

باسمه تعالی

گزارش پروژه درس رمزنگاری پیشرفته

**دانشگاه صنعتي شريف**

عنوان:

جستجوی خودکار تمایزگر تفاضل ناممکن

انجام‌دهنده:

پوریا دادخواه  
401201381

# بهار 1401

چکیده

رمزهای قالبی یکی از مهم‌ترین اولیه رمزنگاری برای ایجاد محرمانگی در کاربردهای مختلف می‌باشند که از قدیم مورد استفاده‌ بوده و در طول زمان با وجود پیدایش روش‌های دیگر رمزنگاری همچنان به دلیل ویژگی‌های بهینه در کاربرد‌های زیادی استفاده‌ می‌شوند و امنیت آن‌ها با حملات جدید مورد بررسی قرارگرفته است. یکی از مهم‌ترین حملات موجود برای رمزهای قالبی، حمله تفاضل ناممکن است که اولین اعمال شده بر الگوریتم 7 دوری AES نیز می‌باشد. چالش اصلی این حمله یافتن زوج مناسب برای ایجاد تناقض و بازیابی صحیح کلید از آن دور است که انجام این کار دستی توسط انسان نیازمند زمان و هزینه زیاد بوده و احتمال خطای محاسباتی نیز زیاد می‌باشد. در این راستا روش‌های مختلفی برای انجام این کار به صورت خودکار پیشنهاد شده‌اند که یکی از آن‌ها استفاده از مدل‌های حل مسائل بهینه‌سازی محدودیت است که در این پروژه با به‌کارگیری یک مدل آن‌ها به یک رمز قالبی سبک به نام simon ،که به دلیل ساختار دور ساده‌ای که دارد در کاربرد‌های با منابع محدود است می‌توان استفاده‌کرد، حمله می‌کنیم.

# فهرست مطالب

[1 مقدمه 1](#_Toc99753053)

[1-1 مرور ادبیات 1](#_Toc99753054)

[1-2 سازمان‌دهی گزارش 1](#_Toc99753055)

[2 حمله خودکار تفاضل ناممکن 4](#_Toc99753056)

[2-1 حمله تفاضل ناممکن 4](#_Toc99753057)

[2-2 بهینه‌سازی محدودیت 5](#_Toc99753062)

2-3 انکودکردن تریل‌های تراکم‌شده قطعی..........................................................................................................................................................6

[3 الگوریتم رمز simon 8](#_Toc99753063)

3-1 توصیف الگوریتم رمزگذاری 8

3-2 توصیف برنامه کلید .......................................................................................................................................................................................................9.

[4 خودکارسازی یافتن تمایزگر بر روی رمز simon 10](#_Toc99753065)

[5 جمع‌بندی و مراجع 12](#_Toc99753067)

[5-1 کارهای آتی 12](#_Toc99753068)

[5-2 مراجع 12](#_Toc99753069)

# فهرست اشکال

[شکل ‏1-2 شمای کلی حمله تفاضل ناممکن روی rd دور 4](#_Toc99753047)

شکل 3-1 تابع دور رمز 8…..……………………………………………………………………………………..………………………………………………………………………………………………simon

شکل 4-1 یافتن تمایزگر تفاضلی 11…….………………………………………………………………………………..……………………………………………………………………………………………

# فهرست جداول

[جدول 3-1: اندازه کلید‌های انواع رمز simon...................................................................................................................................................................8](#_Toc99753044)

جدول 3-2: دنباله اولیه تولید کلید ....................................................................................................................................................................................9

# مقدمه

فصل 1

در این پروژه قصد داریم یک تمایزگر خودکار- به این معنی استفاده از یک خانواده الگوریتم مشخص قابل پیاده‌سازی کامپیوتری به جای انجام اعمال دستی و یا نیمه دستی – برای حمله تفاصل ناممکن به الگوریتم رمز simon پیدا کنیم.

به این منظور ابتدا به جزییات بیشتری از حمله تفاضل ناممکن و روند کلی حمله از جمله یافتن بهترین زوج ورودی از طریق جدول DDT، رسیدن به شرط تناقض در تعداد دور مشخص و سپس توسعه به مرحله ورودی-خروجی به صورت قطعی و با احتمال 1، از طریق الگوریتم بازیابی کلید پرداختیم.

سپس با مطالعه مروری مقاله 5 از مراجع، با روند پیاده‌سازی متفاوت حمله تفاضلی با استفاده از constraint programing آشنا شدیم که به جای قرار دادن شرط خود روی تفاضل ورودی و خروجی و استفاده از روش‌های معمول-دستی، نیمه خودکار و وابسته به الگوریتم هدف- برای یافتن تناقض و سپس بازیابی کلید با روش‌های نه‌چندان کارا، مساله را به صورت واحد دیده و با مدل سازی درست به تعریف متغیرها، دامنه‌ها و نهایتا قیود متناسب با وقوع تناقض موردنیاز می‌پردازد و از یک الگوریتم حل مساله بهینه‌ساز- معمولا به صورت جعبه ‌سیاه- مانندSAT-MILP به یافتن تمایزگر مطلوب می‌پردازد.

در آخر با مطالعه مقاله الگوریتم simon [7]با این الگوریتم fiestel و کلمه محور و تابع دور این الگوریتم-متشکل از شیفت های چرخشی، اند بیت‌وایز و xor با کلید دور- و نحوه تولید زیرکلید‌های آشنا شدیم.

هم‌چنین علاوه بر مطالعات نظری فوق سورس‌کد‌های پیاده‌سازی الگوریتم simon، نمونه کد اعمال حمله جستجو خودکار تمایزگر تفاضل ناممکن بر رمز skinny و minizinc به عنوان یکی از ابزارهای موجود برای اعمال الگوریتم cp و حل مساله بهینه‌سازی را بررسی کردیم.

