ارائه یک مکانیزم درون تراشه برای تشخیص حملات زنجیره پویش در تراشه‌های رمزنگاری

فاطمه جمالی زواره و حاکم بیت‌الهی

چکیده: با پیدایش تراشه‌های رمزنگاری، حملات کانال جانبی تهدید جدیدی علیه الگوریتم‌های رمزنگاری و سیستم‌های امنیتی به شمار می‌روند. حملات کانال جانبی به ضعف‌های محاسباتی الگوریتم‌ها کاری نداشته و از ضعف‌های پیاده‌سازی استفاده می‌نمایند. زنجیره پویش که در آزمون تراشه‌ها کاربرد گسترده‌ای دارد، یکی از این کانال‌های جانبی است. برای جلوگیری از حمله با استفاده از زنجیره پویش، می‌توان ارتباط زنجیره‌های پویش را پس از آزمون ساخت از بین برد اما این روش، امکان آزمون پس از ساخت و همچنین به‌روزرسانی مدارها را غیر ممکن می‌سازد. بنابراین باید علاوه بر حفظ آزمون‌پذیری زنجیره پویش، به دنبال روشی برای جلوگیری از حملات کانال جانبی ناشی از آن بود. در این مقاله روشی ارائه شده که بتواند حمله مهاجم را شناسایی کند و از حمله با استفاده از زنجیره پویش جلوگیری نماید. در این روش با مجازشماری کاربر، خروجی متناسب، تولید شده و از دسترسی مهاجم به اطلاعات حساس جلوگیری خواهد گردید. روش ارائه‌شده با سربار مساحت کمتر از ۱٪، سربار توان مصرفی ایستای حدود ۱٪ و سربار تأخیر ناچیز، قابلیت آزمون‌پذیری را حفظ کرده و می‌تواند از حملات مبتنی بر زنجیره پویش تفاضلی و مبتنی بر امضا بهتر از روش‌های پیشین جلوگیری کند.

کلیدواژه: **امنیت سخت‌افزار، آزمون‌پذیری، حملات مبتنی بر زنجیره پویش.**

۱- مقدمه

طراحی برای آزمون، یک استاندارد صنعتی برای بهبود کنترل‌پذیری و مشاهده‌پذیری درون تراشه‌ها است. این روش، مخصوصاً برای مدارهای دیجیتال با مقیاس بزرگ و سیستم‌های بر روی تراشه که ورودی و خروجی‌ها اطلاعات محدودی درباره عملکرد داخلی مدار می‌دهند، ضروری است. زنجیره پویش، همراه با روش تولید الگوی آزمون خودکار به صورت گسترده مورد استفاده قرار می‌گیرد زیرا می‌تواند به پوشش اشکال بالا، هزینه طراحی پایین، پیاده‌سازی ساده و سرعت آزمون بالا دست پیدا کند. ابزارهای خودکار طراحی الکترونیک (EDA)[1] سودمند همچنین به منظور خودکارکردن پروسه درج زنجیره پویش ایجاد شده‌اند.

تراشه‌های رمزنگاری نیز مانند دیگر تراشه‌ها به درج این زنجیره به منظور تضمین کارکرد صحیح مدار احتیاج دارند، اما احتیاج آزمون به دسترسی به مقادیر داخلی تراشه‌ها به چالش برای تولیدکنندگان این مدارها تبدیل شده است زیرا زنجیره پویش راهی راحت را به منظور استخراج‌کردن اطلاعات حساس داخلی فراهم می‌کند و می‌تواند توسط مهاجمین مورد سوء استفاده قرار گیرد. در واقع هرچه در مدارهای امن باید دسترسی محدود باشد اما برای آزمون یک مدار احتیاج دسترسی به مقادیر داخلی لازم است. در حال حاضر روش‌های مختلف حمله توسط زنجیره پویش برای ربودن اطلاعات حساس داخلی مدارهای رمزنگاری به وجود آمده‌اند. به دنبال آن روش‌هایی نیز برای مقابله با حملات و همچنین محافظت از آزمون‌پذیری زنجیره پویش پیشنهاد شده است. اما روش مقابله مذکور پس از مدتی مورد حمله جدید قرار گرفته و شکسته شده است. علاوه بر این، سربار بالا از نظر مساحت اشغالی و زمان انجام آزمون، پیچیدگی ساخت و غیر قابل سنتز بودن این روش‌ها سبب شده تا همچنان به روشی جدید و کارا برای مقابله با حملات زنجیره پویش احتیاج باشد. در واقع از یک سو باید مطمئن بود که امنیت سیستم از بین نمی‌رود و از طرف دیگر، آزمون‌پذیری سیستم نباید در معرض خطر قرار گیرد. در این مقاله سعی داریم روشی ارائه کنیم تا علاوه بر پوشش خطای مناسب، میزان امنیت قابل قبولی نیز ارائه دهد. برای این منظور، مداری درون تراشه رمزنگاری شده قرار خواهیم داد. وظیفه این مدار، تشخیص مهاجم می‌باشد. سپس در صورت تشخیص وقوع حمله احتمالی توسط مهاجم، با تولید خروجی تصادفی، از حمله مهاجم جلوگیری خواهیم کرد. این روش علاوه بر امنیت بالا، به دلیل نمایش کامل خروجی به مهندس آزمون، پیچیدگی آزمون را افزایش نمی‌دهد.

نتایج شبیه‌سازی نشان می‌دهد روش ما دارای سربار مساحت ۰/۹٪ است و همچنین توان مصرفی ایستا را نیز به میزان ۱٪ اضافه کرده و تقریباً سربار تأخیر ندارد. این در حالی است که از لحاظ امنیتی بهتر از فناوری‌های اخیر عمل می‌کند.

در ادامه این مقاله، ابتدا در بخش ۲ به معرفی مفاهیم مرتبط با موضوع خواهیم پرداخت و با زنجیره پویش و حمله‌های مبتنی بر آن بیشتر آشنا خواهیم شد. پس از آن در بخش ۳، مروری بر روش‌های موجود برای جلوگیری از حملات خواهیم کرد. در بخش ۴، روش پیشنهادی را ارائه می‌کنیم و نهایتاً در بخش ۵ به ارزیابی و مقایسه آن با روش‌های موجود می‌پردازیم.

۲- مفاهیم پایه

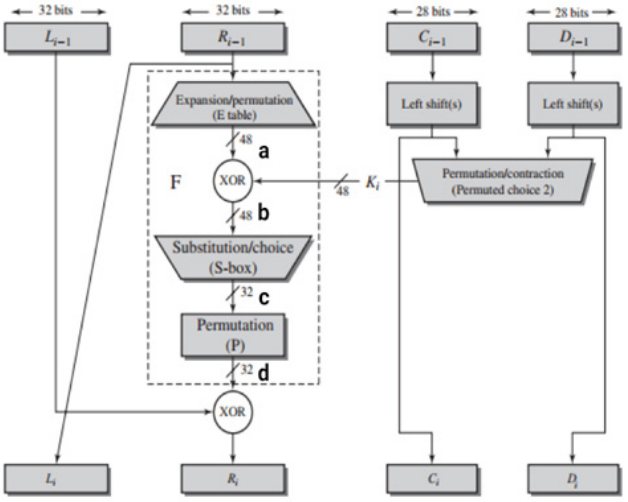
امروزه در طراحی‌های VLSI، تعداد ترانزیستورها بر روی یک تراشه در حال افزایش است. افزایش تعداد ترانزیستورها به معنی افزایش احتمال وقوع خطا بر روی یک تراشه پس از تولید می‌باشد. به منظور تشخیص خرابی، مهندسان آزمون باید روشی سریع و قابل اعتماد برای این طرح‌های بزرگ پیش از آن که تراشه‌ها به دست مشتری برسد پیدا کنند. آزمون سیستم‌های بر تراشه سخت‌تر نیز می‌باشد زیرا مهندسان دسترسی بسیار محدودی به تمام تراشه دارند و به همین دلیل باید روشی داشته باشند که به آنها اجازه کنترل‌پذیری و مشاهده‌پذیری بالای تمام تراشه را بدهد. به همین منظور، طراحی برای آزمون از دو دهه پیش مورد استفاده

این مقاله در تاریخ ۱۰ تیر ماه ۱۳۹۷ دریافت و در تاریخ ۱ اسفند ماه ۱۳۹۷ بازنگری شد.

