نشریه علمی پدافند غیرعامل

سال دوازدهم، شماره ۲، تابستان ۱۴۰۰، (پیاپی ۴۶): صص ۵۲-۴۱

علمی- ترویجی

مقدمه‌ای بر مقاوم‌سازی الگوریتم‌های رمزنگاری در برابر حملات کانال جانبی با استفاده از روش پیاده‌سازی آستانه‌ای

جواد علیزاده[1*]، حمید قنبری[2]

تاریخ دریافت: ۱۳۹۹/۰۹/۰۲
تاریخ پذیرش: ۱۴۰۰/۰۳/۰۲

چکیده

برای تأمین امنیت اطلاعات و ارتباطات لازم است تا یک الگوریتم رمزنگاری به‌صورت نرم‌افزاری یا سخت‌افزاری پیاده‌سازی و به‌کار گرفته شود. در سال ۱۹۹۶ کوچر، حملاتی روی سامانه‌های رمزنگاری مطرح کرد که در آن‌ها از نشت اطلاعات مربوط به پیاده‌سازی الگوریتم‌های رمز استفاده می‌شد. از این نوع حملات که با نام حملات کانال جانبی شناخته شده‌اند، می‌توان به حمله تحلیل توان اشاره کرد. برای مقابله با حملات کانال جانبی، روش‌های مقاوم‌سازی مانند نقاب‌گذاری یا نهان کردن ارائه شد ولی بعدها نشان داده شد که این نوع روش‌ها در حضور گلیچ اثربخشی لازم را ندارند. جهت برطرف کردن این مشکل و مقاوم‌سازی سامانه‌های رمزنگاری در برابر حملات کانال جانبی، حتی در حضور گلیچ، روش پیاده‌سازی آستانه‌ای در سال ۲۰۰۶ توسط نیکووا و همکاران ارائه شد. این‌روش کاربردی از سه مبحث رمزنگاری آستانه‌ای، سهم نهان و محاسبه چندجانبه تشکیل شده است. در واقع خود این‌روش هم نوعی مقاوم‌سازی به روش نقاب‌گذاری است که شرط‌هایی اضافه برای تأمین امنیت در حضور گلیچ دارد. در سال‌های اخیر موسسه استانداردسازی NIST فعالیت‌هایی در حوزه پیاده‌سازی آستانه‌ای شروع کرده است که یکی از اهداف آن‌ها، تدوین یک استاندارد در این زمینه است. این موضوع باعث شده است تا در حال حاضر رمزنگاران موضوع پیاده‌سازی آستانه‌ای را به‌عنوان یک موضوع مهم در نظر بگیرند. در این مقاله روش رمزنگاری آستانه‌ای به‌عنوان یک روش جهت مقاوم‌سازی سامانه‌های رمزنگاری در برابر حملات کانال جانبی توصیف و به نکات برتری و چالش‌های آن در مقایسه با روش‌های مقاوم‌سازی قبلی مانند نقاب‌گذاری اشاره می‌شود.

کلیدواژه‌ها: حمله کانال جانبی، حمله تحلیل توان، پیاده‌سازی آستانه‌ای

۱- استادیار دانشگاه جامع امام حسین (ع)، jaalizadeh@ihu.ac.ir - نویسنده مسئول

۲- دانشجوی کارشناسی ارشد دانشگاه جامع امام حسین (ع)

۱- مقدمه

در حال حاضر اهمیت اطلاعات و تأمین امنیت آن بر کسی پوشیده نیست. برای روشن شدن این موضوع می‌توان به گسترش اینترنت اشیا[1] در بسیاری از ابعاد زندگی انسان در سال‌های اخیر اشاره کرد. در استانداردهای امنیتی، امنیت اطلاعات به‌عنوان تأمین محرمانگی[2]، جامعیت[3] و دسترس‌پذیری[4] اطلاعات تعریف شده و از الگوریتم‌های رمزنگاری به‌عنوان یک ابزار مهم جهت برقراری امنیت اطلاعات و ارتباطات یاد می‌شود [۱]. برای استفاده عملی از یک الگوریتم رمزنگاری، لازم است تا این الگوریتم با استفاده از یک روش نرم‌افزاری یا سخت‌افزاری پیاده‌سازی شود. الگوریتم رمزنگاری همراه با پیاده‌سازی آن، یک سامانه یا سامانه رمزنگاری نامیده می‌شود.

همراه با تلاش‌های طراحان جهت طراحی و پیاده‌سازی سامانه‌های رمزنگاری امن، مهاجمان و تحلیلگران نیز دانش‌های خود جهت حمله به این سامانه‌ها و خنثی کردن اثر محافظتی آن‌ها را توسعه می‌دهند. حملات روی یک سامانه رمزنگاری را می‌توان در دو بخش اصلی حملات مبتنی بر ضعف‌های الگوریتم‌های رمزنگاری و حملات مبتنی بر ضعف‌های پیاده‌سازی این الگوریتم‌ها در نظر گرفت. همچنین با توجه به اینکه یک سامانه رمزنگاری می‌تواند از نوع نرم‌افزاری یا سخت‌افزاری باشد، حملات مربوط به پیاده‌سازی سامانه نیز از دو جهت حملات مربوط به پیاده‌سازی نرم‌افزاری و حملات مربوط به پیاده‌سازی سخت‌افزاری قابل تأمل است. تمرکز این مقاله روی حملات سخت‌افزاری و روش‌های مقاوم‌سازی در برابر این حملات است. از حملات مبتنی بر پیاده‌سازی سخت‌افزاری، تحت عنوان حملات کانال جانبی نیز نام برده می‌شود. به‌عنوان مثال از این نوع حملات می‌توان به حمله کانال جانبی تحلیل توان اشاره کرد که در سال ۱۹۹۶ توسط کوچر و همکاران پیشنهاد شد [۲]. از روش‌های مقاوم‌سازی سامانه‌های رمز در برابر این نوع حمله نیز می‌توان روش‌های نقاب‌گذاری[5] [۳]، نهان‌سازی[6] [۴] و پیاده‌سازی آستانه‌ای[7] [۵] را نام برد.

هدف از این مقاله توصیف روش مقاوم‌سازی با عنوان پیاده‌سازی آستانه‌ای است. برای این منظور ابتدا روش‌های مقاوم‌سازی در برابر حملات کانال جانبی به‌صورت کلی توصیف شده و سپس روش پیاده‌سازی آستانه‌ای با جزئیات بیشتر مورد بررسی قرار می‌گیرد. همچنین برای آشنایی بیشتر با روش پیاده‌سازی آستانه‌ای، کلیات این‌روش روی رمز سبک‌وزن KATAN بیان می‌شود. ادامه مطالب مقاله به‌صورت زیر سازمان‌دهی شده است.

در بخش دوم، حملات کانال جانبی روی سامانه‌های رمزنگاری معرفی می‌شوند. در بخش سوم انواع روش‌های مقاوم‌سازی در برابر حملات کانال جانبی مورد مطالعه قرار گرفته و نقاط قوت و ضعف آن‌ها بیان می‌شود. در این بخش همچنین اثر گلیچ در برابر روش‌های مقاوم‌سازی مورد بررسی قرار می‌گیرد. در بخش چهارم روی روش پیاده‌سازی آستانه‌ای، به‌عنوان یک روش مقاوم‌سازی در برابر حملات کانال جانبی، حتی با حضور گلیچ، تمرکز شده و جزئیات این‌روش بیان می‌شود. در این بخش به برخی نقاط قوت و ضعف روش پیاده‌سازی آستانه‌ای در مقایسه با روش‌های مقاوم‌سازی دیگر اشاره می‌شود. در بخش پنجم جهت آشنایی بیشتر با روش پیاده‌سازی آستانه‌ای، استفاده از این‌روش برای مقاوم‌سازی رمز KATAN در برابر حملات کانال جانبی توضیح داده می‌شود. در نهایت جمع‌بندی و نتیجه‌گیری مقاله در بخش ششم ارائه می‌شود.

۲- حملات کانال جانبی روی سامانه‌های رمزنگاری

همان‌طور که در بخش اول توضیح داده شد، حملات روی یک سامانه رمزنگاری را می‌توان در دو بخش حملات در سطح الگوریتم و حملات در سطح پیاده‌سازی (نرم‌افزاری یا سخت‌افزاری) الگوریتم رمزنگاری در نظر گرفت. در این بخش روی حملات مبتنی بر پیاده‌سازی سخت‌افزاری بحث می‌شود که در ادامه تحت نام حملات کانال جانبی آورده خواهد شد. حملات کانال جانبی در حوزه امنیت فیزیکی یک سامانه رمزنگاری قرار می‌گیرند. منظور از امنیت فیزیکی یک سامانه، تأمین امنیت آن در برابر تهدیدات سخت‌افزاری است که می‌توانند به‌صورت تهاجمی، نیمه تهاجمی یا غیرتهاجمی صورت گیرند. از تهدیدات سخت‌افزاری شناخته‌شده می‌توان به حمله القا خطا[8]، حمله زمان[9]، حمله الکترومغناطیس[10] و حمله تحلیل توان[11] اشاره کرد.

