معرفی حمله DCA روی الگوریتم‌های رمزنگاری جعبهٔ سفید

جواد علیزاده*[1]، محسن صدیقی[2] و هادی سلیمانی[3]

[1و2] دانشگاه جامع امام حسین(ع)، مرکز علم و فناوری فتح، تهران، ایران
jaalizadeh@ihu.ac.ir , mseddighi@ihu.ac.ir

[3] دانشگاه شهید بهشتی، پژوهشکده فضای مجازی، تهران، ایران
h_soleimany@sbu.ac.ir

چکیده

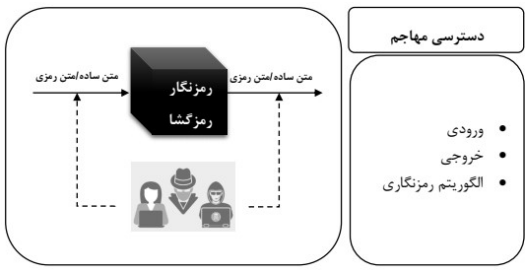
پیشرفت فناوری‌های اطلاعاتی و ارتباطی در عصر حاضر سبب استفاده از سامانه‌های نوین مانند تلفن همراه هوشمند شده است. این سامانه‌ها که دسترسی و آزادی عمل بیشتری در مقایسه با سایر سامانه‌ها به مهاجم می‌دهند، منجر به تعریف مدل رمزنگاری جعبهٔ سفید می‌شوند. در این مدل تلاش می‌شود تا کلید رمزنگاری در یک نوع پیاده‌سازی الگوریتم رمزنگاری پنهان شود. روش تحلیل محاسبات تفاضلی یک نوع حمله کانال جانبی است که روی طرح‌های رمزنگاری جعبهٔ سفید مطرح شده است. اهمیت این روش تحلیل از این جهت قابل تأمل است که توانست تمام طرح‌های جعبهٔ سفید ارائه‌شده در زمان خودش را بشکند. روش تحلیل محاسبات تفاضلی که یک روش تحلیل نرم‌افزاری است، شباهت‌هایی با روش تحلیل توان تفاضلی دارد که یک نوع حمله کانال جانبی شناخته‌شده در حوزه سخت‌افزاری است. در این مقاله به معرفی اصول کلی و نحوه انجام عملی روش تحلیل محاسبات تفاضلی پرداخته شده است. برای این منظور ابتدا این روش تحلیل به‌صورت نظری معرفی، سپس مراحل انجام آن تشریح می‌شود.

واژگان کلیدی: رمزنگاری جعبهٔ سفید، رمز قالبی، حمله کانال جانبی، حمله تحلیل توان تفاضلی، حمله تحلیل محاسبات تفاضلی.

۱- مقدمه

الگوریتم‌های رمزنگاری نقش مهم و اساسی در تأمین امنیت اطلاعات و ارتباطات دارند. برای اطمینان از این امر لازم است تا الگوریتم مورد استفاده امن باشد. در حالت کلی منظور از امنیتِ یک الگوریتم رمزنگاری، مقاومت آن در برابر مهاجمی است که می‌خواهد اثر حفاظتی این الگوریتم را از بین ببرد؛ بنابراین سؤالی که در این مورد مطرح می‌شود این است که این مهاجم چه توانمندی و چه امکاناتی در اختیار دارد. بسته به پاسخ این سؤال مدل‌های مختلف حمله به الگوریتم‌های رمزنگاری مطرح می‌شود. این مدل‌ها را می‌توان در سه دسته کلی جعبهٔ سیاه[1]، جعبهٔ خاکستری[2] و جعبهٔ سفید[3] در نظر گرفت [۱].

در مدل حمله جعبهٔ سیاه مطابق شکل (۱) مهاجم تنها می‌تواند از سامانهٔ رمزنگاری استفاده کند و به مجموعه‌ای از ورودی‌ها و خروجی متناظرشان دسترسی داشته و به‌جز الگوریتم رمزنگاری، هیچ اطلاعی از جزئیات و ساختار داخلی سامانه ندارد؛ بنابراین لازم است تا پیاده‌سازی سامانه رمزنگاری در مدل جعبهٔ سیاه در محیط امن و قابل اعتماد انجام شود [۲].



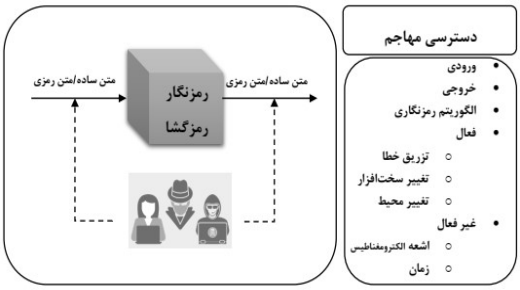
(شکل-۱): مدل جعبهٔ سیاه

مدل حمله جعبهٔ خاکستری با توجه به وجود برخی ضعف‌ها در مدل جعبهٔ سیاه، توسط کوچر[4] و همکاران معرفی شد [۳]. در این مدل حمله، مطابق شکل (۲) مهاجم علاوه‌بر ورودی و خروجی، اطلاعات محدودی نیز در مورد

[1] Black-Box
[2] Gray-Box
[3] White-Box
[4] Kocher

*نویسندهٔ مسؤول

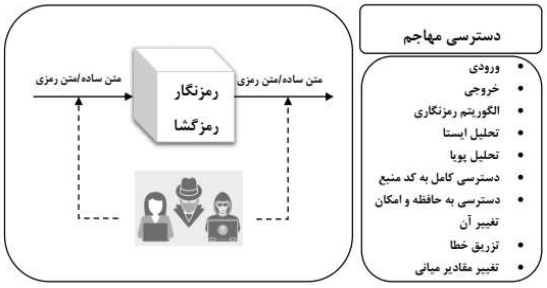
جزئیات داخلی سامانه دارد. به‌عبارت دیگر مهاجم دارای قدرت بیشتری نسبت به مدل حمله جعبهٔ سیاه است. برای مثال، می‌تواند توان مصرفی یک سامانهٔ رمزنگاری را اندازه‌گیری و یا یک نوع خطا در فرآیند اجرای سامانه رمزنگاری اعمال کند. حملات ارائه‌شده در این مدل تحت عنوان حملات کانال جانبی نامیده می‌شوند[۳ و ۴].



(شکل-۲): مدل جعبهٔ خاکستری

ایده مدل حمله جعبهٔ سفید این است که مهاجم می‌تواند در جایگاه مالک دستگاه در حال اجرای نرم‌افزار پیاده‌سازی‌شده باشد. ازاین‌رو مهاجم دارای توانایی بیشتری نسبت به مدل‌های قبلی بوده و فرض می‌شود که کنترل کامل بر محیط اجرا دارد. در این مدل مطابق شکل (۳) مهاجم از تمام جزئیات سامانهٔ رمزنگاری مانند مقادیر داده‌ها، منابع حافظه و حتی نتایج میانی و غیره اطلاع داشته و می‌تواند از ابزارهای مورد نظر خود به‌منظور بازیابی کلید محرمانه استفاده کند. همچنین مهاجم می‌تواند تغییرات دلخواه خود را روی سامانهٔ مورد نظر اعمال کند. علاوه‌بر دسترسی‌های بیان‌شده، در این مدل دسترسی کامل به پیاده‌سازی الگوریتم‌ها وجود داشته و الگوریتم داخلی به‌طورکامل قابل مشاهده و تغییر است. برخلاف مدل جعبهٔ سیاه پیاده‌سازی سامانهٔ رمزنگاری در مدل جعبهٔ سفید در یک محیط باز و بدون نیاز به اعتماد انجام می‌شود.