فاطمه جمالی زواره، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران، (email: f_jamali@comp.iust.ir).

حاکم بیت‌الهی (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران، (email: beitollahi@iust.ac.ir).

1. Electronic Design Automation



شکل ۱: یک دور از الگوریتم DES.

قرار می‌گیرد. طراحی برای آزمون به مهندس آزمون اجازه دسترسی را به نقاطی از تراشه که دور از دسترس است می‌دهد. یکی از مرسوم‌ترین روش‌ها برای آزمون مدارات دیجیتالی، قراردادن زنجیره‌های پویش است که پوشش اشکال بالا را تضمین می‌کند و بنابراین کیفیت محصولات نهایی نیز بالا خواهد بود. آزمون پویش یک روش آزمون قوی می‌باشد که می‌توان وضعیت داخلی مدار را با استفاده از زنجیره‌های پویش کنترل و مشاهده کرد. در واقع، آزمون پویش یک روش طراحی برای آزمون می‌باشد که به مهندسان توانایی کنترل و مشاهده مدار تحت آزمون را می‌دهد که باعث پوشش خطای بسیار بالایی به روشی قابل اعتماد و سریع می‌شود. این روش شامل قراردادن زنجیره‌های پویش به منظور افزایش مشاهده‌پذیری و کنترل‌پذیری گره‌های درون مدار می‌باشد.

به منظور فعال‌کردن آزمون پویش برای طرح تراشه، منطق‌هایی برای آزمون باید در طرح درج شود که به آنها "درج پویش" می‌گویند. درج پویش شامل دو مرحله می‌باشد:

۱) جایگزین‌کردن سلول‌های ساده حافظه مانند فلیپ‌فلاپ‌ها یا لچ‌ها با سلول‌های پویش.

۲) اتصال سلول‌های پویش به یکدیگر به منظور تشکیل یک یا چند زنجیره.

سلول‌های پویش می‌توانند در دو حالت کار کنند: حالت عملکردی که در عملکرد عادی مدار استفاده می‌شود و حالت پویش که اجازه شیفت در زنجیره‌های پویش را می‌دهد.

در صورت وجود سلول‌های پویش که از دو حالت عملکرد و پویش پشتیبانی می‌کنند، آزمون پویش به صورت زیر انجام می‌شود:

۱) با شیفت در زنجیره‌های پویش، طرح زیر آزمون مستقیماً مقداردهی می‌شوند.

۲) سپس به مدت یک تا چند سیکل ساعت، مدار در حالت عملکردی کار می‌کند و خروجی‌ها از زنجیره استخراج می‌شوند.

۳) در آخر، نتایج از طریق زنجیره پویش خوانده می‌شود تا با خروجی‌های مورد انتظار مقایسه شوند و رفتار طرح زیر آزمون بررسی شود.

همان طور که اشاره شد فعال‌کردن زنجیره پویش، کنترل‌پذیری و مشاهده‌پذیری سلول‌های حافظه را به طور کامل فراهم می‌کند. از نقطه نظر امنیت، آزمون زنجیره پویش مانند یک شمشیر دولبه می‌ماند. از یک سو یک روش آزمون قوی می‌باشد اما از سوی دیگر به همان قدر می‌توان از آن به عنوان یک ابزار قوی برای حمله استفاده کرد. حمله زنجیره پویش یکی از حملات معروف کانال جانبی می‌باشد که در آن مهاجم بدون احتیاج به از بین بردن بسته‌بندی تراشه برای قراردادن پروب، می‌تواند به داده‌های حساس درون تراشه دسترسی پیدا کند. این حملات احتیاجی به تجهیزات گران‌قیمت ندارد و نرخ پیروزی آن، حتی اگر هسته رمزنگاری در یک سیستم پیچیده قرار گرفته باشد بالا است.

حملات زنجیره پویش

حملات زنجیره پویش در سامانه‌های دیجیتال به صورت کلی به دو دسته حملات زنجیره پویش تفاضلی و حملات زنجیره پویش امضا تقسیم می‌شوند. حملات زنجیره پویش تفاضلی، حملاتی هستند که به نگهداری مقادیر ثبات‌ها در هنگام تغییر وضعیت تراشه از حالت عملکردی به حالت آزمون نیازمندند. برای یک تراشه، مهاجم می‌تواند رمز را در حالت عملکرد مدار با ورودی دلخواه و برای چند سیکل اجرا کند و سپس با تغییر وضعیت به حالت آزمون، محتویات ثبات‌های داخلی را مشاهده کند. این ثبات‌ها نتایج میانی اجرای رمز را نگهداری می‌کنند. بدین صورت، مهاجم می‌تواند به نتایج میانی دست پیدا کند و با تحلیل تفاضلی، کلید امن را به دست آورد. این دسته از حملات بر روی الگوریتم‌های رمزنگاری مختلفی از جمله الگوریتم استاندارد رمزنگاری داده [۱] و الگوریتم استاندارد رمزنگاری پیشرفته [۲] صورت گرفته است. دسته دیگر حملات زنجیره پویش، حملات مبتنی بر امضا می‌باشند که در این حملات، هدف به چندین بخش تقسیم می‌شود و سپس با استفاده از آزمون و خطا، هر بخش شکسته می‌گردد. در [۳] حمله امضا به رمزنگاری DES نشان داده شده است. این دسته از حملات بر روی رمزنگاری‌های استاندارد رمزنگاری داده [۴]، رمزنگاری خم بیضوی [۵] و RSA [۶] انجام شده است.

اولین حمله مبتنی بر زنجیره پویش، حمله تفاضلی [۱] روی الگوریتم DES می‌باشد. در این حمله، مهاجم پس از استخراج ساختار زنجیره پویش، تنها با استفاده از ۹ جفت متن ساده می‌تواند کلید رمزنگاری را استخراج کند.

