F, ν)
Output: SAT یا UNSAT

CDCL(F, ν)
  if `UnitPrppagation`$(F, \nu) =$ CNFLICT **then**
    return UNSAT
  end if
  $dl \leftarrow 0$: سطح تصمیم
  while $\neg$`AllVariablesAssigned`(F, ν) **do**
    $(X, v) \leftarrow$ `PickBrnchingVariable`(F, ν)
    $dl \leftarrow dl + 1$
    $\nu \leftarrow \nu \cup \{(X, v)\}$
    if `UnitPrppagation`$(F, \nu) =$ CONFLICT **then**
      $(C_L, \beta) \leftarrow$ `AnalyzeConflict`(F, ν)
      if $\beta = 0$ **then**
        return UNSAT
      else
        $F \leftarrow F \cup \{C_L\}$
        `BacktrackTo`(F, ν, β)
        $dl \leftarrow \beta$
      end if
    end if
  end while
  return SAT
end

الگوریتم ۳۲ الگوریتم تبدیل دستگاه معادلات چندجمله‌ای روی $\mathbb{F}_۲$ به مسئله صدق‌پذیری

ورودی $S : f_۱ = \cdots = f_m = ۰ \; s.t \; f_۱, ..., f_m \in \mathbb{F}_۲[x_۱, ..., x_n], l \geq ۳$

خروجی گزاره منطقی K، به طوری که جواب‌های دستگاه S در تناظر ۱-۱ با چندتایی‌های منطقی هستند که گزاره K را ارضا می‌کنند.

۱: قرار می‌دهیم $G = \emptyset$. گام‌های ۲ تا ۵ را به ازای $i = ۱, ..., m$ اجرا می کنیم.

۲: گام ۳ را تا زمانی که هیچ چندجمله‌ای g با خاصیت ذکر شده در گام ۳ یافت نشود اجرا می‌کنیم.

۳: چندجمله‌ای g که از جمع اعضای زیرمجموعه‌ای از $\mathrm{Supp}(f_i)$ حاصل می‌شود را به نحوی می‌یابیم که در شرایط روش تبدیل انتخاب شده (روش استاندارد، روش شریک خطی و ...) صدق کند. متغیر جدید y_j را معرفی کرده و f_i را با $f_i - g + y_j$، جایگزین کرده و $g + y_j$ را به مجموعه G ضمیمه می‌کنیم.

۴: گام ۵ را تا زمانی که $\# \mathrm{Supp}(f_i) \leq l$ اجرا می‌کنیم و سپس f_i را به G ضمیمه می‌کنیم.

۵: اگر $\# \mathrm{Supp}(f_i) > l$ آن‌گاه متغیر جدید y_j را معرفی می‌کنیم. اگر g مجموع $l - ۱$ جمله اول f_i باشد، f_i را با $f_i - g + y_j$، جایگزین کرده و سپس $g + y_j$ را به مجموعه G اضافه می‌کنیم.

۶: به ازای هر چندجمله‌ای در G، نمایش منطقی به صورت ترکیب عطفی نرمال متناظر با آن را که با K نمایش می‌دهیم را محاسبه می‌کنیم. K خروجی الگوریتم خواهد بود.

کتاب‌نامه

[1] Albrecht, Martin. Algebraic attacks on the courtois toy cipher. *Cryptologia*, **32**(3):220–276, 2008.

[2] Albrecht, Martin. *Algorithmic algebraic techniques and their application to block cipher cryptanalysis*. PhD thesis, University of london, 2010.

[3] Albrecht, Martin and Cid, Carlos. Algebraic techniques in differential cryptanalysis. Cryptology ePrint Archive, Report 2008/177, 2008. http://eprint.iacr.org/2008/177.

[4] Arabnezhad-Khanoki, Hossein, Sadeghiyan, Babak, and Pieprzyk, Josef. Algebraic attack efficiency versus s-box representation. Cryptology ePrint Archive, Report 2017/007, 2017. http://eprint.iacr.org/2017/007.

[5] Ars, Gwénolé; Faugere, Jean-Charles; Imai Hideki; Kawazoe Mitsuru and Sugita, Makoto. Comparison between xl and gröbner basis algorithms. In *Advances in Cryptology-ASIACRYPT 2004*, pages 338–353. Springer, 2004.