مدل رمزنگاری جعبهٔ سفید برای بار نخست در سال ۲۰۰۲ توسط چاو[1] و همکاران به همراه نخستین پیشنهاد پیاده‌سازی جعبهٔ سفید برای رمز AES و DES معرفی شد [3,4]. به این دلیل که مدل رمزنگاری جعبهٔ سفید و فرضیات در مورد توانمندی‌های مهاجم در مدل یادشده در سال ۲۰۰۲ (با توجه به نوع سامانه‌های ارتباطی) تا حدی از واقعیت و عملی‌بودن فاصله داشت، این موضوع تا سال ۲۰۱۴ مورد استقبال چندانی قرار نگرفت؛ هر چند که در این مدت پیشرفت‌هایی در حوزه صنعتی رمزنگاری جعبهٔ سفید حاصل شد؛ اما در سال‌های اخیر با توجه به رشد اینترنت اشیا از یک سمت و افزایش استفاده از فناوری‌های نوین مانند تلفن هوشمند از سمت دیگر، رمزنگاری جعبهٔ سفید اهمیت بیشتری پیدا کرده است. برای تأکید روی این موضوع می‌توان به چالش‌هایی اشاره کرد که در کنفرانس‌های CHES 2017 [۵] و CHES 2019 [۶] در رابطه با رمزنگاری جعبهٔ سفید مطرح شده‌اند.



(شکل-۳): مدل جعبهٔ سفید

با توجه به مقالات و گزارش‌های ارائه‌شده، تاکنون بیشتر طرح‌های رمزنگاری جعبهٔ سفید بعد از گذشت زمان به‌نسبه کوتاهی از ارائه آنها شکسته شده‌اند [۵، ۶]. حملات ارائه‌شده روی این طرح‌ها به‌طورکلی به دو دستهٔ حملات نظری و حملات کانال جانبی دسته‌بندی می‌شوند. حملات کانال جانبی به‌دلیل عملی‌بودن، داشتن پیچیدگی کم، قابلیت اجرا روی چندین طرح و همچنین داشتن قابلیت خودکارسازی از اهمیت خاصی برخوردار هستند. یکی از حملات کانال جانبی معرفی‌شده در حوزه رمزنگاری جعبهٔ سفید حمله تحلیل محاسبات تفاضلی[2] است. این حمله توسط باس[3] و همکاران در سال ۲۰۱۶ به‌منظور ارزیابی امنیت پیاده‌سازی‌های جعبهٔ سفید ارائه شد [۷، ۸]. تحلیل محاسبات تفاضلی را می‌توان نوع نرم‌افزاری تحلیل توان تفاضلی[4] (DPA) در نظر گرفت. با این وجود تفاوت‌های مهمی بین استفاده از نرم‌افزار و ردیابی‌های سخت‌افزاری وجود دارد. این حمله روی برخی پیاده‌سازی‌های نرم‌افزاری جعبهٔ سفید مانند چالش Wyseur [۹]، چالش Hack.lu [۱۰]، چالش SSTIC 2012 [۱۱] و غیره اعمال و موفق به شکستن آنها شده است.

سلیمانی و صادقی در [۱۲] رویکردها، کاربردها و چالش‌های رمزنگاری جعبهٔ سفید را با تمرکز بر پیاده‌سازی امن رمزهای قالبی استاندارد بیان کرده‌اند. در ادامه کار

[1] Chow

[2] Differential Computation Analysis
[3] Bos
[4] Differential Power Analysis (DPA)

ایشان، در این مقاله به بررسی روش‌های تحلیل و ارزیابی طرح‌های رمزنگاری جعبهٔ سفید با تمرکز بر حمله تحلیل محاسبات تفاضلی پرداخته می‌شود. بدین منظور بقیه مطالب این مقاله به‌صورت زیر سازمان‌دهی شده است. در بخش دوم، برخی کاربردهای رمزنگاری جعبهٔ سفید بیان می‌شود. در بخش سوم، طرح‌های رمزنگاری جعبهٔ سفید، از نقطه نظر طراحی، در نظر گرفته شده و یک دید کلی از اصول طراحی این طرح‌ها ارائه می‌شود. در بخش چهارم، برخی حملات ارائه‌شده روی طرح‌های رمزنگاری جعبهٔ سفید معرفی و در بخش پنجم مراحل و ویژگی‌های حمله DCA با جزئیات بیشتر شرح داده می‌شود؛ درنهایت جمع‌بندی و نتیجه‌گیری مقاله در بخش ششم ارائه می‌شود.

۲- کاربردهای رمزنگاری جعبهٔ سفید

اگر چه رمزنگاری جعبهٔ سفید با هدف اصلی مدیریت حقوق دیجیتال مطرح شد [۱، ۲]، ولی در حال حاضر با افزایش تقاضا برای ارتقای امنیت در کاربردهای نرم‌افزاری در دستگاه‌هایی مانند رایانه‌های شخصی، تلفن همراه هوشمند، سرور و همچنین زمینه روبه‌رشد خدمات مبتنی بر ابر، هدف رمزنگاری جعبهٔ سفید متنوع شده است. برای مثال، رمزنگاری جعبهٔ سفید شامل مقاومت در برابر بدافزارها و کرم‌ها نیز می‌شود؛ بنابراین، این مدل کاربردهایی در بانکداری و سایر حوزه‌های مهم امنیتی هم خواهد داشت. از دیگر کاربردهای رمزنگاری جعبهٔ سفید می‌توان از انجام معاملات بانکی ایمن از طریق اینترنت، ارتباطات محرمانه، ابزارهای رمزنگاری نظامی، تأیید اعتبار سرویس‌های وب و غیره نام برد. در ادامه دو کاربرد مهم رمزنگاری جعبهٔ سفید به‌اختصار توصیف شده است. برای آشنایی با سایر کاربردهای این مدل رمزنگاری، می‌توان به [۱۲] مراجعه کرد.

۲-۱- مدیریت حقوق دیجیتال[1] (DRM)

ناشران، نویسندگان و سایر سازندگان محتوا از برنامه‌ای استفاده می‌کنند که در آن رسانه‌ها، داده‌ها، کتاب الکترونیکی، محتوا، نرم‌افزار یا سایر مطالب دارای حق چاپ رمزگذاری می‌شوند و فقط کسانی که کلید رمزگشایی را دارند، می‌توانند به محتوا دسترسی پیدا کنند. این مفهوم تحت عنوان مدیریت حقوق دیجیتال نامیده می‌شود [۱۳]. هدف مدیریت حقوق دیجیتال جلوگیری از توزیع مجدد محتوای دیجیتالی و محدودکردن راه‌هایی است که مصرف‌کنندگان می‌توانند محتوای خریداری‌شده خود را کپی کنند.

در دنیای دیجیتالی امروز، اهمیت مدیریت حقوق دیجیتال نه‌تنها برای سازندگان محتوای دیجیتال بلکه برای شرکت‌ها و افرادی که از محتوای دیجیتال استفاده می‌کنند، به‌طور قابل توجه در حال افزایش است. برای تأمین امنیت در این حوزه، از رمزنگاری جعبهٔ سفید استفاده می‌شود. برای این کار لازم است تا در دستگاه کاربر که محتوا را پخش می‌کند، با استفاده از پیاده‌سازی جعبهٔ سفید از کلید مخفی در برابر حملات استخراج کلید، حملات کانال جانبی و غیره محافظت شود.

۲-۲- شبیه‌سازی کارت میزبان[2] (HCE)

مطابق شکل (۴)، HCE یک فناوری برای ایمن‌سازی تلفن همراه به‌منظور انجام معاملات اعتباری است. در هنگام استفاده از این فناوری، رمزنگاری جعبهٔ سفید بر روی تلفن همراه، اساس کلی امنیت است. پیاده‌سازی جعبهٔ سفید باید امنیت را از دو طریق تأمین کند. ابتدا باید از داده‌های حساس مانند رمزهای یک بار مصرف، اطلاعات پرداخت و داده‌های کارت در برابر نرم‌افزارهای مخرب و جاسوسی که به‌احتمال در همان CPU در حال اجرا هستند، محافظت کند. دوم، باید اطمینان حاصل شود که دستگاه‌های معتبر و کاربران با استفاده از احراز هویت امن مطمئن بین ابر و دستگاه، به اعتبارنامه پرداخت خود در ابر دسترسی پیدا می‌کنند [۱۳].

