دوفصل‌نامه علمی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)
سال ۱۴۰۰، جلد ۱۹، شماره ۲ (۱۲-۲۱)
https://monadi.isc.org.ir

ارائه‌شده در هفدهمین کنفرانس بین المللی انجمن رمز ایران در دانشگاه علم و صنعت ایران، ۱۹ و ۲۰ شهریور ۱۳۹۹

ارائه یک نقاب‌گذاری بهینه برای پیاده‌سازی بدون تأخیر زمانی جعبه جانشانی AES★

علی نوری خامنه*[۱،۲]، راضیه سالاری فرد[۲] و هادی سلیمانی[۱]

[۱]پژوهشکده فضای مجازی، دانشگاه شهید بهشتی، تهران، ایران
[۲]دانشکده برق و کامپیوتر، دانشگاه شهید بهشتی، تهران، ایران

اطلاعات مقاله

کلمات کلیدی:
طرح DOM
جعبه جانشانی AES
سهم
پیاده‌سازی آستانه
نقاب‌گذاری
تأخیر زمانی

dor: 10.0000/000000000

نوع مقاله: پژوهشی

چکیده

یکی از روش‌های معمول برای مقابله با حملات کانال جانبی، روش نقاب‌گذاری است. ارائه یک روش نقاب‌گذاری امن و کارا برای پیاده‌سازی سخت‌افزاری الگوریتم‌های رمزنگاری به خاطر وجود گلیچ و تأثیر آن بر نشت اطلاعات، از موضوعات مهم در حوزه رمزنگاری کاربردی است که طی سالیان اخیر توجهات بسیاری را به خود جلب کرده است. یکی از این روش‌های ارائه شده که با فرض رخ دادن گلیچ ایمن است، روش پیاده‌سازی آستانه است که براساس آن، روش‌های متنوعی برای نقاب‌گذاری پیاده‌سازی الگوریتم‌های رمزنگاری در سخت‌افزار ارائه شده است. طرح DOM یکی از روش‌های نقاب‌گذاری بر اساس طرح آستانه است که تا کنون برای پیاده‌سازی الگوریتم‌های رمزنگاری گوناگونی نظیر AES، ارائه شده است. تأخیر زمانی زیاد، یکی از چالش‌هایی است که در این پیاده‌سازی وجود دارد. با توجه به اهمیت تأخیر زمانی در برخی کاربردهای عملی، اخیراً محققین راهکارهایی را به‌منظور کاهش تأخیر زمانی و تعداد بیت‌های تصادفی مورد نیاز برای نقاب‌گذاری به روش DOM در پیاده‌سازی جعبه جانشانی AES ارائه کرده‌اند که مبتنی بر حذف مرحله فشرده‌سازی است که علی‌رغم کاهش تأخیر زمانی و تعداد بیت‌های تصادفی مورد نیاز منجر به افزایش مساحت جعبه جانشانی و همچنین افزایش تعداد سهم‌های خروجی می‌شود. هدف از ارائه این مقاله، بهبود طرح‌های پیشین برای پیاده‌سازی امن و بدون تأخیر زمانی جعبه جانشانی AES براساس طرح DOM است به گونه‌ای که ضمن حفظ ویژگی تأخیر زمانی صفر سیکل، مساحت مورد نیاز و همچنین تعداد سهم‌های خروجی برای پیاده‌سازی کاهش پیدا می‌کند. نتایج به دست آمده، نشان می‌دهد که درطرح پیشنهادی تعداد سهم‌های خروجی ۵۰ درصد کاهش پیدا کرده است. همچنین طرح پیشنهادی در این مقاله در بسترهای ASIC و FPGA پیاده‌سازی شده است. مساحت پیاده‌سازی ASIC با استفاده‌از کتابخانه Nangate 45nm بیش از ۴۶ درصد نسبت به کارهای پیشین کاهش یافته است. علاوه بر این نتایج پیاده‌سازی FPGA با استفاده از دستگاه Xilinx Virtex-5 نشان می‌دهد که تعداد LUTها نیز ۳۹ درصد کاهش پیدا می‌کند.

۱ مقدمه

حملات کانال جانبی[1] یکی از پرکاربردترین حملات فیزیکی بر روی الگوریتم‌های رمزنگاری می‌باشند که از اطلاعات نشتی کانال‌های جانبی نظیر الکترومغناطیس، توان، زمان و غیره انجام می‌گیرند. حملات تحلیل

[1] side channels

★از کمیته علمی هفدهمین کنفرانس بین‌المللی انجمن رمز ایران برای داوری این مقاله تشکر می‌شود.
*نویسنده مسئول
آدرس‌های رایانامه: a.nourikhameneh@mail.sbu.ac.ir (علی نوری خامنه)، salarifard@ce.sharif.edu (راضیه سالاری فرد)، h_soleimany@sbu.ac.ir (هادی سلیمانی)

توان از مهم‌ترین حملات کانال جانبی می‌باشند که از توان مصرفی الگوریتم رمزنگاری استفاده می‌کنند [۱]. هدف از حملات تحلیل توان عموماً یافتن داده‌های میانی حساس یا وابسته به کلید است. دسته‌ای از راهکارهای مقابله با حملات کانال جانبی تحلیل توان، روش نقاب‌گذاری است [۲، ۳] که هدف آن‌ها، تصادفی‌سازی داده‌های میانی و حساس است. ارائه یک روش امن نقاب‌گذاری در پیاده‌سازی سخت‌افزاری الگوریتم‌های رمزنگاری نسبت به پیاده‌سازی‌های نرم‌افزاری به علت رخ دادن گلیچ [1] و تأثیر آن بر نشت اطلاعات، امری چالش‌برانگیز است [۴]. یکی از این روش‌های ایمن و کارا برای پیاده‌سازی سخت‌افزاری الگوریتم‌های رمزنگاری، روش پیاده‌سازی آستانه [2] [۵، ۶] است که می‌تواند امنیت قابل اثبات در مقابل حملات تحلیل توان را ارائه کند. پیاده‌سازی آستانه از رمزنگاری آستانه الهام گرفته و از روش‌هایی نظیر تسهیم راز [3] و محاسبات چندجانبه [4] استفاده می‌کند. روش‌های نقاب‌گذاری گوناگونی برای پیاده‌سازی الگوریتم‌های رمزنگاری استاندارد (مانند AES [5])، بر اساس پیاده‌سازی آستانه ارائه شده‌اند [۷]. این روش‌ها به مدارهای اضافی برای تولید بیت‌های تصادفی و عملیات فشرده‌سازی نیاز دارند. در سالیان اخیر هدف محققین ارائه راهکارهایی به منظور کاهش این سربارهای سخت‌افزاری بوده است.

