دوفصل‌نامه علمی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)
سال ۱۴۰۴، جلد ۲۳، شماره ۱ (۱۲۰–۱۳۵)
https://monadi.isc.org.ir

مروری بر حملات القاء خطا گذرا به رمزهای قالبی

مهدی فرزانگان و هادی سلیمانی*

پژوهشکده فضای مجازی، دانشگاه شهید بهشتی، تهران، ایران

اطلاعات مقاله

کلمات کلیدی:
حمله القاء خطا
حمله فعال
روش‌های مقابله با القاء خطا
اقدام متقابل القاء خطا

نوع مقاله: مروری

چکیده

حملات القاء خطا دسته‌ایی از حملات فعال در مدل جعبه خاکستری هستند که در آن مهاجم با دسترسی فیزیکی به دستگاه رمزنگاری، به صورت عمدی خطایی به دستگاه اعمال می‌کند. این خطا منجر به تغییر در مقادیر میانی و خروجی دستگاه شده و مهاجم با تحلیل تفاوت‌های ایجادشده، اطلاعات حساس مانند کلید رمزنگاری را استخراج می‌کند. به دلیل سهولت اجرا و کارایی بالا، این حملات تهدیدی جدی برای امنیت پیاده‌سازی‌های رمزنگاری محسوب می‌شوند و می‌توانند حتی سیستم‌های دارای مکانیسم‌های محافظتی را نیز تحت تأثیر قرار دهند. روند اخیر نشان می‌دهد که میزان تحقیقات در این حوزه به طور قابل توجهی رشد کرده و به عنوان یک زمینه تحقیقاتی فعال در مبحث پیاده‌سازی الگوریتم‌های رمزنگاری به حساب می‌آید. از این رو نیاز به یک مطالعه جامع که دربرگیرنده حملات القاء خطا و راه‌های مقابله با آن باشد، احساس می‌شود. در این مقاله، ضمن معرفی حمله القاءخطا، به ویژگی‌ها، نقاط قوت و ضعف این حمله می‌پردازیم. سپس، در ادامه، روش‌های تحلیل حملات را بررسی و مقایسه می‌کنیم و در انتها به روش‌های مقابله با این حملات می‌پردازیم.

© ۱۴۰۴ انجمن رمز ایران

۱ مقدمه

در سال‌های اخیر، با گسترش روزافزون تکنولوژی و به دنبال آن نیاز به امنیت اطلاعات در حوزه‌های مختلفی همچون نظامی، صنعتی، اقتصادی و حتی فناوری‌های نوینی مانند اینترنت‌اشیا، استفاده از الگوریتم‌های رمزنگاری به یکی از ضرورت‌های حیاتی تبدیل شده است. این سیستم‌های رمزنگاری در برابر تهاجم‌های فیزیکی آسیب‌پذیر هستند. دسته‌ای از این تهدیدات در حوزه امنیت که توجه جوامع علمی را به طور ویژه به خود جلب کرده، حملات القاء خطا [1] هستند. در این نوع حملات، مهاجم با ایجاد اختلالات عمدی در فرآیندهای پردازشی، با فرض بر اینکه به صورت فیزیکی به دستگاه دسترسی دارد، تلاش می‌کند تا محاسبات داخلی را به گونه‌ای تحت تأثیر قرار دهد که اطلاعات محرمانه‌ای نظیر کلیدهای رمزنگاری را استخراج کند. مهاجمان با استفاده از تکنیک‌هایی نظیر تغییر ناگهانی ولتاژ، اعمال گلیچ در سیگنال ساعت یا استفاده از پالس‌های لیزر، دستگاه را در شرایط غیرعادی قرار می‌دهند و از این طریق، محاسبات داخلی را دچار اختلال می‌کنند. تحلیل خروجی‌های حاصل از این شرایط غیرعادی، به مهاجم امکان می‌دهد تا با مقایسه خروجی‌های صحیح و خطادار، به اطلاعات محرمانه پی ببرد [۱].

سناریوهای تأثیر القاء خطا بر یک سیستم بسیار گسترده هستند و می‌توانند از تغییر عملکرد عادی دستگاه گرفته تا حذف یا دستکاری داده‌ها، دسترسی غیرمجاز به بخش‌های محافظت‌شده حافظه و حتی دور زدن مکانیزم‌های امنیتی را در بر بگیرند. با توجه به افزایش تحقیقات در این زمینه، حملات القاء خطا به عنوان یکی از جدّی‌ترین چالش‌های امنیتی در پیاده‌سازی‌های رمزنگاری سبک‌وزن مطرح شده‌است. انعطاف‌پذیری بالای این حملات، اهمیت اتخاذ تدابیر امنیتی و روش‌های مقاوم‌سازی را بیش از پیش آشکار می‌سازد و لزوم ارائه راهکارهای پیشرفته برای

*نویسنده مسئول
آدرس‌های رایانامه: m.farzanegan@mail.sbu.ac.ir (مهدی فرزانگان)، h_soleimany@sbu.ac.ir (هادی سلیمانی)
© ۱۴۰۴ تمامی حقوق متعلق به انجمن رمز ایران است.

[1] Fault Attack

کاهش تهدیدات بالقوه را برجسته می‌کند [۲].

۲ مدل خطا

برای ارتباط بین یک حمله‌ی فیزیکی و یک حمله‌ی الگوریتمی، از مدل خطا استفاده می‌شود. مدل خطا توصیف‌کننده‌ی نحوه‌ی واکنش یک دستگاه یا بخشی از آن در برابر یک روش خاص از تزریق خطاست [۳].

مدل‌های مختلفی برای القاء خطا بسته به توانایی مهاجم تعریف می‌شوند. هر چه دقت مدل خطا بالاتر باشد، امکان طراحی حملات قدرتمندتری فراهم می‌شود، اما اجرای این حملات در عمل ممکن است پیچیده‌تر و دشوارتر باشد. بر اساس توانایی مهاجم، مدل‌های متفاوتی را می‌توان برای القاء خطا در نظر گرفت.

یکی از معیارهای دسته‌بندی مدل‌های خطا، محدوده‌ی داده‌ای است که تحت تأثیر قرار می‌گیرد؛ در برخی موارد، خطا ممکن است تنها یک بیت را تغییر دهد، در حالی که در موارد دیگر می‌تواند یک واحد داده‌ای بزرگ‌تر مانند یک بایت یا یک کلمه‌ی ۳۲ بیتی را تحت تأثیر قرار دهد [۴]. علاوه بر این، خطا ممکن است تنها یک مقدار یا چندین مقدار در حال پردازش را به طور همزمان دچار اختلال کند [۵].

نحوه‌ی تغییر داده‌ها نیز از دیگر عوامل مهم در طبقه‌بندی مدل‌های خطا است. در برخی موارد، مقدار پردازش‌شده به یک مقدار مشخص و از پیش تعیین‌شده[1] تبدیل می‌شود، در حالی که در سایر موارد مقدار جدید می‌تواند کاملاً تصادفی باشد. این تصادفی بودن ممکن است دارای توزیع یکنواخت یا غیریکنواخت باشد. نوع دیگری از خطا نیز وجود دارد که مقدار جدید دقیقاً معکوس مقدار واقعی[2] است. دقت زمانی و مکانی القاء خطا نیز نقش مهمی در تعیین مدل خطا دارد. در برخی سناریوها، مهاجم باید خطا را دقیقاً در یک بخش مشخص از داده و در یک لحظه‌ی خاص از اجرای الگوریتم القا کند، مانند تغییر مقدار یک بایت خاص در یک دور مشخص از فرآیند رمزنگاری. در مقابل، در برخی مدل‌ها، القاء خطا می‌تواند در یک بازه‌ی زمانی وسیع‌تر رخ دهد و لزوماً محدود به یک مکان خاص نباشد [۴].

مدت زمان اثرگذاری خطا نیز یکی از معیارهای مهم در دسته‌بندی مدل‌های مختلف است. خطا ممکن است گذرا[3] باشد، به این معنا که تنها برای یک لحظه رخ داده و سپس از بین برود. در مقابل، ممکن است خطا به صورت دائمی[4] باشد و مقدار تغییریافته دیگر به حالت اصلی خود بازنگردد. حالت سومی نیز وجود دارد که در آن خطا ماندگار[5] اما غیردائمی است، یعنی برای مدتی باقی می‌ماند اما پس از چند مرحله پردازش، از بین می‌رود [۶، ۷].

[1]Stuck-at Fault [2]FlipFault [3]Transient [4]Permanent [5]Persistent

۳ منابع القاء خطا

در این بخش، به طور مختصر تجهیزات مورد استفاده برای القاء خطا را بررسی می‌کنیم. ابزارهای القاء خطا را می‌توان بر اساس ویژگی‌های مختلفی مقایسه کرد.

یکی از مهم‌ترین ویژگی‌ها، نوع اثر خطا است. خطا می‌تواند محلی[6] یا سراسری[7] باشد. در خطای محلی، مهاجم قادر است تأثیر خطا را به یک بخش یا نقطه خاص از دستگاه محدود کند، در حالی که در خطای سراسری، مهاجم کنترلی بر محل دقیق ایجاد خطا ندارد و خطا به صورت کلی در سیستم القا می‌شود [۴].

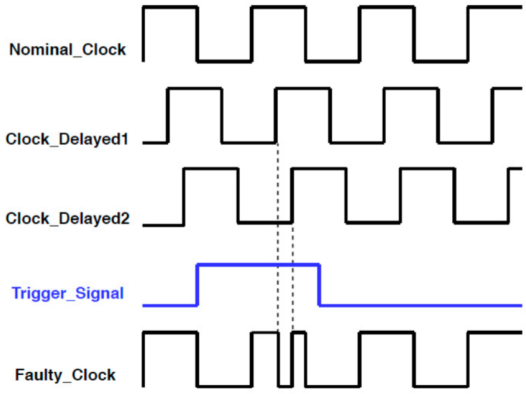
ویژگی دیگر، لزوم دسترسی فیزیکی به تراشه است. در برخی روش‌ها، برای اعمال خطا نیاز است که لایه‌برداری انجام شود، یعنی پوشش بیرونی تراشه با استفاده از روش‌های شیمیایی (مانند اسید) یا مکانیکی برداشته شود تا بتوان به اجزای داخلی دستگاه دسترسی پیدا کرد. در برخی موارد، این فرایند ممکن است باعث تخریب کامل دستگاه شود، بنابراین انتخاب روش القاء خطا باید بر اساس میزان دسترسی به دستگاه و اهمیت آن تنظیم شود. همچنین، میزان نیاز به دانش درباره طراحی داخلی دستگاه یکی دیگر از فاکتورهای مهم است. برخی ابزارهای القاء خطا برای عملکرد صحیح نیاز به آگاهی دقیق از ساختار و معماری تراشه دارند، در حالی که برخی دیگر بدون نیاز به اطلاعات فنی یا با دانش محدودی قابل استفاده هستند [۴].