در حمله القا خطا به‌صورت تعمدی یک خطا در روند اجرای عملیات رمزنگاری ایجاد می‌کنند تا بتوانند از طریق این خطا به کلید رمز دسترسی پیدا کنند [۶- ۸]. این حمله برای بار اول توسط آرلات مطرح شد [۸] و می‌تواند به دو صورت تهاجمی یا غیرتهاجمی اعمال شود. در نوع تهاجمی مهاجم با تراشه ارتباط فیزیکی پیدا کرده و با تغییر ولتاژ یا اعمال جریان از خارج، کار

[1] Internet of things
[2] Confidentiality
[3] Integrity
[4] Availibility
[5] Masking
[6] Hiding
[7] Threshold implementation
[8] Fault Attack
[9] Timing Attack
[10] Electromagnetic Attack
[11] Power Analysis

القای خطا را انجام می‌دهد. اما در روش غیرتهاجمی مهاجم هیچ‌گونه تماس فیزیکی با دستگاه رمزکننده ندارد و یک منبع خارجی با تولید برخی پدیده‌های فیزیکی، مانند تابش یون‌های غلیظ و تداخل الکترومغناطیسی، باعث جریان‌های ساختگی در داخل تراشه هدف می‌شود [۹].

در حمله زمان با استفاده از زمان اجرای الگوریتم به ازای ورودی‌های مختلف به کلید دسترسی پیدا می‌کنند. اولین بار این نوع حمله توسط کوچر [۱۰] مطرح گردید اما بعدها توسط دم برای شکستن رمز RSA به‌کار گرفته شد [۱۱].

در حمله الکترومغناطیس با قرار دادن سیم‌پیچ‌هایی در کنار تراشه رمزنگار به توالی اجرای عملیات پی می‌برند. این حمله برای بار اول توسط سمید و کواتیزر [۱۲] مطرح گردید. سپس توسط گاندولفی و همکاران در [۱۳] گسترش داده شد.

حمله تحلیل توان که یکی از حملات معروف و پرکاربرد در حوزه حملات کانال جانبی است در سال ۱۹۹۰ توسط کوچر [۲] ارائه شد. در این‌روش از وابستگی توان مصرفی تراشه رمزکننده به اطلاعات در حال پردازش و نوع عملیات در حال انجام، برای بازیابی کلید استفاده می‌شود. این دسته از حملات به سه روش حمله تحلیل توان ساده[1]، حمله تحلیل توان تفاضلی[2] و حمله تحلیل توان همبستگی[3] صورت می‌گیرد. برای مقابله با حمله تحلیل توان روش‌های متفاوتی ارائه شده است که در همه آن‌ها سعی بر از بین بردن ارتباط میان توان مصرفی تراشه رمزنگار با اطلاعات در حال پردازش و عملیات در حال انجام است.

با ارائه حملات تحلیل توان روش‌هایی نیز برای مقابله با این حملات ارائه شد. از آنجایی که در حمله تحلیل توان از رابطه بین توان مصرفی تراشه با داده در حال پردازش یا عملیات در حال اجرا برای کشف اطلاعات مخفی مانند کلید استفاده می‌شود، این‌روش‌های مقاوم‌سازی باید به‌گونه‌ای باشد که این رابطه بیان‌شده را از بین ببرد. در برخی مقالات مانند [۴] از دو روش نهان‌سازی[4] و نقاب‌گذاری[5] به‌عنوان روش‌هایی برای مقابله با این نوع حمله نام برده شده است که در بخش بعد توصیف می‌شوند. تفاوت اصلی این دو روش در نحوه از بین بردن وابستگی بین توان مصرفی تراشه و کلید است. در نهان‌سازی با از بین بردن ویژگی توان مصرفی تراشه به این امر دست می‌یابند اما در نقاب‌گذاری با تصادفی‌سازی مقادیر میانی به این هدف می‌رسند.

با طرح شدن گلیچ در سال ۲۰۰۵ [۱۴] و تأثیر آن روی سامانه‌های رمزنگاری سخت‌افزاری و نیز غیر مؤثر بودن آن در مورد سامانه‌های رمزنگاری نرم‌افزاری، موجب شد تا موضوع مقاوم‌سازی در برابر حملات کانال جانبی در شاخه نرم‌افزار و سخت‌افزار از هم جدا شوند. وجود گلیچ سبب می‌شود حتی با نقاب‌گذاری طرح اطلاعات مخفی نشت کند. برای حل این مساله در سال ۲۰۰۶ یک روش مقاوم‌سازی در برابر حملات تحلیل توان به نام پیاده‌سازی آستانه‌ای توسط نیکووا و همکاران [۵] مطرح شد. این‌روش مبتنی بر استفاده از سهم نهان[6] [۱۵, ۱۶]، رمزنگاری آستانه‌ای [۱۷] و پروتکل‌های محاسباتی چندجانبه[7] [۱۸] است. در بخش مربوط به توصیف روش پیاده‌سازی آستانه‌ای در این مقاله، به شرح کامل این‌روش پرداخته شده است. اما در ادامه به برخی فعالیت‌های قبلی و جدید در این مورد اشاره می‌شود که نشان از اهمیت و جایگاه این فناوری در حوزه امن‌سازی سامانه‌های رمزنگاری دارد.

روش‌های پیاده‌سازی آستانه‌ای متفاوتی برای رمزهای مختلف ارائه شده است که در اینجا به چند مورد مهم پیاده‌سازی آستانه‌ای اشاره می‌شود. این‌روش‌ها به مدارهای اضافی برای تولید بیت‌های تصادفی و عملیات فشرده‌سازی نیاز دارند. در سالیان اخیر هدف محققین ارائه راه‌کارهایی به‌منظور کاهش این نوع سربارهای سخت‌افزاری بوده است [۱۹]. برای نخستین بار پیاده‌سازی آستانه‌ای رمز AES توسط مرادی و همکاران [۲۰] ارائه شد. مشکل این‌روش پیاده‌سازی اشغال مساحت بالایی از تراشه بود. بیلگین [۲۱, ۲۲] در سال ۲۰۱۴ یک روش پیاده‌سازی آستانه‌ای جدیدی از AES ارائه داد که در آن مساحت و تعداد بیت‌های تصادفی نسبت به روش مرادی و همکاران کمتر بود. ایده پیاده‌سازی آستانه‌ای به‌صورت یک مدل ریاضیاتی بود که یک روش سخت‌افزاری جامع برای آن توسط بیلگین و همکاران به اسم CMS[8] ارائه شد [۲۳]. این‌روش این قابلیت را دارد که بتوان الگوریتم‌های رمزنگاری متفاوت را در مرتبه‌های بالاتر در برابر حملات امن نگه داشت. این طرح بر روی جعبه جانشانی AES توسط دی‌کنادی[9] [۲۴] پیاده‌سازی و نشان داده شد که مساحت پیاده‌سازی با هزینه افزایش تعداد بیت تصادفی کاهش می‌یابد. البته طرح CMS در سال ۲۰۱۹ توسط مرادی [۲۵] در مرتبه‌های بالا شکسته شد و می‌توان گفت که امنیت سطح بالا در مقابل حملات کانال جانبی را ندارد. موارد ذکرشده در مورد ارائه طرح‌های پیاده‌سازی آستانه‌ای، مساحت اشغالی

[1] Simple Power Analysis (SPA)
[2] Differential Power Analysis (DPA)
[3] Correlation Power Analysis (CPA)
[4] Hiding
[5] Masking

[6] Secret sharing
[7] multi-party computation protocols
[8] Consolidating masking scheme
[9] De Cnudde

زیادی بر سطح تراشه داشت. به همین دلیل محققان سعی کردند تا طرح‌هایی برای پیاده‌سازی رمزهای استاندارد مانند رمز AES با منابع مصرفی پایین‌تر پیدا کنند. در سال ۲۰۱۷ طرح جدیدی توسط ینو[1] [۲۶] ارائه شد که مساحت پیاده‌سازی کمتری نسبت به طرح دی کنادی داشت. پس از آن مقاله‌ای توسط دی کنادی [۲۷] ارائه شد که از معماری جدیدی برای پیاده‌سازی جعبه جانشانی AES استفاده می‌کرد. بعد از این طرح نیز طرح جدیدی مبتنی بر روش DOM[2] در سال ۲۰۱۸ ارائه شد که موجب کاهش مساحت می‌شد. اما این طرح تأخیر زمانی فراوانی در جعبه جانشانی AES داشت [۲۸]. در ادامه تحقیقات روی روش پیاده‌سازی آستانه‌ای مقاله‌ای ارائه شد که از روشی به نام عوض کردن گاردها[3] استفاده می‌کرد [۲۹]. هدف این‌روش آن است که نیازی به استفاده از بیت‌های تصادفی در هر مرحله از عملیات رمزنگاری نباشد. مرادی [۳۰] این‌روش را روی جعبه جانشانی AES پیاده‌سازی کرد. اگر چه تعداد بیت‌های تصادفی مورد نیاز در این طرح صفر است اما مساحت اشغالی و تأخیر زمانی زیادی در این معماری وجود دارد. در سال ۲۰۱۸ روشی توسط گراس[4] [۳۱] معرفی شد که هدف آن کاهش تأخیر زمانی طرح DOM و از بین بردن تصادم[5] سهم‌های یک مقدار در عملیات غیرخطی است. این‌روش تأخیر زمانی و تعداد بیت‌های تصادفی را به صفر کاهش می‌دهد. با این مساحت و تعداد سهم‌های خروجی بالایی نیاز دارد. بالاخره اینکه در سال ۲۰۲۰ یک روش با تعداد بیت تصادفی صفر و با تأخیر زمانی کمتر نسبت به روش ارائه‌شده در [۳۰] و با تعداد سهم‌های ورودی کمتر ارائه شد [۳۲]. این‌روش بهبودی از روش ارائه‌شده در [۳۳] بود که تأخیر زمانی در آن برابر ۲ سیکل است.