حمله در دو مرحله صورت می‌گیرد. در مرحله اول، ساختار زنجیره پویش مشخص می‌شود و در مرحله دوم کلیدهای دور DES و کلید کاربر بازیابی می‌گردد. شکل ۱ ساختار داخلی یک دور از الگوریتم DES را نشان می‌دهد. همان طور که در شکل مشخص است، با نیمه راست و چپ هر ۶۴ بیت، مانند یک ۳۲ بیت مستقل برخورد می‌شود. پردازش کلی در هر دور می‌تواند در فرمول‌های زیر خلاصه شود

$$Li = Ri-1$$
$$Ri = Li-1 \oplus F(Ri-1, Ki)$$

برای شناسایی ساختار، فلیپ‌فلاپ‌ها را به عنوان ورودی قرار می‌دهیم و ثبات‌های L و R در زنجیره پویش هستند. پایه خروجی پویش TDO یک جریان بیت سریال را تولید می‌کند که ارتباط بین بیت‌های درون ثبات‌ها و بیت‌های جریان بیت خروجی مستقیماً مشخص نیست. با تغییر مدار DES بین حالت عادی و حالت آزمون، می‌توانیم ساختار زنجیره پویش را به صورت زیر مشخص کنیم:

۱) بازنشانی تراشه DES و اجرای آن در حالت عادی برای یک سیکل ساعت تا کلمات متن ساده مشخص درون ثبات‌های ورودی بارگذاری شوند.

۲) تغییر به حالت آزمون و خارج‌کردن جریان بیت شماره ۱.

۳) بازگشت به حالت عادی و اجرای یک سیکل ساعت به منظور بارگذاری متن ساده در ثبات‌های L و R.

۴) تغییر به حالت آزمون و خارج‌کردن جریان بیت شماره ۲.
۵) تکرار قدم‌های ۱ تا ۴ با استفاده از متن ساده که تنها در یک بیت با متن ساده اول تفاوت دارد و ذخیره جریان بیت شماره ۳ و ۴.
۶) تکرار ۳۲ بار مرحله ۵ با تغییر یک بیت در R که موجب تغییر فلیپ‌فلاپ متناظر آن در خروجی خواهد شد و ۳۲ بار دیگر با تغییر یک بیت در L که با حذف فلیپ‌فلاپ‌های R، باقی فلیپ‌فلاپ‌ها مشخص می‌شوند.

اکنون جایگاه ثبات‌های L و R در زنجیره پویش مشخص شده است و می‌توانیم الگوریتم DES را با واردکردن ۳ متن ساده مشخص و تحلیل الگوریتم بشکنیم.

با توجه به شکل ۱، یک دور DES به صورت زیر می‌باشد

$$L_1 = R_0 \quad (1)$$

$$R_1 = L_0 \oplus d \quad (2)$$

$$d = permutation(c) \quad (3)$$

$$a = Expand(R_1) \quad (4)$$

$$b = a \oplus k_1 \quad (5)$$

$$c = s - box(b) \quad (6)$$

با بارگذاری کلمات متن ساده (L_0 و R_0) و اجرای ۳ سیکل ساعت و سپس تغییر به حالت آزمون و خارج‌کردن جریان بیت، R_1 و L_1 مشخص می‌شوند. با استفاده از مقادیر R_1 و L_0 و با جایگذاری آنها در (۲)، d و از (۳) نیز c مشخص می‌شود. از (۴)، a محاسبه می‌گردد. از (۵) اگر b را بدانیم، می‌توانیم کلید دور K_1 را به دست آوریم. b را به راحتی می‌توان از (۶) به دست آورد جایی که b ورودی $s-box$ است و c (خروجی $s-box$) را که قبلاً به دست آورده‌ایم.

در این حمله برای شناسایی ساختار تمام زنجیره پویش به ۳۸۴۰۷ سیکل ساعت (۱۹۲×۱۹۹+۱۹۹) برای قراردادن تمام ۱۹۲ فلیپ‌فلاپ در ثبات متن ساده ورودی و ثبات‌های L و R احتیاج است. همچنین ۳۹۹ سیکل ساعت شامل ۲ سیکل برای عملکرد عادی + ۱۹۸ سیکل برای عمل پویش + ۱ سیکل برای عملکرد عادی + ۱۹۸ سیکل برای عمل پویش، برای هر ورودی متن ساده و دسترسی آن به R_0، L_0، R_1 و L_1 احتیاج است. در واقع به ۱۱۹۷ سیکل ساعت (۳×۳۹۹) برای بازیابی کلید دور K_1 احتیاج داریم. به روش مشابه، ۱۱۸۵ سیکل ساعت برای بازیابی کلیدهای دور K_2 و K_3 احتیاج است. در کل، ۴۱۹۷۴ سیکل ساعت (۳۸۴۰۷+۱۱۹۷+۲×۱۱۸۵) برای بازیابی کلید کاربر نیاز است. این حمله تنها به ۷۳ (متن ساده، متن رمزشده) احتیاج دارد که در مقایسه با 2^{64} (متن ساده، متن رمز) که برای حمله به DES لازم است مقدار کمی می‌باشد. در حقیقت، زنجیره پویش باعث می‌شود تا هزینه حمله و استخراج کلید بسیار کاهش پیدا کند. از طرف دیگر به دو دلیل نمی‌توان آزمون تراشه‌های رمزنگاری را نادیده گرفت: از یک طرف، خطاهای آزمون‌نشده احتمالی ممکن است امنیت سیستم را به خطر بیندازد و همچنین از طرف دیگر، مانند هر تراشه دیگری، آزمون، کیفیت محصول را تضمین می‌کند. برای حل مشکل امنیت در زنجیره پویش، روش‌هایی برای جلوگیری از حمله مهاجمین از طریق زنجیره پویش پیشنهاد شده است. در بخش ۳ مروری بر کارهایی که در این زمینه صورت گرفته است خواهیم داشت.

۳- مروری بر کارهای گذشته

در حالی که آزمون‌پذیری یک تراشه به افزایش مشاهده‌پذیری و کنترل‌پذیری آن احتیاج دارد، تراشه‌های امن به صورت معکوس طراحی می‌شوند و به منظور محافظت در برابر کاربران مخرب، تراشه باید تا حد ممکن که قابل استفاده است، کمترین اطلاعات را فاش کند. در نتیجه، استفاده از روش‌های معمول آزمون در هنگام طراحی تراشه‌های رمزنگاری سطح امنیت فراهم‌شده توسط تراشه را به صورت جدی پایین می‌آورد. به عبارت دیگر از یک سو باید مطمئن بود که امنیت سیستم از بین نمی‌رود و از طرف دیگر، آزمون‌پذیری سیستم نباید در معرض خطر قرار گیرد.

یکی از پیشنهادهای اولیه برای رفع نگرانی، سوزاندن فیوزهای مدار آزمون پس از ساخت می‌باشد [۷] که البته باعث از بین رفتن امکان آزمون عملکرد تراشه‌ها پس از ساخت می‌شود. پیشنهاد دیگر استفاده از روش‌های دیگر آزمون به جای زنجیره پویش و یا ترکیب آن با روش‌های دیگر است. در [۸] نویسنده برای آزمون تراشه ترکیبی از زنجیره پویش با BIST را پیشنهاد کرده است. هرچند این روش‌ها از امنیت بسیار بالایی برخوردار هستند اما کاهش چشم‌گیر پوشش خطا را به دنبال دارند [۹]. همچنین برخی مقالات از ساختارهای پیشرفته DFT یا طراحی BIST به عنوان پوسته محافظ در برابر حملات زنجیره پویش استفاده می‌کنند. مهم‌ترین ساختارهای پیشرفته DFT، response compactor و input decompressor می‌باشند که به عنوان روش‌های جلوگیری از حملات به کار گرفته می‌شوند. در [۹] تا [۱۲] نمونه‌هایی از حملات امضا آورده شده که با وجود ساختارهای پیشرفته DFT صورت گرفته‌اند و این روش محافظت، شکست خورده است [۱۲]. راه حل دیگری که کارهای بسیاری نیز در این زمینه انجام شده است، امن‌سازی زنجیره پویش می‌باشد.