(شکل-۴): نحوه کارکرد HCE

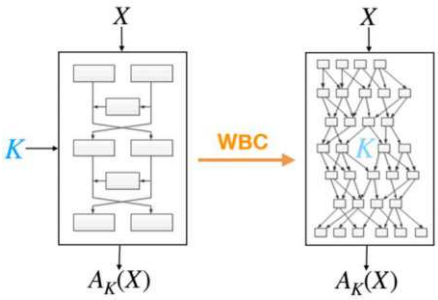
[1] Digital Right Management

[2] Host Card Emulation

۳- طرح‌های رمزنگاری جعبهٔ سفید

در مدل رمزنگاری جعبهٔ سفید، هدف این است که اگر مهاجم دسترسی کامل به یک الگوریتم رمزنگاری جعبهٔ سفید داشته باشد، قادر به کشف هیچ‌گونه اطلاعات در مورد محتوای کلید مخفی با استفاده از بررسی داخلی این پیاده‌سازی نباشد. این امر معادل یک محیط با دسترسی جعبهٔ سیاه است؛ بنابراین چنین پیاده‌سازی جعبهٔ سفیدی باید در برابر تمام حملات شناخته‌شده مقاومت کند.

روش کلی طراحی طرح‌های رمزنگاری جعبهٔ سفید در شکل (۵) نشان داده شده است. روش رایج برای این نوع طراحی، استفاده از شبکه‌ای از جداول مبنا است.



(شکل-۵): پنهان‌کردن کلید در طراحی رمزنگاری جعبهٔ سفید [۶]

در طرح‌های رمزنگاری جعبهٔ سفید کلید به‌صورت مستقیم به سامانه وارد نمی‌شود؛ بلکه در خود ساختار تعبیه شده و یک جزو از آن است. برای مثال برای پیاده‌سازی رمز می‌توان مقادیر را با توجه به یک کلید ثابت محاسبه و در جداول مبنا ذخیره کرد. بدین صورت کلید ثابت فرض‌شده در درون جداول مبنا قرار گرفته و به‌صورت مستقیم به سامانه وارد نمی‌شود؛ بنابراین با این وجود که مهاجم دسترسی جعبهٔ سفید به سامانه دارد، اما به کلید دسترسی ندارد. با توجه به توضیحات در مورد اصول طراحی طرح‌های رمزنگاری جعبهٔ سفید، طرح‌های ارائه‌شده را می‌توان در حالت کلی در دو دسته در نظر گرفت. دسته نخست شامل الگوریتم‌هایی هستند که از یک رمز قالبی موجود مانند AES یا DES استفاده کرده و از روش‌های مختلفی برای مبهم‌کردن[1] فرآیند رمزگذاری آن استفاده می‌کنند؛ به‌طوری‌که یک مهاجم در مدل جعبهٔ سفید قادر به استخراج کلید مخفی نباشد. طرح‌هایی از قبیل چاو و Xiao-Lai و غیره در این دسته قرار می‌گیرند [۱، ۲، ۳، ۱۴، ۱۵، ۱۶]. دسته دوم شامل طراحی رمزهای قالبی جعبهٔ سفید جدید با محافظت داخلی هستند. به‌طورمعمول چنین طرح‌هایی مبتنی بر مؤلفه‌های وابسته به کلید، مانند S-boxهای وابسته به کلید بوده و به‌گونه‌ای طراحی شده‌اند که حتی اگر یک مهاجم جعبهٔ سفید بتواند فهرست کاملی از چنین مؤلفه‌ای را بازیابی کند، باز هم نمی‌تواند کلید مخفی را بازیابی کند. طرح‌های ASASA، SPACE، SPN-Box، WhiteBlock و WEM جزء این دسته هستند [۴، ۱۷، ۱۳، ۱۸، ۱۹].

[1] Obfuscate

۴- روش‌های تحلیل رمز مدل جعبهٔ سفید

همان‌طورکه در مقدمه عنوان شد، حملات ارائه‌شده در حوزه رمزنگاری جعبهٔ سفید به دو دستهٔ حملات نظری و حملات کانال جانبی دسته‌بندی می‌شوند. برخلاف حمله‌های نظری که از روابط و ساختارهای جداول مبنا در پیاده‌سازی استفاده می‌کنند، حملات کانال جانبی از روش‌های توسعه‌یافتهٔ مدل جعبهٔ خاکستری بهره می‌برند. از حملات کانال جانبی مهم روی طرح‌های رمزنگاری جعبهٔ سفید می‌توان به حمله تحلیل خطای تفاضلی[2] [۷، ۸]، حمله شمارش تفاضل صفرها[3] [۲۰] و حمله تحلیل محاسبات تفاضلی [۷، ۸] اشاره کرد. در ادامه این بخش دو روش حمله تحلیل خطای تفاضلی و حمله شمارش تفاضل صفرها که در مقایسه با روش‌های دیگر بیشتر مورد توجه قرار گرفته‌اند، به‌اختصار معرفی می‌شوند. روش حمله تحلیل محاسبات تفاضلی که موضوع اصلی این مقاله است، در بخش ۵ مقاله توصیف می‌شود.

۴-۱- حمله تحلیل خطای تفاضلی (DFA)

باس و همکاران در سال ۲۰۱۶ حمله تحلیل خطای تفاضلی نرم‌افزاری را که مشابه حملات تزریق خطا در حوزه سخت‌افزاری است، معرفی کردند [۷، ۸]. این روش شامل تزریق خطا با استفاده از ابزارهای نرم‌افزاری (و سخت‌افزاری) و مقایسه نتایج به‌دست‌آمده با نتایج مورد انتظار در حالت عادی است. در این روش مهاجم باید موقعیت تزریق خطا را در سامانه مشخص کند. مهاجم می‌تواند یک تحلیل پویا یا ایستا را اجرا یا اینکه به‌طور تصادفی خطا را تزریق کند. یکی از نقاط ضعف این حمله در زمینه جعبهٔ سفید، محدودبودن به پیاده‌سازی‌هایی است که از کدگذاری خارجی در خروجی

[2] Differential Fault Analysis
[3] Zero Difference Enumeration

رمزگذاری یا رمزگشایی استفاده نکرده باشد [۷، ۸]. یک نمونه دیگر از این نوع حملات تغییر فرکانس است که زمان انجام هر عملیات را مشخص می‌کند. بدین منظور فرکانس ساعت در یک دوره مشخص افزایش داده می‌شود تا مانع از اجرای صحیح برخی عملیات شود. این امر منجر به محاسبه نادرست خروجی سامانه می‌شود. درنهایت نتایج به‌دست‌آمده با استفاده از روش‌های تحلیل رمز تفاضلی به‌منظور بررسی وابستگی‌های کلید تحلیل می‌شوند.

۲-۴- حمله شمارش تفاضل صفرها (ZDE)

در سال ۲۰۱۷ بانیک[1] و همکاران در [۲۰] حمله شمارش تفاضل صفرها را معرفی کردند که بر اساس یک آزمون آماری ساده عمل می‌کند. در این حمله زوج‌های خاصی از متن‌های ساده تصادفی انتخاب شده، سپس مقادیر محاسبه‌شده در طی انجام رمزگذاری به‌وسیلهٔ پیاده‌سازی جعبهٔ سفید مورد تحلیل و بررسی قرار می‌گیرد. مطابق شکل (۶) زوج‌های بیان‌شده به‌گونه‌ای انتخاب می‌شوند که در صورت صحیح‌بودن حدس کلید، تعداد قابل توجهی از بایت‌ها در طول رمزگذاری زوج متن ساده انتخاب‌شده، یکسان باشند. تعیین کلید صحیح پس از آزمایش این‌که کدام جفت‌ها در حالت AES بیشترین تعداد بایت برابر را در طول رمزگذاری دارند، صورت می‌گیرد.

۵- حمله تحلیل محاسبات تفاضلی (DCA)

در بخش مقدمه توضیحاتی در رابطه با حمله DCA، اهمیت آن و نیز مشابهت آن با حمله DPA که روی سخت‌افزار انجام می‌شود، آورده شد. در این بخش حمله DCA روی طرح‌های رمزنگاری جعبهٔ سفید با جزییات بیشتر مورد بررسی قرار می‌گیرد. برای این کار ابتدا مراحل انجام حمله تشریح، سپس به ویژگی‌های این حمله اشاره و درنهایت روش‌های مقاوم‌سازی در برابر این حمله معرفی می‌شود.