۱.۱ روش‌های پیشین پیاده‌سازی امن جعبه جانشانی AES

روش‌های نقاب‌گذاری متعددی برای پیاده‌سازی سخت‌افزاری الگوریتم رمزنگاری استاندارد AES ارائه شده‌اند. با توجه به آنکه جعبه جانشانی AES تنها قسمت غیرخطی این الگوریتم است، بخش زیادی از تمرکز محققین ارائه روش‌های نقاب‌گذاری مؤثر برای پیاده‌سازی امن و کارآمد جعبه جانشانی بوده است.

برای نخستین بار پیاده‌سازی آستانه روی الگوریتم رمزنگاری AES توسط مرادی [6] و همکاران در سال ۲۰۱۱ ارائه شد [۸] که در آن مساحت پیاده‌سازی زیاد بود. پس از آن، روشی توسط بیلگین [7] در سال ۲۰۱۴ ارائه شد که دارای مساحت و تعداد بیت‌های تصادفی کمتری بود [۹، ۱۰]. ایده پیاده‌سازی آستانه به‌صورت یک مدل ریاضیاتی بود که یک روش سخت‌افزاری جامع برای آن توسط بیلگین و همکاران به نام طرح CMS [8] در سال ۲۰۱۵ ارائه شد [۱۱]. این روش این قابلیت را ارائه داد تا بتوان الگوریتم‌های رمزنگاری گوناگون را در مرتبه‌های بالاتر در برابر حملات امن نگه داشت. این طرح بر روی جعبه جانشانی AES توسط دی کنادی [9] در سال ۲۰۱۶ پیاده‌سازی و نشان داده شد که مساحت پیاده‌سازی با هزینه‌ی افزایش تعداد بیت تصادفی کاهش می‌یابد [۱۲]. لازم به ذکر است که طرح CMS در سال ۲۰۱۹ توسط مرادی در مرتبه‌های بالا شکسته شد و بنابراین امنیت سطح بالا در مقابل حملات کانال جانبی را ندارد [۱۳]. روش دیگری که برای نقاب‌گذاری در سخت‌افزار برای مرتبه‌های بالاتر بر مبنای پیاده‌سازی آستانه در سال ۲۰۱۶ ارائه شد طرح DOM [10] نام دارد [۱۴]. این روش توسط گراس [11] برای پیاده‌سازی امن جعبه جانشانی AES ارائه شده است که تعداد بیت‌های تصادفی کمتر ولی تأخیر زمانی [12] بیشتری را به همراه دارد.

با توجه به آنکه طرح‌های پیشین عموماً مساحت قابل توجهی مورد نیاز دارند، محققین طی سالیان اخیر تلاش‌هایی به منظور کاهش مساحت مورد نیاز طرح‌های پیاده‌سازی امن جعبه جانشانی AES انجام داده‌اند. در سال ۲۰۱۷ طرح جدیدی توسط ینو [13] ارائه شد که مساحت پیاده‌سازی کمتری نسبت به روش ارائه شده توسط دی کنادی داشت [۱۵]. پس از آن مقاله‌ای توسط دی کنادی در سال ۲۰۱۷ ارائه شد که از معماری جدیدی برای پیاده‌سازی جعبه جانشانی AES استفاده می‌کرد [۱۶]. پس از آن طرح جدیدی مبتنی بر روش DOM در سال ۲۰۱۸ ارائه شد [۱۷] که منجر به کاهش مساحت می‌شد، اما تأخیر زمانی فراوانی در جعبه جانشانی AES داشت. پس از آن مقاله‌ای ارائه شد که از روشی به نام عوض کردن گاردها [14] استفاده می‌کرد [۱۸]. این روش از پیاده‌سازی آستانه گرفته شده است. هدف از این روش این است که دیگر نیازی به استفاده از بیت‌های تصادفی در هر مرحله عملیات رمزنگاری نیست. مرادی این روش را روی جعبه جانشانی AES پیاده‌سازی کرد. بیت‌های تصادفی مورد نیاز در آن صفر است ولی مساحت پیاده‌سازی و تأخیر زمانی بالایی در این معماری وجود دارد [۱۹]. مقاله دیگری در سال ۲۰۱۸ ارائه شد که مخلوطی از نقاب‌گذاری بولی [15] و نقاب‌گذاری ضربی [16] بود [۲۰].

با توجه به آنکه روش‌های نقاب‌گذاری ارائه شده عموماً تأخیر زمانی زیادی دارند، طی سالیان اخیر تحقیقاتی به منظور ارائه راهکاری امن برای پیاده‌سازی جعبه جانشانی AES با تأخیر زمانی کم انجام شد. در سال ۲۰۱۸ توسط گراس روشی ارائه شد [۲۱] که هدف از آن، کاهش تأخیر زمانی طرح DOM و از بین بردن تصادم [17] سهم‌های [18] یک مقدار در عملیات‌های غیرخطی است. این روش تأخیر زمانی و تعداد بیت‌های تصادفی را به صفر کاهش داد. مشکل این پیاده‌سازی، مساحت و تعداد سهم‌های خروجی بسیار بالای آن است. پس از آن روش دیگری در سال ۲۰۱۹ با تعداد بیت تصادفی صفر و با تأخیر زمانی کمتر نسبت به مقاله [۱۹] و تعداد سهم‌های ورودی کمتر ارائه شد [۲۲]. در سال ۲۰۲۰ مقاله‌ای با تأخیر زمانی ۱ سیکل توسط بیلگین ارائه شد [۲۳]. این مقاله از مقاله‌ای در سال ۲۰۱۴ گرفته شده بود که در آن مقدار تأخیر زمانی برابر ۲ سیکل بود [۲۴].

۲.۱ هدف این مقاله

هدف در این مقاله کاهش مساحت پیاده‌سازی جعبه جانشانی AES نقاب‌گذاری شده در مقاله [۲۱] و نصف کردن تعداد سهم‌های خروجی آن با همان تأخیر زمانی و تعداد بیت تصادفی صفر است. این کار با تغییر

[1] glitch [2] threshold implementation [3] secret sharing [4] multi-party computation [5] Advanced Encryption Standard [6] Moradi [7] Bilgin [8] Consolidating Masking Scheme [9] De Cnudde

[10] Domain Oriented Masking [11] Gross [12] latency [13] Ueno [14] changing of the guards [15] boolean masking [16] multiplicative masking [17] collision [18] share

معکوس‌کننده در مد GF(۱۶)[1] در جعبه جانشانی AES انجام می‌گیرد که باعث کاهش مساحت پیاده‌سازی و نصف شدن تعداد سهم‌های خروجی آن می‌شود.