[6] Auzinger, Winfried and Stetter, Hans J. An elimination algorithm for the computation of all zeros of a system of multivariate polynomial equations. In *Numerical Mathematics Singapore 1988*, pages 11–30. Springer, 1988.

[7] Bard, Gregory. *Algebraic cryptanalysis*. Springer Science & Business Media, 2009.

[8] Barkan, Elad; Biham, Eli and Keller, Nathan. Instant ciphertext-only cryptanalysis of gsm encrypted communication. In *Advances in Cryptology-CRYPTO 2003*, pages 600–616. Springer, 2003.

[9] Bigatti, A. M., Abbott J. and Lagorio, G. CoCoA-5: a system for doing Computations in Commutative Algebra. Available at http://cocoa.dima.unige.it.

[10] Biryukov, Alex; Shamir, Adi and Wagner, David. Real time cryptanalysis of a5/1 on a pc. In *International Workshop on Fast Software Encryption*, pages 1–18. Springer, 2000.

[11] Biryukov, Alex and Cannière, Christophe De. Block ciphers and systems of quadratic equations. In Johansson, Thomas, editor, *Fast Software Encryption, 10th International Workshop, FSE 2003, Lund, Sweden, February 24-26, 2003, Revised Papers*, volume 2887 of *Lecture Notes in Computer Science*, pages 274–289. Springer, 2003.

[12] Canniere, C De and Preneel, B. Trivium specifications. *citeseer. ist. psu. edu/734144. html*, 2005.

[13] Cid, Carlos, Murphy, Sean, and Robshaw, Matthew. *Algebraic aspects of the advanced encryption standard*. Springer Science & Business Media, 2006.

[14] Cid, Carlos, Murphy, Sean, and Robshaw, Matthew JB. Small scale variants of the aes. In *International Workshop on Fast Software Encryption*, pages 145–162. Springer, 2005.

[15] Collart, Stéphane; Kalkbrener, Michael and Mall, Daniel. Converting bases with the gröbner walk. *Journal of Symbolic Computation*, **24**(3):465–469, 1997.

[16] Courtois, Nicolas; Klimov, Alexander; Patarin Jacques and Shamir, Adi. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In *Advances in Cryptology—EUROCRYPT 2000*, pages 392–407. Springer, 2000.

[17] Courtois, Nicolas T; Bard, Gregory V and Wagner, David. Algebraic and slide attacks on keeloq. In *International Workshop on Fast Software Encryption*, pages 97–115. Springer, 2008.

[18] Courtois, Nicolas. Ctc2 and fast algebraic attacks on block ciphers revisited. *IACR Cryptology ePrint Archive*, 2007.

[19] Courtois, Nicolas T. Fast algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology-CRYPTO 2003*, pages 176–194. Springer, 2003.

[20] Courtois, Nicolas T. How fast can be algebraic attacks on block ciphers? In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2007.

[21] Courtois, Nicolas T and Meier, Willi. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology—EUROCRYPT 2003*, pages 345–359. Springer, 2003.

[22] Courtois, Nicolas T and Pieprzyk, Josef. Cryptanalysis of block ciphers with overdefined systems of equations. In *International Conference on the Theory and*

Application of Cryptology and Information Security, pages 267–287. Springer, 2002.

[23] Cox, David A.; Little, John and O'Shea, Donal. *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015. An introduction to computational algebraic geometry and commutative algebra.

[24] Davis, Martin, Logemann, George, and Loveland, Donald. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[25] Davis, Martin and Putnam, Hilary. A computing procedure for quantification theory. *Journal of the ACM (JACM)*, 7(3):201–215, 1960.

[26] Decker, Wolfram, Greuel, Gert-Martin, Pfister, Gerhard, and Schönemann, Hans. Singular 4-1-0 — A computer algebra system for polynomial computations. http://www.singular.uni-kl.de, 2016.

[27] Diem, Claus. The xl-algorithm and a conjecture from commutative algebra. In *Advances in Cryptology-ASIACRYPT 2004*, pages 323–337. Springer, 2004.

[28] Diffie, Whitfield and Hellman, Martin. New directions in cryptography. *IEEE transactions on Information Theory*, **22**(6):644–654, 1976.