۱-۵- مراحل انجام حمله DCA

حمله DCA نیز همانند حمله DPA شامل دو قسمت کلی جمع‌آوری داده و تحلیل داده است. در این طرح به‌منظور تحلیل داده‌ها از روش تفاضل میانگین ردها[2] استفاده می‌شود. منظور از ردها در اینجا مقادیر محاسبه و استفاده‌شده هنگام اجرای نسخه نرم‌افزاری الگوریتم رمزنگاری جعبهٔ سفید است. این مقادیر شامل اطلاعات در مورد محتویات حافظه‌اند که قابل دسترسی هستند. در این بخش حمله DCA با تمرکز روی پیاده‌سازی جعبهٔ سفید رمز قالبی AES-128 تشریح می‌شود. یکی از ویژگی‌های این حمله عدم نیاز به دانش در مورد نحوه پیاده‌سازی سامانهٔ رمزنگاری است؛ بنابراین توضیحات داده شده برای هر پیاده‌سازی جعبهٔ سفید از رمز AES-128 قابل اجرا است.

روش ارائه‌شده برای ردهای به‌دست‌آمده از پیاده‌سازی سایر رمزها نیز به همین ترتیب است. هدف حمله تعیین کلید دور نخست AES است. کلید دور نخست AES، ۱۲۸ بیت طول دارد که با یک حمله مبتنی بر بایت، می‌توان آن را بازیابی کرد. در ادامه روی بازیابی نخستین بایت کلید دور نخست تمرکز شده است. برای بازیابی بایت‌های دیگر نیز به‌صورت مشابه عمل می‌شود.

۱. جمع‌آوری رد

پس از مشخص‌شدن الگوریتم مورد استفاده، مطابق شکل ۷، n متن ساده مختلف p_e $(1 \le e \le n)$ به‌عنوان ورودی به سامانهٔ جعبهٔ سفید در نظر گرفته شده و در هر اجرا، یک رد نرم‌افزاری t_e ثبت می‌شود. شکل (۸) یک رد نرم‌افزاری را متشکل از صد بیت نشان می‌دهد. هر نمونه مربوط به یک بیت از نشانی‌های حافظه قابل دسترس در طول اجرا است.

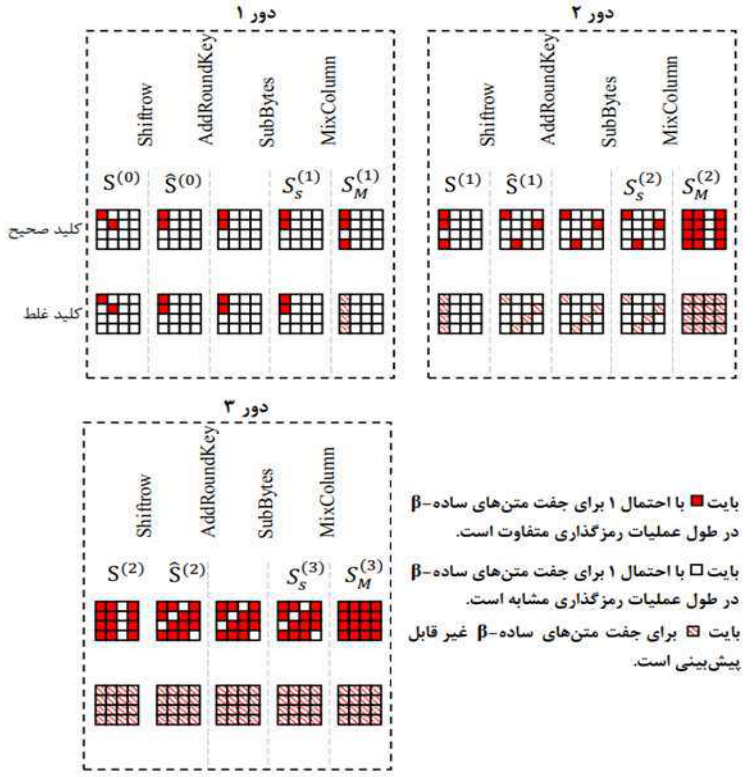
در هنگام ثبت رد، اگر اجرای کامل زمان زیادی ببرد می‌توان دامنه ردیابی را همانند حملات شبه DPA محدود کرد و بسته به اینکه حمله در کدام قسمت از رمز انجام می‌شود، فقط در همان قسمت ردیابی را انجام داد. به‌طور مثال می‌توان روی دور نخست یا دور آخر تمرکز کرد. تفاوت این مرحله با مرحله قبل در میزان اطلاعات ثبت شده است. در مرحله نخست تمام اطلاعات مربوط به حافظه ثبت می‌شود؛ درحالی‌که در این مرحله تنها محدوده‌ای خاص از حافظه به‌صورت اختیاری ثبت می‌شود. ردهای ثبت‌شده به‌طورمعمول شامل بایت‌هایی از حافظه که خوانده و یا در پشته[3] (یک نوع ساختمان داده که برای نگهداری داده‌ها استفاده می‌شود) نوشته شده و یا دست‌کم تعداد قابل توجهی از نشانی‌های حافظه قابل دسترس هستند. استفاده از این روش در ثبت ردهای نرم‌افزاری علاوه‌بر دادن نتیجه مطلوب، ذخیره‌سازی ردهای نرم‌افزاری را به کمینه

[1] Banik
[2] Traces
[3] Stack

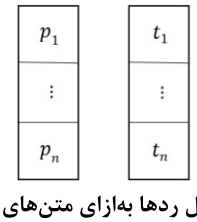
می‌رساند. البته در صورت نبود مشکل از بابت ذخیره‌سازی، می‌توان اجرای کامل را به‌عنوان رد ثبت کرد؛ درنهایت مجموعه‌ای از ردیابی‌های نرم‌افزاری که شامل قسمتی از نشانی‌ها یا داده‌های واقعی می‌شوند به‌دست می‌آیند. برای تبدیل به یک نمایش مناسب برای انجام تحلیل‌های توان مرسوم، داده‌ها به‌صورت بردارهای صفر و یک مرتب می‌شود.

۲. محاسبه مقدار میانی

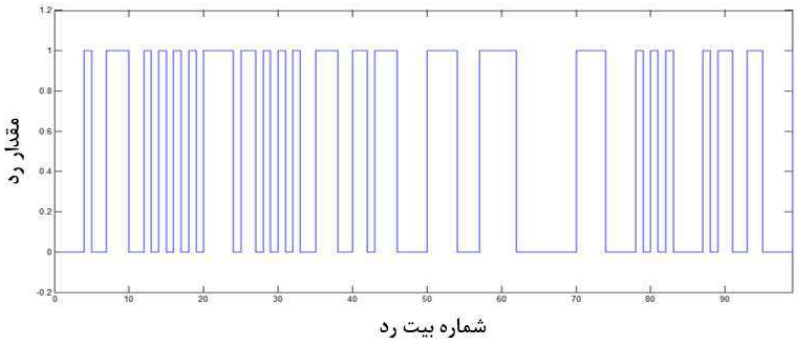
در این مرحله یک تابع انتخاب برای محاسبه یک بایت مقدار میانی از فرآیند محاسبه AES تعریف می‌شود. به‌عبارت دیگر، یک بایت مقدار میانی محاسبه می‌شود که در تکرار واقعی این حمله، وابسته به بایت کلیدی که تحلیل می‌شود، است. این بایت مقدار میانی z نامیده می‌شود. تابع انتخاب تنها یک بیت از مقدار میانی z را به‌عنوان بیت هدف برمی‌گرداند. این مقدار به‌عنوان یک تمایزگر در مراحل بعدی استفاده می‌شود. در ادامه مقدار میانی z بر اساس ورودی مختلف و کلیدهای ممکن نشان داده می‌شود.