در نهایت، دقت ابزارهای القاء خطا از دو جنبه زمانی و مکانی مورد بررسی قرار می‌گیرد. دقت زمانی به میزان هماهنگی ابزار با فرآیندهای داخلی دستگاه هنگام القاء خطا اشاره دارد، در حالی که دقت مکانی مربوط به میزان کنترل مهاجم بر محل دقیق اعمال خطا در مدار است. هر چه این دو پارامتر دقیق‌تر باشند، احتمال موفقیت حمله نیز افزایش می‌یابد. حال در ادامه منابع تزریق خطا به همراه چگونگی القاء و اثرگذاری آنها را بررسی می‌کنیم.

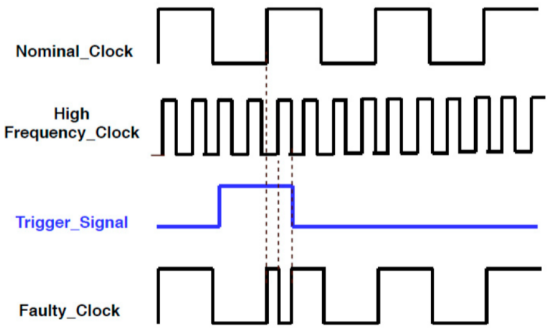
۱.۳ گلیچ فرکانس

در حالت کلی، گلیچ‌های ساعت می‌توانند موجب تغییر در مسیر بحرانی مدار، جریان برنامه و داده‌ها (جابه‌جایی دستورات، تغییر در مقدار خطاهای محاسباتی) شوند. در این روش مهاجم زمان‌بندی جریان اعمال خطا را کاملاً تحت کنترل داشته و می‌تواند زمان‌بندی حمله‌ی خود را متناسب با آن تنظیم کند [۹]. برای ایجاد گلیچ ساعت از دو روش ترکیب ساعت‌های شیفت‌یافته و یا ترکیب ساعت با فرکانس‌های متفاوت استفاده می‌شود [۸]. در شکل ۱ نحوه ایجاد گلیچ با دو روش ذکر شده نشان داده شده است.

[6]Local [7]Global



(آ) روش ترکیب ساعت‌های شیفت‌یافته

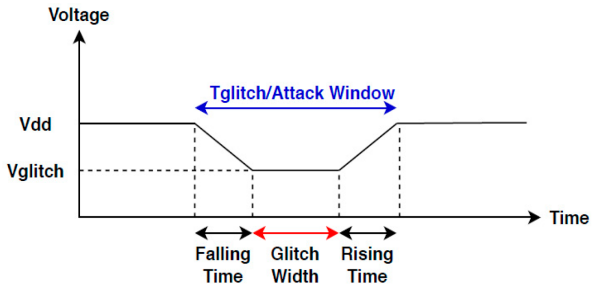


(ب) روش ترکیب ساعت با فرکانس‌های متفاوت

شکل ۱. نحوه ایجاد گلیچ با دو روش معرفی شده [۸]

۲.۳ گلیچ ولتاژ

تغییرات شدید و ناگهانی ولتاژ می‌توانند موجب تغییراتی همانند تغییرات ناشی از گلیچ‌های ساعت شوند، و یا حتی می‌توانند میزان تاخیر انتشار[1] را تغییر دهند. برای اعمال این روش، ولتاژ منبع تغذیه‌ی مدار را به صورت گذرا یا دائمی کاهش و یا افزایش می‌دهند [۱۰]. در شکل ۲ شماتیک خطا گلیچ ولتاژ نشان داده شده‌است.



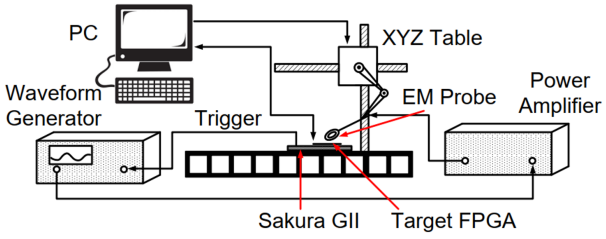
شکل ۲. شماتیک کلی خطای گلیچ ولتاژ [۸]

۳.۳ تغییرات دمایی

تغییرات دمایی نیز می‌توانند در شرایطی خاص و البته با کنترل‌پذیری کمتر در مدار خطا ایجاد کنند (تأثیر آن‌ها بر تاخیرها و زمان‌بندی‌های مدار است). این تغییرات می‌توانند شامل افزایش یا کاهش دما باشند. در مواردی، از این روش به منظور تسریع و یا تقویت اثر روش‌های دیگر استفاده می‌شود. افزایش دما معمولا با روش‌های غیرمستقیم، مانند قراردادن تراشه بر روی صفحات حرارتی انجام می‌شود [۱۱].

۴.۳ پالس‌های‌الکترومغناطیس

یکی دیگر از روش‌های ایجاد خطا استفاده از پروب‌های مغناطیسی برای ایجاد پالس‌های الکترومغناطیسی در مدار است. تعیین دقیق محدوده‌ی اثرگذاری این پالس‌ها بر روی مدار کار ساده‌ای نیست. این پالس‌ها موجب ایجاد جریان‌های گردابی[2]، سوییچینگ ترانزیستورها و اختلال در مسیر بحرانی[3] مدار می شوند [۱۱]. مواردی مانند نوع پروب‌های مورد استفاده، مدت زمان و دامنه‌ی پالس‌های مورد اعمال، در میزان اثرگذاری این روش نقش دارند [۱۲]. در شکل ۳ نحوه راه‌اندازی خطای الکترومغناطیس نشان داده شده است.



شکل ۳. شماتیک راه‌انداز خطای الکترومغناطیس [۹]

۵.۳ اشعه لیزر

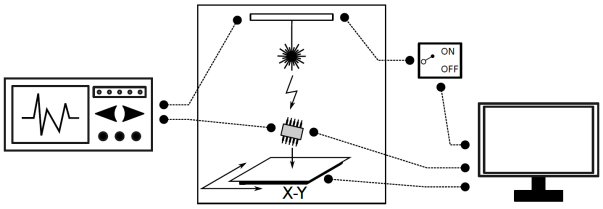
تزریق خطا با لیزر یکی از دقیق‌ترین و قدرتمندترین روش‌ها برای اعمال خطا در مدارهای الکترونیکی است، اما اجرای آن هزینه‌ی بالایی دارد و فرایند نمونه‌برداری از نتایج آن نیز پیچیده و گران‌قیمت است. در این روش، مهاجم برای اعمال خطا نیاز به دسترسی مستقیم به محل مورد نظر در تراشه دارد، بنابراین باید لایه‌برداری از دستگاه انجام شود تا قسمت‌های داخلی آن در معرض تابش لیزر قرار بگیرند. این روش دقت بسیار بالایی دارد و امکان اعمال خطا در موقعیتی مشخص و کنترل‌شده را فراهم می‌کند. علاوه بر این، توانایی ایجاد انواع مختلفی از خطاها را دارد و مقابله با آن دشوار است. اجرای این روش به تجهیزات پیشرفته و تخصصی نیاز دارد که هزینه‌ی آن بالاست. عملکرد این روش تحت تأثیر عوامل متعددی قرار دارد، از جمله نوع پالس لیزر، طول موج، میزان انرژی و محل اعمال خطا که هر یک نقش مهمی در میزان موفقیت حمله ایفا می‌کنند [۱۲]. در شکل ۴ شمای کلی از تزریق خطا توسط لیزر نشان داده شده است.

[1] Propagation Delay

[2] Eddy Currents [3] Critical Path

جدول ۱. مقایسه منابع القاء خطا [۴]

روش القا	تأثیر	دستگاه			دقت		هزینه تجهیزات	مهارت فنی
		لایه‌برداری	آسیب	جزئیات طراحی	زمان	موقعیت		
گلیچ کلاک ابزار	سراسری	خیر	خیر	ضروری	بالا	پایین	پایین	متوسط
اسپایک ولتاژ	سراسری	خیر	خیر	خیر	متوسط	پایین	پایین	متوسط
تغییرات دما	سراسری	خیر	خیر	ضروری	پایین	پایین	پایین	پایین
پالس الکترومغناطیسی	محلی	خیر	احتمالی	تا حدی ضروری	متوسط	پایین	متوسط	متوسط
پرتو لیزر	محلی	بله	احتمالی	ضروری	بالا	بالا	بالا	بالا



شکل ۴. شمای کلی تزریق خطا توسط لیزر [۱۳]

در جدول ۱ منابع القا خطا از نظر ویژگی‌های مختلف مورد مقایسه قرار گرفته است.

۴ روش‌های تحلیل خطا

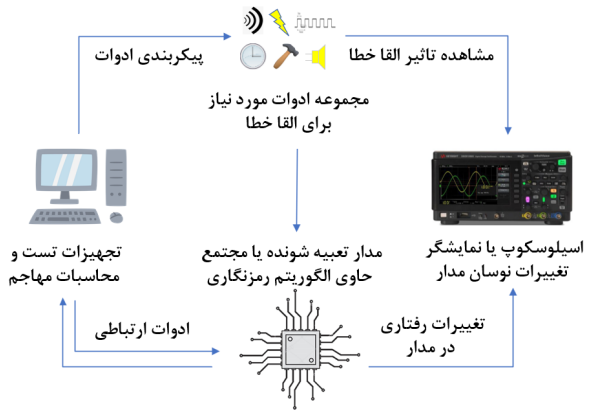
پس از القاء خطا با یکی از روش‌های موجود، مهاجم باید روشی مناسب برای تحلیل نتایج انتخاب کند. همان‌طور که در شکل ۵ نشان داده شده، مهاجم ابتدا الگوریتم مورد نظر را به عنوان هدف حمله انتخاب کرده و منابع مورد نیاز تزریق خطا را آماده می‌کند. این تجهیزات با دقت و پارامترهای از پیش تعیین‌شده پیکربندی می‌شوند. مدار مورد حمله ممکن است از یک میکروکنترلر ساده تا یک تراشه‌ی پیشرفته متغیر باشد. در طول حمله، مدار قربانی و تجهیزات مهاجم به ابزارهای اندازه‌گیری مانند نمایشگر یا اسیلوسکوپ متصل می‌شوند تا هرگونه نشتی اطلاعاتی به‌دقت ثبت و تحلیل شود.

مهاجم با بررسی این داده‌ها، با استفاده از روش‌های تحلیل، تأثیر خطا را ارزیابی کرده و تلاش می‌کند اطلاعات حساس سیستم را استخراج کند. به طور کلی، تکنیک‌های تجزیه و تحلیل خطا را می‌توان در سه دسته عمده طبقه‌بندی کرد: مبتنی بر تفاضل [۱]، مبتنی بر برخورد [۲] و مبتنی بر تحلیل آماری [۳].

۱.۴ تحلیل خطا مبتنی بر تفاضل

همه روش‌ها در دسته تحلیل مبتنی بر تفاضل تا حدودی مشابه هستند و می‌توان آن‌ها را به عنوان بسط تحلیل خطای تفاضلی در نظر گرفت. این روش‌ها از ویژگی‌های آشفتگی و انتشار یک رمز برای القاء یک

[1]Differential [2]Collision [3]Statistical



شکل ۵. مراحل تحلیل حملات القاخطا

خطای مناسب استفاده می‌کنند که به مهاجم این امکان را می‌دهد که توزیع تفاضل دلخواه خود را در خروجی الگوریتم به‌دست آورد. به طور کلی، تحلیل خطای تفاضلی و انواع آن نسبت به مدل‌های غیرمؤثر یا آماری کاربردی‌تر هستند زیرا تعداد خطاهای کمتری نیاز دارند.