با این مطالعات نهایتا هدف این است که بتوانیم با توجه الگوریتم هدف علاوه بر مشخص کردن متغیر‌ها و فراپارامتر‌های مساله قیود متناسب این الگوریتم را برای دستیابی به یک تناقض پیدا کرده و با استفاده از یک الگوریتم cp حالت مختلف ورودی دادن را تست کرده ( مانند تعداد دور های هربخش از حمله، یا تعداد جفت ورودی‌های ورودی الگوریتم) و تمایزگری را انتخاب کنیم که پس از توسعه و بازیابی کلید، زمان و پیچیدگی مطلوبی در یافتم کلید درست داشته باشد.

## مرور ادبیات

حمله تفاضلی غیرممکن (ID)، که به صورت مستقل توسط بیهم و همکاران[1] و نیوودسن معرفی شده است، یکی از حملات مهم بر روی رمزگذارهای بلوکی است. به عنوان مثال، حمله ID به عنوان اولین حمله این نوع، موفق به شکست 7 دور از AES-128 شده است. حمله ID از تفاوت غیرممکن در یک رمزگذار بلوکی که معمولاً ناشی از انتشار کندی است، به منظور بازیابی کلید اصلی استفاده می‌کند.

فرآیند جستجو برای حمله ID روی رمزگذارهای بلوکی شامل دو فاز اصلی است: یافتن تمایزگر و انجام بازیابی کلید بر اساس تمایزگری که شناسایی شده است. یکی از روش‌های اصلی برای یافتن تشخیص‌دهنده‌های ID تکنیک «اشتباه در میانه» است [5]. ایده این روش این است که دو تفاوت (ماسک‌های خطی) را شناسایی کنیم که در نصفه‌ی راه از رمزگذار به طور قطعی به سمت جلو و عقب انتقال پیدا کنند، اما در نقطه میانی با یکدیگر در تناقض باشند. با این حال، اجرای این تکنیک نیازمند ردیابی انتشار تفاوت‌ها (یا ماسک‌های خطی) در سطح واژه یا بیت رمزگذار است که می‌تواند زمان‌بر و ممکن است خطاهایی داشته باشد. درباره بازیابی کلید، باید تشخیص‌دهنده را در دو طرف گسترش داده و پخش ویژگی‌های رمزنگاری بیشتری را با در نظر گرفتن پارامترهای حیاتی ردیابی کنیم. به طور کلی، یافتن یک حمله کامل بهینه ID معمولاً به یک مسئله بهینه‌سازی ترکیبیاتی دشوار منجر می‌شود، به خصوص زمانی که اندازه بلوک بزرگ است و تعداد راه‌حل‌های ممکن زیاد است. بنابراین، توسعه ابزارهای خودکار برای ارزیابی امنیت رمزگذارهای بلوکی در برابر این حملات حائز اهمیت است، به ویژه در طراحی و تحلیل ابزارهای رمزنگاری سبک، زیرا تحلیل دقیق امنیت به ما کمک می‌کند تا حاشیه‌های امنیتی را به حداقل برسانیم.

رویکرد دیگری برای حل مسائل بهینه‌سازی ناشی از حملات رمزنگاری، توسعه الگوریتم‌های اختصاصی است. به عنوان مثال، در CRYPTO ۲۰۱۶، دِربِز و فوک الگوریتم اختصاصی [2] را برای یافتن حملات DS-MITM و ID ارائه کردند. با این حال، توسعه و پیاده‌سازی الگوریتم‌های کارآمد مشکل است و نیاز به برنامه‌نویسی پیچیده دارد. علاوه بر این، سایر محققان ممکن است با چالش‌ها روبه‌رو شوند وقتی می‌خواهند این الگوریتم‌ها را به مسائل مشابه با ویژگی‌های متفاوتی تطبیق دهند، که ممکن است وقت‌گیر و دشوار باشد.

رویکرد دیگری، تبدیل مسئله رمزنگاری به یک مسئله رضایت محدود[[1]](#footnote-1) (CSP) یا مسئله بهینه‌سازی محدودیت دار[[2]](#footnote-2) (COP) است و سپس با استفاده از حل‌کننده‌های قابل استفاده برای برنامه‌نویسی محدودیت (CP) آن را حل کند. در سال‌های اخیر، چندین رویکرد مبتنی بر CP برای حل مسائل پیچیده تحلیل تمامی رمزنگاری‌های همسانی معرفی شده است که از نظر دقت و کارایی بهتر از روش‌های دستی یا اختصاصی است[3]. به عنوان مثال، در EUROCRYPT ۲۰۱۷، ساساکی و تودو ابزار خودکار جدیدی بر اساس حل‌کننده‌های برنامه‌نویسی خطی بهینه مختلط (MILP) [[3]](#footnote-3)برای یافتن تشخیص‌دهنده‌های ID ارائه کردند[4]. کوئی و همکاران رویکرد مشابهی برای کشف تشخیص‌دهنده‌های ID و ZC ارائه کردند [5]. اخیراً، سان و همکاران روش مبتنی بر CP جدیدی برای جستجوی تشخیص‌دهنده‌های ID و ZC در کنفرانس ToSC ۲۰۲۰ پیشنهاد کردند[6].

اگرچه روش‌های خودکار برای جستجوی حمله ID در سال‌های گذشته پیشرفت قابل توجهی داشته‌اند، اما هنوز محدودیت‌های پایه‌ایی وجود دارند:

- مدل‌های CP برای یافتن تشخیص‌دهنده‌های ID/ZC که در [3,4,5] پیشنهاد شده است، بر پایه غیرممکن بودن در هنگام ثابت بودن تفاوت/ماسک ورودی/خروجی است. با این حال، یافتن یک حمله بهینه بازیابی کلید مسئله‌ای بهینه‌سازی بر پایه ناممکن بودن است. بنابراین، مدل‌های CP قبلی برای یافتن تشخیص‌دهنده‌های ID نمی‌توانند به یک مدل یکپارچه بهینه‌سازی برای یافتن یک حمله کامل گسترش یابند. همچنین، مدل‌های CP قبلی برای یافتن تمایزگر‌های ID نیازمند بررسی جداگانه هر ویژگی ورودی/خروجی هستند. به عبارت دیگر، یافتن تمایزگر‌های ممکن زمانی که اندازه بلوک کافی بزرگ است، محاسباتی دشوار است.