(شکل-۶): نحوه انجام حمله ZDE



(شکل-۷): ماتریس شامل ردها به‌ازای متن‌های ساده تصادفی متفاوت

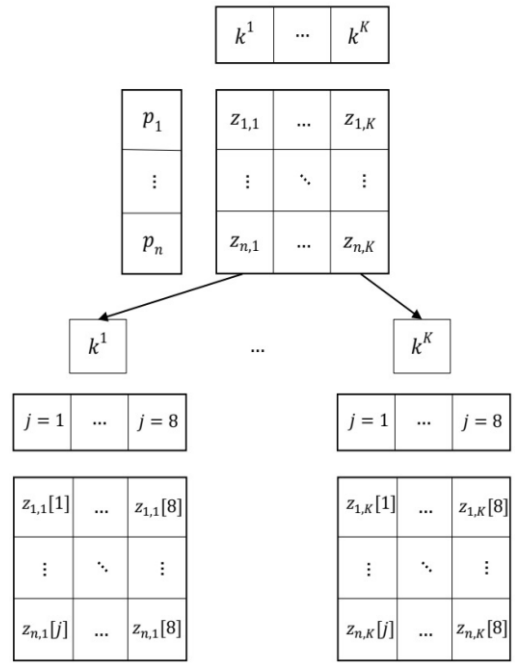


(شکل-۸): نمونه‌ای از یک رد نرم‌افزاری صد بیتی

در رمز AES، تابع انتخاب $Sel(p_e, k^h, j)$، بایت مقدار میانی z را در خروجی S-Box دور اول محاسبه کرده و ژامین بیت از z را مطابق رابطه (۱) برمی‌گرداند ($1 \le j \le 8$).

$$Sel(p_e, k^h, j) := SBox(p_e \oplus k^h)[j] = b \in \{0,1\} \quad (۱)$$

بسته به پیاده‌سازی جعبهٔ سفید که مورد تحلیل قرار می‌گیرد، ممکن است، بسته به این که روی کدام بیت j تمرکز می‌شود، همبستگی قوی بین b و ردهای نرم‌افزاری تنها برای بعضی از بیت‌های z قابل مشاهده باشد؛ بنابراین، مراحل ۳، ۴ و ۵ برای هر بیت j از z مطابق شکل (۹) تکرار می‌شود.



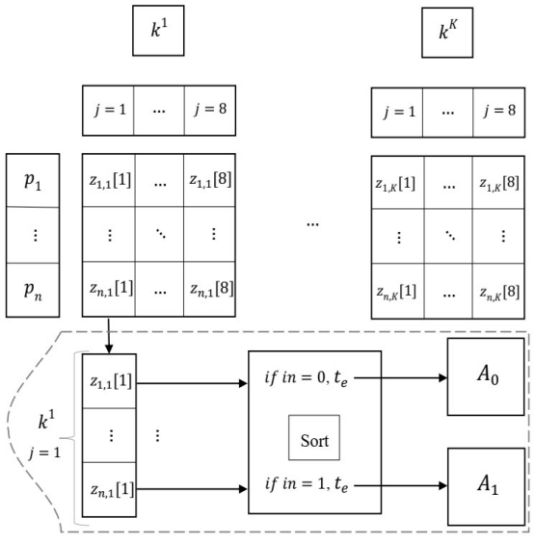
(شکل-۹): نحوه انتخاب بیت‌های مختلف مقدار میانی

۳. دسته‌بندی ردها

در این مرحله هر یک از ردهای t_e به یکی از دو مجموعه A_0 یا A_1 بر اساس مقدار $Sel(p_e, k^h, j) = b$ مطابق رابطه (۲) طبقه‌بندی می‌شود:

$$A_b := \{t_e | 1 \le e \le n, Sel(p_e, k^h, j) = b\} \quad For\ b \in \{0,1\} \quad (۲)$$

نحوه محاسبه رابطه (۲) برای فرض کلید k^1 و بیت j = 1 در شکل (۱۰) نشان داده شده است. برای ستون‌های دیگر نشان داده‌شده نیز به‌صورت مشابه عمل می‌شود.



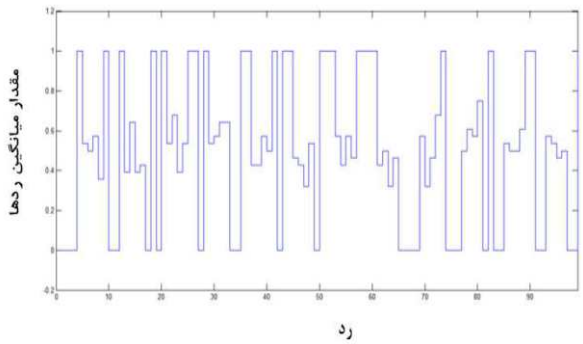
(شکل-۱۰): نحوه دسته‌بندی ردها

۴. میانگین مجموعه‌ها

در این مرحله برای هر یک از دو مجموعه ردهای به‌دست‌آمده در مرحله قبل یک رد میانگین مطابق رابطه (۳) محاسبه می‌شود؛ بدین منظور تمام ردهای معقول از هر مجموعه جمع و بر مجموع تعداد آثار آن مجموعه تقسیم می‌شود؛ بنابراین برای $b \in \{0,1\}$:

$$\bar{A}_b := \frac{\sum_{t \in A_b} t}{|A_b|} \quad (۳)$$

برای هر یک از دو مجموعه، می‌توان میانگین ردها را مانند شکل (۱۱) نشان داد.

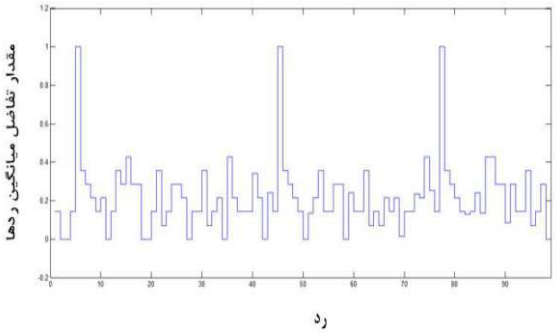


(شکل-۱۱): نمونه‌ای از میانگین ردها یک دسته

۵. تفاضل میانگین‌ها

در این مرحله تفاضل بین میانگین مجموعه ردهای به‌دست‌آمده در مرحله قبل مطابق رابطه (۴) محاسبه می‌شود. شکل (۱۲) تفاضل ناشی از میانگین‌ها را نشان می‌دهد:

$$\Delta = |\bar{A}_0 - \bar{A}_1| \quad (۴)$$



(شکل-۱۲): نمونه‌ای از تفاضل میانگین ردها

۶. بهترین بیت هدف

در این مرحله تفاضل میانگین ردهای به‌دست‌آمده برای همهٔ بیت‌های هدف j برای یک فرض کلید k^h داده شده مقایسه می‌شود. Δ^j تفاضل بین میانگین ردهای به‌دست‌آمده برای بیت هدف j بوده و $H(\Delta^j)$ بالاترین پیک در رد Δ^j باشد؛ سپس، Δ^j به‌عنوان بهترین تفاضل میانگین ردها برای k^h مطابق رابطه (۵) انتخاب می‌شود؛ به‌طوری‌که $H(\Delta^j)$ بیشینه مقدار بین بالاترین پیک‌ها از همه تفاضل میانگین‌های ردهای دیگر است؛ برای مثال:

$$\forall\, 1 \le j' \le 8, H(\Delta^{j'}) \le H(\Delta^j) \qquad (۵)$$

به‌عبارت دیگر، به بالاترین پیک به‌دست‌آمده از هر تفاضل میانگین‌های رد نگاه می‌شود؛ درنهایت تفاضل میانگین‌های رد با بالاترین پیک $H(\Delta^j)$ به‌عنوان تفاضل میانگین‌های به‌دست‌آمده برای فرض کلید k^h در تکرار واقعی حمله مورد تحلیل قرار می‌گیرد؛ به‌طوری‌که $\Delta^h := \Delta^j$.