۳.۱ ساختار مقاله

در بخش ۲ مفاهیم اولیه مورد نیاز مانند پیاده‌سازی آستانه و طرح DOM معرفی خواهند شد. در بخش ۳، روش معرفی شده در [۲۱] و کاربرد آن برای پیاده‌سازی امن جعبه جانشانی AES معرفی خواهد شد. در بخش ۴، یک روش مؤثر و جدید برای پیاده‌سازی امن و بدون تأخیر جعبه جانشانی AES معرفی خواهد شد که نسبت به روش‌های پیشین نیاز به مساحت پیاده‌سازی و تعداد سهم‌های خروجی کمتری دارد. در بخش ۵ روش پیشنهادی نسبت به کارهای پیشین از دیدگاه تعداد بیت تصادفی و تأخیر زمانی و مساحت پیاده‌سازی مقایسه خواهد شد. در بخش ۶، نتیجه‌گیری مقاله ارائه خواهد شد.

۲ مفاهیم اولیه

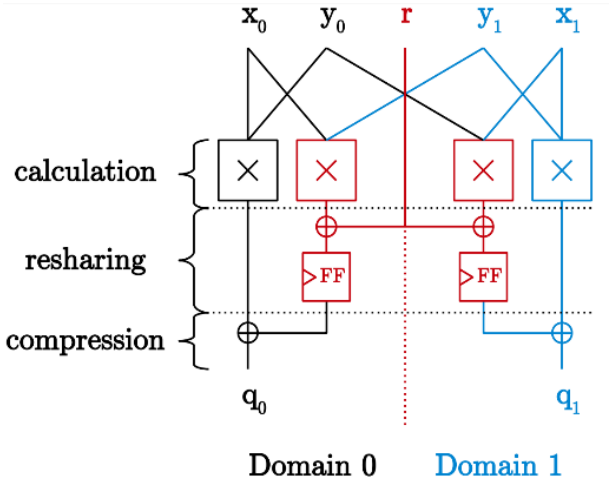
در این بخش نخست پیاده‌سازی آستانه و سپس طرح DOM معرفی می‌شوند.

۱.۲ پیاده‌سازی آستانه

به طور کلی نقابگذاری با استفاده از تصادفی کردن مقادیر میانی حساس انجام می‌شود. این کار می‌تواند به این صورت انجام شود که ورودی را به $d+1$ سهم با مقادیر تصادفی تقسیم کند. به مقدار d مقدار آستانه یا مرتبه‌ی اطمینان[2] می‌گویند. بدین معنا است که اگر مهاجم d سهم یا کمتر از آن مقدار حساس را داشته باشد نمی‌تواند اطلاعاتی از این مقدار به دست آورد. به این مدل که در بالا ارائه شد مدل d-probing گفته می‌شود [۲۵]. حال اگر این تعداد ورودی به همین تعداد تابع وارد شود که مجموع این توابع برابر تابع کل باشد. به این نقابگذاری پیاده‌سازی آستانه می‌گویند [۵، ۶]. یکی از ویژگی‌های مهم این نقابگذاری غیر کامل بودن[3] است. این ویژگی به این معنا است که هر تابع باید از حداقل یکی از سهم‌های ورودی مستقل باشد.

۲.۲ طرح DOM

ایده اصلی طرح DOM [۱۴] به این صورت است که هر معماری (و هر مقدار ورودی به دنبال آن) می‌تواند به $d+1$ معماری کاملاً مستقل از هم تقسیم شود. به هرکدام از این بخش‌ها (معماری‌ها)، یک منطقه[4] گفته می‌شود. هر منطقه، تنها به یک سهم از ورودی‌ها وابسته است. به طور مثال برای مرتبه اول در منطقه صفر تنها سهم‌های x_0 و y_0 حضور دارند که اندیس صفر به معنای شماره منطقه است.



شکل ۱. ضربکننده طرح DOM [۱۴]

عملیات‌های خطی را بدون تصادم سهم‌ها می‌توان انجام داد ولی در عملیات‌های غیرخطی نیاز است که میان منطقه‌های مختلف تصادم سهم‌ها اتفاق افتد.

به‌طور مثال در عمل ضرب $x \cdot y$ که مقادیر x و y به صورت سهم‌های مختلف تقسیم می‌شوند عملیات ضرب به‌صورت $\sum_{i=0}^{d} \sum_{i=0}^{d} x_i \cdot y_i$ انجام می‌گیرد. این مشکل تصادم در ضربکننده‌های طرح DOM حل‌شده است. ضربکننده مرتبه اول طرح DOM در شکل ۱ آمده است.

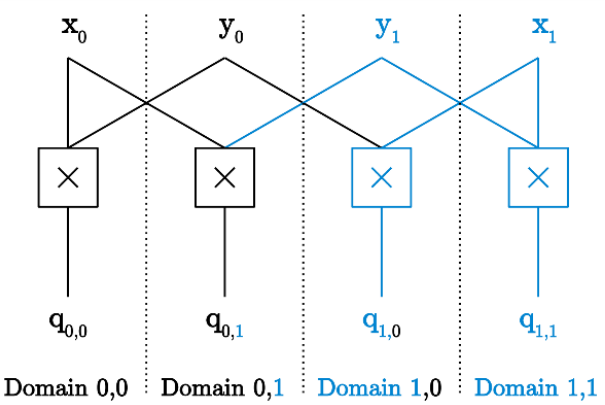
در این طرح در مرتبه اول معماری به دو منطقه تقسیم می‌شود. این طرح سه مرحله دارد. مرحله اول برای محاسبه‌ی ضرب سهم‌ها است. در این مرحله بعضی سهم‌ها در منطقه خود در دیگری ضرب می‌شوند مانند $x_0 \cdot y_0$ و $x_1 \cdot y_1$. به این دلیل که ضرب این سهم‌ها در همان منطقه‌ی خود انجام می‌گیرد بنابراین نشت اطلاعات وجود ندارد.

برخی سهم‌ها در سهم‌های مناطق دیگر ضرب می‌شوند، مانند $x_1 \cdot y_0$ و $x_0 \cdot y_1$ که به این خاطر ممکن است نشت اطلاعات صورت گیرد. چون اگر مقدار $x \cdot x$ محاسبه شود و سهم‌های این دو باهم برابر باشند یک نشت اطلاعات به‌صورت $x_0 \cdot x_1$ به وجود می‌آید. این مشکل در بخش ۲.۳ برطرف می‌شود. مرحله سوم برای این است که تعداد سهم‌های خروجی به $d+1$ کاهش یابد. برای جلوگیری از نشت اطلاعات در این مرحله به دلیل گلیچ، احتیاج به تصادفی‌سازی وجود دارد. این کار در مرحله‌ی دوم با اضافه شدن بیت‌های تصادفی و استفاده از رجیسترها برای همگام‌سازی انجام می‌گیرد. این رجیسترها باعث یک سیکل تأخیر زمانی می‌شوند که هدف در بخش ۱.۳ حذف این رجیسترها و به دنبال آن مرحله فشرده‌سازی است.