- مدل CP پیشنهاد شده در [6] از تکنیک اشتباه در میان برای یافتن تشخیص‌دهنده‌های ID/ZC استفاده می‌کند. این رویکرد تفاوت‌ها/ماسک‌های ورودی/خروجی را ثابت نمی‌کند. با این حال، سازگاری بین دو بخش تمایزگر‌ در خارج از مدل CP با ایجاد یک حلقه بررسی می‌شود که الگوی فعالیت یک سلول حالت در نقطه دیدار باید در هر دوره ثابت باشد.

- تمام مدل‌های CP قبلی مربوط به حمله ID تنها بر روی یافتن تمایزگر‌های بلندتر تمرکز دارند. با این حال، عوامل مهم دیگری وجود دارند که بر پیچیدگی نهایی این حملات تأثیر می‌گذارند و با مدل کردن تنها بخش تشخیص‌دهنده قابل در نظر گرفتن نیستند. عواملی مانند موقعیت و تعداد سلول‌های فعال در ورودی/خروجی تمایزگر‌، تعداد فیلترها برای بررسی خاصیت‌های مطلوب در ورودی/خروجی تشخیص‌دهنده و تعداد بیت‌های کلید مشارکت شده در بازیابی کلید، تنها چندین پارامتر بحرانی هستند که بر پیچیدگی نهایی حمله تأثیر می‌گذارند اما تنها با مدل کردن بخش بازیابی کلید قابل توجه هستند. نشان داده شده است که بهترین حمله نیازمند تمایزگر‌ بلندتر نیست. بنابراین، یکپارچگی بخش بازیابی کلید و تشخیص برای یافتن حملات بهتر ID حائز اهمیت است.

- ابزار معرفی شده توسط دِربِز و فوک [2] تنها ابزاری است که برای یافتن حملات ID کامل وجود دارد. با این حال، این ابزار بر اساس یک الگوریتم اختصاصی پیاده‌سازی شده در C/C++ است و به اندازه روش‌های مبتنی بر CP گسترده نیست. علاوه بر این، این ابزار نمی‌تواند تمام پارامترهای بحرانی حملات ID را در نظر بگیرد تا پیچیدگی نهایی را به حداقل برساند. همچنین، این ابزار قادر به یافتن حملات ID مرتبط با کلید(یا tweakey) نیست.

ما یک روش خودکار جدید عمومی، مبتنی بر CP و با استفاده آسان را برای یافتن حملات ID کامل پیشنهاد می‌کنیم که محدودیت‌های فوق را برطرف می‌کند. بر خلاف تمام مدل‌های قبلی CP برای این متمایزکننده‌ها، که مبتنی بر عدم رضایت هستند، مدل CP ما برای یافتن متمایزکننده‌ها بر رضایت‌پذیری تکیه می‌کند. به این ترتیب، هر راه حل مدل های CP ما با یک ID مطابقت دارد. این ویژگی کلیدی ما را قادر می سازد تا مدل های متمایز خود را به یک مدل یکپارچه برای یافتن یک حمله بازیابی کلید بهینه گسترش دهیم. علاوه بر این، مدل CP یکپارچه ما از پل زدن کلید و تکنیک‌های میانی استفاده می‌کند. برای نشان دادن سودمندی روش خود، آن را روی Simon[7] می‌کنیم.

## سازمان‌دهی گزارش

در ادامه مقاله ابتدا در فصل 2 به جزییات حمله تفاضل ناممکن می‌پردازیم و با معرفی روش‌های بهینه‌سازی محدودیت روش کلی یافتن تمایزگر با آن‌ها را بیان می‌کنیم. در فصل سوم الگوریتم رمز قالبی سبک simon را معرفی می‌کنیم تا با جزییات هر دور آن و اعمال قیود مناسب آشنا شویم و در فصل 4 روش یافتن تمایزگر روی الگوریتم خاص simon را بررسی می‌کنیم. فصل 5 نیز جمع‌بندی مقاله و ادامه راه را بیان می‌کند.

# حمله خودکار تفاضل ناممکن

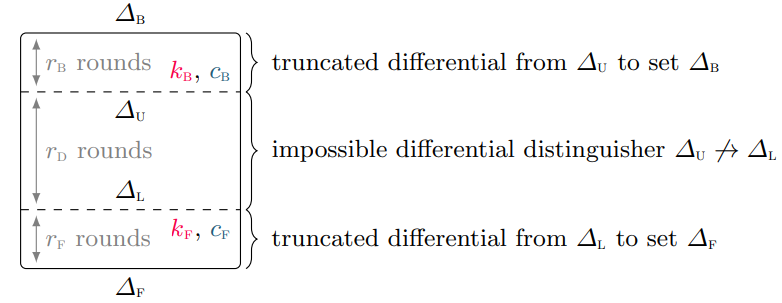
فصل 2

در اینجا، اصول حمله ID یاداوری می‌کنیم و همچنین، نمادهایی که در بقیه‌ی این مقاله استفاده می‌شوند را معرفی می‌کنیم. سپس مدل‌های حل بهینه‌سازی محدودیت را معرفی می‌کنیم و اصول تعریف قیود عمومی را بررسی می‌کنیم.

## حمله تفاضلی ناممکن

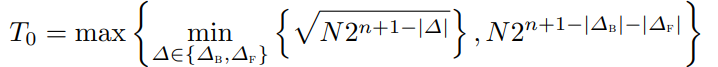
حمله تفاضلی غیرممکن به طور مستقل توسط بیهم و همکاران [1] و نیز نیوودسن معرفی شد. ایده اصلی یک حمله تفاضلی غیرممکن، بهره‌برداری از یک تفاضل غیرممکن در یک رمزگذار برای بازیابی کلید است، با دوران تمامی کلیدهای ممکن که به چنین تفاضل غیرممکنی منجر می‌شوند، کنار می‌آییم. اولین نیاز حمله ID یک تمایزگر ID است، به عبارتی تفاوت ورودی که هرگز به یک تفاوت خروجی خاص نمی‌تواند انتشار پیدا کند. سپس، ما تمایزگر ID را به عقب و به جلو توسعه می‌دهیم. کلیدی که یک جفت داده را به تفاضل غیرممکنی جزئی رمزنگاری/رمزگشایی می‌کند، قطعاً معتبر نیست. هدف، دور انداختن تعداد زیادی از کلیدهای اشتباه است. در نهایت، ما با جستجوی جامع در میان کلیدهای باقی‌مانده، کلید را به صورت یکتا بازیابی می‌کنیم.