در کنار فعالیت‌های دانشگاهی و علمی و مقالات ارائه‌شده در حوزه پیاده‌سازی آستانه‌ای، شرکت‌ها و موسسه‌های صنعتی و استانداردسازی نیز فعالیت‌هایی در این زمینه داشته‌اند. از جمله این فعالیت‌ها می‌توان به اقدامات موسسه ملی استاندارد و فناوری آمریکا[6] (NIST) اشاره کرد که در سال‌های ۲۰۱۸ و ۲۰۱۹ مساله رمزنگاری آستانه‌ای را مورد توجه قرار داد که پیاده‌سازی آستانه‌ای نیز یکی از محورهای آن است [۳۴, ۳۵]. در سال ۲۰۲۰ (بهار ۱۳۹۹)، این موسسه گزارش جدیدی در خصوص پیاده‌سازی آستانه‌ای الگوریتم‌های رمزنگاری منتشر کرد که نشان‌دهنده تمایل آن برای استانداردسازی روش پیاده‌سازی آستانه‌ای الگوریتم‌های رمزنگاری است (گزارش NIST IR-8214A). هم‌زمان در کارگاهی که توسط دانشگاه KUL بلژیک و همکاری NIST برگزار شد، مسئولان این موسسه اعلام کردند که در حال بررسی ساز و کارهای اجرائی استانداردسازی پیاده‌سازی آستانه‌ای هستند. اتفاقی که در صورت وقوع، سبب خواهد شد تعداد زیادی از محصولات از این‌روش برای امن‌سازی سامانه‌های رمزنگاری استفاده کنند. بر اساس، اعلام محققین NIST، استانداردسازی مورد هدف برای پیاده‌سازی آستانه‌ای الگوریتم‌های رمزنگاری می‌تواند اشکال مختلفی باشد. استانداردسازی مورد نظر NIST لزوماً به شکل استانداردهای FIPS[7] نخواهد بود بلکه می‌تواند از نوع SP باشد. لازم به ذکر است در حالی که FIPS مجموعه‌ای است که به‌طور خاص به‌دستورالعمل‌ها و استانداردها مربوط می‌شود، SP ماحصل تعامل بخش‌های صنعتی، دولتی و دانشگاهی است که در تلاش مشترک برای انتشار دستورالعمل‌ها، یافته‌ها و توصیه‌ها با یکدیگر همکاری هدفمند داشته‌اند.

۳- روش‌های مقاوم‌سازی در برابر حملات کانال جانبی

با توجه به تهدیدات و حملات سخت‌افزاری که در بخش ۲ توصیف شدند، لازم است تا طراحان و پیاده‌سازان سامانه‌های رمزنگاری از امنیت سامانه‌های خود در برابر این حملات مطمئن شوند. برای مقابله با حملات کانال جانبی روش‌های شناخته‌شده‌ای مطرح شده‌اند که از آن‌ها می‌توان به روش‌های نقاب‌گذاری، نهان‌سازی و پیاده‌سازی آستانه‌ای اشاره کرد. مهم‌ترین کار در این‌روش‌ها، از بین بردن ارتباط میان توان مصرفی تراشه رمزنگار با اطلاعات در حال پردازش یا عملیات در حال انجام است. در ادامه این بخش دو روش نهان‌سازی و نقاب‌گذاری به‌صورت مختصر معرفی می‌شود. همچنین توضیح داده می‌شود که فرآیند گلیچ روی مدارات سخت‌افزاری چگونه می‌تواند باعث حذف اثر محافظتی روش نقاب‌گذاری شود.

۳-۱- نهان‌سازی

پیشنهادهایی که در رابطه با نهان‌سازی اطلاعات برای از بین بردن ارتباط میان توان مصرفی، اطلاعات در حال پردازش و عملیات در حال اجرا است را می‌توان در دو دسته در نظر گرفت. در دسته اول توان مصرفی مربوط به عملیات اجرایی الگوریتم رمزنگاری در طول فرآیند اجرای الگوریتم، تصادفی‌سازی می‌شود. این نوع روش بر بعد زمانی مصرف توان تأثیر می‌گذارد. در دسته دوم پیاده‌سازی الگوریتم را طوری انجام می‌دهند که در هر

[1] Ueno
[2] Domain Oriented Masking
[3] Changing of the Guards
[4] Gros
[5] Collision
[6] NIST
[7] Federal Information Processing Standards

چرخه اجرا الگوریتم[1]، توان مصرفی متفاوت باشد یا مصرف توان تمام عملیات با هر مقدار داده یکسان باشد. این نوع روش نیز بر بعد دامنه مصرف توان تأثیر می‌گذارد.

در بعد زمانی، اجرای عملیات‌هایی در لحظات مختلف زمانی (در فرآیند الگوریتم رمزنگاری)، باعث تصادفی شدن مصرف توان می‌شود. این کار با اجرای تصادفی یک سری عملیات تصنعی یا به هم ریختن ترتیب اجرای عملیات الگوریتم رمزنگاری انجام می‌شود. ایده اصلی که در این‌روش وجود دارد این است که طراح و پیاده‌ساز پس از آنکه الگوریتم رمزنگاری را پیاده‌سازی نمود؛ در نهایت با جاسازی و اجرای تصادفی یک سری عملیات تصنعی در قبل، حین و پس از اجرای الگوریتم رمزنگاری باعث تصادفی‌سازی مصرف توان می‌شود. نکته قابل توجه در این‌روش نیاز به استفاده از مولدهای اعداد تصادفی و ایجاد تصمیم‌گیری بر اساس آن‌ها است [۴]. در این‌روش، برای آن‌که کارآیی سامانه‌های رمزنگاری کاهش نیابد، می‌توان به‌جای استفاده از عملیات تصنعی و اضافی، ترتیب اجرای دستورالعمل‌های الگوریتم رمزنگاری را به‌طور تصادفی تغییر داد.

دومین دسته از پیشنهادها در زمینه نهان‌سازی، روی سطح دامنه مصرف توان تأثیرگذار هستند. در این پیشنهادها روش‌هایی برای تغییر مستقیم ویژگی‌های مصرف توان عملیات محاسباتی الگوریتم بیان می‌شود. هدف تمامی این‌روش‌ها کاهش نسبت سیگنال به نویز در اجرای الگوریتم است که در حالت کلی این نسبت می‌تواند با کاهش سیگنال و یا افزایش نویز کاهش یابد[۴]. ساده‌ترین روش برای افزایش نویز عملیات، انجام چندین عملیات مستقل به‌صورت موازی است. برای مثال حمله تحلیل توان به تک بیت در یک معماری ۱۲۸بیتی رمز AES سخت‌تر از حمله به یک معماری ۳۲ بیتی آن است. در روند کاهش سیگنال، هدف ساخت ابزاری است که در آن، اجرای تمام عملیات نیاز به مقدار یکسانی از توان برای هر نوع داده ورودی داشته باشد. در تأکید بر انجام این کار می‌توان این نکته را ذکر نمود که در حملات DPA و CPA از تعداد مشاهدات بالایی از نمونه توان استفاده می‌شود. بنابراین به‌وسیله یک محاسبه آماری ساده می‌توان تغییرات توان را در نقاط مختلفی از زمان به‌دست آورد. لذا این بخش از کار که بتوان مداری ساخت که در آن تمام عملیات به‌طور یکسانی توان مصرف کنند کار به نسبت دشواری است. روش‌های مختلفی برای نهان‌سازی در سطح دامنه وجود دارد. این‌روش‌ها معمولاً می‌توانند یک سامانه رمزنگاری را در برابر حملات SPA مقاوم کنند. با این وجود این‌روش‌ها برای مقابله با حملات DPA کافی نیستند. با توجه به این امر و نیز موضوعاتی مانند تأخیر ایجادشده در استفاده از روش نهان‌سازی [۳۶]، روش‌های مقاوم‌سازی دیگری مانند نقاب‌گذاری مورد توجه قرار می‌گیرد.

۳-۲- نقاب‌گذاری

در این‌روش مقادیر میانی الگوریتم رمزنگاری که توسط آن در حال پردازش هستند به‌صورت تصادفی تغییر می‌کنند. از مزیت‌های این‌روش می‌توان به اجرای آن در سطح الگوریتم اشاره کرد که می‌تواند باعث عدم تغییر در توان مصرفی تراشه شود.