[29] Ding, Jintai; Buchmann, Johannes; Mohamed Mohamed Saied Emam; Mohamed Wael Said Abd Elmageed and Weinmann, Ralf-Philipp. Mutantxl. *Proc. Symbolic Computation and Cryptography*, 2008.

[30] Faugere, Jean-Charles. A new efficient algorithm for computing gröbner bases (f4). *Journal of pure and applied algebra*, 139(1):61–88, 1999.

[31] Faugère, Jean Charles. A new efficient algorithm for computing gröbner bases without reduction to zero (f5). In *Proceedings of the 2002 International Symposium on Symbolic and Algebraic Computation*, ISSAC '02, pages 75–83, New York, NY, USA, 2002. ACM.

[32] Faugère, Jean-Charles and Ars, Gwénolé. *Comparison of XL and Gröbner basis algorithms over Finite Fields*. PhD thesis, INRIA, 2004.

[33] Faugere, Jean-Charles; Gianni, Patrizia; Lazard Daniel and Mora, Teo. Efficient computation of zero-dimensional gröbner bases by change of ordering. *Journal of Symbolic Computation*, **16**(4):329–344, 1993.

[34] Faugère, Jean-Charles. FGb: A Library for Computing Gröbner Bases. In Fukuda, Komei, Hoeven, Joris, Joswig, Michael, and Takayama, Nobuki, editors, *Mathematical Software - ICMS 2010*, volume 6327 of *Lecture Notes in Computer Science*, pages 84–87, Berlin, Heidelberg, September 2010. Springer Berlin / Heidelberg.

[35] Frank, Jeremy, Cheeseman, Peter, and Stutz, John. When gravity fails: Local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.

[36] Hawkes, Philip and Rose, Gregory G. Rewriting variables: The complexity of fast algebraic attacks on stream ciphers. In *Advances in Cryptology–CRYPTO 2004*, pages 390–406. Springer, 2004.

[37] Jean, Jérémy. TikZ for Cryptographers. http://www.iacr.org/authors/tikz/, 2016.

[38] Jovanovic, Philipp and Kreuzer, Martin. Algebraic attacks using sat-solvers. *Groups–Complexity–Cryptology*, 2(2):247–259, 2010.

[39] Kahn, D. The code breakers–the story of secret writing. scribner, new york, usa 1996. Technical report, ISBN 0-684-83130-9, 1996.

[40] Katz, Jonathan and Lindell, Yehuda. *Introduction to modern cryptography*. CRC press, 2014.

[41] Kehrein, Achim and Kreuzer, Martin. Computing border bases. *Journal of Pure and Applied Algebra*, **205**(2):279–295, 2006.

[42] Kipnis, Aviad and Shamir, Adi. Cryptanalysis of the hfe public key cryptosystem by relinearization. In *Annual International Cryptology Conference*, pages 19–30. Springer, 1999.

[43] Kreuzer, Martin. Algebraic attacks galore! *Groups–Complexity–Cryptology*, **1**(2):231–259, 2009.

[44] Kreuzer, Martin et al. ApCoCoA-1.9: Approximate Computations in Commutative Algebra. Available at http://www.apcocoa.org.

[45] Kreuzer, Martin and Robbiano, Lorenzo. *Computational commutative algebra. 2*. Springer-Verlag, Berlin, 2005.

[46] Kreuzer, Martin and Robbiano, Lorenzo. *Computational commutative algebra 1*. Springer-Verlag, Berlin, 2008. Corrected reprint of the 2000 original.

[47] Maplesoft. Maple 2015.0, Maplesoft, a division of Waterloo Maple Inc. http://www.maplesoft.com.

[48] Marques-Silva, João P and Sakallah, Karem A. Grasp: A search algorithm for propositional satisfiability. *IEEE Transactions on Computers*, 48(5):506–521, 1999.

[49] Mohamed, Mohamed Saied Emam; Cabarcas, Daniel; Ding Jintai; Buchmann Johannes; Bulygin Stanislav. Mxl3: An efficient algorithm for computing gröbner bases of zero-dimensional ideals. In *International Conference on Information Security and Cryptology*, pages 87–100. Springer, 2009.