۳ طرح DOM بدون تأخیر زمانی

طرح DOM که در بخش قبل معرفی شد، دارای تأخیر زمانی بالائی است. بر همین اساس در [۲۱]، دو راهکار به‌منظور کاهش تأخیر زمانی طرح DOM ارائه شد: ۱) حذف کردن مرحله فشرده‌سازی و تصادفی‌سازی که

[1] Galois field [2] certain degree [3] non-completeness [4] domain



شکل ۲. طرح DOM بدون تأخیر زمانی [۲۱]

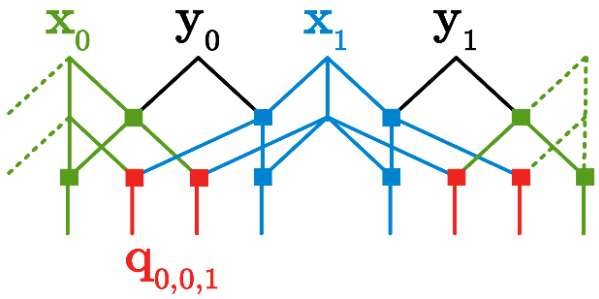
نیاز به رجیستر دارد. حذف این رجیستر باعث کاهش تأخیر زمانی می‌شود. ۲) از بین بردن تصادم مقدارهای ورودی در عملیات‌های غیرخطی وقتی این مقدارها به هم وابسته باشند. این دو راهکار در این بخش به صورت مختصر شرح داده خواهد شد. در ادامه پیاده‌سازی این طرح بر روی جعبه جانشانی AES توضیح داده خواهد شد.

۱.۳ حذف فشرده‌سازی و تصادفی‌سازی

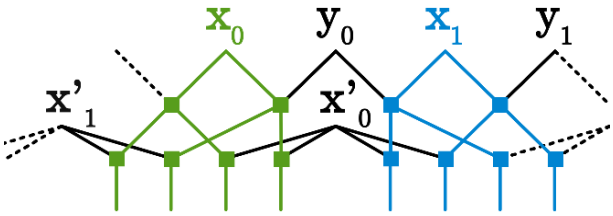
در طرح DOM مرحله‌ی فشرده‌سازی و تصادفی‌سازی برای مدل حمله الزامی نیست، چون خروجی هر کدام از ضرب‌کننده‌ها فقط به یک سهم وابسته است. حذف مرحله فشرده‌سازی موجب می‌شود که دیگر به مرحله تصادفی‌سازی احتیاجی نداشته باشد. این کار باعث کاهش تأخیر زمانی و حذف بیت‌های تصادفی می‌شود. حذف این مراحل، باعث افزایش تعداد سهم‌های خروجی می‌شود. به‌عنوان مثال همان‌گونه که در شکل ۲ قابل مشاهده است، در عمل ضرب $x \cdot y$ در مرتبه اول با تعداد سهم ورودی ۲، تعداد سهم خروجی برابر ۴ است. هرگونه عملیات خطی بعد از آن روی هرکدام از سهم‌های q به صورت مستقل انجام می‌گیرد. اگر مقدار q در مقدار دیگری ضرب شود این کار باعث افزایش نمایی تعداد سهم‌های خروجی آن می‌شود. برای کاهش افزایش نمایی سهم‌های خروجی، باید تعداد عملیات‌های غیرخطی به حداقل برسد.

۲.۳ جلوگیری از تصادم مقدارهای ورودی یکسان

حذف مراحل تصادفی‌سازی و فشرده‌سازی در صورتی که ورودی‌ها از هم مستقل باشند، خللی در امنیت پیاده‌سازی ایجاد نمی‌کند. به‌طور مثال محاسبه‌ی $(x \cdot y) \cdot z$ در صورتی که سهم‌های تمام این مقدارها از هم مستقل باشند، ایرادی ندارد. حال اگر ورودی‌های عملیات ضرب به‌صورت $(x \cdot y) \cdot x$ باشند در این صورت ورودی‌ها به هم وابسته هستند و ممکن است باعث نشت اطلاعات شود. به عنوان مثال همان‌گونه که در شکل ۳ قابل مشاهده است، یکی از سهم‌های خروجی ضرب‌کننده‌ی طرح DOM بدون تأخیر زمانی مقدار $q_{0,0,1} = x_0 \cdot y_0 \cdot x_1$ است. این سهم خروجی دو سهم از مقدار x را کنار هم می‌آورد که می‌تواند منجر به نشت اطلاعات می‌شود (بارنگ قرمز نشان داده شده است).



شکل ۳. تصادم سهم‌های یک مقدار یکسان در طرح DOM بدون تأخیر زمانی [۲۱]



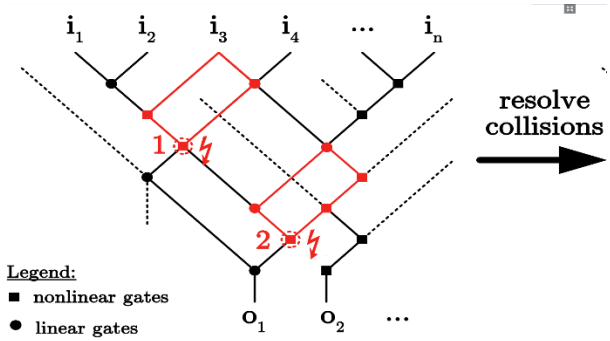
شکل ۴. حل شدن مشکل تصادم سهم‌های یک مقدار یکسان در طرح DOM بدون تأخیر زمانی [۲۱]

برای غلبه بر این مشکل، می‌توان سهم‌های یکی از مقادیر ورودی x را که موجب این تصادم شده است، تغییر داد. به این معنا که به جای محاسبه‌ی $(x \cdot y) \cdot x$ مقدار برابر آن $(x \cdot y) \cdot x'$ را محاسبه کرد. مقدار x' به‌صورت $x' = x = \sum_{i=0}^{d} x_i = \sum_{i=0}^{d} x'_i$ است. به طور مثال سهم‌های x به‌صورت $x_1 = x \oplus r_1$ و $x_2 = r_1$ باشد و سهم‌های x' به صورت $x'_1 = x \oplus r_2$ و $x'_2 = r_2$ باشد. همان‌طور که در شکل ۴ نشان داده شده است $q_{0,0,1} = x_0 \cdot y_0 \cdot x'_1$ دیگر باعث نشت اطلاعات نمی‌شود.