ما تحلیل پیچیدگی حمله ID را بر اساس منابع [8,9] بیان می‌کنیم و برای جزییات بیشتر به آن ارجاع می‌دهیم. فرض کنید E یک رمزگذار بلوکی با اندازه بلوک n بیت و کلید k بیت باشد. همانطور که در شکل 1 نشان داده شده است، فرض کنید برای rd دور E به عنوان Ed، یک تفاضل غیرممکن وجود داشته باشد. فرض کنید (resp) با احتمال 1 به صورت عقب (resp.جلو) از طریق (resp. EF) به (resp) انتقال پیدا کند و (resp)ابعاد فضای برداری (resp) را نشان دهد. فرض کنید cb (resp. cf) تعداد شرایط بیتی باشد که باید برای (resp. ) برآورده شود، به عبارت دیگر علاوه بر این، فرض کنید KB (resp. KF) اطلاعات کلید را، به طور کلی بیت‌های کلید فرعی، نشان دهد که در Eb (resp. Ef) درگیر هستند. با این فرض‌ها می‌توانیم حملات ID را به سه مرحله تقسیم کنیم:

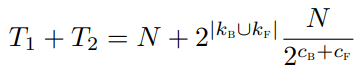


شکل2-1. شمای کلی حمله تفاضل ناممکن روی rd دور

- مرحله 1: تولید جفت. با دسترسی به رمزگذار رمزنگاری (و احتمالاً رمزگشایی)، ما N جفت را تولید می‌کنیم به گونه‌ای که x⊕y ∈ ∆B و E(x)⊕E(y) ∈ ∆F و آن‌ها را ذخیره می‌کنیم. این یک مسئله تولد تولد تولد محدود است و مطابق با [9]پیچیدگی این مرحله عبارت است از:



- مرحله 2: حدس زدن و فیلتر کردن. هدف از این مرحله، دور انداختن تمامی زیرکلیدهای موجود در KF ∪ KB است که توسط حداقل یکی از جفت‌های تولید شده نامعتبر شده‌اند. بجای حدس زدن همه‌ی زیرکلیدهای KF ∪ KB به صورت یکجا و آزمودن آن‌ها با تمام جفت‌ها، می‌توانیم این مرحله را با استفاده از تکنیک قطع زودرس [10] بهینه کنیم: KF ∪ KB را به زیرمجموعه‌های کوچکتری، به طور معمول کلیدهای دوره، تقسیم می‌کنیم و آن‌ها را مرحله به مرحله حدس زده و آزمایش می‌کنیم. در هر مرحله، با بررسی اینکه آیا جفت‌ها شرایط مسیر تفاضلی میانبری قطع شده را ارضا می‌کنند، تعداد حداقلی از رمزنگاری/رمزگشایی جزئی را کاهش می‌دهیم [10]:



- مرحله 3: جستجوی جامع. احتمال اینکه یک کلید اشتباه از طریق مرحله حدس زدن و فیلتر کردن باقی بماند، برابر است با بنابراین، تعداد کلیدهای باقی‌مانده پس از انجام مرحله حدس زدن و فیلتر کردن برابر است با P · در میانگین. از سوی دیگر، مرحله حدس زدن و فیلتر کردن شامل بیت از اطلاعات کلید نیست. به عبارت دیگر، برای تعیین یکتای کلید، ما باید به طور جامع در فضایی به اندازه جستجو کنیم.

سپس، پیچیدگی زمان کلی حمله ID به صورت زیر است:



که در آن CE هزینه یک رمزنگاری کامل است و CE’ نسبت هزینه یک رمزنگاری جزئی به رمزنگاری کامل را نشان می‌دهد.

برای حفظ پیچیدگی داده کمتر از کتابخانه‌ی کد کامل، ما نیاز داریم T0 < 2^n باشد. علاوه بر این، برای بازیابی حداقل یک بیت از اطلاعات کلید در مرحله حدس زدن و فیلتر کردن، باید P < 1/2 بود. توجه کنید که معادله 2 پیچیدگی زمانی میانگین مرحله حدس زدن و فیلتر کردن است؛ برای هر حمله ID، باید پیچیدگی آن را به طور دقیق ارزیابی کنیم تا اطمینان حاصل کنیم که این محدودیت در عمل برقرار است.

## مسئله بهینه‌سازی محدودیت

مسئله رضایت از محدودیت (CSP) یک مسئله ریاضی است که شامل مجموعه‌ای از محدودیت‌ها بر روی مجموعه‌ای از متغیرهاست که باید برآورده شوند. به طور دقیق‌تر، یک CSP به صورت سه‌تایی(X,D,C) تعریف می‌شود، جایی که مجموعه‌ای از متغیرها، مجموعه‌ای از دامنه‌ها به طوری کهXi ∈ Di است و مجموعه‌ای از محدودیت‌ها را نشان می‌دهد. هر محدودیت Cj ∈ C به صورت یک تاپل تعریف می‌شود، جایی که

و Rj یک رابطه بر روی دامنه‌های مربوطه است، به عبارت دیگر، .

هر نحوه‌ی تخصیص مقادیر به متغیرها که تمام محدودیت‌ها را برآورده کند، به عنوان یک راه‌حل امکان‌پذیر برای یک مسئله CSP در نظر گرفته می‌شود. مسئله بهینه‌سازی محدودیت (COP) با اضافه کردن یک تابع هدف جهت کمینه کردن یا بیشینه کردن معیاری، CSP را توسعه می‌دهد.