برای امن‌سازی زنجیره پویش و جلوگیری از حمله مهاجم به زنجیره پویش، سه رویکرد وجود دارد [۱۳]:

۱) مسدودسازی ورودی‌های مشکوک: این روش، باعث تأخیر در ورودی مدار و در نتیجه سربار و پایین‌آمدن کارایی زنجیره پویش خواهد شد.
۲) مسدودسازی کلید رمز و جلوگیری از ورود اطلاعات حساس به زنجیره پویش.
۳) مبهم‌سازی خروجی پویش و به هم زدن ترتیب خروجی پویش.

به طور کلی می‌توان روش‌های ارائه‌شده در مقالات را به سه دسته مبهم‌سازی خروجی، مقایسه روی تراشه و پوشش‌گذاری تقسیم کرد. در ادامه به معرفی این دسته‌ها و روش‌های ارائه‌شده برای هر یک می‌پردازیم.

۳-۱ روش‌های مبهم‌سازی

در این روش‌ها همه کاربران می‌توانند از زنجیره پویش استفاده کنند ولی اطلاعات محرمانه برای همه قابل بازیابی نخواهد بود. در این دسته از روش‌ها فرض می‌شود که مهاجم اطلاعات ساختار تراشه را در اختیار ندارد و نمی‌تواند به آن دست پیدا کند. مقالات مختلفی در این زمینه وجود دارد که روش‌هایی به دو صورت ایستا و پویا ارائه کرده‌اند. در روش‌های مبهم‌سازی ایستا، درهم‌سازی خروجی پس از ساخت همیشه به یک شکل می‌باشد، مانند مدارهای ترکیبی. البته این روش‌ها در برابر حملات امضا که به ساختار زنجیره پویش احتیاج ندارند مقاوم نیستند. دسته دیگر در هر بار اجرا ترتیب متفاوتی دارند و به آنها روش‌های پویا گفته می‌شود. این روش‌ها برای پویاکردن تغییرات، معمولاً وابسته به وضعیت داخلی مدار هستند.

نویسنده در [۱۴] در مسیر زنجیره پویش، یک معکوس‌کننده قرار داده

است. اضافه‌کردن معکوس‌کننده‌ها تأثیری بر روی عملکرد عادی مدار نخواهد داشت اما در حالت آزمون به دلیل عدم اطلاع مهاجم از مکان آنها، نمی‌تواند از خروجی زنجیره پویش برای بازیابی کلید استفاده کند. با این حال در [۱۵] امنیت این روش شکسته شده است. در [۱۶] نویسنده برای مبهم‌کردن خروجی مقدار دو ثبات پشت سر هم را XOR می‌کند تا مهاجم نتواند از آنها استفاده کند. با این حال، مبهم‌سازی به صورت ایستا در برابر حملات امضا آسیب‌پذیر است. برای پویاکردن تغییرات، [۱۷] برخی از SFFها را با سلول SDSFF جایگزین می‌کند. سلول SDSFF در خروجی SFFها یک latch قرار می‌دهد تا مقدار قبلی آن را نگهداری کند و یک XOR که خروجی latch و SFF را XOR می‌کند. مقاله [۱۸] زنجیره پویش را به چند زیرزنجیره تقسیم می‌کند و با استفاده از Enable زیرزنجیره‌ها ترتیب نمایش آنها در خروجی را مشخص می‌کند. مدار کنترل Enableها در این روش، مشخص‌کننده میزان امنیت و آزمون‌پذیری زنجیره پویش خواهد بود.

۳-۲ روش‌های مقایسه روی تراشه

در [۷] و [۱۹] روش مقایسه روی تراشه ارائه شده است. در این روش، خروجی زنجیره پویش به کاربر نشان داده نمی‌شود. هر چند این روش‌ها از نظر امنیت بسیار بالا می‌باشند اما تشخیص مکان اشکال در مدار، با این روش بسیار سخت خواهد بود. در [۷] مداری طراحی شده که خروجی مورد نظر را از کاربر دریافت می‌کند و با خروجی تولیدشده توسط زنجیره پویش مقایسه می‌نماید، سپس دو خروجی را با هم مقایسه می‌کند و اعلام می‌کند که آیا تراشه سالم است یا خیر. با این روش، خروجی زنجیره پویش به دست مهاجم نخواهد رسید تا بتواند از آن جهت سرقت کلید رمزنگاری استفاده کند. نویسنده در [۷] روش [۲۰] را بهبود داده است. در این مقاله به جای مقایسه مستقیم خروجی با خروجی دلخواه، از خروجی طلایی که توسط LFSR[1] تولید می‌شود استفاده می‌کند. کاربران نیز به جای واردکردن خروجی اصلی باید خروجی طلایی را که توسط کارخانه تهیه شده است به دستگاه وارد کنند تا دستگاه این دو مقدار را با هم مقایسه کند و صحیح یا غلط‌بودن عملکرد مدار را اطلاع دهد. در این روش علاوه بر کم‌کردن حجم داده و بهبود زمان آزمون، کاربر نیز به دلیل در اختیار نداشتن مقادیر زنجیره پویش نمی‌تواند به کلید دسترسی پیدا کند، با این حال آزمون‌پذیری این روش از [۲۰] کمتر است.

۳-۳ روش‌های پوشش‌گذاری

این دسته از روش‌ها، معمولاً از کلید برای تشخیص هویت استفاده می‌کنند و دو وضعیت امن و ناامن دارند. از ویژگی‌های دیگر این دسته ایجاد برخی محدودیت‌ها به منظور غیر قابل استفاده کردن مسیر پویش برای مهاجمین است. چگونگی ایجاد کلید و محافظت از آن، سربار مساحت و زمان بالای آزمون از چالش‌های این دسته از روش‌ها می‌باشد. مرجع [۱۹] زنجیره پویش را به چند زیرزنجیره تقسیم کرده و توسط یک تسهیم‌کننده، ترتیب اجرای زیرزنجیره‌ها مشخص می‌شود. هنگام شروع به کار زنجیره پویش، کلید آزمون از کاربر دریافت می‌شود. اگر کلید غلط باشد، خط انتخاب تسهیم‌کننده از تابع تصادفی تغذیه می‌شود و در غیر این صورت، زیرزنجیره‌ها به ترتیب فعال خواهند شد. هرچقدر تعداد زیرزنجیره‌ها بیشتر باشد، امنیت بالاتر خواهد بود اما به دنبال آن پیچیدگی مدار و همچنین سربار مساحت هم زیاد می‌شود. مقاله [۲۱] روش MKR را ارائه کرده است. هدف این روش این است که داده‌های به دست آمده از زنجیره پویش، شامل اطلاعات حساس و مربوط به کلید رمزنگاری نباشد. برای این منظور، دو وضعیت متفاوت برای تراشه تعریف شده است. در وضعیت امن که تراشه با کلید رمزنگاری اصلی کار می‌کند، مدار فقط در حالت عملکردی خواهد بود. برای کارکرد مدار در حالت آزمون، مدار از کلید دیگری برای رمزنگاری استفاده می‌کند تا نتوان از داده‌های زنجیره پویش کلید اصلی را بازیابی کرد. هنگام تغییر از وضعیت امن به ناامن، زنجیره پویش فعال شده و مدار بازنشانی می‌شود تا مقادیر ذخیره‌شده در زنجیره پاک شوند و مهاجم نتواند از آنها سوء استفاده کند. البته در این روش به دلیل ذخیره دو کلید، سربار مساحت بالایی دارد و همچنین در برخی از مدارها کلید در حافظه جداگانه وجود ندارد و درون خود مدار می‌باشد که در این صورت نمی‌توان کلید دیگری را جایگزین کلید اصلی کرد [۲۲]. در [۲۳] روش قفل و کلید ارائه شده است. این روش دو وضعیت مختلف را برای تراشه تعریف می‌کند. اول وضعیت امن که کاربر تأییدشده می‌تواند آزمون را انجام دهد و وضعیت ناامن که خروجی غلط به مهاجم داده خواهد شد.