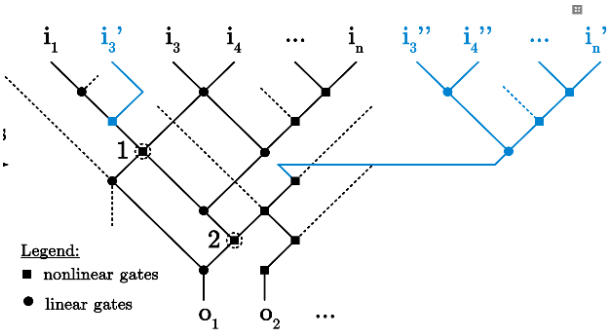
۳.۳ تصادم مقادیر یکسان پس از عبور از چند عملیات خطی و غیرخطی

در صورتی که دو مقدار یکسان پس از عبور از چند عملیات خطی و غیرخطی در یک عملیات غیرخطی دیگر با هم تصادم داشته باشند، باز هم این کار باعث نشت اطلاعات می‌شود. به طور مثال همان طور که در شکل ۵ با رنگ قرمز نشان داده‌شده است مقدار i_3 در نقطه‌ی ۱ با خودش تصادم دارد. در نقطه‌ی ۲ نیز از i_3 تا i_n در یک عملیات غیرخطی با خودشان تصادم دارند.

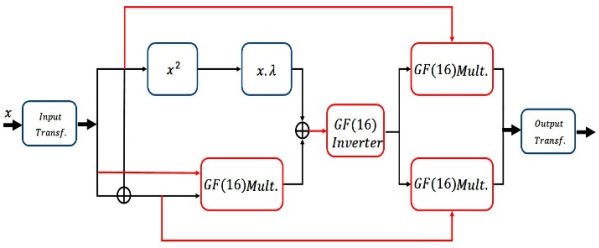
برای حل این مشکل در نقاط ۱ و ۲ این مقادیر را به همراه عملیات‌های خطی و غیرخطی با سهم‌های متفاوت باید تکرار کرد. این کار در شکل ۶ با رنگ آبی نشان داده شده است. مقدار i'_4 در نقطه ۱ و مقادیر i''_4 تا i''_n در نقطه ۲ به همراه عملیات‌های خطی و غیرخطی آن دوباره تا این نقاط تکرار شده‌اند.



شکل ۵. تصادم مقادیر یکسان پس از عبور از چند عملیات خطی و غیرخطی [۲۱]



شکل ۶. حل مشکل تصادم مقادیر یکسان پس از عبور از چند عملیات خطی و غیرخطی [۲۱]



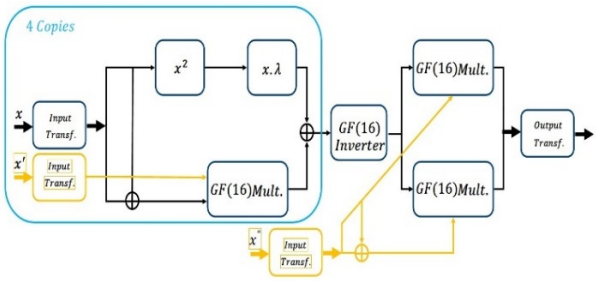
شکل ۷. جعبه جانشانی AES در طرح ادوین موی [۲۶]

۴.۳ پیاده‌سازی طرح DOM بدون تأخیر زمانی بر روی جعبه جانشانی AES

جعبه جانشانی AES به روش‌های گوناگونی به‌صورت سخت‌افزاری پیاده‌سازی شده است. یکی از طرح‌هایی که برای پیاده‌سازی طرح DOM بدون تأخیر زمانی[1] بر روی آن از لحاظ مساحت مناسب است طرح ادوین موی[2] است [۲۶] که در شکل ۷ نشان داده‌شده است.

برای پیاده‌سازی طرح DOM بدون تأخیر زمانی بر روی این جعبه جانشانی در عملیات‌های خطی مشکلی وجود ندارد اما در عملیات‌های غیرخطی تصادم وجود دارد. در ادامه این تصادم‌ها و روش حل آن‌ها آمده است.

[1] zero-latency [2] Edwin NC Mui



شکل ۸. جعبه جانشانی AES نقاب‌گذاری شده با ضرب‌کننده‌های طرح DOM بدون تأخیر زمانی [۲۱]

۱.۴.۳ سه ضرب‌کننده‌ی داخل جعبه جانشانی AES طرح ادوین موی

در طرح DOM بدون تأخیر زمانی در مرتبه اول مقدار x به ۲ سهم تقسیم می‌شود و وارد جعبه جانشانی می‌شود. در عملیات‌های نگاشت ورودی، توان رسانی، XOR و ضرب در یک عدد ثابت تصادم دو مقدار یکسان وجود ندارد، زیرا تمام عملیات‌ها خطی هستند اما در عملیات ضرب‌کننده قبل از معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ این مشکل وجود دارد. در شکل ۷ این مشکل که با رنگ قرمز نشان داده شده است. این تصادم از ورود سهم‌های بیت‌های پایین x، و XOR سهم‌های بیت‌های پایین و بالای x به وجود آمده است.

برای حل این مشکل بیت‌های پایین x را به همراه عملیات نگاشت ورودی با سهم‌های متفاوت دوباره تکرار می‌کند. این کار در شکل ۸ با رنگ زرد نشان داده شده است. سهم‌های x' با x متفاوت است ولی در XOR سهم‌ها، باهم برابر هستند.

در دو ضرب‌کننده‌ی پس از معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ نیز تصادم دو مقدار یکسان وجود دارد. در شکل ۷ این مشکل با رنگ قرمز نشان داده شده است. برای حل این مشکل مقدار x را به همراه عملیات نگاشت ورودی و XOR پس از آن را، با سهم‌های متفاوت دوباره تکرار می‌کند. این کار در شکل ۸ با رنگ زرد نشان داده شده است. سهم‌های x'' با x نیز متفاوت است ولی در XOR سهم‌ها، باهم برابر هستند.

۲.۴.۳ معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ در طرح ادوین موی

معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ در طرح ادوین موی [۲۶] به صورت زیر است:

$$
\begin{aligned}
a' &= a \oplus abc \oplus ad \oplus b \\
b' &= abc \oplus abd \oplus ad \oplus b \oplus bc \\
c' &= a \oplus abc \oplus acd \oplus b \oplus bd \oplus c \\
d' &= abc \oplus abd \oplus ac \oplus acd \oplus ad \oplus b \oplus bc \oplus bcd \oplus c \oplus d
\end{aligned}
\qquad (۱)
$$

همان‌طور که در رابطه (۱) نشان داده شده است بیت‌های یک مقدار یکسان در هم ضرب می‌شوند. این خود باعث تصادم میان مقادیر یکسان می‌شود. بنابراین باید هر بیت ورودی را تکرار کرد یعنی به تعداد ۴ کپی

عملیات‌های قبل از معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ را با سهم‌های متفاوت تکرار کرد که در شکل ۸ با رنگ آبی نشان داده شده است.