فرایند جستجو برای یافتن راه‌حل یک مسئله CSP یا COP به عنوان برنامه‌نویسی محدودیت (CP) شناخته می‌شود و ابزارهایی که جستجو را انجام می‌دهند به عنوان حل‌کننده‌های CP نامیده می‌شوند.

یک روش حل،استفاده از MiniZinc [11] برای مدل‌سازی و حل مسائل CSP و COP با اعداد صحیح و اعداد حقیقی است. MiniZinc امکان مدل‌سازی مسائل CSP و COP را به صورتی سطح بالا و مستقل از حل‌کننده‌ها فراهم می‌کند. این ابزار مدل را به زبان FlatZinc ترجمه می‌کند که یک زبان استاندارد است و توسط مجموعه‌ی گسترده‌ای از حل‌کننده‌های CP پشتیبانی می‌شود.

برای مسائل CSP/COP با اعداد صحیح، از Or-Tools [12] و برای مسائل CSP/COP با اعداد حقیقی، از Gurobi [17] به عنوان حل‌کننده استفاده می‌شود.

مسائل بهینه سازی خطی که نیاز به اعداد صحیح برخی از متغیرها دارند، برنامه های عدد صحیح مختلط (MIP) نامیده می شوند.

این متغیرها می توانند به چند صورت ایجاد شوند:

* متغیرهای عدد صحیحی که تعداد اقلام را نشان می‌دهند، مانند ماشین‌ها یا دستگاه‌های تلویزیون، و مشکل این است که تصمیم می‌گیریم چه تعداد از هر اقلام را تولید کنیم تا سود را به حداکثر برسانیم. به طور معمول، چنین مسائلی را می توان به عنوان مسائل بهینه سازی خطی استاندارد تنظیم کرد، با این شرط که متغیرها باید اعداد صحیح باشند. بخش بعدی نمونه ای از این نوع مشکلات را نشان می دهد.
* متغیرهای بولی که تصمیماتی را با مقادیر 0-1 نشان می دهند. به عنوان مثال، مشکلی را در نظر بگیرید که شامل انتساب کارگران به وظایف است. برای راه‌اندازی این نوع مشکل، می‌توانید متغیرهای بولی xi,j را تعریف کنید که اگر کارگر i به وظیفه j اختصاص داده شود برابر با 1 و در غیر این صورت 0 است.

گوگل چند راه برای حل مشکلات MIP ارائه می دهد:

* MPSolver : پوششی برای چندین حل کننده MIP شخص ثالث که از تکنیک های استاندارد شاخه و کران استفاده می کنند.
* حل کننده CP-SAT : یک حل کننده برنامه نویسی محدودیت که از روش های SAT (رضایت پذیری) استفاده می کند.
* حل کننده اصلی CP : حل کننده برنامه نویسی محدودیت.

هیچ قانون اساسی برای تصمیم گیری در مورد استفاده از حل کننده MIP یا حل کننده CP-SAT وجود ندارد. به عنوان یک راهنمای تقریبی:

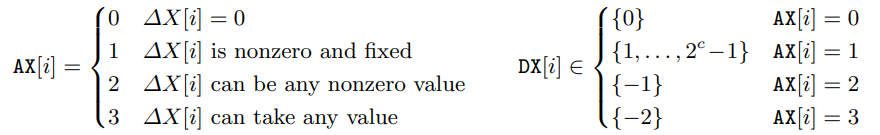
* حل‌کننده‌های MIP برای مسائلی مناسب‌تر هستند که می‌توانند به عنوان یک LP استاندارد تنظیم شوند، اما با متغیرهای عدد صحیح دلخواه، مانند مثال اول بالا.
* حل کننده CP-SAT برای مسائلی که اکثر متغیرها در آنها Boolean هستند، مناسب تر است.

برای MIP های معمولی که دارای هر دو متغیر اعداد صحیح و بولی هستند، اغلب تفاوت واضحی در سرعت بین دو حل کننده وجود ندارد، بنابراین انتخاب شما ممکن است به ترجیح شخصی بستگی داشته باشد.

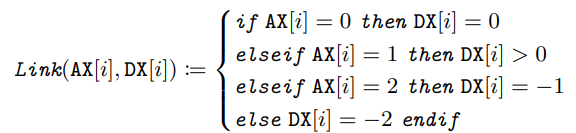
## انکود کردن تریل‌های تراکم شده قطعی

در این قسمت، روشی که در [6] پیشنهاد شده برای انکودکردن تریل‌های تراکم شده قطعی را بازبینی می‌کنیم. در اینجا دو نوع متغیر برای رمزگذاری تریل‌های تراکم شده قطعی تعریف می‌کنیم. فرض کنید نشان دهنده تفاضل حالت داخلی X در یک رمزگذار بلوکی n-بیتی E باشد، که در آن n = m · c و

ما از متغیر صحیح برای انکود الگوی فعالیت و متغیر صحیح دیگر برای انکود مقدار واقعی تفاضل c-بیتی استفاده می‌کنیم:



سپس AX[i] و DX[i] را برای تمام به صورت زیر مرتبط می‌کنیم:

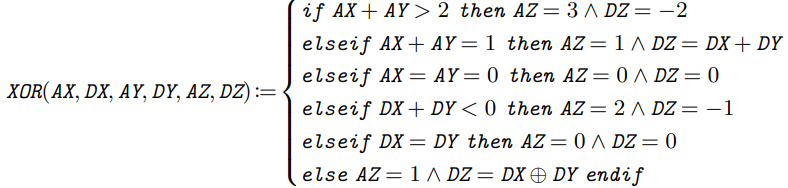


زبان MiniZinc عبارت شرطی if-then-else-endif را پشتیبانی می‌کند، بنابراین نیازی به تبدیل به نامساوی‌های عدد صحیح نداریم. حال، می‌خواهیم به طور خلاصه قواعد انتشار تریل‌های تراکم شده قطعی را تشریح کنیم.