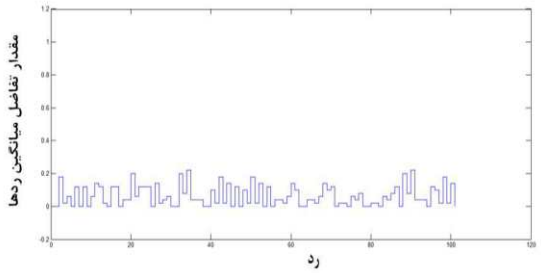
۷. بهترین فرض بایت کلید

فرض کنید Δ^h تفاضل میانگین‌های رد برای فرض کلید h و $H(\Delta^h)$ بالاترین پیک در رد Δ^h باشد. طبق رابطه (۶)، k^h طوری انتخاب می‌شود که $H(\Delta^h)$ بیشینه مقدار در میان سایر تفاضل میانگین‌های ردها Δ^h باشد. به‌عبارت دیگر بیشترین مقدار را بین همه مقادیر با درنظرگرفتن کلیدهای مختلف داشته باشد:

$$\forall\, 1 \le h' \le 256, H(\Delta^{h'}) \le H(\Delta^h) \qquad (۶)$$

برای $H(\Delta^h)$ بالاتر، احتمال بیشتری دارد که حدس کلید درست باشد. برای توضیح این موضوع فرض کنید در فرآیند حمله، ردها در دو مجموعه A_0 و A_1 بر اساس این‌که آیا بیت z، صفر یا یک است تقسیم‌بندی شوند. اگر فرض کلید صحیح باشد، انتظار می‌رود که دسترسی به حافظه در A_0 متفاوت از A_1 باشد، زیرا مقدار بیت هدف متفاوت است. ازاین‌رو انتظار می‌رود که این تفاوت در میانگین ردها برای A_0 و A_1 منعکس شود که منجر به پیک در تفاضل میانگین‌های رد می‌شود. در مقابل اگر فرض کلید درست نباشد، مجموعه‌های A_0 و A_1 را می‌توان به‌عنوان یک مجموعه تصادفی از ردها در نظر گرفت. بنابراین z می‌تواند هر مقدار دلخواه را در A_0 و A_1 بگیرد؛ ازاین‌رو، انتظار تفاوت زیادی بین ردهای اجراها از A_0 و A_1 وجود ندارد؛ به‌طوری‌که منجر به یک تفاضل میانگین‌های رد با مقادیر کم مطابق شکل (۱۳) شود.

داشتن بیشینه مقدار یک بدین معنی است که برای تمام ردها در A_0، بیت نشانی حافظه مورد نظر برابر با صفر و این بیت برای تمام ردها در A_1 برابر با یک (یا برعکس) است. به‌عبارت دیگر، بیت هدف z[j] به‌صورت مستقیم یا در شکل منفی آن در نشانی حافظه در دسترس، در پیاده‌سازی موجود است.



(شکل-۱۳): نمونه‌ای از تفاضل میانگین ردها برای حدس کلید نادرست

در ادامه برای آشنایی بیشتر با حمله DCA، روند پردازش، چگونگی به‌دست‌آوردن رد نرم‌افزاری و همچنین نحوه تحلیل ردها با توجه به شکل (۱۴)، به‌صورت زیر توصیف می‌شوند:

۱- تعدادی متن ساده به‌عنوان ورودی به سامانهٔ رمزنگاری (رمزگشایی) داده شده و در حین اجرا تغییرات حافظه ثبت می‌شود. این تغییرات رد متناظر با آن ورودی است.

۲- ورودی داده‌شده به سامانهٔ رمزنگاری، به الگوریتم در حال اجرا (AES-128) داده شده و یک مقدار میانی، با توجه به بایت فرضی کلید محاسبه می‌شود. این مقدار میانی به‌عنوان تمایزگر عمل می‌کند.

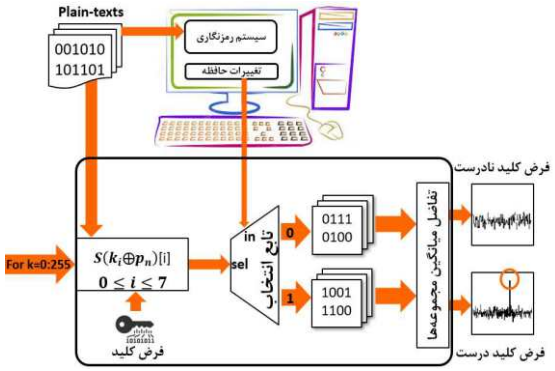
۳- با توجه به تمایزگر حاصل در مرحله دو، ردهای ذخیره‌شده در مرحله یک به دو دسته تقسیم‌بندی می‌شوند.

۴- برای هر یک از دو مجموعه، میانگین ردها حساب شده و از هم کم می‌شوند.

تکرارها

۱- با توجه به این‌که مقدار به‌دست‌آمده در مرحله دو، اندازه بایتی دارد، بنابراین هشت حالت ممکن برای انتخاب تمایزگر وجود دارد که برای هر یک از هشت حالت ممکن مراحل سه و چهار را محاسبه کرده و بیشترین مقدار به‌عنوان احتمال درست‌بودن آن کلید در نظر گرفته می‌شود.

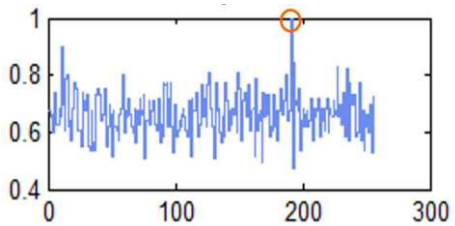
۲- برای حمله به هر بایت از کلید با توجه به ۲۵۶ حالت ممکن برای آن، مراحل دو، سه و چهار اجرا و بیشترین مقدار موجود در نتیجه به‌عنوان احتمال درست‌بودن آن کلید در نظر گرفته می‌شود.



(شکل-۱۴): مراحل انجام حمله DCA

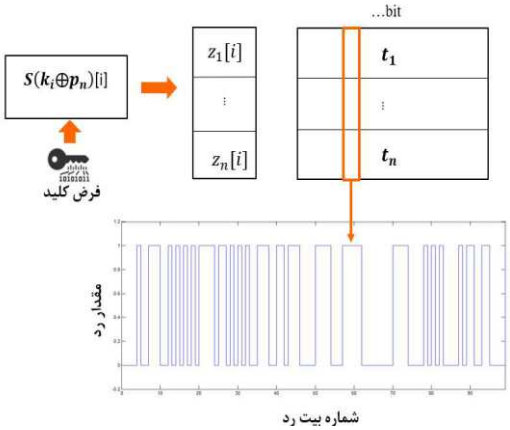
انتخاب کلید درست

با توجه به مراحل توصیفی برای ۲۵۶ حالت ممکن کلید و هشت حالت ممکن برای انتخاب تمایزگر، مقدار تفاضل میانگین مجموعه‌ها به‌دست آورده شده است. احتمال درست‌بودن هر فرض کلید برابر با بیشترین مقدار موجود در تفاضل میانگین مجموعه‌های به‌دست‌آمده برای آن کلید در هر هشت حالت ممکن است. احتمال حاصل برای هر یک از ۲۵۶ حالت ممکن برای کلید در یک مثال خاص در شکل (۱۵) نمایش داده شده است. کلید درست دارای بیشترین مقدار در بین دیگر کلیدها است.



(شکل-۱۵): احتمال حاصل برای ۲۵۶ حالت ممکن کلید

به‌منظور نحوه انجام حمله و تشخیص کلید درست مطابق شکل (۱۶)، تمایزگر انتخاب‌شده در حین اجرای حمله، در الگوریتم مورد نظر، در یک مکان از نشانی حافظه نوشته می‌شود؛ بنابراین با تطابق مقادیر نوشته‌شده در حافظه (ردها) با مقادیر میانی می‌توان کلید درست را تشخیص داد. برای کلید درست در یک بیت مشخص از ردهای محاسبه‌شده، شباهتی میان تمایزگر و مقدار آن بیت در ردها وجود دارد؛ درحالی‌که در حالت کلید نادرست مقادیر به‌صورت تصادفی انتخاب می‌شوند.



(شکل-۱۶): علت کار کردن حمله DCA

۲-۵- ویژگی‌های حمله DCA در مقایسه با حملات دیگر

همان‌طورکه گفته شد، حمله DCA شباهت‌هایی با حمله DPA دارد؛ بنابراین به‌منظور توضیح بهتر ویژگی‌ها، روی مقایسه حمله DCA با حمله DPA تمرکز می‌شود.

۱-۲-۵- انجام محاسبات با ابزارهای DPA

با توجه به مراحل انجام حمله DCA و DPA، مشاهده می‌شود که مراحل کلی انجام دو حمله به‌طورتقریبی با هم یکی بوده و فقط در چگونگی یافتن رد با هم متفاوت هستند. در حمله DPA ردها مقدار توان مصرفی هستند؛ درحالی‌که در حمله DCA بیت‌های حافظه به‌عنوان ردها مورد استفاده قرار می‌گیرند؛ بنابراین با توجه به نکات بیان‌شده، می‌توان پس از به‌دست‌آوردن ردها در حمله DCA، از همان روش‌های مورد استفاده در حمله DPA با کمی تغییر (به‌دلیل بیتی‌بودن ردها) به‌منظور محاسبه کلید استفاده کرد.