مشکل دیگری که وجود دارد این است که باز امکان تصادم به هنگام XOR در رابطه (۱) وجود دارد. به طور مثال فرض کنید قسمتی از رابطه (۱) به صورت زیر است:

$$q = abc \oplus abd \oplus acd \qquad (۲)$$

اگر سهم‌های q در رابطه (۲) را محاسبه کنید، یکی از سهم‌های خروجی به‌صورت زیر است:

$$q_{(i,j,k)} = a_i b_j c_k \oplus a_i b_j d_k \oplus a_i c_j d_k \qquad (۳)$$

در رابطه‌ی (۳) سهم‌های b_j و c_j اگر گلیچ اتفاق بیفتد با هم تصادم دارند. برای جلوگیری از این اتفاق تعداد سهم‌های خروجی به صورت رابطه زیر افزایش می‌یابد:

$$q_{(i,j,k)} = \{a_i b_j c_k \oplus a_i b_j d_k, a_i c_j d_k\} \qquad (۴)$$

همان‌طور که در رابطه‌ی (۴) مشخص است یک XOR حذف‌شده و تعداد سهم‌های خروجی افزایش یافته است.

۲ سهم به جعبه جانشانی وارد می‌شود. این ۲ سهم به ضرب‌کننده‌ی پس از آن وارد می‌شود که در ۲ سهم از x' ضرب می‌شود. پس تعداد سهم‌های خروجی ضرب‌کننده در مرتبه اول برابر ۴ می‌شود که با ۲ سهم که از عملیات‌های توان رسانی و ضرب‌در عدد ثابت می‌آید XOR شده و با ۴ سهم وارد معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ می‌شود. در ورودی معکوس‌کننده ۴ سهم هر بیت، از ۴ سهم بیت دیگر مستقل است. تعداد سهم‌های خروجی به ازای بیت‌های دوم و سوم ۱۲۸ سهم و به ازای بیت‌های صفرم و اول ۶۴ سهم است (این تفاوت سهم‌ها به خاطر تصادم به هنگام عملیات خطی است که در بیت‌های دوم و سوم بیشتر است). ۱۲۸ سهم به ضرب‌کننده‌های پایانی وارد می‌شود که در ۲ سهم از x'' ضرب می‌شوند. خروجی این ضرب‌کننده‌ها و خروجی کل ۲۵۶ سهم است. جعبه جانشانی AES در حالت بدون تأخیر زمانی به‌صورت شکل ۸ است.

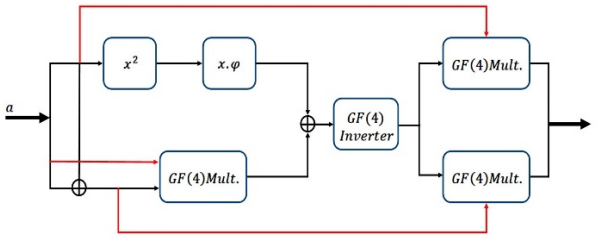
در این نقاب‌گذاری، تأخیر زمانی جعبه جانشانی AES و تعداد بیت‌های تصادفی آن برابر صفر سیکل است. اندازه مساحت پیاده‌سازی آن برابر 47.66 kGE [1] است.

۴ راهکار پیشنهادی برای کاهش مساحت و تعداد سهم‌های خروجی

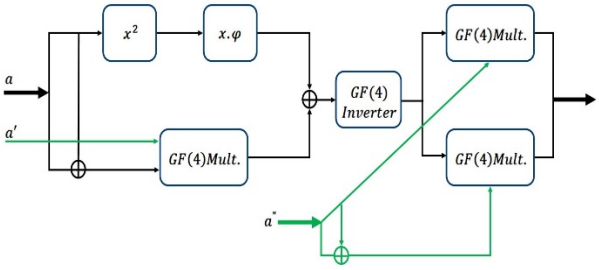
اصلی‌ترین مشکلات پیاده‌سازی ارائه‌شده در بخش ۴.۳ مساحت آن و تعداد سهم‌های خروجی بالای آن است. برای حل این مشکلات به‌جای استفاده از معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ باید آن را به‌صورت $\mathrm{GF}((۲^۲)^۲)$ تجزیه کرد. این معکوس‌کننده در شکل ۹ آمده است.

معماری ارائه شده در بخش ۴.۳ تا قبل از این معکوس‌کننده تغییر نمی‌کند. به جز اینکه ۴ کپی معماری‌های پیش از آن حذف می‌شود. در

[1] Kilo Gate equivalent



شکل ۹. معکوس‌کننده در مد $\mathrm{GF}((۲^۲)^۲)$



شکل ۱۰. حل مشکل تصادم ضرب‌کننده‌ها در معکوس‌کننده در مد $\mathrm{GF}((۲^۲)^۲)$

این قسمت نیز ابتدا ۲ سهم به جعبه جانشانی وارد می‌شود. سپس، بعد از عبور از ضرب‌کننده در ۲ سهم x' ضرب می‌شود. در نهایت ۴ سهم با ۲ سهمی که از عملیات‌های خطی آمده XOR شده و با ۴ سهم وارد معکوس‌کننده در مد $\mathrm{GF}((۲^۲)^۲)$ می‌شود.

در این معکوس‌کننده در مد $\mathrm{GF}((۲^۲)^۲)$ در عملیات‌های خطی مانند توان‌رسانی و غیره مشکلی وجود ندارد. در معکوس‌کننده در مد $\mathrm{GF}(۴)$ نیز به خاطر خطی بودن مشکلی وجود ندارد چون سهم ها در عملیات‌های خطی با هم ترکیب نمی‌شوند:

$$\Delta_۰ = g_۰ \oplus g_۱ \qquad (۵)$$

$$\Delta_۱ = g_۱ \qquad (۶)$$

در این قسمت تصادم دو مقدار یکسان پس از عبور از چند عملیات خطی و غیرخطی در ضرب‌کننده‌های در مد $\mathrm{GF}(۴)$ وجود دارد. مشکل وجود تصادم در شکل ۹ با رنگ قرمز نشان داده شده است.

۱.۴ ضرب‌کننده پیش از معکوس‌کننده در مد $\mathrm{GF}(۴)$

برای حل مشکل تصادم، بیت‌های پایین a را به همراه معماری قبل از معکوس‌کننده در مد $\mathrm{GF}((۲^۲)^۲)$ با سهم‌های متفاوت دوباره تکرار شده است. اجرای این روش در شکل ۱۰ با رنگ سبز نشان داده شده است. سهم‌های a' با a متفاوت هستند ولی در XOR سهم‌ها، با هم برابر هستند.