گزاره ۱ (شاخه‌بندی): برای که و Z = Y = X، انتقالات معتبر برای تریل‌های تراکم شده قطعی تفاضلی با شرایط زیر صادق است:



گزاره ۲ (XOR): برای که و Z = X ⊕ Y، انتقالات معتبر برای تریل‌های تراکم شده قطعی با شرایط زیر صادق است:



گزاره ۳ :(s-box)فرض کنید یک جعبه‌ی -cبیتی S باشد و Y = S(X) باشد، انتقالات معتبر برای تریل‌های تراکم شده قطعی تفاضلی با شرایط زیر صادق است:



برای انکود کردن ماتریس‌های غیر-MDS مانند ماتریس استفاده شده در simon می‌توانیم از قواعد XOR و شاخه‌بندی برای انکود کردن انتشار استفاده کنیم.

# الگوریتم رمز simon

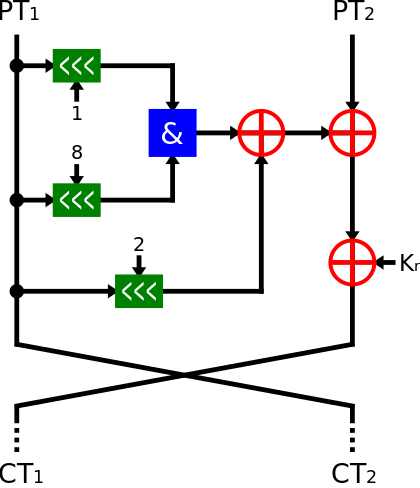
فصل 3

Simon خانواده‌ای از رمزنگارهای سبک وزن است که توسط آژانس امنیت ملی (NSA) در ژوئن 2013 منتشر شد. Simon برای عملکرد در پیاده‌سازی سخت‌افزار بهینه‌سازی شده است، در حالی که الگوریتم خواهرش، Speck، برای پیاده‌سازی نرم‌افزار بهینه شده است. [7]

NSA کار روی رمزهای Simon و Speck را در سال 2011 آغاز کرد. این آژانس پیش بینی کرد که برخی از آژانس ها در دولت فدرال ایالات متحده به رمزی نیاز دارند که در مجموعه متنوعی از دستگاه های اینترنت اشیا به خوبی عمل کند و در عین حال سطح قابل قبولی از امنیت را حفظ کند.

## توصیف الگوریتم رمزگذاری

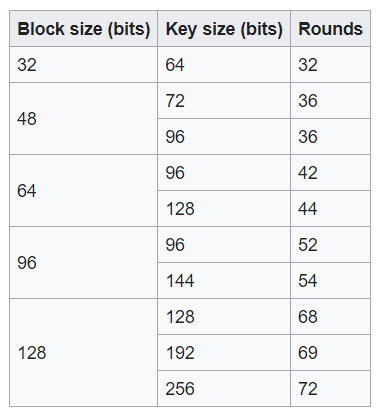
رمزگذار Simon یک رمزگذار Feistel متعادل با واژه‌ای n-بیتی است و به همین دلیل طول بلوک برابر با 2n است. طول کلید نیز ضریبی از n با مقادیر 2، 3 و 4 می‌باشد و به عنوان m نمایش داده می‌شود. بنابراین، یک پیاده‌سازی رمزگذار Simon با علامت Simon2n/nm نشان داده می‌شود. به عنوان مثال، Simon64/128 به رمزگذاری اشاره دارد که بر روی بلوک متن ساده 64 بیتی (n = 32) عمل می‌کند و از یک کلید 128 بیتی استفاده می‌کند.[13] بخش بلوک رمزگذار در تمام پیاده‌سازی‌های Simon یکسان است؛ اما منطق تولید کلید بستگی به پیاده‌سازی از 2، 3 یا 4 کلید دارد.



شکل 3-1: تابع دور رمز simon

Simon از ترکیب‌های زیر از اندازه بلوک، اندازه کلید و تعداد دورهای پشتیبانی می‌کند:

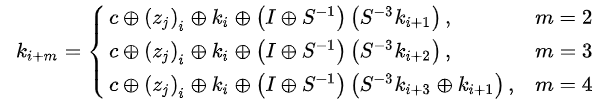
جدول 3-1: اندازه کلید‌های انواع رمز simon



## توصیف برنامه کلید

ماژول برنامه کلید به صورت ریاضی به صورت زیر توصیف می‌شود:

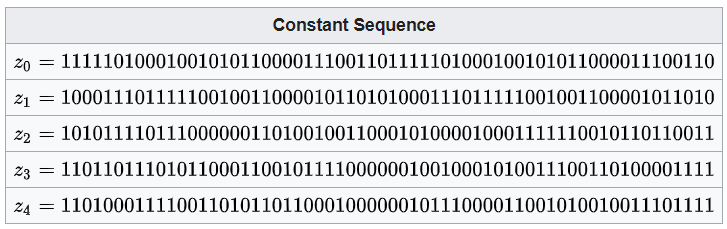
SJ نشان‌دهنده یک شیفت چرخشی به چپ با اندازه j بیت است.



ساختار برنامه کلید ممکن است متعادل باشد یا نباشد. تعداد کلمات کلید m برای تعیین ساختار توسعه کلید استفاده می‌شود، که نتیجه آن عرض بیت کلید است که برابر است با m\*n توسعه کلمه کلید شامل یک شیفت به راست، XOR و یک دنباله ثابت است که با نمایش داده می‌شود. بیت یکبار در هر دور بر روی بیت پایین‌تر کلمه کلید عمل می‌کند.[7]

دنباله ثابت توسط یک LFSR ایجاد می‌شود. توالی منطقی ثابت بر اساس مقدار اندازه کلید و بلوک تعیین می‌شود. LFSR توسط یک فیلد 5 بیتی ایجاد می‌شود. بیت ثابت در هر دور بر روی بیت پایین‌تر کلمه کلید عمل می‌کند تا آنتروپی غیر وابسته به کلید به برنامه کلید اضافه شود. هر دنباله منطق متفاوتی دارد؛ با این حال، شرایط اولیه برای رمزگذاری برابر است. شرایط اولیه برای LFSR در هر دور برای رمزگشایی متغیر است.