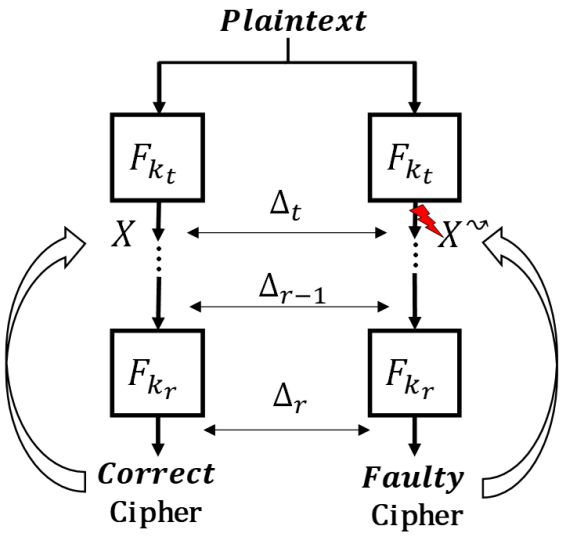
۱.۱.۴ حمله تحلیل خطای تفاضلی (DFA)

اجرای حمله‌ی القاءخطا بر روی رمزهای متقارن با ایده‌ی اولیه‌ی که توسط دن بونه [۴] ارائه شده بود [۱۴]، به دلیل ساده نبودن ساختار جبری رمزهای متقارن با چالش‌هایی همراه بود. اما بیهام [۵] و شمیر [۶] توانستند با الهام از روش بونه، اولین حمله‌ی القاءخطای تفاضلی (DFA) را طراحی و به رمزهای متقارن نیز اعمال کنند [۱۵].

ساختار کلی حمله‌ی تحلیل خطای تفاضلی در حمله‌ی تحلیل خطای تفاضلی، با القاء خطا در یکی از دورهای میانی، تفاوتی بین مقادیر میانی در حالت عادی و خطادار ایجاد می‌شود. اگر این تفاضل با احتمال بالایی به خروجی الگوریتم منتقل شود، امکان بازیابی کلید رمزنگاری فراهم می‌شود. همانطور که در شکل ۶ مشاهده می‌شود این روش مشابه تحلیل تفاضلی استاندارد عمل می‌کند، به این صورت که مهاجم زیرکلید دورهای آخر را حدس می‌زند و با رمزگشایی بررسی می‌کند که آیا تفاضل ایجادشده

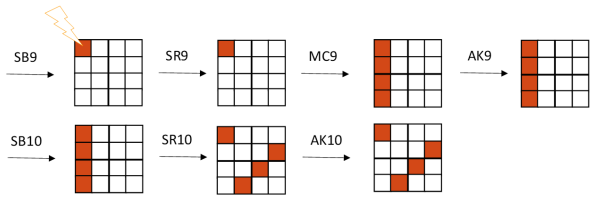
[4]Dan Boneh [5]Biham [6]Shamir

در دور میانی همچنان برقرار است یا نه. در صورتی که تفاضل مورد نظر برقرار باشد، احتمالا کلید حدس‌زده‌شده درست خواهد بود، اما اگر این شرط برقرار نباشد، آن کلید قطعاً نادرست است [۱۶–۱۸].



شکل ۶. ساختار کلی حمله تحلیل خطا تفاضلی

حمله تحلیل خطای تفاضلی به AES برای درک بهتر نحوه عملکرد حمله القاء خطای تفاضلی، اجرای این حمله را روی الگوریتم استاندارد AES با جزئیات مورد بررسی قرار می‌دهیم. اگر مهاجم یک خطا در یکی از بایت‌های ابتدای دور نهم تزریق کند، این خطا باعث می‌شود که در متن رمزشده‌ی خطادار، طبق شکل ۷ چهار بایت، تفاضل غیرصفر نسبت به متن رمزشده‌ی صحیح داشته باشند.



شکل ۷. نحوه انتشار خطا در بایت صفرم دور نهم الگوریتم AES

برای استخراج اطلاعاتی درباره‌ی زیرکلید دور آخر، ابتدا چهار بایت مرتبط به خطا از زیرکلید دور دهم را حدس می‌زنیم و یک دور رمزگشایی معکوس انجام می‌دهیم تا مقادیر ستون اول در انتهای دور نهم به صورت روابط زیر به دست آید:

$$X = SB^{-1} \circ SR^{-1}\left(C[0,7,10,13] \oplus sk_{10}[0,7,10,13]\right)$$
$$X^{\rightsquigarrow} = S_B^{-1} \circ S_R^{-1}\left(C^{\rightsquigarrow}[0,7,10,13] \oplus sk_{10}[0,7,10,13]\right)$$

که در آن sk_{10} زیرکلید دور دهم، C متن رمزشده، و SR^{-1} و SB^{-1} به ترتیب توابع معکوس شیفت سطری و جعبهٔ جانشانی AES می‌باشد. در ادامه، تفاضل این مقادیر را قبل از اعمال مرحله‌ی مخلوط‌سازی ستون‌ها محاسبه می‌کنیم و بررسی می‌کنیم که آیا رابطه‌ی زیر برقرار است یا خیر:

$$MC^{-1}(X \oplus X^{\rightsquigarrow}) =? [*,0,0,0]$$

اگر تنها یک بایت فعال داشت، کلید ممکن است صحیح باشد. در غیر این صورت کلید غلط است و حذف می‌شود. با استفاده از یک جفت متن رمزشده، فضای جستجوی کلید برای این چهار بایت از 2^{32} به $2^{8} = (2^{-8})^{3} \times 2^{32}$ کاهش می‌یابد. در نتیجه، پیچیدگی زمانی حمله برابر با $(2^{8})^{4} = 2^{32}$ خواهد بود، که این مقدار در مقایسه با جستجوی جامع، بسیار کارآمدتر است [۱۹].

۲.۱.۴ حمله تحلیل خطای تفاضلی غیرممکن (DFIA)

حمله‌ی خطای تفاضلی غیرممکن[1] از همان ایده‌ی حمله‌ی تفاضلی غیرممکن کلاسیک (IDA) پیروی می‌کند، با این تفاوت که مهاجم به جای بررسی تفاضل‌هایی با احتمال بالا، به دنبال تفاضل‌هایی با احتمال صفر است. مراحل اجرای این حمله مشابه حمله‌ی خطای تفاضلی (DFA) است، با این تفاوت که، مهاجم با شناسایی زیرکلیدهایی که منجر به تفاضل صفر می‌شوند و حذف آن‌ها، فضای جستجوی کلید را محدود می‌کند. زیرا آن‌ها نمی‌توانند گزینه‌ی درستی برای کلید صحیح باشند. در حالی که IDA از تفاوت در ورودی برای یافتن تفاضل استفاده می‌کند، IDFA همان هدف را از طریق القاء خطا دنبال می‌کند [۲۰، ۲۱].

۲.۴ تحلیل خطا مبتنی بر برخورد

برخلاف حملات تحلیل خطا تفاضلی که عمدتاً بر شناسایی تفاوت‌های خروجی‌های خطادار متمرکزند، برخی از روش‌های پیشرفته‌تر به دنبال شرایط خاصی نظیر تصادم در خروجی یا عدم تغییر خروجی نهایی هستند. ما در این راستا، سه نوع حمله‌ی مهم شامل حمله‌ی تحلیل خطا برخورد (CFA)، تحلیل خطای ایمن (SEA) و تحلیل خطای غیرمؤثر (IFA) را بررسی می‌کنیم.

۱.۲.۴ حمله تحلیل خطای برخورد (CFA)

حمله‌ی خطای برخورد[2] روشی برای تحلیل خطا است که در آن مهاجم به دنبال ورودی‌هایی می‌گردد که خروجی‌های یکسانی در حالت عادی و خطادار تولید کنند. اگر دو متن رمزشده برابر باشند، مهاجم می‌تواند اطلاعاتی درباره‌ی وضعیت‌های میانی و کلید مخفی استخراج کند. برخلاف حمله‌ی DFA که خطا در دورهای انتهایی تزریق می‌شود، در CFA خطا در مراحل اولیه اعمال می‌شود تا پیش از انتشار کامل، شرایط تصادم فراهم شود. این حمله ابتدا در [۲۲، ۲۳] مطرح شد و بر پایه‌ی وقوع تصادم در خروجی رمزنگاری است. مهاجم ابتدا نسخه‌ی خطادار یک متن دلخواه M_0 را رمزگذاری کرده و خروجی آن را دریافت می‌کند ($C^{\rightsquigarrow} = E^{\rightsquigarrow}(M_0)$). سپس متن M را می‌یابد که بدون خطا، همان خروجی C را تولید کند ($E(M) = C = C^{\rightsquigarrow}$).

[1] Impossible Differential Fault Attack [2] Collision Fault Analysis

۲.۲.۴ حمله تحلیل خطای غیر موثر (IFA)

حمله‌ی خطای غیرمؤثر[1] زمانی انجام می‌شود که مهاجم متنی را پیدا کند که پس از اعمال خطا، داده‌های میانی هدف‌گیری‌شده تغییری نکنند و در نتیجه، متن رمزشده‌ی خروجی بدون تغییر باقی بماند. این حمله از خطاهایی که تأثیری بر مقادیر داخلی ندارند، برای استخراج اطلاعات استفاده می‌کند و به همین دلیل به آن «خطای غیرمؤثر» گفته می‌شود. IFA و CFA از نظر اجرا تفاوت‌های چشمگیری دارند. در CFA تنها با یک خطا می‌توان بخشی از کلید را افشا کرد، اما IFA به خطاهای متعددی نیاز دارد تا یکی از آن‌ها غیرمؤثر باشد. با این حال، IFA نیازی به اعمال خطا در مراحل ابتدایی ندارد و مهاجم می‌تواند در هر مرحله از رمزنگاری، خطاهای غیرمؤثر را شناسایی کند. علاوه بر این، در IFA دانستن مقدار دقیق متن رمزشده‌ی خطادار ضروری نیست و تنها کافی است مشخص شود که خطا بر خروجی تأثیر گذاشته یا نه، که همین ویژگی آن را در برابر روش‌های تشخیص خطا مقاوم می‌سازد. در اجرای این حمله، مهاجم باید مطمئن شود که علت بی‌اثر بودن خطا، مقدار طبیعی داده‌ها بوده است و نه شکست در تزریق خطا. به همین دلیل، ابزار القاء خطا باید با دقت بالا عمل کند تا نتایج قابل اعتماد و دقیق به دست آید [۲۴].