مقدار a که ورودی است ۴ سهم دارد و a' که تکرار آن است نیز ۴ سهم دارد. به این ترتیب از ضرب‌کننده ۱۶ سهم خارج می‌شود و با ۴ سهم عملیات‌های توان رسانی و ضرب در یک عدد ثابت، XOR می‌شود. درنهایت ۱۶ سهم وارد معکوس‌کننده در مد $\mathrm{GF}(۴)$ می‌شود. به خاطر خطی بودن معکوس‌کننده در مد $\mathrm{GF}(۴)$ ۱۶ سهم از آن خارج می‌شود.

۲.۴ دو ضرب‌کننده پس از معکوس‌کننده در مد $GF(۴)$

در دو ضرب‌کننده موجود پس از معکوس کننده که در مد $GF(۴)$ عمل می‌کند، مانند بخش ۱.۴ تصادم دو مقدار یکسان وجود دارد که می‌تواند منجر به نشت اطلاعات شود. در شکل ۹ این مسئله با رنگ قرمز نشان داده شده است. برای برطرف کردن این معضل، مقدار a به همراه معماری قبل از معکوس‌کننده در مد $GF((۲^۲)^۲)$ دوباره تکرار می‌شود. این کار در شکل ۱۰ با رنگ سبز نشان داده شده است. سهم‌های a'' با a متفاوت است ولی در XOR سهم‌ها، با هم برابر هستند.

در این قسمت ۱۶ سهم که از معکوس‌کننده در مد $GF(۴)$ خارج می‌شوند در ۴ سهم از a'' ضرب می‌شود. بدین ترتیب خروجی کل معکوس‌کننده در مد $GF((۲^۲)^۲)$ برابر ۶۴ سهم می‌شود.

پس از این معکوس‌کننده در مد $GF((۲^۲)^۲)$ مدار مانند بخش ۴.۳ است. به این صورت که ۲ سهم x'' در ۶۴ سهم خروجی معکوس‌کننده ضرب می‌شود. در نهایت سهم‌های خروجی در این مقاله برابر ۱۲۸ است. در این طرح ۴ کپی (از هرکدام از ۴ بیت ورودی معکوس‌کننده) پیش از معکوس‌کننده به ۳ کپی (a, a', a'') کاهش یافت.

این روش سبب می‌شود که تأخیر زمانی جعبه جانشانی AES و تعداد بیت تصادفی آن همچنان صفر باقی بماند ولی مساحت پیاده‌سازی کاهش یابد. همان‌طور که در بخش بعد دیده خواهد شد، نتایج عملی نشان می‌دهد که اندازه مساحت پیاده‌سازی روش ارائه شده در این مقاله برابر 25.67 kGE است که به مراتب کوچک‌تر از روش‌های پیشین است.

۵ مقایسه و ارزیابی

همان‌طور که مشاهده می‌شود، در روش ارائه شده در این مقاله تعداد سهم‌های خروجی جعبه جانشانی AES که با طرح DOM بدون تأخیر زمانی نقاب‌گذاری شده است از ۲۵۶ به ۱۲۸ سهم (معادل ۵۰ درصد) کاهش پیدا کرد که باعث نصف شدن محاسبات در دو ضرب‌کننده‌ی پایانی، نگاشت خروجی و تبدیل آفینی می‌شود. این عامل همچنین سبب کاهش مساحت پیاده‌سازی جعبه جانشانی AES با طرح DOM بدون تأخیر زمانی می‌شود. هم‌چنین تعداد کپی‌های موردنیاز از مدارهای پیش از معکوس‌کننده در مد $GF(۱۶)$ از ۴ به ۳ کاهش پیدا کرد.

به‌منظور ارائه یک مقایسه منصفانه روش پیشنهادی در این مقاله و همچنین روش ارائه شده در مقاله‌ی [۲۱] بر روی کتابخانه Nangate 45nm با نرم‌افزار Design compiler Synopsys پیاده‌سازی شد که نتایج آن در جدول ۱ نمایش داده شده است. در این جدول همچنین نتایج پیاده‌سازی با تکنولوژی‌های متفاوت که در مقالات منتشرشده، آورده شده است. همان‌گونه که در جدول ۱ قابل مشاهده است، روش پیشنهادی در این مقاله با حفظ ویژگی‌های روش ارائه شده در مقاله‌ی [۲۱] به لحاظ تأخیر زمانی و تعداد بیت‌های تصادفی، مساحت موردنیاز از 47.66 kGE به 25.67 kGE (بیش از ۴۶ درصد) کاهش پیدا کرده است.

جدول ۱. مقایسه ایده پیشنهادی و مقالات ارائه‌شده بر روی ASIC

تکنولوژی	بیت تصادفی (bits)	تأخیر زمانی (cycle)	مساحت (kGE)	طرح
UMC 180nm	۸	۳۶	۱/۳۷	[۱۷]
UMC 180nm	۰	۱۶	۴/۲۰	[۱۹]
UMC 180nm	۳۲	۸	۲/۳۲	[۱۷]
UMC 180nm	۱۸	۸	۲/۶۰	[۱۴]
Nangate 45nm	۵۴	۶	۱/۹۸	[۱۲]
TSMC 65nm	۶۴	۵	۱/۴۲	[۱۵]
Nangate 45nm	۱۹	۲ + ۳	۱/۶۹	[۲۰]
UMC 180nm	۴۸	۴	۴/۲۴	[۸]
Nangate 45nm	۰	۴	۳/۵۰	[۲۲]
UMC 180nm	۴۴	۳	۳/۷۱	[۹]
Nangate 45nm	۲۰	۳	۲/۹۱	[۱۶]
UMC 180nm	۳۲	۳	۲/۸۴	[۱۰]
UMC 90nm	۴۱۶	۲	۶/۷۴	[۲۱]
created a Liberty-format	۳۶	۲	۲/۸۳	[۲۴]
UMC 90nm	۲۰۴۸	۱	۶۰/۷۳	[۲۱]
Global Foundry 28nm	۳۶	۱	۳/۴۸	[۲۳]
UMC 90nm	۰	۰	۱۷/۸۳	[۲۱]
Nangate 45nm	۰	۰	۴۷/۶۶	[۲۱]
Nangate 45nm	۰	۰	۲۵/۶۷	روش پیشنهادی با تأخیر زمانی صفر سیکل

جدول ۲. مقایسه پیاده‌سازی ایده پیشنهادی و مقاله [۲۱] بر روی دستگاه Xilinx Virtex-5

طرح	LUT	IOB	فرکانس (MHz)	Flip Flop
[۲۱]	۸۹۸۸	۲۱۸۴	۱۰۲	۰
روش‌پیشنهادی با تأخیر زمانی صفر سیکل	۵۴۹۰	۱۱۳۰	۷۰	۰

مقاله [۲۱] بر روی کتابخانه UMC 90nm پیاده‌سازی شده است. در این مقاله برای مقایسه منصفانه این مقاله بر روی کتابخانه Nangate 45nm پیاده سازی شده است.