[DOR: 20.1001.1.24763047.1398.8.2.4.2]

۵-۲-۲- فاقد نوفه

یکی از تفاوت‌های مهم در مقایسه حمله DCA و حمله DPA مقدار نوفه ردهای اندازه‌گیری شده است. با توجه به این‌که در حمله DPA به‌طورمعمول نمونه‌های آنالوگ به‌عنوان رد ثبت می‌شوند، وجود نوفه امری طبیعی است؛ اما در حمله DCA که در حوزه نرم‌افزار انجام می‌شود، بیت‌هایی از حافظه به‌عنوان رد ثبت می‌شوند. با توجه به اینکه بیت‌ها تنها می‌توانند دو مقدار صفر یا یک را داشته باشند، ردهای ثبت‌شده فاقد هرگونه نوفه اندازه‌گیری هستند. نبود نوفه اندازه‌گیری در ردها، فرصت مناسبی را برای اعمال روش‌هایی به‌منظور اجرای خودکار، کاهش اندازه ردها و همچنین افزایش سرعت حمله فراهم کرده است.

۵-۲-۳- قابلیت اجرای خودکار و اجرا بدون نیاز به حضور سامانه

در حملات DPA نیاز به تعداد زیادی رد برای انجام حمله است. با توجه به فیزیکی‌بودن سامانه در حمله DPA، حضور سامانه امری بدیهی است که همین امر موجب محدودشدن حمله می‌شود؛ اما نرم‌افزار، رایانه‌ها، گوشی‌های هوشمند و غیره بسترهای ناامنی برای ذخیره اطلاعات هستند؛ زیرا بدافزارها و تروجان‌ها از طریق اینترنت به‌راحتی گسترش می‌یابند و به‌طور مداوم تعداد زیادی از رایانه‌های شخصی را در مدت کوتاهی آلوده می‌کنند. با توجه به اینکه برای انجام حمله DCA فقط به تعدادی رد نرم‌افزاری نیاز است، می‌توان از کرم‌ها و بدافزارها به‌عنوان ابزاری برای ثبت رد مورد نیاز استفاده کرد؛ بنابراین در حمله DCA نیازی به حضور سامانه نبوده که نسبت به نوع سخت‌افزاری آن یک مزیت است. یکی دیگر از مزیت‌های این امر قابلیت اجرای خودکار حمله است که یک موضوع مهم در حوزه تحلیل و ارزیابی سامانه‌های رمزنگاری است.

۵-۴- روش‌های مقاوم‌سازی در برابر حمله DCA

همراه با پیشرفت‌ها در حوزه روش‌های تحلیل و ارزیابی طرح‌های رمزنگاری جعبهٔ سفید، روش‌های مقاوم‌سازی این طرح‌ها در برابر حملات یادشده نیز بهبود پیدا کرده‌اند. برای مقاوم‌سازی این نوع رمزها از روش‌های حفاظتی مانند مبهم‌کردن روند کنترل، مقاومت در برابر دست‌کاری و نقاب‌گذاری استفاده می‌شود [۲۰]. این روش‌ها را می‌توان به‌صورت زیر تعریف کرد.

مبهم‌کردن جریان کنترل[1]

در این روش، برای هر اجرا از الگوریتم و حمله روی آن مسیر محاسبات به‌منظور ایجاد اغتشاش تصادفی شده و مهاجم را مجبور می‌کند که یک نمودار گره پیچیده را که شامل زمان اجرا تصادفی و عملیات ساختگی است، مهندسی معکوس کند [۲۱]. با توجه به این کار دیگر نمی‌توان انتظار داشت که حمله DCA مطابق آنچه که توضیح داده شده کارساز باشد.

مقاومت در برابر دست‌کاری[2]

در این روش، ویژگی حفظ یک‌پارچگی پیاده‌سازی الگوریتم، برای اطمینان از این‌که کد برنامه و داده‌های فقط خواندنی دست‌کاری نشده باشند، تأمین می‌شود [۲۲].

نقاب‌گذاری

حمله DCA از دسترسی‌ها به نشانی حافظه پیاده‌سازی‌های نرم‌افزاری به‌عنوان رد در حمله استفاده می‌کند. از آنجا که این نشانی‌های حافظه به‌طوراساسی ورودی جدول‌های مختلف را نشان می‌دهند، اطلاعات کافی برای انجام حمله DCA را فراهم می‌کنند [۲۰]. یکی از روش‌های پیشنهادشده برای مقابله با حمله DCA در حوزه جعبهٔ سفید، استفاده از نقاب‌گذاری است که برای محافظت در برابر حملات کانال جانبی استفاده می‌شود. با این حال، با توجه به محیط جعبهٔ سفید حملات جدیدی که قادر به شکستن طرح‌های نقاب‌گذاری باشند، ارائه شده‌اند [۲۳]. نقاب‌گذاری یک متغیر x را به n بخش تقسیم می‌کند، طوری که x با استفاده از d+1 بخش ($n \geq d+1$) قابل بازیابی است و توجه شود که با استفاده از d بخش هیچ اطلاعاتی از x به‌دست نمی‌آید [۲۳]. برای آشنایی با برخی روش‌های نقاب‌گذاری می‌توان به [۲۴، ۲۵، ۲۶، ۲۷، ۲۸] مراجعه کرد. برای مثال در روش ارائه‌شده در [۲۸] قبل از کدگذاری مقادیر میانی در طول تولید جداول مبنا از نقاب‌گذاری استفاده می‌شود؛ یا طرح نقاب‌گذاری ارائه‌شده در [۲۳] ترکیبی از اجزای خطی و غیر خطی برای دست‌یابی به مقاومت در برابر حملات محاسباتی و جبری است.

۶- نتیجه‌گیری

ایجاد امنیت در محیط‌های با دسترسی جعبهٔ سفید یکی از مهم‌ترین چالش‌های موجود در استفاده از این گونه سامانه‌ها است. اگرچه طرح‌های رمزنگاری جعبهٔ سفید زیادی تاکنون

[1] Control Flow Obfuscation
[2] Tamper Resistance

[DOR: 20.1001.1.24763047.1398.8.2.4.2]

ارائه شده است، اما این طرح‌ها هنوز به وضعیت مطلوبی نرسیده‌اند؛ به‌طوری‌که بیشتر طرح‌ها در مدت زمان کوتاهی شکسته شده‌اند. بنابراین شناخت حملات موجود در این زمینه یکی از الزامات مهم در پیاده‌سازی یک طرح رمزنگاری جعبهٔ سفید جدید و یا شکستن یک طرح موجود است. به‌منظور مقابله با طرح‌های رمزنگاری جعبهٔ سفید، حملات متفاوتی ارائه شده است. برخی از حملات از نقطه‌ضعف موجود در یک طرح استفاده کرده و برخی دیگر یک روش کلی برای حمله به این سامانه‌ها ارائه داده‌اند.

یکی از حملات ارائه‌شده در حوزه رمزنگاری جعبهٔ سفید، حمله DCA است. با توجه به شباهت‌های موجود میان این حمله و حمله شناخته شدهٔ DPA، در این مقاله به بررسی این حمله پرداخته شد. مطالعات زیادی در زمینه DPA ارائه شده است که می‌توان با کمی تغییر و به اصطلاح شخصی‌سازی، در حوزه حمله DCA از آن‌ها استفاده کرد؛ بنابراین نخستین گام در این زمینه شناخت قواعد، اصول و همچنین نحوه انجام عملی حمله DCA است. این حمله و نیز حمله DPA به‌طورکلی به دو قسمت ثبت رد و تحلیل و ارزیابی ردها به‌منظور محاسبه کلید تقسیم می‌شوند. با بررسی‌های انجام‌شده مشاهده شد که قسمت تحلیل و ارزیابی ردها در دو حملهٔ یادشده تا حد زیادی باهم مشابه هستند. به‌طورکلی می‌توان گفت، عملکرد دو حمله باهم مشابه است و ممکن است فقط در برخی مراحل میانی تغییری ایجاد شود. با توجه به نقش مهم حمله DPA در ارزیابی سخت‌افزاری الگوریتم‌های رمزنگاری می‌توان نتیجه گرفت که حمله DCA قابلیت استفاده را به‌عنوان یک ابزار مهم جهت بررسی امنیت طرح‌های رمزنگاری جعبهٔ سفید دارد.