در این روش، زنجیره پویش به چند زیرزنجیره تقسیم می‌شود و یک واحد TSC نیز به مدار اضافه می‌شود. واحد TSC شامل ۴ بخش FSM، Comparator، LFSR و Decoder می‌باشد. پس از فعال‌شدن TC کاربر باید کلید k بیتی را وارد کند. پس از k کلاک بخش Test Key Comparator جواب مقایسه را به FSM می‌دهد. اگر کلید درست وارد شده باشد، مدار وارد وضعیت امن می‌شود و در غیر این صورت مدار به وضعیت ناامن خواهد رفت. Decoder که وظیفه تغذیه Enableها را بر عهده دارد توسط LFSR مقداردهی می‌شود. LFSR به گونه‌ای برنامه‌ریزی شده که اگر وضعیت مدار امن باشد، هر زیرزنجیره تنها یک بار فعال شود اما اگر وضعیت ناامن بود، ترتیب فعال‌شدن زیرزنجیره‌ها و تعداد فعال‌شدن آنها مشخص نیست. با بیشترکردن تعداد زیرزنجیره‌ها، امنیت بیشتر می‌شود اما باعث بزرگ‌ترشدن تسهیم‌کننده و پیچیده‌ترشدن LFSR و در نتیجه سربار مساحت خواهد شد. در [۲۲] برای کم‌کردن سربار مساحت روش [۲۳]، روش جدیدی ارائه شده است. برای شناسایی کاربر، مدار m کلید n بیتی را تولید می‌کند و در صورت غیر صحیح بودن کلیدهای ورودی توسط کاربر، scan out غیر فعال شده و مقدار صفر را تولید می‌کند. در [۱۳] به جای مقایسه کلید، کلید ورودی، فعال‌کننده برخی از عناصر زنجیره پویش خواهد بود. یک شیفت رجیستر به مدار اضافه می‌شود که خروجی‌های آنها به خط انتخاب عناصر زنجیره وارد می‌شود و در صورتی که مقدار واردشده غلط باشد، عنصر زنجیره پویش فعال نخواهد شد و در نتیجه داده‌های قبل از آن عنصر به خروجی منتقل نمی‌شود.

با مقایسه این سه دسته مشاهده می‌کنیم که امنیت و آزمون‌پذیری در تقابل با یکدیگر قرار دارند و همچنین سربار طرح یکی از چالش‌های ارائه راه حل می‌باشد.

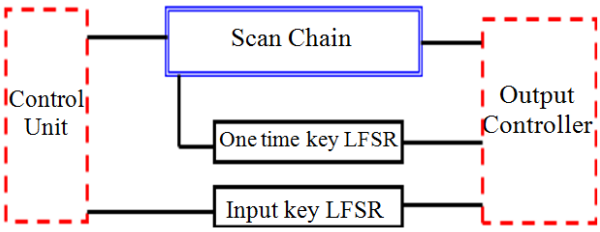
٤- روش پیشنهادی

در روش پیشنهادی برای تشخیص حمله، از کاربر کلیدی دریافت می‌شود که در صورت صحیح‌بودن کلید، کاربر خروجی صحیح دریافت می‌کند و در غیر این صورت مدار، خروجی تصادفی تولید می‌کند.

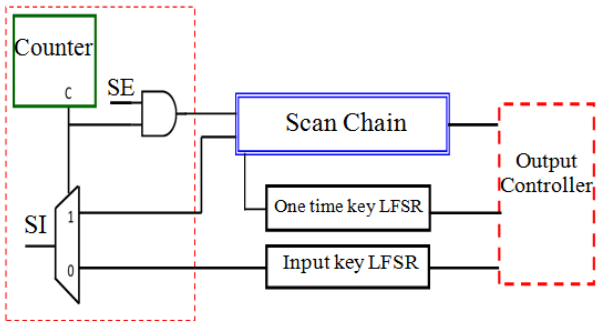
همان طور که در شکل ۲ مشاهده می‌کنید طرح، دارای یک واحد کنترل ورودی، یک واحد کنترل خروجی و دو LFSR می‌باشد.

واحد کنترل، شامل یک شمارنده kبیتی است که در آن k، طول

1. Linear Feedback Shift Register



شکل ۲: معماری کلی طرح پیشنهادی.



شکل ۳: مدار کنترل ورودی.

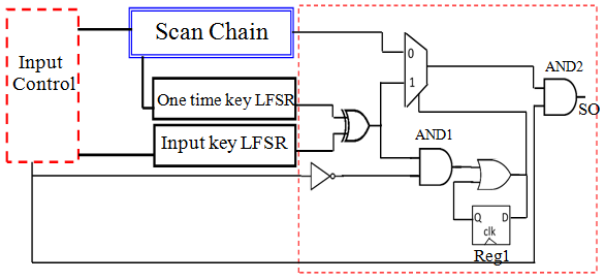
کلید خواهد بود. شمارنده در ابتدا توسط سیگنال SE بازنشانی می‌شود و تا هنگامی که کلید از طریق SI وارد input key LFSR می‌شود، شمارش را انجام می‌دهد. پس از k تا سیکل ساعت، خط c شمارنده، یک می‌شود و تا هنگامی که دوباره بازنشانی نشده است مقدار آن یک باقی می‌ماند. سیگنال c وارد تسهیم‌کننده می‌شود و از این طریق SI را کنترل می‌کند. در صورتی که $c=0$ باشد، مدار در مرحله دریافت و مقایسه کلید خواهد بود و کاربر از طریق SI کلید را وارد LFSR می‌کند. پس از یک‌شدن سیگنال c، مدار وارد مرحله آزمون می‌شود و در نتیجه SI وارد زنجیره پویش می‌شود. مدار کنترل ورودی در شکل ۳ نشان داده شده است.

برای تغییر حالت زنجیره از حالت عملکردی به آزمون و برعکس، از andکردن سیگنال SE با سیگنال c استفاده می‌شود تا در طول مرحله دریافت و مقایسه کلید، زنجیره پویش در حالت عملکردی خود باشد.