جدول 3-2: دنباله اولیه تولید کلید



# خودکارسازی یافتن تمایزگر بر روی رمز simon

فصل 4

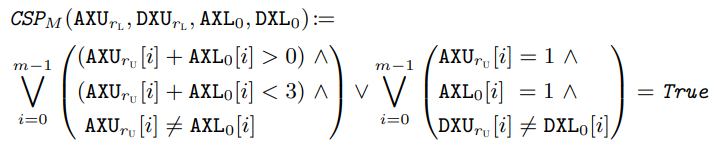
در این بخش، به مدل‌سازی تمایزگر برای حمله ID می‌پردازیم. ساختار متمایزکننده‌های ID بر اساس رویکرد همین‌میان است که به تکنیک Miss-in-the-Middle برمی‌گردد [5،6]. ایده این روش، پیدا کردن دو تفاوت (ماسک‌های خطی) است که با اطمینان به صورت ایابی و برگشتی از طریق رمزگذاری منتقل می‌شوند اما در مرکز با یکدیگر در تناقض هستند. تضاد بین این انتقال‌ها منجر به وجود یک تفاضل غیرممکن (یا ترکیب خطی بی‌طرف) می‌شود.

فرض کنید دنباله Ed را در نظر بگیرید که rd مرحله از یک رمزگذار E را نشان می‌دهد. همچنین فرض می‌کنیم اندازه بلوک E برابر با n بیت است، جایی که n = m · c با c اندازه سلول و m تعداد سلول‌ها است. ما تکنیک Miss-in-the-Middle را به یک مسئله ریاضی مبتنی بر محدودیت تبدیل می‌دهیم تا به صورت خودکار متمایزکننده‌های ID را پیدا کنیم. ما Ed را به دو بخش تقسیم می‌کنیم: بخش بالا Eu که ru مرحله را پوشش می‌دهد و بخش پایین El که rl مرحله را پوشش می‌دهد. ما به مسیرهای بالا و پایینی که برای Eu و El کشف شده‌اند، به ترتیب به عنوان مسیر بالا و پایین اشاره می‌کنیم. حالت داخلی Eu پس از r مرحله را با XUr نشان می‌دهیم و حالت داخلی El پس از r مرحله را با XLr نشان می‌دهیم. نقطه تلاقی که Eu و El به آن می‌رسند، با XUru یا (XL0) نشان داده می‌شود.

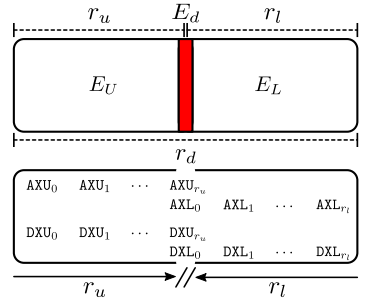
ما از متغیرهای صحیح AXUr و AXLr برای رمزگذاری الگوی فعالیت XUr و XLr استفاده می‌کنیم و از DXUr و DXLr برای رمزگذاری مقادیر تفاوت c-بیتی واقعی در مرحله r Eu و El استفاده می‌کنیم.

ما انتقال تفاضل‌های خطی کوتاه شده و قطعی از طریق Eu و El را به عنوان دو مسئله مستقل CSP با استفاده از قوانینی که در بخش2.3 توضیح داده شد، رمزگذاری می‌کنیم. برای جلوگیری از راه‌حل‌های ساده، محدودیت‌هایی را اضافه می‌کنیم که تضاد بین مسیرهای بالا و پایین در حداقل یک سلول در نقطه تلاقی تضمین کنند.

در نهایت، دو محدودیت زیر را اضافه کرده و با تعریف CSPهای زیر به یافتن تمایزگر می‌پردازیم. CSPd را به عنوان اتحاد CSPu، CSPl و CSPM ( وقوع تناقض در نقطه برخورد) تعریف می‌کنیم. هر راه‌حل قابل قبول در CSPD متناظر با یک تفاضل غیرممکن است.



روش ما برای جستجوی متمایزکننده‌ها تفاوت‌های مهمی با روش ارائه شده در[6] دارد. در حالی که [6] CSPu و CSPl را به صورت مجزا در یک حلقه حل می‌کند و الگوی فعالیت یک سلول در نقطه تلاقی را در هر تکرار ثابت می‌کند، مدل ما اطمینان حاصل می‌کند که هر راه‌حل CSPd متناظر با یک متمایزکننده ID است. به علاوه، ما ارزش مدل خود را در ورودی / خروجی یا در نقطه تلاقی محدود نمی‌کنیم، که امکان گسترش مدل ما برای بازیابی کلید و ساخت یک مسئله بهینه‌سازی محدودیتی جامع برای یافتن حملات ID نزدیک به بهینه را فراهم می‌کند.



شکل 4-1: یافتن تمایزگر تفاضلی

مدل ما برای متمایزکننده‌ها فرض می‌کند که کلید‌های مرحله مستقل هستند و همه انتشارات ممکن، حتی انتشاراتی که به دلیل وابستگی کلید‌های مرحله به یکدیگر امکان ندارد، را در نظر می‌گیرد. ما همچنین از جعبه S به عنوان یک جعبه سیاه استفاده می‌کنیم و از ساختار داخلی آن بهره‌برداری نمی‌کنیم. هرچند در simon عملا sbox نداریم و با شیفت مدل می‌شود.