برای نقاب‌گذاری یک مقدار میانی الگوریتم مانند v از یک مقدار تصادفی مانند m استفاده شده و مقدار جدید v_m به‌صورت $v_m = v * m$ پیاده‌سازی می‌شود. عمل $*$ یک عمل محاسباتی است که با توجه به عملیات‌هایی که در الگوریتم رمزنگاری استفاده می‌شود، مشخص می‌شود. با توجه به نوع عملیات استفاده‌شده در الگوریتم‌های رمزنگاری متقارن، می‌توان گفت عمل $*$ بیشتر می‌تواند یک تابع بولی، عمل XOR، جمع پیمانه‌ای یا ضرب پیمانه‌ای باشد. توجه شود که در استفاده از نقاب‌گذاری حتماً می‌بایست تمام مقادیر میانی نقاب‌گذاری شوند. حتی باید این مساله تضمین گردد که اگر یک مقدار میانی از مقادیر میانی قبل حاصل می‌شود، این مقدار نیز نقاب‌گذاری شود. برای مثال اگر یک مقدار میانی، خود حاصل XOR دو مقدار میانی قبلی باشد، باید هم این دو مقدار میانی و هم مقدار میانی حاصل نقاب‌گذاری شوند. به همین دلیل معمولاً از چند نوع نقاب[2] برای نقاب‌گذاری استفاده می‌شود. باید این موضوع را نیز مد نظر قرار داد که با افزایش تعداد نقاب کارآیی سامانه رمزنگاری کاهش می‌یابد. به این معنی که سرعت محاسباتی سامانه رمزنگاری کاهش پیدا کرده و هزینه و منابع لازم برای پیاده‌سازی آن افزایش می‌یابد.

روش مقاوم‌سازی نقاب‌گذاری را می‌توان در دو دسته نقاب‌گذاری بولی و نقاب‌گذاری محاسباتی در نظر گرفت. در نقاب‌گذاری محاسباتی معمولاً مقادیر میانی توسط جمع و ضرب پیمانه‌ای نقاب‌گذاری می‌شوند. در مقابل، در نقاب‌گذاری بولی عمل نقاب‌گذاری توسط تابع بولی XOR صورت می‌گیرد. در برخی موارد نیاز است تا از هر دو نوع نقاب‌گذاری برای مقاوم‌سازی سامانه رمز استفاده شود. از آنجا که این کار نیاز به عملیات اضافی دارد می‌توان انتظار داشت که کارآیی سامانه رمز کاهش پیدا کند [۳۷, ۳۸]. علاوه بر این در یک الگوریتم رمزنگاری، توابع خطی و غیرخطی مورد استفاده قرار می‌گیرد. اگر تابع f نسبت به عملگر $*$ خطی باشد، می‌توان رابطه زیر را نوشت:

[1] منظور در هر Clock

[2] Mask

$$f(x * y) = f(x) * f(y)$$

حال اگر به‌جای عملگر $*$ عملگر $\oplus$ در نظر گرفته شود، می‌توان نوشت:

$$f(x \oplus y) = f(x) \oplus f(y)$$

به‌عبارت دیگر در یک طرح نقاب‌گذاری بولی، عملیات خطی، مانند عملگر XOR، نقاب m را به‌گونه‌ای تغییر می‌دهد که به‌راحتی قابل محاسبه باشد. به این معنی که نقاب‌گذاری یک تابع خطی به‌صورت بولی راحت است.

در مقابل راحتی نقاب‌گذاری برای توابع خطی، نقاب گذاری توابع غیرخطی نسبتاً سخت است. برای مثال جعبه جانشانی رمز قالبی AES را در نظر بگیرید که یک تابع غیرخطی است و نقاب‌گذاری آن با روش نقاب‌گذاری بولی نمی‌تواند کار ساده‌ای باشد.[1] در [۳۹] یک روش کارآمد برای تغییر بین نقاب‌گذاری بولی و ضربی (محاسباتی) ارائه شده است.

یک روش نقاب‌گذاری را می‌توان به این صورت انجام داد که هر متغیر میانی به چند سهم تقسیم شود. به‌طوری که با داشتن اطلاع از هر کدام از سهم‌ها به تنهایی، نتوان اطلاعاتی از مقادیر میانی به‌دست آورد. این کار (تجزیه هر متغیر میانی به چندین سهم) ارتباط میان توان مصرفی لحظه‌ای و مقادیر میانی را از بین می‌برد. درنتیجه می‌تواند باعث مقاوم‌سازی در برابر حملات کانال جانبی مرتبه اول شود [۴۰]. به‌عنوان مثال اگر مقدار میانی نقاب‌گذاری شده v_m برابر $v_m = (v \oplus m)$ باشد، پس مقدار میانی v می‌تواند توسط v_m و m محاسبه گردد. یعنی مقدار میانی v می‌تواند توسط دو سهم (v_m, m) نمایش داده شود. به‌عنوان مثال دیگر، نقاب‌گذاری بولی متغیر x به دو سهم را به‌صورت $x = x_1 \oplus x_2$ در نظر بگیرید. جدول (۱) نشتی اطلاعات مربوط به وزن همینگ این دو متغیر را به‌صورت $L(x) = HW(x_1, x_2)$ نشان می‌دهد. ستون چهارم این جدول میانگین اطلاعات نشتی به ازای xهای متفاوت را نشان می‌دهد. همان‌طور که مشاهده می‌شود، در این مورد نقاب‌گذاری به ازای xهای متفاوت هیچ‌گونه اطلاعاتی راجع به وزن همینگ نشان نمی‌دهد.

روش نقاب‌گذاری که در بالا توصیف شد، می‌تواند باعث مقاوم‌سازی در برابر حملات کانال جانبی مرتبه اول شود. برای مقاوم‌سازی یک سامانه رمز در برابر حملات کانال جانبی مرتبه بالا، لازم است تا مرتبه نقاب‌گذاری نیز بالا برده شود. یعنی لازم است تا تعداد سهم‌های بیشتری برای نقاب‌گذاری استفاده شود [۴۰]. روشن است که این کار باعث کاهش بیشتر کارآیی سامانه رمزنگاری می‌شود. از نواقص دیگر مقاوم‌سازی با روش نقاب‌گذاری، عدم کارآیی آن در حضور گلیچ[2] است که این موضوع در بخش بعد توضیح داده می‌شود.

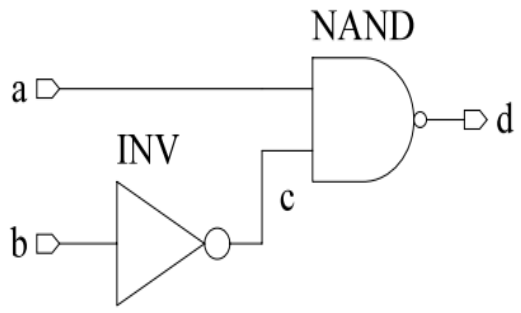
جدول (۱): رفتار نشتی اطلاعات یک بیت در قبال تقسیم آن به دو سهم.

X	X_1	X_2	L(x)	mean(L(x))
۰	۰	۰	۰	۱
	۱	۱	۲	
۱	۰	۱	۱	۱
	۱	۰	۱	

۳-۳- گلیچ

دروازه‌های استاندارد مکمل اکسید فلز نیمه رسانا[3] (CMOS) سبب رخداد گلیچ می‌شوند که این امر می‌تواند باعث وابستگی مقادیر نقاب‌گذاری نشده به توان مصرفی شود. در مدارات ترکیبی CMOS چندین سلول منطقی[4] به یکدیگر متصل شده‌اند. خروجی هر سلول به‌عنوان ورودی سلول بعدی مورد استفاده قرار می‌گیرد و همیشه ورودی هر مدار به‌صورت آنی از خروجی مدار دیگری دریافت نمی‌شود. علت این امر وجود تأخیرهای متفاوت در عملیات و مسیرها است. بنابراین تغییر خروجی یک مدار بر اثر تأخیر ورودی آن مدتی طول می‌کشد.

به‌عنوان مثال شکل (۱) را در نظر بگیرید. در این شکل a به‌صورت مستقیم به ورودی گیت NAND وارد می‌شود. b نیز به ورودی گیت معکوس‌کننده می‌رود. خروجی گیت معکوس‌کننده را c و خروجی کل مدار را d در نظر بگیرید.



شکل (۱): مدار ترکیبی کوچک [۴]

[1] در جعبه‌هایی که به روش معکوس ضربی در میدان ساخته می‌شوند بهتر است نقاب‌گذاری به روش ضرب پیمانه‌ای انجام شود زیرا

$$f(x \times m) = (x \times m)^{-1} = f(x) \times f(m)$$

[2] Glitch
[3] Complementary Metal Oxide Semicondutor
[4] Logic Cell

جدول (۲)، جدول درستی مدار ترکیبی شکل (۱) را نشان می‌دهد. با فرض صفر بودن تأخیر سیم‌ها، فرض کنید ورودی‌های a و b از ۰ به ۱ تغییر کنند. تأخیر ناشی از گیت‌های NAND و معکوس‌کننده را به ترتیب به‌صورت $t_{prop,NAND}$ و $t_{prop,INV}$ در نظر بگیرید. همان‌طور که در شکل (۲) مشاهده می‌شود یک گلیچ در خروجی d اتفاق می‌افتد. این خروجی در این حالت باید مقدار ۱ باشد اما به‌دلیل دیرتر رسیدن سیگنال ورودی c به گیت NAND نسبت به سیگنال a برای مدتی موقت مقدار d به ۰ تغییر می‌کند که مدت این حالت وابسته به $t_{prop,INV}$ است.