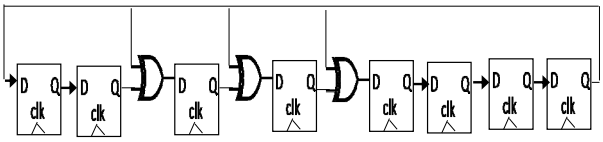
مدار کنترل خروجی در شکل ۴ نشان داده شده است. تا هنگامی که شمارنده در حال شمارش باشد، این مدار دو کلید را مقایسه و نتیجه را در ثبات Reg۱ ذخیره می‌کند. مقدار ثبات برای انتخاب خروجی از تسهیم‌کننده استفاده می‌کند.

همان طور که بیان شد در مرحله دریافت کلید، مقدار c شمارنده صفر می‌باشد و بنابراین مقدار گیت NOT، یک خواهد بود و گیت AND۱ مقدار صفر مقایسه را به گیت OR می‌رساند. با هر سیکل یک بیت از دو LFSR توسط گیت XOR با هم مقایسه می‌شوند. اگر در طول مقایسه دو بیتی وجود داشته باشند که برابر نباشند (کلید غلط وارد شده است)، مقدار XOR یک خواهد شد و به دنبال آن گیت OR نیز یک می‌شود. مقدار یک در ثبات ذخیره می‌شود و تا هنگامی که ثبات بازنشانی نشود یک باقی خواهد ماند اما اگر تمام بیت‌ها صحیح باشند، مقدار ثبات تغییری نمی‌کند و صفر می‌ماند. پس از اتمام مقایسه کلید، مقدار گیت NOT، صفر می‌شود و ارتباط گیت XOR با ثبات قطع می‌شود.

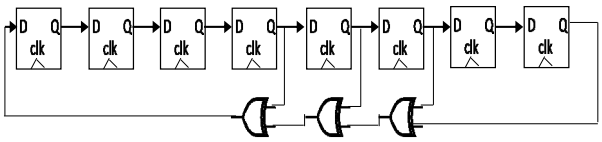
تا زمانی که مدار در مرحله مقایسه کلید است، خروجی AND۲ صفر خواهد بود زیرا مقدار ورودی c آن صفر می‌باشد. تسهیم‌کننده با ورود به مرحله آزمون، خروجی را انتخاب می‌کند. در صورتی که حمله صورت نگرفته باشد، خروجی زنجیره پویش به مهندس آزمون داده می‌شود اما در



شکل ۴: مدار کنترل خروجی.



شکل ۵: مدار LFSR ۸بیتی گالویس.

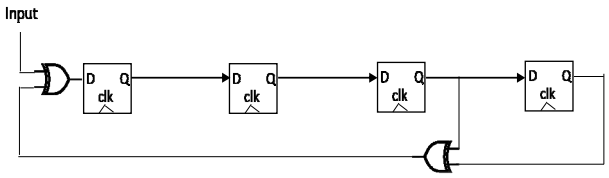


شکل ۶: مدار LFSR ۸بیتی از نوع فیبوناچی.

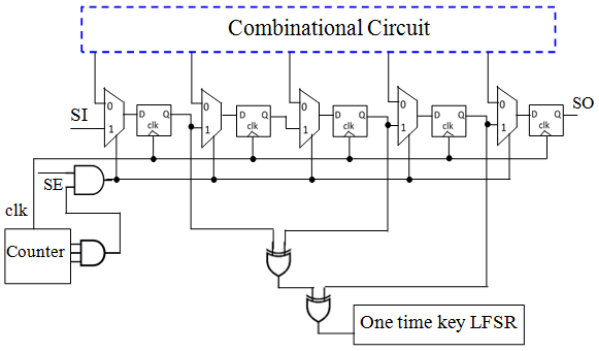
صورت بروز حمله، خروجی تصادفی که توسط LFSRها تولید می‌شود، به مهاجم نمایش داده می‌شود. به این صورت جلوی دستیابی مهاجم به اطلاعات حساس درون تراشه گرفته خواهد شد.

LFSR واحد تولید خروجی تصادفی است تا حمله‌کننده را گمراه کند. این واحد ساختاری شبیه به شیفت رجیستر دارد که در آن تابع خطی از وضعیت قبلی ثبات‌ها به عنوان ورودی اولین ثبات استفاده می‌شود. تابع خطی معمولاً XOR می‌باشد و از LFSRها برای تولید اعداد شبه‌تصادفی استفاده می‌شود. بسته به تعداد، نحوه قرارگرفتن XORها و نحوه مقداردهی اولیه شیفت رجیستر، توابع متنوعی از LFSR برای تولید خروجی تصادفی موجود است. شکل ۵ مدار LFSR ۸بیتی از نوع گالویس را برای تولید اعداد تصادفی نشان می‌دهد در حالی که شکل ۶ مدار LFSR ۸بیتی از نوع فیبوناچی را برای تولید اعداد تصادفی نشان می‌دهد. برای یک چندجمله‌ای داده‌شده، خروجی تولیدشده توسط هر دو مدار یکسان خواهد بود اما مدار نوع گالویس از لحاظ سرعت بر مدار نوع فیبوناچی برتری دارد. دلیل این برتری در این واقعیت نهفته است که در مدار فیبوناچی یا چندین XOR به شکل آبشار پشت سر هم بسته شده‌اند یا این که از XORهای با چندین ورودی باید استفاده شود، در حالی که در مدل نوع گالویس از XORهای دو ورودی بدون بستن آبشاری پشت سر هم استفاده شده است. در نتیجه تأخیر انتشار در مدار گالویس کمتر از مدار فیبوناچی است و مدار گالویس با فرکانس بالاتری می‌تواند کار کند.

از آنجایی که هدف ما در این مقاله به کارگیری یک مدل LFSR برای تولید اعداد تصادفی است، انتخاب بین هر دو مدل فیبوناچی و گالویس آزاد است و در بحث امنیت تراشه نیز سرعت تولید اعداد تصادفی حاد نیست و ما می‌توانیم از هر دو مدل استفاده کنیم. شکل ۷، مدار LFSR از نوع فیبوناچی ۴بیتی را نشان می‌دهد که ما در این مقاله برای تولید اعداد تصادفی استفاده کرده‌ایم. این LFSR علاوه بر وضعیت قبلی خود، از خط ورودی هم تغذیه می‌شود تا از تولید الگوهای تکراری جلوگیری کند. در هنگام تغییر حالت تراشه از عملکردی به آزمون، مقادیر تمام ثبات‌های LFSR صفر می‌شود. پس از آن، one time key LFSR توسط زنجیره پویش و input key LFSR توسط ورودی SI مقداردهی می‌شوند.



شکل ۷: مدار LFSR استفاده‌شده در مقاله.



شکل ۸: مدار تولید کلید.

نحوه تولید کلید

برای تولید کلید ابتدا در حالت عملکرد مدار، آزمون‌گر ورودی دلخواه را وارد می‌کند. سپس با توجه به اطلاع آزمون‌گر از کلید رمزنگاری، باید از مقادیر زنجیره پویش اطلاع داشته باشد و در نتیجه با محاسبه مقدار XOR می‌تواند کلید تولیدشده را محاسبه کند. مدار تولید کلید در شکل ۸ مشاهده می‌شود.