۳.۲.۴ حمله تحلیل خطای ایمن (SEA)

سومین روش تحلیل خطا که از تطابق خروجی‌های الگوریتم‌های رمزنگاری بهره می‌برد، تحلیل خطای ایمن (SEA)[2] است. این روش ابتدا برای شکستن کلید خصوصی سیستم رمزنگاری RSA معرفی شد. در نگاه اول، تحلیل خطای ایمن و تحلیل خطای غیرمؤثر (IFA) مشابه به نظر می‌رسند، زیرا هر دو با بررسی تأثیر خطا بر خروجی، اطلاعاتی درباره‌ی کلید مخفی به دست می‌آورند. با این حال، تفاوت مفهومی مهمی بین آن‌ها وجود دارد. در SEA، خطا داده‌های هدف را تغییر می‌دهد، اما چون این داده‌های تغییریافته در ادامه‌ی محاسبات استفاده نمی‌شوند، خروجی بدون تغییر باقی می‌ماند. در مقابل، در IFA خطا مستقیماً داده‌هایی را که در محاسبه به کار می‌روند هدف قرار می‌دهد، اما به دلیل ماهیت خطا و مقدار داده، این داده‌ها تغییری نمی‌کنند. به بیان ساده، IFA بر بررسی مقدار داده‌ها تمرکز دارد، در حالی که SEA به نحوه‌ی استفاده از داده‌ها توجه می‌کند [۲۵].

۳.۴ تحلیل خطا مبتنی بر تحلیل آماری

در این دسته از تجزیه و تحلیل خطا، توزیع آماری یک متغیر در نظرگرفته می‌شود که به دلیل ماهیت القاء خطا، غیریکنواخت است. به طور کلی، این دسته در مقایسه با روش‌های مبتنی بر تفاضل، نیاز به کنترل ورودی ندارد و تنها به متن رمزشده نیاز دارد. همچنین در این سناریو دشمن قدرتمندتری فرض می‌شود. جزئیات پیاده‌سازی‌های عملی در چنین مدل‌های خطا را می‌توان در [۲۶] یافت.

[1] Ineffective Fault Analysis [2] Safe Error Analysis

۱.۳.۴ حمله آماری القاء خطا (SFA)

فرض کنید خطای القاءشده موجب شود که متغیر میانیِ m بیتیِ x، با یک توزیع احتمال غیریکنواخت طبق جدول ۲ به مقدار x' تبدیل شود.

جدول ۲. توزیع احتمال تزریق خطا [۲۷]

$x \to x'$	۰۰	۰۱	۱۰	۱۱
۰۰	۱	۰	۰	۰
۰۱	$\frac{1}{2}$	$\frac{1}{2}$	۰	۰
۱۰	$\frac{1}{2}$	۰	$\frac{1}{2}$	۰
۱۱	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$

در چنین شرایطی اگر مهاجم مجموعه‌ای از متون رمزشده‌ی خطادار را به دست آورد $(C_1^{\rightsquigarrow}, \ldots, C_N^{\rightsquigarrow})$، و سپس (بخشی از) زیرکلید دور (sk) را حدس زده و مقادیر میانی معادل آن متون رمزشده $(x_1^k, \ldots, x_N^k)$ را با رمزگشایی به‌دست آورد، مهاجم می‌تواند با استفاده از تست‌های آماری بررسی کند که آیا توزیع احتمال مقادیر میانی محاسبه‌شده با توزیع غیریکنواخت مورد انتظار مطابقت دارد یا نه. اگر تطابق وجود داشته باشد، احتمالاً زیرکلید به درستی حدس زده شده است. در غیر این صورت، زیرکلید حدس‌زده‌شده نادرست خواهد بود. بر اساس میزان توانایی مهاجم، می‌توان سه سناریو را در نظر گرفت:

(۱) **بدون کنترل:** تنها چیزی که مهاجم می‌داند این است که توزیع احتمال مقدار میانی پس از اعمال خطا، یکنواخت نیست و هیچ اطلاعات بیش‌تری درباره‌ی مدل تأثیر خطا ندارد. بر این اساس، فاصله‌ی توزیع احتمال مقدار میانی به ازای کلید حدس‌زده‌شده از توزیع احتمال یکنواخت را می‌توان با استفاده از پارامتر (SEI)[3] تخمین زد:

$$SEI(k) = \sum_{\delta=0}^{2^m-1} \left(\frac{\#\{\, i : x_i^k = \delta \,\}}{N} - \frac{1}{2^m} \right)^2$$

برای حالتی که کلید درست حدس‌زده‌شده باشد، انتظار می‌رود که مقادیر پارامتر به حالت یکنواخت (یعنی $\frac{1}{2^m}$) نزدیک باشند. در صورت در اختیار داشتن تعداد کافی متن رمزشده‌ی خطادار، مقدار SEI به ازای کلید صحیح نسبت به مقدار SEI به ازای کلید غلط بیشتر می‌شود.

(۲) **کنترل جزئی:** توزیع مقدار میانی پس از اعمال خطا برای مهاجم مشخص نیست، اما مدل تأثیر خطا مشخص است. در این سناریو می‌توانیم بر اساس مدل تأثیر خطا از یک تمایزگر مخصوص به همان مدل برای تشخیص کلید درست استفاده کنیم.

(۳) **کنترل کامل:** حالتی که توزیع مقدار میانی پس از اعمال خطا برای مهاجم مشخص است. در این سناریو، مهاجم با حدس تمام حالت‌های مختلف کلید، توزیع مقدار میانی به ازای کلید حدس‌زده‌شده را محاسبه می‌کند $(q_\delta^k = \frac{\#\{\, i : x_i^k = \delta \,\}}{N})$ و با توزیع مقدار میانی

[3] Squared Euclidean Imbalance

مشخص‌شده ($p_\delta = Pr(x' = \delta)$) با کمک تست آماری LLR مقایسه می‌کند:

$$LLR(\mathbf{q}^k, \mathbf{p}) = N \sum_{\delta=0}^{2^m-1} q_\delta^k \frac{p_\delta}{\theta_\delta}$$

اگر مقدار LLR به ازای توزیع q^k حداکثر شود، k کلید صحیح است.

۲.۳.۴ حمله حساسیت به خطا (FSA)

در حمله‌ی تحلیل حساسیت به خطا (FSA [1])، مهاجم از نحوه‌ی تغییر رفتار مدار در مواجهه با خطا برای استخراج اطلاعات حساس استفاده می‌کند، بدون اینکه نیاز به مقدار دقیق متن رمزشده‌ی خطادار داشته باشد. در این روش، مهاجم ابتدا چندین متن اصلی را رمزگذاری کرده و خروجی‌های بدون خطای آن‌ها را ثبت می‌کند. سپس، با اعمال خطا به صورت تدریجی، شدت آن را افزایش می‌دهد تا زمانی که خروجی دیگر با متن رمزشده‌ی بدون خطا مطابقت نداشته باشد. مقدار شدت خطایی که باعث این تغییر می‌شود، به عنوان آستانه‌ی حساسیت به خطا در نظر گرفته می‌شود. در مرحله‌ی بعد، مهاجم یک مقدار میانی خاص را انتخاب کرده و با حدس کلید، مقدار آن را برای همه‌ی متون محاسبه می‌کند. سپس همبستگی بین این مقادیر و میزان حساسیت به خطا بررسی می‌شود و کلیدی که بیشترین همبستگی را نشان دهد، به عنوان کلید صحیح در نظر گرفته می‌شود. این روش به یک فاز اولیه برای جمع‌آوری داده نیاز دارد اما نیازی به ابزارهای پیچیده و گران‌قیمت ندارد و تنها دانستن سطح تأثیر خطا کافی است [۲۸].

۳.۳.۴ حمله شدت خطای تفاضلی (DFIA)

حمله شدت خطای تفاضلی [2]، بر اساس این اصل که شدت خطا بر روی تعداد خطاهای ایجادشده تأثیر دارد، طراحی شده است. ابتدا به ازای یک متن ثابت M، عملیات رمزنگاری را N بار اجرا و خطا را اعمال می‌کنیم؛ به‌نحوی‌که شدت خطا در اجرای jام نسبت به شدت آن در اجرای $(j-1)$ام بیشتر باشد. انتظار داریم مقدار میانی در اجرای jام یا با مقدار میانی در اجرای $(j-1)$ام برابر باشد و یا این‌که فاصله‌ی همینگ آن یک واحد بیشتر باشد. سپس مطابق معمول، کلید را حدس زده و مقدار میانی را به ازای متون رمزشده محاسبه می‌کنیم. کلید صحیح، کلیدی است که به ازای آن مقدار فاصله‌ی همینگ مینیمم شود.

$$\sum_{i=1}^{N-1} HD(X^j, X^{j+1})$$

این حمله نسبت به حمله حساسیت خطا ، نیاز به فاز اولیه برای جمع‌آوری اطلاعات در خصوص حساسیت داده‌های مختلف ندارد اما در این حمله به متن رمزشده خطادار نیاز داریم [۲۶].

[1] Fault Sensitivity Analysis [2] Differential Fault Intensity Analysis

۴.۳.۴ حمله آماری القاءخطای غیرمؤثر (SIFA)

حمله‌ی SIFA [3] ترکیبی از حملات القاء خطای غیرمؤثر (IFA) و حملات آماری (SFA) است که با تحلیل توزیع خطاهای ناموفق، اطلاعات حساس سیستم رمزنگاری را استخراج می‌کند. در این روش، مهاجم از این واقعیت استفاده می‌کند که احتمال وقوع خطا در پردازش رمزنگاری همیشه به صورت یکنواخت توزیع نمی‌شود. اگر کلید حدس‌زده‌شده صحیح باشد، یک الگوی خاص در توزیع خطاهای ناموفق مشاهده می‌شود، اما اگر کلید اشتباه باشد، چنین الگویی وجود ندارد.

در این حمله، ابتدا تعدادی متن اصلی را رمزگذاری کرده و همزمان، خطایی را در مقدار میانی خاصی با توزیع غیریکنواخت اعمال می‌کنیم. سپس، فقط متون رمزشده‌ای را که خطا بر آن‌ها بی‌اثر بوده‌اند، انتخاب کرده و فرآیند رمزگشایی را روی آن‌ها انجام می‌دهیم. برای این کار، بخش‌هایی از کلید را حدس زده و مقادیر میانی مربوطه را محاسبه می‌کنیم. سپس، با استفاده از آزمون‌های آماری بررسی می‌شود که آیا توزیع احتمال این مقادیر یکنواخت است یا خیر. اگر یکنواخت باشد، کلید حدس‌زده‌شده نادرست است، اما اگر توزیع غیریکنواخت باشد، احتمالا کلید صحیح است [۲۷].

۵.۳.۴ حمله آماری القاء خطا موثر (SEFA)

حمله‌ی SEFA [4] یک روش تحلیل آماری مبتنی بر القاء خطا است که با بررسی تأثیر خطاهای مؤثر در فرآیند رمزنگاری، اطلاعات حساس مانند کلید مخفی را استخراج می‌کند. این حمله مشابه SIFA عمل می‌کند اما از اطلاعات حاصل از خطاهای مؤثر استفاده می‌کند.