جدول (۲): جدول درستی مدار ترکیبی شکل (۱)

a	b	c=INV(b)	d=NAND(a,c)
۰	۰	۱	۱
۰	۱	۰	۱
۱	۰	۱	۰
۱	۱	۰	۱

به‌عنوان مثالی دیگر فرض کنید رابطه (۱) با استفاده از دو سهم نقاب‌گذاری شده باشد.

$$S(y,z) = yz \qquad (۱)$$

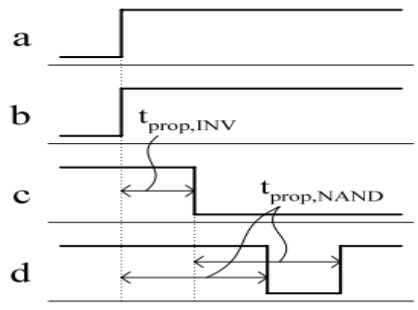
رابطه (۲) نشان می‌دهد که ورودی‌ها و خود تابع S چگونه نقاب‌گذاری می‌شوند.

مقدار y با دو سهم y_1 و y_2 و مقدار z نیز با دو مقدار z_1 و z_2 در نظر گرفته شده است.

$$S(y_1,z_1) = y_1z_1 \qquad (۲)$$
$$S'(y_1,y_2,z_1,z_2) = y_1z_2 \oplus y_2z_1 \oplus y_2z_2$$

حال فرض کنید مقدار y_2 به‌علتی با تأخیر برسد و گلیچ در این مدار رخ دهد. پس بنا بر معادلات زیر مقدار میانی z آشکار می‌گردد.

$$\Delta S' = \Delta y_2 z_1 \oplus \Delta y_2 z_2 = \Delta y_2 (z_1 \oplus z_2)$$
$$= (z_1 \oplus z_2) = z$$



شکل (۲): نمودار زمانی سیگنال‌های ورودی و خروجی و گلیچ [۴]

با توجه به توضیحاتی که ارائه شد، روشن است که در صورت رخداد گلیچ، تأثیر حفاظتی روش نقاب‌گذاری از بین می‌رود. روش پیاده‌سازی آستانه‌ای که یک روش مقاوم‌سازی آستانه‌ای است تلاش می‌کند تا از یک سامانه رمز در برابر حملات کانال جانبی، حتی در حضور گلیچ نیز مقاومت کند. در بخش بعد روش پیاده‌سازی آستانه‌ای تشریح می‌شود.

۴- پیاده‌سازی آستانه‌ای

در بخش‌های قبل این مقاله دو روش نهان‌سازی و نقاب‌گذاری برای مقابله با حملات کانال جانبی معرفی و نقاط قوت و ضعف آن‌ها بیان شد. مهم‌ترین ضعف روش نقاب‌گذاری، بدون اثر بودن آن در حضور گلیچ است که وجود آن در مدارات سخت‌افزاری غیرقابل اجتناب است. طرح شدن گلیچ در سال ۲۰۰۵ [۱۴] و تأثیر آن روی سامانه‌های رمزنگاری سخت‌افزاری و نیز غیر مؤثر بودن آن در مورد سامانه‌های رمزنگاری نرم‌افزاری، موجب شد تا موضوع مقاوم‌سازی در برابر حملات کانال جانبی در شاخه نرم‌افزار و سخت‌افزار از هم جدا شوند. به‌عبارت دیگر برای مقاوم‌سازی سامانه‌های رمزنگاری نرم‌افزاری در برابر حملات کانال جانبی می‌توان همچنان از روش‌های نهان‌سازی و نقاب‌گذاری استفاده کرد. در حالی که در مورد سامانه‌های رمزنگاری سخت‌افزاری می‌بایست از روش‌های بهبود یافته دیگری استفاده کرد. یکی از این‌روش‌ها، پیاده‌سازی آستانه‌ای است که در سال ۲۰۰۶ توسط نیکووا و همکاران [۵] ارائه شد و از آن موقع تاکنون، به‌خصوص در دو سال اخیر، مورد توجه جامعه رمزنگاری قرار گرفته است [۴۱]. این‌روش مبتنی بر استفاده از سهم نهان[1] [۱۵, ۱۶]، رمزنگاری آستانه‌ای [۱۷] و پروتکل‌های محاسباتی چندجانبه[2] [۱۸] است. روش پیاده‌سازی آستانه‌ای که در واقع یک نوع بهبودیافته و پیشرفته از روش نقاب‌گذاری است، می‌تواند ضعف‌های مربوط به روش نقاب‌گذاری در برابر گلیچ را پوشش دهد. علاوه بر این، روش ذکرشده این مزیت را دارد که می‌توان با استفاده از آن پیاده‌سازی‌های امن، سبک‌وزن و همراه با تأخیر کم را برای الگوریتم‌های رمزنگاری متصور بود. برتری دیگر روش پیاده‌سازی آستانه‌ای در مقایسه با روش‌های قبلی این است که روش‌های قبلی بعد از پایان هر مرحله غیرخطی نیاز به یک مقدار تصادفی جدید دارند اما در روش پیاده‌سازی آستانه‌ای تنها در شروع کار نیاز به مقدار تصادفی است [۵].

در یک پیاده‌سازی آستانه‌ای از یک الگوریتم رمزنگاری می‌بایست سه شرط صحت[3]، عدم کامل بودن[1] و یکنواختی[2]

[1] Secret sharing
[2] multi-party computation protocols
[3] Correctness

تأمین شود که این ویژگی‌ها در ادامه این مقاله توصیف می‌شوند. توجه شود در مقاوم‌سازی به روش نقاب‌گذاری نیز لازم بود تا دو شرط صحت و یکنواختی تأمین شوند اما بحثی روی ویژگی عدم کامل بودن صورت نمی‌گرفت.

۴-۱- نمادگذاری برای پیاده‌سازی آستانه‌ای

برای پیاده‌سازی آستانه‌ای یک تابع مانند f از F^n به F^m، با ضابطه $f(X)=A$، هر متغیر x به s_x سهم x_i تقسیم می‌شود که در آن $i \in \{1,2,\dots,s_x\}$ است. برای انجام این امر به‌صورت تصادفی یک توزیع یکنواخت روی سهم‌های $x_1, x_2, \dots, x_{s_x-1}$ در نظر گرفته می‌شود. سپس x_{s_x} به‌گونه‌ای انتخاب می‌شود که XOR این سهم‌ها برابر خود x شود. بردار سهم‌های صحیح یعنی $\mathbf{x}=(x_1, x_2, \dots, x_{s_x})$ و عملیات تقسیم به سهم‌ها، سهم‌بندی یا نقاب‌گذاری مقدار سهم‌بندی نشده x نامیده می‌شود. همچنین از عبارت $s_x - sharing$ برای نشان دادن تعداد سهم‌ها استفاده می‌شود.

برای تمام مقادیر x با احتمال $\Pr(X=x)>0$ مقدار Sh(x) نشان‌دهنده بردار سهم صحیح $\mathbf{x}$ برای تمام مقادیر x است که به‌صورت رابطه (۳) قابل بیان است.

$$Sh(x)=\{\boldsymbol{x}\in F^{ns_x} \mid x_1 \oplus x_2 \oplus \dots \oplus x_{s_x}=x\} \quad (۳)$$

قبل از وارد شدن به توصیف ویژگی‌های مربوط به پیاده‌سازی آستانه‌ای، لازم است تا نقاب‌گذاری یکنواخت تشریح شود.

۴-۲- نقاب‌گذاری یکنواخت

نقاب‌گذاری $\mathbf{X}$ یکنواخت است اگر و تنها اگر مقدار ثابت p به نحوی وجود داشته باشد که برای تمام مقادیر x بتوان نوشت [۴۲]:

اگر $\boldsymbol{x}\in Sh(x)$ آنگاه $\Pr(\boldsymbol{X}=\boldsymbol{x}|X=x)=p$

در غیر این صورت

$$\Pr(\boldsymbol{X}=\boldsymbol{x}|X=x)=0$$

و

$$\sum_{\boldsymbol{x}\in Sh(x)} \Pr(\boldsymbol{X}=\boldsymbol{x})=\Pr(X=x)$$

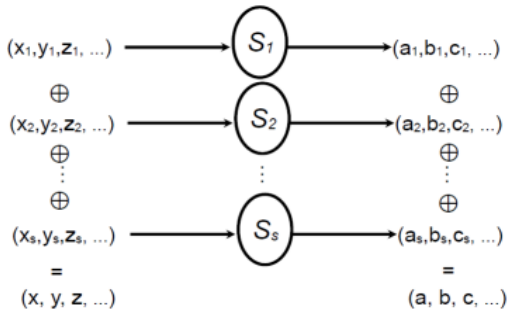
به‌عبارت دیگر یک نقاب‌گذاری یکنواخت است اگر برای هر مقدار x، مقادیر بردار نقاب آن با احتمال یکسان قابل حدس باشند. از این‌رو داشتن اطلاعات درباره s_x-1 از سهم‌ها هیچ نوع اطلاعی درباره x آشکار نمی‌کند.