۷- مراجع

[1] Ch. Stanley, E. Philip, H. Johnson, and P.C. Van Oorschot, "White-box cryptography and an AES implementation," *In International Workshop on Selected Areas in Cryptography*, pp. 250-270. Springer, Berlin, Heidelberg, 2002.

[2] Ch. Stanley, E. Philip, H. Johnson, and P.C. Van Oorschot, "A white-box DES implementation for DRM applications," *In ACM Workshop on Digital Rights Management*, pp. 1-15. Springer, Berlin, Heidelberg, 2002.

[3] Y. Xiao, and L. Xuejia "A secure implementation of white-box AES," *In 2009 2nd International Conference on Computer Science and its Applications*, pp. 1-6. IEEE, 2009.

[4] A.Biryukov, Ch. Bouillaguet, and D. Khovratovich, "Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key," *In International Conference on the Theory and Application of Cryptology and Information Security*, pp. 63-84, Springer, Berlin, Heidelberg, 2014.

[5] CHES Challenge 2017.

[6] CHES Challenge 2019.

[7] W. Joppe, B. Charles Hubain, W. Michiels, and Ph. Teuwen, "Differential computation analysis: Hiding your white-box designs is not enough," *In International Conference on Cryptographic Hardware and Embedded Systems*, pp. 215-236, Springer, Berlin, Heidelberg, 2016.

[8] W. Joppe, B. Bock, E. Alpirez, Ch. Brzuska, Ch. Hubain, W. Michiels, C. Mune, E. Gonzalez, P. Teuwen, and A. Treff, "White-box cryptography: don't forget about grey-box attacks," *Journal of Cryptology 32*, no. 4, pp.1095-1143, 2019.

[9] http://whiteboxcrypto.com/challenges.php.

[10] J.-B. B´edrune, Hack.lu 2009 reverse challenge 1. Online, 2009. http://2009.hack.lu/index.php/ .

[11] F. Marceau, F. Perigaud, and A. Tillequin. Challenge SSTIC 2012. Online, 2012. http://communaute. sstic.org/ChallengeSSTIC2012.

[12] H. Soleimany, and M. Sadeghi, "Approaches and challenges of the white-box model of the block cipher schemes implementation," *Biannual Journal Monadi for Cyberspace Security (AFTA)* 7, no. 1, pp.3-20, 2019.

[13] A. Bogdanov, I. Takanori, and E. Tischhauser, "Towards practical whitebox cryptography: optimizing efficiency and space hardness," *In International Conference on the Theory and Application of Cryptology and Information Security*, pp. 126-158. Springer, Berlin, Heidelberg, 2016.

[14] J.Bringer, H. Chabanne, and E. Dottax, "White Box Cryptography: Another Attempt," IACR Cryptology ePrint Archive 2006, no. 2006 (2006): 468.

[15] M.Karroumi, "Protecting white-box AES with dual ciphers," *In International Conference on Information Security and Cryptology*, pp. 278-291, Springer, Berlin, Heidelberg, 2010.

[16] E. L. Hamilton, and D. Neumann, "Clarifying obfuscation: improving the security of white-box DES," *In International Conference on Information Technology: Coding and Computing (ITCC'05),* vol. 1, pp. 679-684, IEEE, 2005.

[DOR: 20.1001.1.24763047.1398.8.2.4.2]

[17] A.Bogdanov, and I.Takanori, "White-box cryptography revisited: Space-hard ciphers," *In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 1058-1069. 2015.

[18] A. Fouque, P. Karpman, P. Kirchner, and B. Minaud, "Efficient and provable white-box primitives," *In International Conference on the Theory and Application of Cryptology and Information Security*, pp. 159- 188. Springer, Berlin, Heidelberg, 2016.

[19] Ch. Jihoon, K. Young Choi, I. Dinur, O. Dunkelman, N. Keller, D. Moon, and A. Veidberg, "WEM: a new family of white-box block
ciphers based on the even-mansour construction," *In Cryptographers' Track at the RSA Conference*, pp. 293-308, Springer, Cham, 2017.

[20] B. Subhadeep, A. Bogdanov, T. Isobe, and M. Jepsen, "Analysis of software countermeasures for whitebox encryption," *IACR Transactions on Symmetric Cryptology*, pp. 307-328, 2017.

[21] ARXAN. TransformIT: Software-based Key Protection, 2014.

[22] Microsemi, WhiteboxCRYPTO: Cryptographic key hiding with tunable security and performance, 2015.

[23] O. Seker, T. Eisenbarth and M. Liskiewicz.: A White-Box Masking Scheme Resisting Computational and Algebraic Attacks. Cryptology ePrint Archive, Report 2020/443, 2020.

[24] I, Yuval, A. Sahai, and D.Wagner, "Private circuits: Securing hardware against probing attacks," In *Annual International Cryptology Conference*, pp. 463-481, Springer, Berlin, Heidelberg, 2003.

[25] R. Matthieu, and E.Prouff, "Provably secure higher-order masking of AES," *In International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 413-427. Springer, Berlin, Heidelberg, 2010.

[26] N. Svetla, V. Rijmen, and M. Schläffer, "Secure hardware implementation of non-linear functions in the presence of glitches," *In International Conference on Information Security and Cryptology*, pp. 218-234, Springer, Berlin, Heidelberg, 2008.

[27] R. Thomas, and E. Prouff, "Higher-order glitch free implementation of the aes using secure multi-party computation protocols," *Journal of Cryptographic Engineering 2*, no. 2, pp.111-127, 2012.

[28] L.Seungkwang, T. Kim, and Y. Kang, "A masked white-box cryptographic implementation for protecting against differential computation analysis," *IEEE Transactions on Information Forensics and Security* 13, no. 10, pp. 2602-2615, 2018.

[29] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *In Annual International Cryptology Conference*, pp. 388-397, Springer, Berlin, Heidelberg, 1999.

جواد علیزاده، دکترای خود را در رشته رمزنگاری از دانشگاه جامع امام حسین (ع)، در سال ۱۳۹۵ تحت راهنمایی آقای دکتر محمدرضا عارف و مشاوره آقای دکتر نصور باقری اخذ کردند. در حال حاضر ایشان استادیار دانشکده فناوری اطلاعات و ارتباطات دانشگاه جامع امام حسین(ع) هستند. علاقه‌مندی‌های پژوهشی وی شامل طراحی و تحلیل پروتکل‌های رمزنگاری و الگوریتم‌های رمزنگاری متقارن است.

محسن صدیقی، کارشناسی مهندسی برق گرایش الکترونیک را در سال ۱۳۹۶ از دانشگاه گیلان و کارشناسی ارشد مهندسی مخابرات امن و رمزنگاری را در سال ۱۳۹۸ از دانشگاه جامع امام حسین (ع) اخذ کرد. در حال حاضر ایشان به‌عنوان پژوهش‌گر در حوزه پیاده‌سازی امن و حملات کانال جانبی در دانشگاه جامع امام حسین(ع) مشغول به فعالیت است.

هادی سلیمانی، کارشناسی را در رشته مخابرات از دانشگاه علم و صنعت ایران در سال ۱۳۸۷ اخذ کرد و کارشناسی ارشد را با گرایش مخابرات رمز در سال ۱۳۸۹ در دانشگاه جامع امام حسین(ع) به پایان رساند؛ سپس مدرک دکترای خود را از دانشکده علوم رایانه دانشگاه آلتوی فنلاند در سال ۱۳۹۴ اخذ کرد. وی همچنین طی یک دوره کوتاه مدت پسادکترا در گروه رمزنگاری دانشگاه DTU دانمارک درخصوص تحلیل و طراحی رمزهای قالبی نوین مشغول به پژوهش شد. نامبرده هم‌اکنون، ضمن همکاری با پژوهشکده‌ها و مراکز پژوهشی مختلف در حوزه رمزنگاری و امنیت اطلاعات، به‌عنوان استادیار گروه امنیت شبکه و رمزنگاری پژوهشکده فضای مجازی دانشگاه شهید بهشتی مشغول به کار است. زمینه‌های پژوهشی مورد علاقه ایشان تحلیل و طراحی اولیه‌های رمزنگاری متقارن و همچنین پیاده‌سازی امن است.

[DOR: 20.1001.1.24763047.1398.8.2.4.2]