در این روش، مهاجم ابتدا چندین متن اصلی را رمزگذاری کرده و سپس خطاهای مختلفی را به فرآیند رمزنگاری اعمال می‌کند. اگر مدار از روش‌های مقابله‌ای استفاده کند، در بسیاری از موارد خطای اعمال‌شده شناسایی شده و خروجی رمزنگاری مسدود می‌شود. در این شرایط، مهاجم همان عملیات رمزگذاری را بدون اعمال خطا انجام می‌دهد و متن رمزشده‌ی صحیح را دریافت می‌کند. در مرحله‌ی بعد، با حدس بخش‌هایی از کلید، فرآیند رمزگشایی روی این متون رمزشده‌ی صحیح انجام شده و مقادیر میانی مربوط به آن‌ها محاسبه می‌شود. سپس، از روش‌های آماری برای بررسی توزیع احتمالات این مقادیر استفاده می‌شود. اگر توزیع به‌دست‌آمده یکنواخت باشد، کلید حدس‌زده‌شده نادرست است، اما اگر توزیع غیریکنواخت باشد، احتمالاً کلید صحیح استخراج شده است [۲۹].

۴.۴ دیگر حملات معرفی‌شده

۱.۴.۴ حمله خطا مرتبط‌شده (LFA)

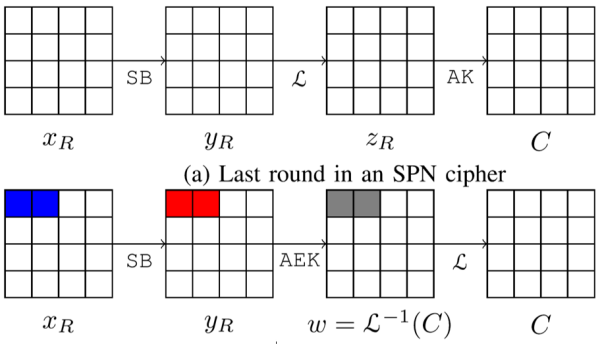
حمله خطای مرتبط شده [5] که با آگاهی مهاجم از زمان‌بندی دقیق القاء خطا (محل‌خطا) انجام می‌شود، رمزهای قالبی کلمه‌گرا را مورد هدف

[3] Statistical ineffective Fault Attack [4] Statistical Effective Fault Attack

[5] Linked Fault Attack

قرار می‌دهد که در آن‌ها لایه غیرخطی با استفاده از یک یا چند جدول جستجوی از پیش محاسبه‌شده پیاده‌سازی شده است [۳۰].

مکانیزم حمله از طریق دستکاری دستورالعمل‌های پردازشی عمل می‌کند، به طوری که با ایجاد پرش هدفمند در جریان اجرای دستورات، هنگامی که پردازش متوالی مقادیر میانی v و u انجام می‌شود، مقدار خطادار u به مقدار میانی v مرتبط می‌شود. به طور مثال اگر دستگاه در حال محاسبه مقادیر خروجی بایت‌های صفر و یک، جعبه جانشانی باشد و مهاجم به هر نحوی، بتواند در زمان فراخوانی مقدار بایت یکم، خطایی تزریق کند به نحوی که این فراخوانی به درستی انجام نشود، مقدار جدید بایت یکم، بدون تغییر و برابر با بایت صفرم می‌باشد. در نهایت مهاجم به کمک این ارتباط و تحلیل آماری آن‌ها قادر به استخراج بخش‌هایی از کلید مخفی خواهد بود.



شکل ۸. نحوه اجرای حمله خطای مرتبط‌شده تفاضلی [۳۰]

در ابتدا، با توجه به اینکه هم لایه‌ی اضافه‌شدن کلید و هم لایه‌ی جایگشت L خطی هستند، می‌توان با استفاده از یک تکنیک شناخته‌شده برای ساده‌سازی تحلیل در دور آخر، زیرکلید sk را با $\mathrm{esk}_R = L^{-1}(\mathrm{sk}_R)$ که به آن زیرکلید معادل گفته می‌شود، جایگزین کرد. بر اساس معادله‌ی زیر، یک نمایش معادل از دور آخر شبکه‌ی SPN در شکل ۸ نشان داده شده است.

$$\begin{aligned}\mathcal{L}(y_R) \oplus sk_R &= \mathcal{L}(y_R) \oplus \mathcal{L}(\mathcal{L}^{-1}(sk_R)) \\ &= \mathcal{L}(y_R \oplus \mathcal{L}^{-1}(sk_R)) = \mathcal{L}(y_R \oplus esk_R)\end{aligned}$$

کدر آن $x_R[i]$ و $y_R[i]$ به ترتیب i–امین کلمه‌ی ورودی و i–امین کلمه‌ی خروجیِ دور r–ام هستند، و حالتِ b–بیتی از L کلمه با طولِ یکسانِ m بیت $(m = \frac{b}{L})$ تشکیل شده است.

هم‌چنین می‌توان جایگشت دور آخر را با به‌دست‌آوردن مقدار $\omega_r = L^{-1}(C)$ حذف‌شده در نظر گرفت. از آن‌جا که رابطه‌ی بین زیرکلید sk_r و زیرکلید معادل esk_r یک رابطه‌ی یک‌به‌یک است، اگر مهاجم m بیت اطلاعات در مورد esk_r به‌دست آورد، درواقع m بیت اطلاعات در مورد sk_r نیز به‌دست آورده است. حال اگر با پرش یک دستورالعمل، کلمه‌ای مانند $x_R[j]$ برابر با $x_R[i]$ یا $y_R[j]$ برابر با خروجی S-Box یعنی $S(y_R[i])$ شود، مهاجم می‌تواند با استفاده از رابطه‌ی $\omega_r = L^{-1}(C)$، کلمه‌های $\omega'_r[i]$ و $\omega'_r[j]$ را با کمک متن رمزشده‌ی خطادار C' به‌دست آورد. سپس رابطه‌ی زیر بین کلمات مرتبط برقرار است:

$$\begin{aligned}&\omega'_r[i] \oplus \omega'_r[i+1] \\ &= (y'_r[i] \oplus esk_r[i]) \oplus (y'_r[i+1] \oplus esk_r[i+1]) \\ &= \Delta esk_r[i, i+1] \qquad (1)\end{aligned}$$

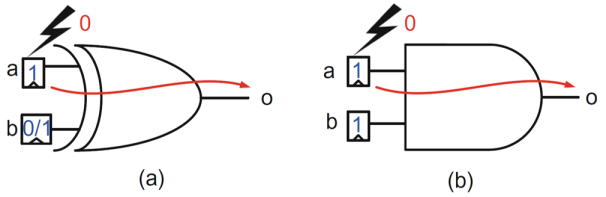
این حمله را می‌توان برای کلمات مختلف تکرار کرد. اگر S-Boxهای دور آخر به ترتیب اجرا شوند، $L-1$ خطای پیوندی بین $x[i]$ و $x[i+1]$ برای $0 \le i \le L-2$ قابل تزریق است. هر خطا m بیت اطلاعات دربارهٔ زیرکلید esk_r فاش کرده و مجموعاً $m \times (L-1)$ بیت اطلاعات قابل استخراج است. با افشای مقدار $\Delta esk_r[i, i+1]$ طبق رابطهٔ (۱)، مهاجم در صورت حدس مقدار $esk_r[0]$ می‌تواند $esk_r[j]$ را برای $1 \le j \le L$ طبق معادلهٔ زیر محاسبه کند:

$$esk_r[j] = esk_r[0] \oplus \bigoplus_{l=0}^{j-1} \Delta esk_r[l, l+1]$$

۲.۴.۴ حمله الگو خطا (FTA)

حملات الگوی خطا (FTA [1]) با تحلیل نحوه‌ی انتشار خطا در مدارهای دیجیتال، اطلاعات حساس مانند کلید رمزنگاری را استخراج می‌کنند. این حمله بر این واقعیت متکی است که تأثیر خطا بر خروجی مدار وابسته به داده‌های پردازش‌شده است و مهاجم می‌تواند از این وابستگی برای نشت اطلاعات استفاده کند. در این روش، مهاجم خطا را در بخش‌های مشخصی از مدار تزریق کرده و تغییرات خروجی را بررسی می‌کند تا مقادیر داخلی یا کلید را بازیابی کند [۳۱].

برخلاف روش‌هایی مانند SIFA این حمله به متن رمزگذاری‌شده نیازی ندارد و می‌تواند در مراحل میانی رمزنگاری انجام شود. موفقیت این حمله به دو عامل بستگی دارد: ابتدا خطا باید در شرایطی تزریق شود که مقدار اصلی و مقدار خطادار متفاوت باشند، سپس این خطا باید از دروازه‌های منطقی عبور کرده و به خروجی برسد تا قابل تحلیل باشد.



شکل ۹. انتشار خطا در گیت‌های XOR و AND [۳۱]

به طور مثال همان‌طور که در شکل ۹ دیده می‌شود، در دروازهٔ XOR اگر ورودی a دارای یک خطای گیرکردهٔ «صفر» باشد، این خطا تنها زمانی فعال می‌شود که مقدار واقعی $a = 1$ باشد. از آنجا که در XOR تغییر هر ورودی مستقیماً خروجی را تغییر می‌دهد، این خطا مستقل از مقدار b به خروجی منتقل می‌شود. در مقابل، در دروازهٔ AND برای آنکه

[1] Fault Template Attack

خطا به خروجی برسد، علاوه بر $a = ۱$، مقدار b نیز باید برابر ۱ باشد؛ در غیر این صورت خطا در همان مرحله حذف می‌گردد.

با بررسی نحوهٔ انتشار خطا در این دروازه‌ها می‌توان حملهٔ الگوی خطا (FTA) را اجرا نمود. این حمله شامل دو فاز است: فاز آفلاین که در آن الگوهای انتشار خطا شناسایی می‌شوند، و فاز آنلاین که در آن مهاجم با مقایسهٔ خروجی‌ها، کلید یا اطلاعات داخلی مدار را استخراج می‌کند.

در فاز آفلاین، مهاجم پیش از اجرای حملهٔ واقعی، یک مدل دقیق از نحوهٔ انتشار خطا در مدار ایجاد می‌کند. در این مرحله مهاجم به یک نسخهٔ مشابه از سیستم هدف با کلید شناخته‌شده دسترسی دارد و با اجرای آزمایش‌های متعدد، نحوهٔ انتشار خطا در ساختار مدار را بررسی می‌کند. مهاجم در مکان‌های مختلف مدار مانند ورودی‌های S-Box یا دروازه‌های منطقی، خطاهایی القا کرده و مشاهده می‌کند که آیا این خطاها به خروجی انتشار پیدا می‌کنند یا نه. نتایج این آزمایش‌ها به صورت مجموعه‌ای از الگوهای خطا در یک جدول ثبت می‌گردد؛ به طوری‌که برای هر ورودی و یک کلید معلوم، مشخص می‌شود خروجی چگونه تحت تأثیر قرار می‌گیرد. این اطلاعات به مهاجم امکان می‌دهد رابطهٔ بین مقادیر داخلی مدار و وضعیت انتشار خطا را مدل‌سازی کند و در نهایت مجموعه‌ای از الگوهای خطا ساخته شود که در فاز آنلاین برای استخراج کلید مورد استفاده قرار می‌گیرند.