۴-۳- صحت

در این بخش ویژگی صحت پیاده‌سازی آستانه‌ای تعریف می‌شود. منظور از ویژگی ذکرشده این است که سهم‌های تعریف‌شده برای یک مقدار میانی، به‌صورت صحیح مقدار میانی مورد نظر را نتیجه دهند [۵]. یعنی برای هر $a\in F_2^m$ و $\boldsymbol{A}=\boldsymbol{f}(\boldsymbol{X})$ که در آن برای تمام مقادیر $\boldsymbol{x}$، $\sum_i a_i=\sum_i f_i(\boldsymbol{x})$ برقرار است، در خروجی رابطه $\sum_i x_i = x$ حاصل شود ($x\in F_2^n$).

مشخص است که تعداد سهم‌های خروجی s_a برابر تعداد توابع تشکیل‌دهنده s_f است. از این به بعد برای اشاره به تعداد سهم ورودی از s_x و برای اشاره به تعداد سهم خروجی از s_f استفاده می‌شود. در صورتی که تعداد سهم ورودی و خروجی برابر باشند از s برای نشان دادن تعداد سهم استفاده می‌شود. برای درک بهتر توضیحات به شکل (۳) توجه کنید.

ویژگی‌هایی که تاکنون توضیح داده شدند، یعنی ویژگی‌های صحت و یکنواختی، ویژگی‌هایی هستند که در مورد استفاده از نقاب‌گذاری نیز لازم بودند. ویژگی مهم پیاده‌سازی آستانه‌ای که آن را از روش نقاب‌گذاری متمایز می‌کند و در واقع مهم‌ترین دلیل برای اثربخش بودن آن در حضور گلیچ است، ویژگی عدم کامل بودن است که در بخش بعد توضیح داده می‌شود.



شکل (۳): ویژگی صحت در پیاده‌سازی آستانه‌ای

۴-۴- عدم کامل بودن

برای اینکه مهاجم با پروب‌گذاری[3] در طرح نقاب‌گذاری شده نتواند اطلاعاتی در مورد اطلاعات محرمانه به‌دست آورد، می‌بایست سهم‌های ورودی یکنواخت باشد و ویژگی عدم کامل بودن را داشته باشد. منظور از ویژگی عدم کامل بودن این است که هر تابع سهم باید مستقل از حداقل یکی از سهم‌های هر ورودی باشد [۵]. به‌عنوان مثال اگر z=f(x,y) باشد و ورودی‌های x و y با n سهم نشان داده شود، برای تأمین این ویژگی، تابع‌های

[1] Non-completeness
[2] Uniformity
[3] Prob

f_1 تا f_n به ترتیب با خروجی‌های z_1 تا z_n به‌صورت زیر تعریف می‌شود.

$$z_1 = f_1(x_2, x_3, \dots, x_n, y_2, y_3, .., y_n)$$
$$z_2 = f_2(x_1, x_3, \dots, x_n, y_1, y_3, .., y_n)$$
$$\dots$$
$$z_n = f_n(x_1, x_2, \dots, x_{n-1}, y_1, y_2, .., y_{n-1})$$

در [۴۲] یک طرح سهمی شده با d تابع مؤلفه‌ای که ویژگی‌های صحت و عدم کامل بودن را تأمین کند، یک طرح پیاده‌سازی آستانه‌ای مرتبه d نامیده شده و امنیت آن مطابق قضیه ۱ ثابت‌شده است.

قضیه ۱. اگر نقاب‌گذاری ورودی X برای تابع سهمی شده f یک نقاب‌گذاری یکنواخت باشد و f یک پیاده‌سازی آستانه‌ای مرتبه dام باشد، آنگاه تحلیل توان مصرفی مرتبه dام یک مدار که تابع f را اجرا می‌کند، هیچ نوع اطلاعاتی در مورد ورودی بدون نقاب x، حتی در حضور گلیچ آشکار نمی‌کند.

اثبات: [۴۲]

یک سؤال مهم که در مورد پیاده‌سازی آستانه‌ای مرتبه d مطرح می‌شود این است که حداقل چه تعداد سهم برای این کار لازم است. پاسخ این سؤال مطابق قضیه ۲ در [۴۲] ارائه شده است.

قضیه ۲. حداقل تعداد سهم‌های ورودی مورد نیاز برای پیاده‌سازی آستانه‌ای مرتبه d (s_{in}) با $d+1$ برابر است. یعنی

$$s_{in} \geq d + 1$$

اثبات: [۴۲]

با توجه به قضیه ۲، انتخاب $s_{in} = d + 1$، باعث می‌شود با افزایش تعداد قالب‌ها در نمایش تابعی یک تابع، افزایش قابل ملاحظه‌ای در تعداد سهم‌های خروجی روی دهد. زیرا هر تابع مؤلفه‌ای تنها از یک سهم هر متغیر استفاده می‌کند. این چالش و راه‌حل‌هایی برای آن در [۴۲] مورد بررسی قرار گرفته‌است.

۴-۵- یکنواختی

منظور از یکنواختی این است که تمام حالت‌های ممکن برای سهم‌بندی یک متغیر، احتمال رخداد یکسانی داشته باشند. با توجه به این ویژگی اگر متغیر x به s سهم تقسیم شود، با داشتن هر ترکیب از s-1 تا سهم نتوان مقدار x را به‌دست آورد. البته در برخی پیاده‌سازی‌ها چون خروجی یک بخش از الگوریتم به‌عنوان ورودی بخش دیگر آن مورد استفاده قرار می‌گیرد، لازم است تا ویژگی یکنواختی در خروجی نیز رعایت شود تا ورودی بخش بعدی نیز خاصیت یکنواختی را داشته باشد. در توابع خطی اگر ورودی یکنواخت باشد، خروجی نیز یکنواخت است اما در توابع غیرخطی مشکل یکنواخت کردن خروجی نیز مطرح می‌شود. برای توضیح بیشتر فرض کنید.

$$(X,Y) \in F_2^2, F_2 \ni A = f(X,Y) = XY$$

باشد. تابع‌های مؤلفه‌ای f_i مربوط به پیاده‌سازی آستانه‌ای مرتبه اول f را به‌صورت زیر در نظر بگیرید.

$$\begin{aligned} A_1 &= f_1(X_2, X_3, Y_2, Y_3) \\ &= X_2Y_2 \oplus X_2Y_3 \oplus X_3Y_2 \\ A_2 &= f_2(X_1, X_3, Y_1, Y_3) \\ &= X_3Y_3 \oplus X_1Y_3 \oplus X_3Y_1 \\ A_3 &= f_3(X_1, X_2, Y_1, Y_2) \\ &= X_1Y_1 \oplus X_1Y_2 \oplus X_2Y_1 \end{aligned} \quad (۴)$$

اگر نقاب‌گذاری ورودی یکنواخت باشد، نقاب‌گذاری A دارای توزیع جدول (۳) خواهد بود. برای تأمین یکنواختی نقاب‌گذاری خروجی تابع تعریف‌شده **A** (روابط (۴)) نیاز به ۱۶ مقدار غیرصفر مساوی در جدول هست. منظور از ۱۶ مقدار (از صفر تا ۱۵)، مقادیری است که با چهار بیت می‌توان نشان داد. قضیه ۲ بیان می‌کند که از این تابع هیچ اطلاعاتی نشت نمی‌کند. اما اگر **A** به‌عنوان ورودی مدار دوم استفاده شود، قضیه ۲ برای آن صادق نیست زیرا ورودی آن دیگر یکنواخت نمی‌باشد.(۰،۰)

جدول (۳): تعداد دفعاتی که a_1 و a_2 و a_3 به ازای ورودی (x,y) مشخص اتفاق می‌افتد

	a_1, a_2, a_3							
(X,Y)	۰۰۰	۰۱۱	۱۰۱	۱۱۰	۰۰۱	۰۱۰	۱۰۰	۱۱۱
(۰،۰)	۷	۳	۳	۳	۰	۰	۰	۰
(۰،۱)	۷	۳	۳	۳	۰	۰	۰	۰
(۱،۰)	۷	۳	۳	۳	۰	۰	۰	۰
(۱،۱)	۰	۰	۰	۰	۵	۵	۵	۱

در زمان‌های زیادی این اتفاق می‌افتد که خروجی مدار اول باید به‌عنوان ورودی مدار دوم مورد استفاده قرار گیرد. در این موارد نیاز است تا خروجی مدار (تابع **A**) نیز یکنواخت شود.

روش‌های متفاوتی برای این کار مانند نقاب‌گذاری مجدد[1] [۲۰]، بالا بردن تعداد سهم‌های ورودی [۵]، کاهش تعداد سهم‌های خروجی [۴۲]، استفاده از قالب صحیح[2] [۵] و موارد دیگر مطرح شده است.