فرض می‌کنیم کلید مدار مقدار f۱۵۷۱c۹۴۷d۹e۸۵۹۰ باشد و آزمون‌گر ورودی ۰۲۴۶۸aceeca۸۶۴۲۰ را وارد کند. طبق شکل ۶ مقادیر سه بیت از زنجیره پویش XOR شده و به صورت سریال وارد LFSR تولید کلید می‌شوند. اگر فرض کنیم این سه بیت، ۳ بیت آخر باشند، مقدار کلید به صورت زیر محاسبه می‌شود:

- در سیکل اول مقدار زنجیره a۰۰۵a۰۰۳cf۰۳c۰f۵ می‌باشد که بنابراین مقادیر سه بیت ۰۱۰ خواهد بود و مقدار XOR آن ۱ می‌شود.
- در سیکل دوم زنجیره مقدار cf۰۳c۰fbad۲۲۸۴۵۳ را خواهد داشت که XOR سه بیت آخر آن نیز ۱ خواهد شد.
- به همین ترتیب سیکل‌های بعدی نیز محاسبه خواهند شد.

اگر فرض کنیم کلید ما ۴بیتی باشد مقدار آن برابر با ۱۰۱۱ خواهد بود. در این روش، کلید در هر دور اجرا یکسان نیست و بسته به ورودی‌های PI می‌تواند تغییر کند. از طرف دیگر هرچه طول کلید بیشتر باشد، احتمال صحیح واردشدن آن توسط مهاجم کمتر خواهد بود.

۵- شبیه‌سازی و ارزیابی

در شبیه‌سازی از الگوریتم تکرارشونده رمزنگاری DES که در شکل ۱ نشان داده شده است استفاده کرده‌ایم. با استفاده از Synopsys Design Compiler رمزنگاری را سنتز کرده و با استفاده از Synopsys DFT Compiler زنجیره پویش را به آن اضافه کرده‌ایم. در ادامه، طرح پیشنهادی را به طرح اضافه کردیم و با ابزار Synopsys Design Compiler طرح را سنتز نمودیم. در آخر با استفاده از ابزار سنتز، مساحت، توان و زمان هر طرح را به دست آوردیم. شبیه‌سازی برای کلیدهای متفاوت با طول ۶۴، ۱۲۸ و ۲۵۶بیتی انجام شده است.

جدول ۱: مساحت طرح پیشنهادی.

طرح	مساحت (um^2)	درصد مساحت
رمزنگاری بدون زنجیره پویش	۲۵۱۳۳۶٫۱۳۵	۸۶٫۵۶٪
رمزنگاری با زنجیره پویش عادی	۲۸۷۷۷۶٫۸۴۹	۱۲٫۵۵٪
رمزنگاری با زنجیره پویش امن	۲۹۰۴۵۱٫۲۷۴	۰٫۸۸٪

جدول ۲: سربار مساحت کارهای پیشین.

روش پیشنهادی	MKR [۲]	قفل و کلید [۲۳]	VIM [۲۲]	DOSD [۱۳]	Flipped Scan [۱۴]
سربار مساحت	۱٫۳۲٪	۶٫۶٪	٪۱	۲٫۲۸٪	۰٫۱۳٪

جدول ۳: تأثیر طول کلید بر مساحت.

شمارنده	تعداد درگاه‌ها	تعداد سلول‌ها	تعداد نت‌ها	مساحت
۶۴بیتی	۱۹۷	۱۴۸۸۹	۱۵۰۱۵	۲۹۰۱۹۱٫۸۱۵
۱۲۸بیتی	۱۹۷	۱۴۸۹۵	۱۵۰۲۰	۲۹۰۳۴۴٫۸۳۰
۲۵۶بیتی	۱۹۷	۱۴۸۹۹	۱۵۰۲۵	۲۹۰۴۵۱٫۲۷۴

جدول ۴: مصرف توان طرح پیشنهادی.

طرح	توان پویا	توان ایستا
رمزنگاری بدون زنجیره پویش	۵٫۵۳۲۹ mW	۹۳۶٫۲۱۴۱ nW
رمزنگاری با زنجیره پویش عادی	۷٫۵۱۱۰ mW	۱٫۰۸۲۵ uW
رمزنگاری با زنجیره پویش امن	۶٫۱۴۷۱ mW	۱٫۰۹۳۴ uW

در جدول ۱ مساحت طرح بدون زنجیره پویش، طرح همراه با زنجیره پویش عادی و ناامن و طرح با زنجیره پویش امن نشان داده شده است. مقدار درصد بیان شده، مساحت هر قسمت نسبت به مساحت کل تراشه امن می‌باشد.

همان طور که مشاهده می‌کنید مساحت اضافه‌شده توسط طرح پیشنهادی بسیار کم و تنها ۰٫۹٪ می‌باشد که نسبت به سربار مساحت زنجیره پویش، ۱۲٫۵٪ ناچیز است.

در جدول ۲ درصد سربار مساحت کارهای پیشین نشان داده شده است. همان طور که مشاهده می‌کنید سربار روش پیشنهادی نسبت به روش‌های مبهم‌سازی خروجی [۱۴] زیاد می‌باشد اما در میان روش‌های پوشش‌گذاری دارای سربار مساحت کم است.

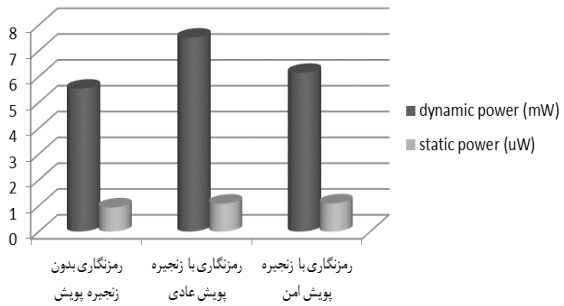
جدول ۳ نیز مساحت طرح بر اساس تغییر طول کلید را نشان می‌دهند. درگاه‌ها شامل ورودی‌ها و خروجی‌های مدار، ورودی‌ها و خروجی زنجیره پویش است که همان طور که در جدول ۳ مشخص است، طرح پیشنهادی درگاهی به طرح اضافه نخواهد کرد زیرا تمام سیگنال‌های کنترل و ورودی‌ها از طریق همان پایه‌های زنجیره پویش مقداردهی می‌شوند و نیازی به درگاه اضافه نمی‌باشد.

با افزایش طول کلید، شمارنده بزرگ‌تری لازم است و در نتیجه به تعداد سلول‌ها و تعداد نت‌ها، اضافه می‌شود و در نتیجه آن، مساحت طرح افزایش خواهد یافت.

توان‌های پویا و ایستای طرح پیشنهادی در مقایسه با زنجیره پویش عادی و طرح بدون زنجیره پویش در جدول ۴ نشان داده شده است. توان پویا شامل توان ناشی از سوییچ نت‌ها و توان مصرفی سلول‌ها می‌باشد. توان ایستا نیز ناشی از توان نشتی ترانزیستورها است. در این جدول برای طرح پیشنهادی ما طول کلید ۲۵۶بیتی انتخاب شده است.

در شکل ۹ نمودار تغییرات توان نشان داده شده است. همان طور که

توان مصرفی طرح پیشنهادی



شکل ۹: توان مصرفی (استاتیک و دینامیک) طرح پیشنهادی.