در فاز آنلاین، مهاجم خطا را در همان بخش مدار تزریق کرده و بدون نیاز به مشاهدهٔ مقدار دقیق خروجی، بررسی می‌کند که آیا تغییری رخ داده است یا خیر. نتایج به‌دست‌آمده سپس با الگوهای ذخیره‌شده مقایسه می‌شوند و در صورت تطابق، کلید رمزنگاری حدس زده می‌شود. در مواردی که چندین نامزد کلید وجود دارد، مهاجم با انجام حملات تکمیلی و ترکیب نتایج می‌تواند کلید صحیح را تشخیص دهد.

۵.۴ مقایسه روش‌های تحلیل خطا

حملات القاء خطا روش‌های مختلفی برای استخراج اطلاعات حساس از پیاده‌سازی‌های رمزنگاری ارائه می‌دهند که هرکدام ویژگی‌های خاص خود را دارند. حمله‌ی القاء خطای تفاضلی (DFA) به تعداد کمی متن منتخب نیاز دارد و از مقایسه بین متن رمزشده‌ی صحیح و خطادار برای استخراج کلید استفاده می‌کند، اما نیاز به دسترسی به هر دو نسخه‌ی صحیح و خطادار متن رمزشده دارد که جز محدودیت های آن به شمار می‌رود.

حمله‌ی القاء خطای آماری (SFA) تنها از متن رمزشده‌ی خطادار استفاده می‌کند، اما اجرای موفق آن به دانستن توزیع دقیق مقدار میانی وابسته است که پیچیدگی حمله را افزایش می‌دهد.

حمله‌ی حساسیت به خطا (FSA) بدون نیاز به مقدار متن رمزشده‌ی خطادار، تنها شدت تأثیر خطا را بررسی می‌کند. این روش به تجهیزات گران‌قیمت نیاز ندارد، اما به یک فاز اولیه برای جمع‌آوری داده‌ها وابسته است و برخی روش‌های نقاب‌گذاری می‌توانند در برابر آن مؤثر باشند. مقایسه این روش با حمله‌ی تفاضلی شدت خطا (DFIA) نشان می‌دهد که DFIA نیازی به فاز اولیه جمع‌آوری اطلاعات ندارد اما همچنان به متن رمزشده‌ی خطادار وابسته است، در حالی که FSA بدون نیاز به این اطلاعات، صرفاً از میزان حساسیت خطا برای انجام حمله استفاده می‌کند. با این‌حال، هر دو روش در مقایسه با DFA و SFA از مزیت عدم نیاز به تجهیزات گران‌قیمت و سناریوهای دشوار برخوردارند.

تمام حملات DFA، SFA، FSA، DFIA مبتنی بر استفاده از متن‌های رمزشده‌ی خطادار هستند اما برخی روش‌های دفاعی می‌توانند وجود خطا در دستگاه را شناسایی کرده و با مسدود کردن خروجی یا ارسال مقدار تصادفی، امنیت سیستم را در برابر این حملات افزایش دهند. در چنین شرایطی، حمله‌ی SIFA توانایی غلبه بر این مکانیزم‌های تشخیص خطا را دارد. در ادامه، حمله‌ی SEFA به عنوان نسخه‌ی پیشرفته‌تر و مقاوم‌تر از SIFA در برابر نویز معرفی شده است که نسبت به SIFA در محیط‌های نویزی، به داده‌ی کمتری برای موفقیت در حمله نیاز دارد، که باعث می‌شود در شرایط واقعی، عملی‌تر و کارآمدتر باشد. علاوه بر این، برخلاف بسیاری از روش‌های تحلیل خطا که در برابر روش‌های مقابله مبتنی بر افزونگی ناکارآمد هستند، SEFA همچنان قابلیت اجرا در چنین محیط‌هایی را دارد. همچنین، این حمله در برخی موارد می‌تواند حتی در حضور روش‌های نقاب‌گذاری که برای محافظت در برابر حملات القاء خطا طراحی شده‌اند، اعمال شود. با این‌حال، یکی از چالش‌های اصلی این روش نیاز به کنترل ورودی است، یعنی مهاجم باید بتواند ورودی‌های مشخصی را در فرآیند رمزنگاری اعمال کند که در برخی شرایط عملی ممکن است محدودیت‌هایی ایجاد کند. و در انتها حمله الگوی خطا (FTA) برای استخراج اطلاعات حساس نیازی به مقدار دقیق خروجی ندارد و تنها با بررسی تغییر یا عدم تغییر خروجی مدار، کلید را بازیابی می‌کند. برخلاف بسیاری از حملات دیگر، این روش می‌تواند در دورهای میانی رمزنگاری اجرا شود. علاوه بر این، FTA قادر است برخی از مکانیزم‌های محافظتی را دور بزند و حتی در سخت‌افزارهای دارای تدابیر امنیتی نیز موفق عمل کند.

۵ روش‌های مقاوم‌سازی

معمولاً در مرحله‌ی طراحی الگوریتم‌های رمزنگاری، تدابیری برای مقابله با حملات القاء خطا در نظر گرفته نمی‌شود، بلکه وظیفه‌ی مقابله با آن‌ها بر عهده‌ی پیاده‌ساز سخت‌افزار یا توسعه‌دهنده‌ی نرم‌افزار امن است. به همین دلیل، مکانیزم‌های مقابله به طور جداگانه از الگوریتم‌های رمزنگاری طراحی و تحلیل می‌شوند. با کمک این روش‌ها می‌توان اعمال حملات القاء خطا را سخت‌تر کرد. در این بخش مهم‌ترین روش‌های مقاوم‌سازی [1] را در سه سطح بررسی می‌کنیم.

- سطح فیزیکی
- سطح پروتکل
- سطح پیاده‌سازی الگوریتم

[1] Countermeasures

جدول ۳. طبقه‌بندی حملات خطای گذرا به رمزهای قالبی [۳۰]

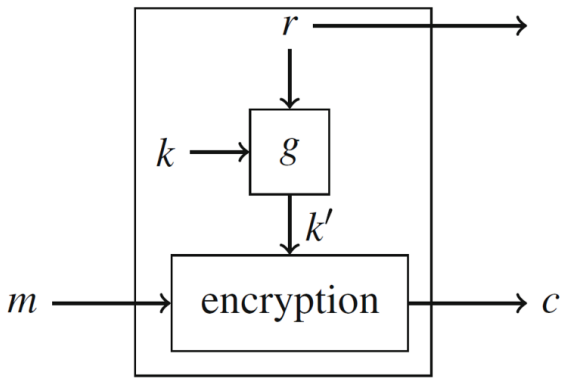
نام حمله	کنترل ورودی	دور زدن اقدامات متقابل افزونگی	اعمال بر پیاده‌سازی نقاب‌گذاری	تجهیزات دقیق
DFA	✓			
IDFA	✓			
DFIA	✓			
CFA	✓			
FTA	✓	✓	✓	✓
FSA	✓	✓		
SEFA	✓	✓	✓	
SFA				
LDFA				
SEA		✓		
IFA		✓		✓
SIFA		✓	✓	
LIFA		✓	✓	

۱.۵ سطح فیزیکی

در این سطح، مکانیزم‌هایی وجود دارند که بدون ارتباط با الگوریتم رمزنگاری، می‌توانند به عنوان یک محافظ از تداخل‌های خارجی، در دو نوع فعال و غیرفعال جلوگیری کنند [۴]. در دسته‌ی غیرفعال، مدارها به گونه‌ای طراحی می‌شوند که دسترسی فیزیکی به آن‌ها دشوار باشد. این روش‌ها شامل استفاده از سپرهای محافظ [1] یا لایه‌های پوششی هستند که برای جلوگیری از نفوذ مهاجم و ایجاد تداخل در عملکرد مدار استفاده می‌شوند. با این حال، این محافظت‌ها ممکن است با استفاده از تکنیک‌هایی مانند لایه‌برداری شیمیایی یا ابزارهای دقیق تراشکاری [2]، دور زده شوند. این ابزارها می‌توانند به صورت بسیار دقیق لایه‌های محافظ را برداشته و دسترسی مستقیم به مدار فراهم کنند [۳۲]. در مقابل، دسته‌ی فعال شامل استفاده از حسگرهایی است که به طور مداوم شرایط فیزیکی مدار را پایش می‌کنند. این حسگرها می‌توانند هرگونه تلاش برای دستکاری یا اعمال فشار فیزیکی را شناسایی کنند. به عنوان مثال، در صورت تشخیص هرگونه تنش، تغییر دما، یا تغییرات مکانیکی، این حسگرها هشدار داده یا حتی عملکرد مدار را متوقف می‌کنند تا از افشای اطلاعات جلوگیری شود [۱۳، ۳۳].

۲.۵ سطح پروتکل

در این سطح، کلید پس از انجام تعداد مشخصی از عملیات، به‌روزرسانی می‌شود. این روش یک رویکرد در سطح پروتکل است که فرض می‌کند بخش کوچکی از دستگاه قابل محافظت است، در حالی که سایر بخش‌ها همچنان در معرض حملات خطا قرار دارند [۳۴، ۳۵]. همان‌طور که در شکل ۱۰ نشان داده شده است، برای ایجاد یک کلید جلسه‌ی جدید (k') از یک تابع مشتق‌گیری (g) استفاده می‌شود که کلید اصلی (k) و یک مقدار تصادفی (r) را به عنوان ورودی می‌گیرد و خروجی آن ($k' = g(k, r)$) به صورت شکل ۱۰ است.



شکل ۱۰. به‌روز رسانی کلید در سطح پروتکل [۳۶]

۳.۵ سطح پیاده‌سازی الگوریتم

روش‌هایی که در سطح پیاده‌سازی الگوریتم می‌باشند، ابتدا خطا را با کمک تکنیک‌هایی تشخیص می‌دهند سپس خطای اعمال شده را یا تصحیح می‌کنند یا خروجی را بلوکه می‌کنند و یا خروجی را به صورت تصادفی ایجاد می‌کنند.

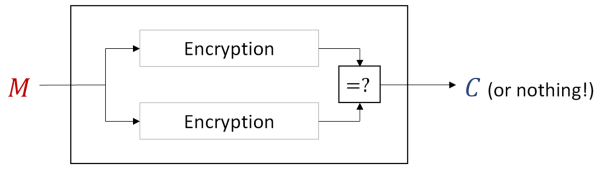
[1] Shield [2] Milling

۱.۳.۵ تشخیص خطا

روش‌های تشخیص خطا مبتنی بر افزونگی [1] می‌باشند. در روش افزونگی، از تکنیک تکرار عملیات استفاده می‌شود. دو روش مهم مورد استفاده عبارتند از:

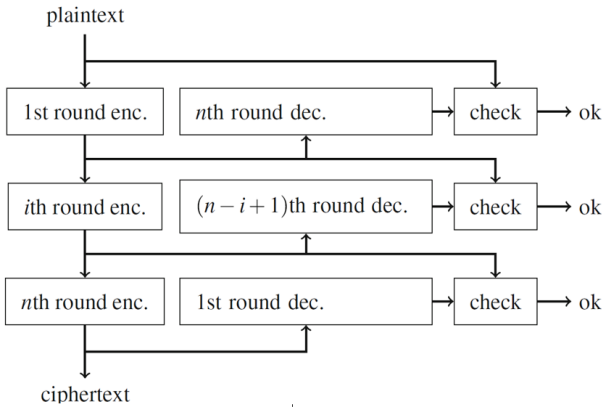
(۱) **اجرای موازی یا متوالی رمزگذاری:** در این روش، عملیات رمزگذاری می‌تواند به صورت موازی یا متوالی انجام شود. سپس خروجی‌های به‌دست‌آمده از هر دو حالت با یکدیگر مقایسه می‌شوند. در صورت تطابق خروجی‌ها، دستگاه اجازه‌ی ارائه‌ی متن رمزشده را صادر می‌کند، اما در صورت عدم تطابق، دستگاه خطا را شناسایی کرده و خروجی را مسدود می‌کند.