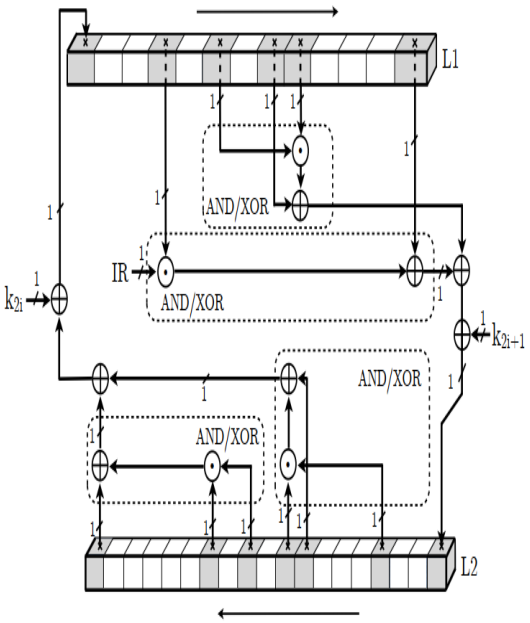
۵- پیاده‌سازی آستانه‌ای رمز KATAN

برای آشنایی بیشتر با بحث پیاده‌سازی آستانه‌ای و ویژگی‌های آن، در این بخش یک نوع پیاده‌سازی آستانه‌ای رمز قالبی KATAN که در [۴۳] معرفی شده است، آورده می‌شود.

رمز KATAN یک خانواده از رمزهای قالبی است. این رمز به نحوی طراحی شده است تا جهت پیاده‌سازی سخت‌افزاری بهینه باشد. این رمز در انواع مختلف ۳۲ بیتی، ۴۸ بیتی و ۶۴ بیتی ارائه شده است. کلید این رمز ۸۰ بیت است و از این نظر انواع مختلف آن دارای امنیت یکسان می‌باشند. این مرز ۲۵۴ دور دارد و تابع دور آن در همه انواع ۳۲،۴۸ و ۶۴ بیتی آن شبیه یکدیگر هستند و از تعدادی دروازه AND و XOR تشکیل شده است.

در شکل (۴) یک دور از رمز KATAN ۳۲ بیتی آورده شده است. هر بلوک در شکل (۴) نمایانگر یک بیت است. متن ورودی ۳۲ بیتی به دو بخش ۱۳ و ۱۹ بیت تقسیم می‌شود و به ترتیب در ثبات‌های L_1 و L_2 قرار می‌گیرد. در هر دور بیت‌های زیادی مورد استفاده قرار می‌گیرند تا اولین بیت ثبات‌های L_1 و L_2 تغییر کنند. همچنین ثبات‌های L_1 و L_2 به اندازه یک بیت به ترتیب شیفت به راست و چپ دارند. بیتی که با IR نمایش داده شده است، آخرین بیت از یک شمارنده دور است که تصمیم می‌گیرد بیت چهارم از ثبات L_1 برای به‌روزرسانی دور مورد استفاده قرار بگیرد یا نه. برای دورهای اول تا چهلم $(i \leq 40)$ k_{2i} و k_{2i+1}، به ترتیب 2iامین و (2i+1)امین بیت از ۸۰ بیت کلید رمز هستند. برای باقی دورها این دو از کلید اصلی توسط یک LFSR تولید می‌شوند. برای اطلاعات بیشتر در مورد این رمز می‌توان به [۴۳] رجوع کرد.

همان‌طور که در شکل (۴) مشاهده می‌شود، بخش غیرخطی این رمز شامل (۴) دروازه AND/XOR است. این تبدیل‌ها را می‌توان با تابع $f(X,Y,Z) = X \oplus YZ$ نمایش داد. در [۵] نشان داده شده است که یک پیاده‌سازی آستانه‌ای یکسان با ۳ سهم برای یک دریچه AND وجود ندارد. به‌عبارت دیگر یک پیاده‌سازی آستانه‌ای یکسان برای تابع $f(X,Y,Z) = X \oplus YZ$ با سه سهم وجود دارد. بنابراین همیشه دریچه AND را با XOR برای پیاده‌سازی این رمز، هم‌گروه می‌کنیم.



شکل (۴): یک دور از رمز KATAN 32 بیتی [۴۰]

به جز بلوکی که IR را دریافت می‌کند، از سهم‌های مشخص‌شده مطابق رابطه (۵) استفاده می‌شود.

$$A_1 = X_2 \oplus (Y_2Z_2 \oplus Y_2Z_3 \oplus Y_3Z_2)$$
$$A_2 = X_3 \oplus (Y_3Z_3 \oplus Y_3Z_1 \oplus Y_1Z_3) \qquad (۵)$$
$$A_3 = X_1 \oplus (Y_1Z_1 \oplus Y_1Z_2 \oplus Y_2Z_1)$$

با توجه به توضیحات ارائه‌شده در [۴۲] تعداد سهم‌های ورودی برابر $s_{in} = t \times d + 1 = 3$ و تعداد سهم خروجی برابر $s_{out} = \binom{s_{in}}{t} = 3$ است. برای پیاده‌سازی آستانه‌ای بلوک دریافت‌کننده IR از رابطه (۶) استفاده می‌شود. زیرا شمارنده دور و در نتیجه IR سهم‌بندی نمی‌شود.

$$A_i = X_i + IR \times Y_i \quad where\ i \leq s_i \qquad (۶)$$

توجه شود که در [۴۲] رمز KATAN ۳۲ بیتی با استفاده از سه سهم پیاده‌سازی شده است. در نتیجه به‌صورت هم‌زمان سه حالت در حال اجرا است. زمان مورد نیاز برای اجرای این رمز در صورت پیاده‌سازی آستانه‌ای برابر حالت پیاده‌سازی بدون حفاظت است. تنها تفاوت این دو حالت مقدار منابع سخت‌افزاری مصرفی است. مجموع هزینه سخت‌افزاری پیاده‌سازی آستانه‌ای رمز KATAN ۳۲ بیتی برابر ۱۷۲۰ دریچه[3] است [۴۲].

[1] Re-masking
[2] Using correction term
[3] 1720 GE

۶- نتیجه‌گیری

هدف از ارائه این مقاله، آشنایی با روش پیاده‌سازی آستانه‌ای به‌عنوان یک روش مهم مقاوم‌سازی سامانه‌های رمز در برابر حملات کانال جانبی بود. برای این منظور ابتدا به‌صورت مختصر حملات کانال جانبی و روش‌های مقاوم‌سازی شناخته‌شده معرفی شدند و نقاط قوت و ضعف روش‌های مقاوم‌سازی بیان شدند. مهم‌ترین ضعف این روش‌ها، عدم کارآیی آن‌ها در حضور پدیده گلیچ است. در ادامه روی روش مقاوم‌سازی پیاده‌سازی آستانه‌ای که می‌تواند ضعف‌های روش‌های قبلی را پوشش دهد و به‌خصوص در حضور گلیچ نیز مقاومت ایجاد کند، تمرکز شد. روش پیاده‌سازی آستانه‌ای، نسخه ارتقا یافته روش نقاب‌گذاری است که به‌منظور به‌کارگیری در سخت‌افزارها مناسب‌سازی شده است. این روش مبتنی بر همان مفاهیم تسهیم راز و محاسبات چندبخشی (MPC[1]) بنا شده است و چهار ویژگی کلی دارد که عبارت‌اند از: یکنواختی نقاب‌گذاری، صحت، عدم کامل بودن و یکنواختی خروجی توابع پردازشگر. توجه شود که دو ویژگی اول میان تمام طرح‌های نقاب‌گذاری مشترک هستند، اما ویژگی‌های سوم و چهارم تنها مختص روش پیاده‌سازی آستانه‌ای هستند.

مزیت‌های روش پیاده‌سازی آستانه‌ای نسبت به سایر روش‌های پیشنهادی را می‌توان در سه حوزه خلاصه کرد [۴۴]. نخست آن‌که برخلاف سایر راه‌کارهایی که ارائه شده‌اند، امنیت اثبات‌پذیر ایجاد می‌کند. دوم آن‌که قابل اعمال به طیف وسیعی از پلت‌فرم‌ها، از جمله مدارات شبه CMOS[2] است که در برخی از طرح‌های پیشنهادی دیگر، با مشکل مواجه بودند. مزیت سوم آن‌که در مقایسه با سایر روش‌ها، کم‌ترین میزان منابع[3] را به طراح و پیاده‌ساز تحمیل می‌کند. در انتها ذکر این نکته لازم است که با توجه به اهمیت سامانه‌های رمزنگاری امن، توسعه حملات کانال جانبی و فعالیت‌هایی که در سال‌های اخیر NIST در حوزه پیاده‌سازی آستانه‌ای شروع کرده است، می‌توان پیاده‌سازی آستانه‌ای را یک حوزه مهم تحقیقاتی برای زمان کنونی جامعه رمزنگاری در نظر گرفت.

۷- مراجع

[1] W. Cheng, Y. Zhou, and L. Sauvage, “Differential fault analysis on Midori,” in International Conference on Information and Communications Security, Springer, pp. 307-317, 2016.