پیاده‌سازی روش پیشنهادی و مقاله [۲۱] بر روی دستگاه Xilinx Virtex-5 نیز در جدول ۲ قابل مشاهده است. نتایج به دست آمده نشان می‌دهد که روش پیشنهادی در این مقاله نسبت به روش پیشین ارائه شده با تأخیر صفر سیکل، در بستر FPGA نیز ۳۸ درصد کاهش یافته است.

لازم به ذکر است که در مقاله‌های [۱۹] و [۲۲] نیز تعداد بیت‌های تصادفی مورد نیاز صفر است اما تأخیر زمانی آن‌ها صفر نیست. بر اساس اطلاعات، روش ارائه شده در این مقاله، منجر به کمترین مقدار مساحت گزارش‌شده برای پیاده‌سازی امن جعبه جانشانی AES با تأخیر زمانی صفر سیکل می‌شود.

۶ نتیجه‌گیری

در این مقاله روشی ارائه شد که بر اساس آن، مساحت و تعداد سهم‌های خروجی پیاده‌سازی امن جعبه جانشانی AES با تأخیر صفر سیکل (ارائه شده در مقاله [۲۱]) بهبود یافته می‌یابد. در این مقاله نشان داده شد که با تغییر معکوس‌کننده از مد $\mathrm{GF}(۱۶)$ به $\mathrm{GF}((۲^۲)^۲)$، مساحت و تعداد سهم‌های خروجی کاهش چشمگیری پیدا می‌کند، به صورتی که تعداد بیت تصادفی و تأخیر زمانی همچنان صفر باقی ماند. روش ارائه شده در این مقاله، کمترین مساحت پیاده‌سازی با تأخیر زمانی صفر سیکل و تعداد بیت تصادفی صفر را نیاز دارد. هم‌چنین تعداد کپی از مدارهای پیش از معکوس‌کننده در مد $\mathrm{GF}(۱۶)$ از ۴ به ۳ کاهش پیدا کرد.

در کارهای آینده می‌توان به کاهش مساحت پیاده‌سازی و کاهش بیشتر سهم‌های خروجی اشاره کرد. هم‌چنین می‌توان این طرح DOM بدون تأخیر زمانی را بر روی روش‌های دیگر پیاده‌سازی جعبه جانشانی AES اجرا کرد.

مراجع

[1] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual international cryptology conference*, pages 388–397. Springer, 1999.

[2] Suresh Chari, Charanjit S Jutla, Josyula R Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In *Annual International Cryptology Conference*, pages 398–412. Springer, 1999.

[3] Louis Goubin and Jacques Patarin. Des and differential power analysis the "duplication" method. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 158–172. Springer, 1999.

[4] Stefan Mangard and Kai Schramm. Pinpointing the side-channel leakage of masked aes hardware implementations. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 76–90. Springer, 2006.

[5] Svetla Nikova, Christian Rechberger, and Vincent Rijmen. Threshold implementations against side-channel attacks and glitches. In *International conference on information and communications security*, pages 529–545. Springer, 2006.

[6] Svetla Nikova, Vincent Rijmen, and Martin Schläffer. Secure hardware implementation of nonlinear functions in the presence of glitches. *Journal of Cryptology*, 24(2):292–321, 2011.

[7] Joan Daemen and Vincent Rijmen. *The design of Rijndael*, volume 2. Springer, 2002.

[8] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong 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*, pages 69–88. Springer, 2011.

[9] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. A more efficient aes threshold implementation. In *International Conference on Cryptology in Africa*, pages 267–284. Springer, 2014.

[10] Begül Bilgin, Benedikt Gierlichs, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Trade-offs for threshold implementations illustrated on aes. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1188–1200, 2015.

[11] Oscar Reparaz, Begül Bilgin, Svetla Nikova, Benedikt Gierlichs, and Ingrid Verbauwhede. Consolidating masking schemes. In *Annual Cryptology Conference*, pages 764–783. Springer, 2015.

[12] Thomas De Cnudde, Oscar Reparaz, Begül Bilgin, Svetla Nikova, Ventzislav Nikov, and Vincent Rijmen. Masking aes with $d + 1$ shares in hardware. In *International Conference on Cryptographic Hardware and Embedded Systems*, pages 194–212. Springer, 2016.

[13] Thorben Moos, Amir Moradi, Tobias Schneider, and François-Xavier Standaert. Glitch-resistant masking revisited: Or why proofs in the robust probing model are needed. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 256–292, 2019.

[14] Hannes Groß, Stefan Mangard, and Thomas Korak. Domain-oriented masking: Compact masked hardware implementations with arbitrary protection order. *Cryptology ePrint Archive*, 2016.

[15] Rei Ueno, Naofumi Homma, and Takafumi Aoki. Toward more efficient dpa-resistant aes hardware architec-

ture based on threshold implementation. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 50–64. Springer, 2017.

[16] Ashrujit Ghoshal and Thomas De Cnudde. Several masked implementations of the boyar-peralta aes s-box. In *International Conference on Cryptology in India*, pages 384–402. Springer, 2017.

[17] Felix Wegener and Amir Moradi. Yet another size record for aes: A first-order sca secure aes s-box based on $gf(2^8)$ multiplication. In *International Conference on Smart Card Research and Advanced Applications*, pages 111–124. Springer, 2018.

[18] Joan 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*, pages 137–153. Springer, 2017.

[19] Felix Wegener and Amir Moradi. A first-order sca resistant aes without fresh randomness. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 245–262. Springer, 2018.

[20] Lauren De Meyer, Oscar Reparaz, and Begül Bilgin. Multiplicative masking for aes in hardware. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 431–468, 2018.

[21] Hannes Groß, Rinat Iusupov, and Roderick Bloem. Generic low-latency masking in hardware. *IACR transactions on cryptographic hardware and embedded systems*, pages 1–21, 2018.

[22] Takeshi Sugawara. 3-share threshold implementation of aes s-box without fresh randomness. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 123–145, 2019.

[23] Pascal Sasdrich, Begül Bilgin, Michael Hutter, and Mark E Marson. Low-latency hardware masking with application to aes. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 300–326, 2020.

[24] Andrew J Leiserson, Mark E Marson, and Megan A Wachs. Gate-level masking under a path-based leakage metric. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 580–597. Springer, 2014.

[25] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *Annual International Cryptology Conference*, pages 463–481. Springer, 2003.

[26] Edwin NC Mui, R Custom, and D Engineer. Practical implementation of rijndael s-box using combinational logic. *Custom R&D Engineer Texco Enterprise Pvt. Ltd*, 2007.