اجرای این روش در حالت موازی، زمان پردازش را تقریباً دو برابر می‌کند، در حالی که در حالت متوالی، مساحت مورد نیاز برای پیاده‌سازی تقریباً دو برابر می‌شود. با این حال، مهاجم می‌تواند با القاء یک خطای یکسان در هر دو اجرا، این مکانیزم را دور بزند [۳۷].



شکل ۱۱. مقاوم‌سازی تشخیص خطا مبتنی بر اجرای موازی عملیات رمزگذاری [۳۷]

(۲) **اجرای توأمان عملیات رمزگذاری و رمزگشایی:** در این روش، ابتدا عملیات رمزگذاری انجام می‌شود و سپس متن رمزشده حاصل، مجدداً رمزگشایی می‌شود. اگر متن اصلی بازیابی‌شده با متن اولیه مطابقت داشته باشد، دستگاه اجازه‌ی خروج متن رمزشده را صادر می‌کند [۳۸].



شکل ۱۲. مقاوم‌سازی تشخیص خطا مبتنی بر بررسی توامان عملیات رمزگذاری و رمزگشایی [۳۶]

در این روش اعمال خطای مشابه برای مهاجم کار سختی است و برای دور زدن ساز و کار امنیتی، ناگزیر به القاء خطا متفاوت است. زمان اجرا برابر با مجموع زمان عملیات‌های رمزنگاری و رمزگشایی است، که تقریبا معادل با زمان دو عملیات رمزنگاری خواهد بود.

ضعف روش‌های تشخیص خطا: اگر دستگاه با شناسایی خطا، خروجی را مسدود کند، مهاجم می‌تواند با بررسی متن رمزشده تشخیص دهد که آیا خطا بر مقدار میانی مرتبط با آن متن تأثیر گذاشته است یا خیر. این وضعیت منجر به نشت اطلاعات می‌شود. در نتیجه، حملاتی مانند SIFA که بر اساس متون رمزشده‌ی غیرمؤثر عمل می‌کنند، می‌توانند علیه پیاده‌سازی‌هایی که از این روش‌های مقابله استفاده می‌کنند، به کار گرفته شوند.

۲.۳.۵ تصادفی سازی خروجی

در این روش، پس از شناسایی خطا، دستگاه، یک مقدار تصادفی تولید کرده و ارائه می‌دهد. در سناریوی متن رمزشده‌ی معلوم، مهاجم قادر نخواهد بود تشخیص دهد که آیا خطا تأثیرگذار بوده است یا خیر. اما در سناریوی متن اصلی منتخب، مهاجم می‌تواند با مقایسه‌ی خروجی‌های عادی و خطادار، به این موضوع پی ببرد. بنابراین، این راهکار به طور کلی اثربخشی لازم را ندارد [۳۹].

۳.۳.۵ تصحیح خطا

این روش‌ها بر پایه‌ی کدینگ طراحی شده‌اند و می‌توانند پیچیدگی اجرای حملات تزریق خطا را به میزان قابل توجهی افزایش دهند. همچنین در مقابله با حملات غیرمؤثر نیز کارآمد هستند. ایده‌ی اصلی که در [۴۰] مطرح شده است، استفاده از مدارهای تصحیح خطا است. این مدارها توانایی شناسایی و اصلاح خطاها را در نقاط مختلف الگوریتم دارند. با استفاده از این روش، مقادیر میانی که دچار خطا شده‌اند، پیش از آن‌که در مدار گسترش یابند و بر خروجی تأثیر بگذارند، به مقدار صحیح خود بازگردانده می‌شوند. طراحی این مدارها معمولاً بر اساس کدهای خطی سیستماتیک انجام می‌شود. این کدها با توجه به پارامتر فاصله‌ی حداقل [2] یا d، قادر به شناسایی حداکثر d خطا و تصحیح تا $d/2$ خطا هستند. با این حال، پیاده‌سازی این مدارها باعث افزایش مساحت مورد نیاز خواهد شد.

۴.۵ مقایسه روش‌های مقاوم‌سازی

برای مقابله با حملات القاء خطا، روش‌های متعددی پیشنهاد شده است که هرکدام دارای نقاط ضعف و قوت خاص خود هستند. روش‌های مقاوم‌سازی در سطح فیزیکی و پروتکل به طور مستقیم از اعمال خطا جلوگیری نمی‌کنند، بلکه تنها فرایند حمله را برای مهاجم دشوارتر می‌کنند. در مقابل، روش‌های مبتنی بر تشخیص خطا که معمولاً از افزونگی استفاده می‌کنند، اجرای متوالی عملیات رمزنگاری را به همراه دارند که این امر در حالت اجرای متوالی، زمان پردازش را تقریباً دو برابر کرده و در حالت

[1] Redundancy

[2] Minimum Distance

اجرای موازی، مساحت سخت‌افزاری را افزایش می‌دهد.

مزیت این روش‌ها در این است که اجرای حملات القاء خطا را برای مهاجم پیچیده‌تر می‌کند، اما همچنان احتمال دور زدن مکانیزم‌های امنیتی توسط مهاجم وجود دارد. اگر سیستم در هنگام تشخیص خطا، خروجی را مسدود کند، مهاجم می‌تواند از همین رفتار برای تحلیل سیستم استفاده کند. در این شرایط، حملاتی مانند SIFA که از متون رمزشده‌ی غیرمؤثر استفاده می‌کنند، می‌توانند حتی در پیاده‌سازی‌هایی که از این نوع روش‌های مقابله بهره می‌برند، اعمال شوند.

در روش‌های تصادفی‌سازی، اگر مهاجم تنها به متن رمزشده‌ی نهایی دسترسی داشته باشد (سناریوی متن رمزشده‌ی معلوم)، قادر نخواهد بود تشخیص دهد که آیا خطا تأثیرگذار بوده است یا خیر. اما اگر مهاجم بتواند متن اصلی را انتخاب کند (سناریوی متن منتخب)، همچنان می‌تواند اثربخشی خطا را تحلیل کند. در نتیجه، این روش در تمام سناریوها مؤثر نیست. و در انتها روش‌های تصحیح خطا راهکاری کارآمد برای مقابله با حملات مبتنی بر متون غیرمؤثر محسوب می‌شوند. اما این روش نیز باعث افزایش چشمگیر مساحت سخت‌افزاری پیاده‌سازی می‌شود که ممکن است برای برخی از سیستم‌ها چالش‌برانگیز باشد.

۶ جمع‌بندی

بسیاری از حملات امنیتی که منجر به شکستن الگوریتم‌های رمزنگاری می‌شوند، به دلیل ضعف‌های پیاده‌سازی و آسیب‌پذیری آن‌ها در مدل جعبه خاکستری رخ می‌دهند. در این مقاله، گروهی از حملات فعال تحت عنوان حملات القاء خطا مورد بررسی قرار گرفته‌اند. ضمن تحلیل مدل‌ها و روش‌های مختلف القاء خطا، حملات رایج بر اساس مفروضات مهاجم، میزان کارایی و ایده‌ی اصلی آن‌ها به سه دسته‌ی اصلی تقسیم شده و به صورت اجمالی بررسی شده‌اند.

حملات القاء خطا در عمل از قدرت و کارایی بالایی برخوردارند و مقابله با آن‌ها اغلب مستلزم صرف هزینه‌های بیشتری است. بنابراین، به‌کارگیری راهکارهای محافظتی مؤثر و طراحی مقاوم در برابر خطا، از جمله اقدامات ضروری برای کاهش ریسک این نوع حملات محسوب می‌شود. در حال حاضر، سناریوهای حملات ترکیبی القاء خطا و کانال جانبی، روش‌های مقاوم‌سازی در برابر این حملات، به‌کارگیری هوش مصنوعی و یادگیری ماشین برای تحلیل حملات و همچنین بررسی تأثیر القاء خطا بر الگوریتم‌های پساکوانتومی از موضوعات مهم و جذاب در حوزه پیاده‌سازی الگوریتم‌های رمزنگاری هستند. این مباحث به عنوان مسیرهای پژوهشی آینده، توجه بسیاری از محققان را به خود جلب کرده‌اند.

مراجع

[1] Debdeep Mukhopadhyay and Rajat Subhra Chakraborty. *Hardware security: design, threats, and safeguards*. CRC Press, 2014.

[2] Alessandro Barenghi, Luca Breveglieri, Israel Koren, and David Naccache. Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures. *Proceedings of the IEEE*, 100(11):3056–3076, 2012.

[3] Ingrid Verbauwhede, Dusko Karaklajic, and Jorn-Marc Schmidt. The fault attack jungle-a classification model to guide you. In *2011 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 3–8. IEEE, 2011.

[4] Anubhab Baksi, Shivam Bhasin, Jakub Breier, Dirmanto Jap, and Dhiman Saha. A survey on fault attacks on symmetric key cryptosystems. *ACM Computing Surveys*, 55(4):1–34, 2022.

[5] Hadi Soleimany, Nasour Bagheri, Hosein Hadipour, Prasanna Ravi, Shivam Bhasin, and Sara Mansouri. Practical multiple persistent faults analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 367–390, 2022.

[6] Xue Gong, Fan Zhang, Xinjie Zhao, Jie Xiao, and Shize Guo. Key schedule guided persistent fault attack. *IEEE Transactions on Information Forensics and Security*, 2024.

[7] Nasour Bagheri, Sadegh Sadeghi, Prasanna Ravi, Shivam Bhasin, and Hadi Soleimany. Sipfa: statistical ineffective persistent faults analysis on feistel ciphers. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 367–390, 2022.

[8] Zahra Kazemi, David Hely, Mahdi Fazeli, and Vincent Beroulle. A review on evaluation and configuration of fault injection attack instruments to design attack resistant mcu-based iot applications. *Electronics*, 9(7):1153, 2020.

[9] Johannes Obermaier, Robert Specht, and Georg Sigl. Fuzzy-glitch: A practical ring oscillator based clock glitch attack. In *2017 International Conference on Applied Electronics (AE)*, pages 1–6. IEEE, 2017.

[10] Claudio Bozzato, Riccardo Focardi, and Francesco Palmarini. Shaping the glitch: optimizing voltage fault injection attacks. *IACR transactions on cryptographic hardware and embedded systems*, pages 199–224, 2019.

[11] Michael Hutter and Jörn-Marc Schmidt. The temperature side channel and heating fault attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 219–235. Springer, 2013.