[2] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in Annual international cryptology conference, Springer, pp. 388-397, 1999.

[3] E. Prouff and M. Rivain, “Masking against side-channel attacks: A formal security proof,” in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, pp. 142-159, 2013.

[4] S. Mangard, E. Oswald, and T. Popp, “Power analysis attacks: Revealing the secrets of smart cards,” Springer Science & Business Media, 2008.

[5] S. Nikova, C. Rechberger, and V. Rijmen, “Threshold implementations against side-channel attacks and glitches,” in International conference on information and communications security, Springer, pp. 529-545, 2006.

[6] D. Boneh, R. A. DeMillo, and R. J. Lipton, “On the importance of checking cryptographic protocols for faults,” in International conference on the theory and applications of cryptographic techniques, Springer, pp. 37-51, 1997.

[7] G.-F. Piret, “Block ciphers: security proofs, cryptanalysis, design, and fault attacks,” Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2005.

[8] J. Arlat, “Validation de la sûreté de fonctionnement par injection de fautes: méthode, mise en oeuvre, application,” Toulouse, INPT, 1990.

[9] S. A. T. Nezhad, “Keeloq block cipher power analysis,” Master, Shahid Sattari, 1393.

[10] P. C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” in Annual International Cryptology Conference, Springer, pp. 104-113, 1996.

[11] A. Z. Torbati, “Practical implementation of combined power-error analysis attack against AES cryptography system on PIC microcontroller,” Master, Shahid Sattari, 1392.

[12] P. Chodowiec and K. Gaj, “Very compact FPGA implementation of the AES algorithm,” in International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 319-333, 2003.

[13] K. Gandolfi, C. Mourtel, and F. Olivier, “Electromagnetic analysis: Concrete results,” in International workshop on cryptographic hardware and embedded systems, Springer, pp. 251-261, 2001.

[14] S. Mangard, T. Popp, and B. M. Gammel, “Side-channel leakage of masked CMOS gates,” in Cryptographers’ Track at the RSA Conference, Springer, pp. 351-365, 2005.

[15] G. R. Blakley, “Safeguarding cryptographic keys,” in 1979 International Workshop on Managing Requirements Knowledge (MARK), IEEE, pp. 313-318, 1979.

[16] A. Shamir, “How to share a secret,” Communications of the ACM, vol. 22, no. 11, pp. 612-613, 1979.

[17] Y. Desmedt, “Some recent research aspects of threshold cryptography,” in International Workshop on Information Security, Springer, pp. 158-173, 1997.

[18] A. C. Yao, “Protocols for secure computations,” in 23rd annual symposium on foundations of computer science (sfcs 1982), IEEE, pp. 160-164, 1982.

[1] Multi-Party Computation
[2] CMOS-like
[3] Resources

Transactions on Cryptographic Hardware and Embedded Systems, pp. 1-21, 2018.

[32] P. Sasdrich, B. Bilgin, M. Hutter, and M. E. Marson, “Low-latency hardware masking with application to aes,” IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 300-326, 2020.

[33] A. J. Leiserson, M. E. Marson, and M. A. Wachs, “Gate-level masking under a path-based leakage metric,” in International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 580-597, 2014.

[34] T. De Cnudde, B. Bilgin, O. Reparaz, and S. Nikova, “Higher-order glitch resistant implementation of the PRESENT S-box,” in International Conference on Cryptography and Information Security in the Balkans, Springer, pp. 75-93, 2014.

[35] L. T. Brandão, N. Mouha, and A. Vassilev, “Threshold Schemes for Cryptographic Primitives: Challenges and Opportunities in Standardization and Validation of Threshold Cryptography,” National Institute of Standards and Technology, 2018.

[36] A. Moradi and T. Schneider, “Side-channel analysis protection and low-latency in action,” in International Conference on the Theory and Application of Cryptology and Information Security, Springer, pp. 517-547, 2016.

[37] J.-S. Coron and L. Goubin, “On boolean and arithmetic masking against differential power analysis,” in International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 231-237, 2000.

[38] L. Goubin, “A sound method for switching between boolean and arithmetic masking,” in International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 3-15, 2001.

[39] M.-L. Akkar and C. Giraud, “An implementation of DES and AES, secure against some attacks,” in International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 309-318, 2001.

[40] T. De Cnudde, B. Bilgin, O. Reparaz, V. Nikov, and S. Nikova, “Higher-order threshold implementation of the AES S-box,” in International conference on smart card research and advanced applications, Springer, pp. 259-272, 2015.

[41] A. Aghaie, A. Moradi, S. Rasoolzadeh, A. R. Shahmirzadi, F. Schellenberg, and T. Schneider, “Impeccable circuits,” IEEE Transactions on Computers, vol. 69, no. 3, pp. 361-376, 2019.

[42] B. Bilgin, “Threshold implementations: as countermeasure against higher-order differential power analysis,” 2015.

[43] C. De Canniere, O. Dunkelman, and M. Knežević, “KATAN and KTANTAN—a family of small and efficient hardware-oriented block ciphers,” in International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 272-288, 2009.

[44] E. Prouff and T. Roche, “Higher-order glitches free implementation of the AES using secure multi-party computation protocols,” in International Workshop on Cryptographic Hardware and Embedded Systems, Springer, pp. 63-78, 2011.

[19] R. S. Ali Noori Khamnaeh and H. Soleymani “Provide an optimal masking for the implementation without delay of AES S-box,” Presented at the ISCISC 2020, Tehran, Iran University of Science and Technology, 1399. [Online]. Available: https://civilica.com/doc/1120276/.

[20] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang, “Pushing the limits: A very compact and a threshold implementation of AES,” in Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, pp. 69-88, 2011.

[21] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, “A more efficient AES threshold implementation,” in International Conference on Cryptology in Africa, Springer, pp. 267-284, 2014.

[22] B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen, “Trade-offs for threshold implementations illustrated on AES,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 7, pp. 1188-1200, 2015.

[23] O. Reparaz, B. Bilgin, S. Nikova, B. Gierlichs, and I. Verbauwhede, “Consolidating masking schemes,” in Annual Cryptology Conference, Springer, pp. 764-783, 2015.

[24] T. De Cnudde, O. Reparaz, B. Bilgin, S. Nikova, V. Nikov, and V. Rijmen, “Masking AES with $$ d+ 1$$ shares in hardware,” in International Conference on Cryptographic Hardware and Embedded Systems, Springer, pp. 194-212, 2016.

[25] T. Moos, A. Moradi, T. Schneider, and F.-X. Standaert, “Glitch-resistant masking revisited,” IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 256-292, 2019.

[26] R. Ueno, N. Homma, and T. Aoki, “Toward more efficient DPA-resistant AES hardware architecture based on threshold implementation,” in International Workshop on Constructive Side-Channel Analysis and Secure Design, Springer, pp. 50-64, 2017.

[27] A. Ghoshal and T. De Cnudde, “Several masked implementations of the boyar-peralta AES s-box,” in International Conference on Cryptology in India, Springer, pp. 384-402, 2017.

[28] F. Wegener and A. Moradi, “Yet Another Size Record for AES: A First-Order SCA Secure AES S-Box Based on $$\mathrm {GF}(2^ 8) $$ Multiplication,” in International Conference on Smart Card Research and Advanced Applications, Springer, pp. 111-124, 2018.

[29] J. Daemen, “Changing of the guards: A simple and efficient method for achieving uniformity in threshold sharing,” in International Conference on Cryptographic Hardware and Embedded Systems, Springer, pp. 137-153, 2017.

[30] F. Wegener and A. Moradi, “A first-order SCA resistant AES without fresh randomness,” in International Workshop on Constructive Side-Channel Analysis and Secure Design, Springer, pp. 245-262, 2018.

[31] H. Groß, R. Iusupov, and R. Bloem, “Generic low-latency masking in hardware,” IACR

An Introduction to Security Enhancement of the Cryptographic Algorithms against Side Channel Attacks Using the Threshold Implementation Approach

J. Alizadeh *, H. Ghanbari

Abstract

In order to establish information security, we need to implement a cryptography algorithm in either the software or the hardware. In 1996, Kocher presented the Side Channel Attacks (SCA) on the cryptography systems in which the leakage of some important information was used. Power analysis is one of these attacks. In order to prevent these kinds of attacks, the designers and implementers presented some countermeasures such as hiding and masking. Afterwards, attackers showed that these countermeasures, especially masking, could not reach the security goals in the presence of Glitch. To resolve this challenge Nikova et al. presented the threshold implementation method in 2006. They used three subjects, namely threshold cryptography, hidden share, and multi-party computation in their new countermeasure. In fact, the threshold implementation is a kind of masking with some extra features to establish the information security in the presence of Glitch. In the recent years, the National Institute of Standards and Technology (NIST) has started some activities in the field threshold implementation. Standardization in this field is the main goal of NIST. In this paper we introduce the threshold implementation method as a countermeasure against side channel attacks and review its challenges and advantages in comparison to the previous countermeasures.

Key Words: *Side Channel Attack, Power Analysis Attack, Threshold Implementation*

* Imam Hossein Comprehensive University (jaalizadeh@ihu.ac.ir)- Writer-in-Charge