[50] Mohamed, Mohamed Saied Emam; Mohamed, Wael Said Abd Elmageed; Ding Jintai and Buchmann, Johannes. Mxl2: Solving polynomial equations over gf (2) using an improved mutant strategy. In *International Workshop on Post-Quantum Cryptography*, pages 203–215. Springer, 2008.

[51] Möller, H Michael. Systems of algebraic equations solved by means of endomorphisms. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pages 43–56. Springer, 1993.

[52] Möller, H Michael and Buchberger, Bruno. The construction of multivariate polynomials with preassigned zeros. In *European Computer Algebra Conference*, pages 24–31. Springer, 1982.

[53] Mourrain, Bernard. A new criterion for normal form algorithms. In *International Symposium on Applied Algebra, Algebraic Algorithms, and Error-Correcting Codes*, pages 430–442. Springer, 1999.

[54] Murphy, Sean and Robshaw, Matthew. Comments on the security of the aes and the xsl technique. *Electronic Letters*, **39**(1):36–38, 2003.

[55] Murphy, Sean and Robshaw, Matthew JB. Essential algebraic structure within the aes. In *Annual International Cryptology Conference*, pages 1–16. Springer, 2002.

[56] Papadimitriou, CH and Steiglitz, K. Combinatorial optimization · prentice-hall. *Inc., Englewood Cliffs, NJ*, 1982.

[57] Raddum, H. Cryptanalytic results on trivium. estream, ecrypt stream cipher project, report 2006/039, 2006.

[58] Selman, Bart, Levesque, Hector J, Mitchell, David G, et al. A new method for solving hard satisfiability problems. In *AAAI*, volume 92, pages 440–446, 1992.

[59] Selman, B., Kautz H. Cohen B. Local search strategies for satisfiability testing. *D.S. Johnson, M.A. Trick (eds.) Cliques, Coloring, and Satisfiability*, 26:521–532, 1996.

[60] Shannon, Claude E. Communication theory of secrecy systems. *Bell system technical journal*, **28**(4):656–715, 1949.

[61] Silva, João P Marques and Sakallah, Karem A. Grasp—a new search algorithm for satisfiability. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*, pages 220–227. IEEE Computer Society, 1997.

[62] Stein, W. A. et al. *SageMath, the Sage Mathematics Software System (Version 7.2.0)*, 2016. http://www.sagemath.org.

[63] Stetter, Hans J. *Numerical polynomial algebra.* Siam, 2004.

[64] Sugita, M; Kawazoe, M and Imai, H. Relation between xl algorithm and gröbner bases algorithms, iacr eprint server, 2004.

[65] The GLPK Developers. GNU Linear Programming Kit. Available at https://www.gnu.org/software/glpk.

[66] Ullah, Ehsan. *New Techniques for Polynomial System Solving.* PhD thesis, Universität Passau, 2012.

[67] Weispfenning, Volker; Becker, Thomas and Kredel, Heinz. Groebner bases: a computational approach to commutative algebra. *Graduate Texts in Mathematics. Springer-Verlag*, 1993.

[68] Xiao, L. Applicability of xsl attacks to block ciphers. *Electronics Letters*, **39**(25):1810–1811, 2003.

Abstract

In this dissertation, based on [43] , we present a collection of techniques in algebraic cryptanalysis. After introducing the basic setup of algebraic attacks and discussing several scenarios for symmetric and public key cryptosystem, we discuss a number of individual methods. In particular, we study the XL, XSL, and MutantXL attacks which are based on linearization techniques for multivariate polynomial system. Next, we study those algebraic attacks which are based on Gröbner, and border bases, as well as, attacks based on integer programming techniques and sat solvers.

Keywords: *Cryptosystem, Algebraic Attack, Polynomial System Solving, Relinearization, Gröbner basis, Border basis, Integer Programming, SAT solver*

University of Tehran

College of Science
School of Mathematic, Statistics and Computer Science

Subject

Topics in Algebraic Attacks on Cryptosystems

By

Hossein Hadipour

Under Supervision of

Dr. Hossein Sabzrou

Under Consultation of

Dr. Hossein Hajiabolhasan

A dissertation submitted to the graduate studies
office for the degree of Master of Science
in
Pure Mathematics

September 2016