[12] Jakub Breier, Dirmanto Jap, and Chien-Ning Chen. Laser-based fault injection on microcontrollers. In *Fault Tolerant Architectures for Cryptography and Hardware Security*, pages 81–110. Springer, 2018.

[13] Wei He, Jakub Breier, Shivam Bhasin, Noriyuki Miura, and Makoto Nagata. Ring oscillator under laser: potential of pll-based countermeasure against laser fault injection. In *2016 Workshop on fault diagnosis and tolerance in cryptography (FDTC)*, pages 102–113. IEEE, 2016.

[14] Dan Boneh, Richard A DeMillo, and Richard J Lipton. On the importance of checking cryptographic protocols for faults. In *International conference on the theory and applications of cryptographic techniques*, pages 37–51. Springer, 1997.

[15] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In *Annual international cryptology conference*, pages 513–525. Springer, 1997.

[16] Navid Vafaei, Maryam Porkar, Hamed Ramzanipour, and Nasour Bagheri. Practical differential fault analysis on skinny. *ISeCure*, 14(3), 2022.

[17] Hamed Ramzanipour, Navid Vafaei, and Nasour Bagheri. Practical differential fault analysis on craft, a lightweight block cipher. *ISeCure*, 14(3), 2022.

[18] Anit Kumar Ghosal, Amit Sardar, and Dipanwita Roy Chowdhury. Differential fault analysis attack-tolerant hardware implementation of aes. *Journal of Supercomputing*, 80(4), 2024.

[19] Amir Moradi, Mohammad T Manzuri Shalmani, and Mahmoud Salmasizadeh. A generalized method of differential fault attack against aes cryptosystem. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 91–100. Springer, 2006.

[20] Eli Biham, Louis Granboulan, and Phong Q Nguyen. Impossible fault analysis of rc4 and differential fault analysis of rc4. In *International Workshop on Fast Software Encryption*, pages 359–367. Springer, 2005.

[21] Patrick Derbez, Pierre-Alain Fouque, and Delphine Leresteux. Meet-in-the-middle and impossible differential fault analysis on aes. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 274–291. Springer, 2011.

[22] Johannes Blömer and Volker Krummel. Fault based collision attacks on aes. In *International Workshop on Fault Diagnosis and Tolerance in Cryptography*, pages 106–120. Springer, 2006.

[23] Ludger Hemme. A differential fault attack against early rounds of (triple-) des. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 254–267. Springer, 2004.

[24] Christophe Clavier. Secret external encodings do not prevent transient fault analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 181–194. Springer, 2007.

[25] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Transactions on computers*, 49(9):967–970, 2000.

[26] Nahid Farhady Ghalaty, Bilgiday Yuce, and Patrick Schaumont. Differential fault intensity analysis on present and led block ciphers. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 174–188. Springer, 2015.

[27] Christoph Dobraunig, Maria Eichlseder, Thomas Korak, Stefan Mangard, Florian Mendel, and Robert Primas. Sifa: exploiting ineffective fault inductions on symmetric cryptography. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 547–572, 2018.

[28] Yang Li, Kazuo Sakiyama, Shigeto Gomisawa, Toshinori Fukunaga, Junko Takahashi, and Kazuo Ohta. Fault sensitivity analysis. In *International workshop on cryptographic hardware and embedded systems*, pages 320–334. Springer, 2010.

[29] Navid Vafaei, Sara Zarei, Nasour Bagheri, Maria Eichlseder, Robert Primas, and Hadi Soleimany. Statistical effective fault attacks: The other side of the coin. *IEEE Transactions on Information Forensics and Security*, 17:1855–1867, 2022.

[30] Ali Asghar Beigizad, Hadi Soleimany, Sara Zarei, and Hamed Ramzanipour. Linked fault analysis. *IEEE Transactions on Information Forensics and Security*, 19:632–645, 2023.

[31] Sayandeep Saha, Arnab Bag, Debapriya Basu Roy, Sikhar Patranabis, and Debdeep Mukhopadhyay. Fault template attacks on block ciphers exploiting fault propagation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 612–643. Springer, 2020.

[32] Hagai Bar-El, Hamid Choukri, David Naccache, Michael Tunstall, and Claire Whelan. The sorcerer's apprentice guide to fault attacks. *Proceedings of the IEEE*, 94(2):370–382, 2006.

[33] Wei He, Jakub Breier, and Shivam Bhasin. Cheap and cheerful: A low-cost digital sensor for detecting laser fault injection attacks. In *International Conference on Security, Privacy, and Applied Cryptography Engineering*, pages 27–46. Springer, 2016.

[34] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In *International Conference on Cryptology in Africa*, pages 279–296. Springer, 2010.

[35] Marcel Medwed, Christoph Petit, Francesco Regazzoni, Mathieu Renauld, and François-Xavier Standaert. Fresh re-keying ii: securing multiple parties against side-channel and fault attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 115–132. Springer, 2011.

[36] Marc Joye, Michael Tunstall, et al. *Fault analysis in cryptography*, volume 147. Springer, 2012.

[37] Jakub Breier, Dirmanto Jap, and Shivam Bhasin. The other side of the coin: Analyzing software encoding schemes against fault injection attacks. In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 209–216. IEEE, 2016.

[38] Ramesh Karri, Kaijie Wu, Piyush Mishra, and Yongkook Kim. Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 21(12):1509–1517, 2002.

[39] Jingyi Feng, Hua Chen, Yang Li, Zhipeng Jiao, and Wei Xi. A framework for evaluation and analysis on infection countermeasures against fault attacks. *IEEE Transactions on Information Forensics and Security*, 15:391–406, 2019.

[40] Batya Karp, Maël Gay, Osnat Keren, and Ilia Polian. Detection and correction of malicious and natural faults in cryptographic modules. *PROOFS*, 7:68–82, 2018.

MonadiAfta-Bianual Scientific Journal for Cyberspace Security
2026, Volume 23, Number 1 (pp. 120–135)
https://monadi.isc.org.ir

An Overview of Transient Fault Attacks on Block Ciphers

Mehdi Farzanegan and Hadi Soleimany *

Cyberspace Research Institute, Shahid Beheshti University, Tehran, Iran

EXTENDED ABSTRACT

Keywords:
Fault attack, Fault injection, Active attack, Fault countermeasures

Type: Review paper

Fault injection attacks constitute a powerful class of active adversarial techniques within the gray-box threat model, where the attacker is assumed to have partial knowledge of the target system and physical access to the cryptographic device. Over the past decades, these attacks have evolved from theoretical constructs into highly practical and efficient tools capable of compromising even well-engineered cryptographic implementations. By deliberately inducing faults during the execution of cryptographic algorithms, an adversary can manipulate intermediate computations and exploit discrepancies between correct and faulty outputs to extract sensitive information, most notably secret keys.

This paper presents a comprehensive survey of transient fault injection attacks targeting block ciphers, focusing on both theoretical foundations and practical considerations. It begins by formalizing fault models, which serve as the critical link between physical fault injection techniques and algorithmic cryptanalysis. Fault models are characterized by several dimensions, including the fault resolution (bit-level, byte-level, or word-level), fault type (e.g., random, stuck-at, or bit-flip), and the number of affected variables. Additionally, spatial and temporal precision play a crucial role: while some attacks require precise targeting of specific operations at exact clock cycles, others operate under more relaxed assumptions. The classification also distinguishes between transient, persistent, and permanent faults, each with different implications for attack feasibility and complexity. A more precise fault model typically enables stronger attacks but increases the difficulty of practical realization.

The paper further explores various fault injection techniques, highlighting their operational principles and trade-offs. Clock glitching and voltage glitching are widely used due to their relatively low cost and ease of implementation, though they offer limited spatial precision. Environmental manipulations, such as temperature variations, can also induce faults but with lower controllability. Electromagnetic fault injection (EMFI) provides improved spatial targeting without requiring invasive access, while laser-based fault injection offers the highest precision, enabling attackers to target specific transistors or memory cells. However, such precision comes at the cost of expensive equipment and invasive chip preparation. These techniques are compared based on criteria such as spatial locality (local vs. global faults), required physical access (invasive vs. non-invasive), cost, and technical expertise.

A central contribution of the work is the systematic classification of fault analysis techniques into three main categories: differential, collision-based, and statistical methods. Differential Fault Analysis (DFA), one of the most influential approaches, leverages the propagation of injected faults through the cryptographic algorithm. By comparing correct and faulty ciphertexts, the attacker can infer information about intermediate states and gradually recover the secret key. The paper illustrates this concept through a detailed analysis of DFA applied to AES, where a single-byte fault injected in an intermediate round can affect multiple output bytes, significantly

* Corresponding author
Email addresses: m.farzanegan@mail.sbu.ac.ir (Mehdi Farzanegan), h_soleimany@sbu.ac.ir (Hadi Soleimany)
© 2026 ISC. All rights reserved.

reducing the key search space. Variants such as Impossible Differential Fault Analysis (IDFA) further refine this approach by exploiting impossible differential patterns to eliminate incorrect key candidates.

Collision-based methods, including CFA and IFA, exploit conditions such as output collisions or unchanged outputs rather than differences. In CFA, the attacker finds inputs that yield identical outputs in both faulty and fault-free executions, revealing constraints on internal states. In contrast, IFA identifies faults that do not alter the output, indicating specific properties of intermediate values. SEA further extends this idea by considering faults that affect unused internal data, leaving the output unchanged. These approaches are particularly useful when differential methods are limited by countermeasures.

Statistical fault analysis techniques form an advanced class of attacks that exploit biases in faulty output distributions, with methods such as SFA, SIFA, and SEFA often relying solely on faulty data, unlike DFA. SIFA is particularly effective against implementations with fault detection, as it leverages ineffective faults—those that do not alter the output—to extract key information even when outputs are suppressed or randomized, challenging the assumption that such faults are harmless. SEFA extends this approach by focusing on effective faults, offering improved robustness in noisy environments and requiring fewer samples for key recovery. In addition, advanced models such as Linked Fault Analysis (LFA) and Fault Template Attacks (FTA) exploit structural properties of implementations and fault propagation patterns, demonstrating that fault attacks can adapt to increasingly sophisticated scenarios, even with only partial system information available.

In the final part, the paper reviews countermeasures against fault injection attacks at three levels: physical, protocol, and algorithmic implementation. Physical-level defenses include shielding, sensors, and tamper-resistant packaging to prevent or detect external interference. Protocol-level approaches, such as periodic key refreshing, aim to limit the usefulness of extracted information. At the implementation level, techniques such as redundancy, concurrent error detection, output randomization, and error correction codes are widely employed. However, each of these methods introduces trade-offs in terms of performance, area overhead, or energy consumption. Moreover, many countermeasures can be bypassed by advanced attacks such as SIFA, which exploit the very mechanisms designed to detect faults.

In conclusion, fault injection attacks remain a critical challenge in the secure implementation of cryptographic systems. Their effectiveness stems from the interplay between physical fault mechanisms and algorithmic vulnerabilities. As the field advances, emerging research directions include combined fault and side-channel attacks, automated fault analysis using machine learning techniques, and the study of fault attacks on post-quantum cryptographic algorithms. Addressing these challenges requires a holistic, multi-layered approach that integrates robust design principles across all levels of the system.