برای پیاده‌سازی ما به مشکل linsence برای استفاده از مدل gurobi خوردیم و نتوانستیم مدل را با روش MILP پیاده‌‌کنیم. در عوض به روش SAT که در فصل دوم معرفی کردیم روی‌آوردیم و با استفاده از کتابخانه cryptoSMT [15] که کتابخانه‌ای جامع برای تحلیل حملات مختلف با استفاده از برنامه‌ریزی محدودیت SAT است، به یافتن تمایز مناسب برای تحویل به بازیابی کلید پرداختیم. نحوه کار با این کتابخانه در فایل‌های ضمیمه آمده و به این صورت است که پس از انتخاب رمز موردنظر، حمله را با مقادیر دلخواه پارس می‌کنیم و در صورت نیاز شروط اولیه نیز بر روی آن قرار می‌دهیم( مثل وزن مدنظر در اختلاف برای موراد خاص). نمونه‌ای از یافتن اختلاف با وزن‌های همینگ مختلف را مشاهده‌ می‌کنیم:

simon - Rounds: 8 Wordsize: 16

-----

Weight: 0 Time: 0.0s

Weight: 1 Time: 0.08s

Weight: 2 Time: 0.16s

Weight: 3 Time: 0.44s

Weight: 4 Time: 0.74s

Weight: 5 Time: 0.89s

در آخر برای مقایسه الگوریتم پیاده‌شده با سایر روش‌ها می‌بایست الگوریتم بازیابی کلید آماده را به آن متصل می‌کردیم که حمله کامل شود ولی نتوانستیم کد پیاده‌شده در مقاله مربوطه در صورت پروژه را استخراج کنیم و از این مقایسه بازماندیم.هرچند طبق روابط نوشته‌شده انتظار می‌رود زمان کل حمله حداقل بدتر از روش ‌های مشابه نبوده چرا که سایر پارامترهای موثر بهینه‌سازی را نیز مدل کرده‌ایم و اختلاف ثابت ورودی را نیز به عنوان ویژگی‌ می‌توانیم تنظیم کنیم.

# جمع‌بندی و مراجع

فصل 5

در این پروژه پس از معرفی حمله تفاضل ناممکن و روند اعمال آن، روش‌های پیاده‌سازی آن را با ابزار بهینه‌سازی محدودیت بیان کردیم و تفاوت روش‌های مختلف این‌ پیاده‌سازی‌ها را اشاره کردیم. تفاوت اصلی روش بیان شده و روش‌های موجود در فیکس نکردن دنباله ورودی و خروجی در یک تفاضل خاص است که باعث میشود برای بازیابی کلید درجه‌ آزادی بیشتری داشته باشیم و پیچیدگی کلی حمله کاهش یابد. درنهایت این روش را برای الگوریتم رمز سبک simon تست کردیم و یک نمونه یافتن تفاضل مشخص را در آن زمانبندی کردیم.

## کارهای آتی

در ادامه باید کد پیاده‌شده را به یک الگوریتم بازیابی کلید مناسب متصل کنیم تا پیچیدگی کل حمله را بتوانیم به درستی مقایسه کنیم و درصورت عدم براوردن نیاز از cp های دیگر و ویژگی‌های بهتر استفاده کنیم.

## مراجع

|  |  |
| --- | --- |
|  | [1] Biham, E., Biryukov, A., Shamir, A.: Cryptanalysis of skipjack reduced to 31 rounds using impossible differentials. In: EUROCRYPT 1999. LNCS, vol. 1592, pp. 12–23. Springer (1999). https://doi.org/10.1007/3-540-48910-X\_2.  [2] Derbez, P., Fouque, P.A.: Automatic search of meet-in-the-middle and impossible differential attacks. In: CRYPTO 2016. LNCS, vol. 9815, pp. 157–184. Springer (2016)  [3] Hadipour, H., Nageler, M., Eichlseder, M.: Throwing boomerangs into feistel structures: Application to CLEFIA, WARP, LBlock, LBlock-s and TWINE. IACR Trans. Symmetric Cryptol. 2022(3), 271–302 (2022). https://doi.org/10.46586/ tosc.v2022.i3.271-302  [4] . Sasaki, Y., Todo, Y.: New impossible differential search tool from design and cryptanalysis aspects. In: EUROCRYPT 2017. pp. 185–215. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3  [5] Cui, T., Chen, S., Jia, K., Fu, K., Wang, M.: New automatic search tool for impossible differentials and zero-correlation linear approximations. IACR Cryptology ePrint Archive, Report 2016/689 (2016), https://eprint.iacr.org/2016/689  [6] Sun, L., Gerault, D., Wang, W., Wang, M.: On the usage of deterministic (relatedkey) truncated differentials and multidimensional linear approximations for spn ciphers. IACR Transactions on Symmetric Cryptology 2020(3), 262–287 (Sep 2020). https://doi.org/10.13154/tosc.v2020.i3.262-287  [7] Observations on the SIMON block cipher family Stefan Kölbl1, Gregor Leander2, and Tyge Tiessen1 2015  [8] Boura, C., Lallemand, V., Naya-Plasencia, M., Suder, V.: Making the impossible possible. Journal of Cryptology 31(1), 101–133 (2018). https://doi.org/10. 1007/s00145-016-9251-7 11.  [9] Boura, C., Naya-Plasencia, M., Suder, V.: Scrutinizing and improving impossible differential attacks: applications to clefia, camellia, lblock and simon. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 179–199. Springer (2014). https://doi.org/10.1007/ 978-3-662-45611-8\_10  [10] Lu, J., Kim, J., Keller, N., Dunkelman, O.: Improving the efficiency of impossible differential cryptanalysis of reduced camellia and MISTY1. In: CT-RSA 2008. LNCS, vol. 4964, pp. 370–386. Springer (2008)  [11] Nethercote, N., Stuckey, P.J., Becket, R., Brand, S., Duck, G.J., Tack, G.: Minizinc: Towards a standard CP modelling language. In: CP 2007. LNCS, vol. 4741, pp. 529–543. Springer (2007)  [12] . Perron, L., Furnon, V.: OR-Tools, https://developers.google.com/ optimization/  [13] Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2022), https: //www.gurobi.com  [14]  [The Simon and Speck Families Of Lightweight Block Ciphers](http://eprint.iacr.org/2013/404.pdf) *(PDF)*. ePrint*. Retrieved 2016-06-16*.  [15] https://en.wikipedia.org/wiki/Simon\_(cipher)#Cryptanalysis |
|  |  |

1. Constraint satisfaction problem [↑](#footnote-ref-1)
2. constraint optimization problem [↑](#footnote-ref-2)
3. mixed integer linear programming [↑](#footnote-ref-3)