مشاهده می‌کنید در طرح پیشنهادی، توان پویا تقریباً بدون تغییر باقی می‌ماند. به عبارت دیگر با اضافه‌شدن مدار کشف و مقابله با حمله، به علت سربار کم تغییری در توان پویا نسبت به حالتی که مدار اضافه نشده است ایجاد نمی‌کند. اما به دلیل افزایش تعداد سلول‌ها و ترانزیستورها توان ایستای بیشتری نسبت به اصل مدار مصرف می‌شود. اما چیزی که حایز اهمیت است آن است که مقدار توان ایستا در مقابل توان مصرفی پویا ناچیز می‌باشد.

در جدول ۵ نیز تأثیر طول کلید بر مقدار توان مصرفی نشان داده شده است. کلید با طول بزرگ‌تر، طرح پیشنهادی ما را موفق‌تر در مقابله حمله قرار می‌دهد اما با نسبت بسیار کمی، مصرف توان ایستا و پویا را افزایش می‌دهد. همچنان که در جدول ۵ دیده می‌شود افزایش طول کلید از ۶۴ بیت به ۲۵۶ بیت، صرفاً مصرف توان پویا و استاتیک را به ترتیب ۰٫۰۵٪ و ۰٫۱۷٪ زیاد خواهد کرد که بسیار ناچیز است. این نشان می‌دهد که در طرح پیشنهادی به راحتی می‌توان طول کلید را بدون در نظر گرفتن اثرات جانبی افزایش داد.

همان طور که در بخش ۴ توضیح داده شد، پس از ورود به حالت آزمون، کاربر باید ابتدا کلید را وارد کند. اگر کلید k بیت باشد، واردکردن کلید، k سیکل ساعت نسبت به زنجیره پویش عادی بیشتر زمان می‌برد. البته طول کلید نسبت به طول الگوهای پویش و تعداد امروزی آنها ناچیز می‌باشد.

به صورت کلی، هر چقدر به امنیت بیشتری نیاز باشد، می‌توان کلید را بزرگ‌تر در نظر گرفت، هر چند این کار سبب سربار بیشتر از نظر مساحت، توان و همچنین زمان خواهد شد که بسته به شرایط و نیاز می‌توان بین امنیت و سربار به مصالحه رسید.

۵-۱ تحلیل امنیت

همان طور که در بخش‌های پیشین گفته شد، هدف از ایجاد تغییر در زنجیره پویش ساده، افزایش امنیت آن در برابر حملات کانال جانبی می‌باشد. در طرح پیشنهادی این مقاله، با تشخیص حمله از وقوع آن پیشگیری می‌شود و همچنین امنیت این طرح می‌تواند با تغییر طول کلید به میزان دلخواه تغییر کند.

در صورت اطلاع کاربران از روش امن‌سازی زنجیره پویش، مهاجم اطلاعی راجع به مکان بیت‌هایی از زنجیره پویش که کلید را تولید می‌کنند، ندارد و حتی اگر بتواند با استفاده از blueprint مکان آنها را به دست آورد، مقادیر این بیت‌ها را نمی‌داند تا بتواند کلید را محاسبه کند. از طرف دیگر اگر مهاجم بخواهد به صورت تصادفی کلید را وارد نماید، چون هر بیت به صورت مستقل می‌تواند صفر یا یک باشد و اگر طول کلید را n بیت در نظر بگیریم، 2^n انتخاب خواهد داشت. در نتیجه، احتمال

جدول ۵: تأثیر طول کلید بر توان مصرفی.

شمارنده	توان پویا	توان ایستا
۶۴بیتی	۶٫۱۴۳۷ mW	۱٫۰۹۱۵ uW
۱۲۸بیتی	۶٫۱۴۵۵ mW	۱٫۰۹۲۸ uW
۲۵۶بیتی	۶٫۱۴۷۱ mW	۱٫۰۹۳۴ uW

صحیح واردنمودن کلید $1/2^n$ خواهد بود. در صورتی که مقدار $n=64$ یا ۱۲۸ باشد، این احتمال به ترتیب $E-205{,}42$ و $E-392{,}94$ خواهد بود.

در صورت اشتباه‌بودن کلید در [۲۳]، مقدار خروجی صفر تولید می‌شود که در نتیجه مهاجم می‌تواند به سرعت تشخیص دهد که کلید را اشتباه وارد کرده است و مدار را بازنشانی می‌کند و دوباره کلید را امتحان خواهد کرد. در [۲۴] در صورتی که کلید درست وارد نشود، ترتیب زیرزنجیره‌ها به هم زده می‌شود. این روش هرچند از حملات تفاضلی جلوگیری می‌کند اما در برابر حملات مبتنی بر امضا آسیب‌پذیر است. در روش این مقاله، مقدار خروجی تصادفی تولید می‌شود که هم سبب می‌شود که مهاجم نتواند تشخیص دهد که کلید اشتباه وارد شده و هم این که اطلاعات صحیحی در اختیار مهاجم قرار نخواهد گرفت تا بتواند به وسیله آن کلید را استخراج نماید.

علاوه بر این تا هنگامی که بیت‌های کلید به صورت کامل و صحیح وارد نشده باشند، خروجی صفر خواهد بود و مهاجم نمی‌تواند مقادیر اولیه LFSR را محاسبه کند. چون مقدار LFSR همواره تابعی از مقدار وضعیت قبلی خود می‌باشد، در نتیجه مهاجم از روی خروجی LFSR نمی‌تواند مقدار آن را محاسبه کند. البته از روی خروجی می‌تواند طول حدودی کلید را حدس بزند اما باز هم برای حدس کلید باید از بین 2^n حالت انتخاب کند.

۵-۲ تحلیل آزمون‌پذیری

برتری زنجیره پویش نسبت به بقیه روش‌های آزمون، پوشش خطای بالا می‌باشد. در صورتی که کلید صحیح وارد شود، عملکرد زنجیره پویش مانند حالت عادی خواهد بود و مهندس آزمون می‌تواند از الگوهای معمول برای آزمون تراشه استفاده کند. بنابراین احتیاجی به تغییر در ابزارهای صنعتی تولید خودکار الگوی آزمون نیست و می‌توان با استفاده از همین ابزارهای موجود، آزمون تراشه را به خوبی انجام داد.

اشکالات ممکن است در مدارهای کنترلی یا LFSRها رخ بدهد که نمی‌توان آنها را با زنجیره پویش بررسی نمود چون این زنجیره پویش نیز ممکن است به عنوان کانال جانبی مورد حمله قرار گیرد. برای تشخیص این خطاها می‌توان از روش‌های دیگر آزمون مانند BIST استفاده کرد. همچنین به دلیل حجم بسیار کم این مدار نسبت به کل مدار، می‌توان از آزمون آن صرف نظر کرد.

۶- نتیجه‌گیری و کار آتی

در این مقاله روشی جدید برای مقابله با حملات کانال جانبی مبتنی بر زنجیره پویش که به تراشه‌های رمزنگاری انجام می‌شود، ارائه کرده‌ایم. این دسته از حملات با استفاده از مقادیر میانی رمزنگاری که از زنجیره پویش استخراج می‌شود صورت می‌گیرد. روش‌هایی را که برای مبارزه با این حملات ارائه شده‌اند بررسی کردیم و به ویژگی‌های هر دسته پرداختیم. در روش پیشنهادی این مقاله، مدار کنترلی با دریافت کلید از کاربر، حمله را تشخیص داده و جلوی انجام آن را می‌گیرد. کلیدی که

باعث تشخیص حمله می‌شود در درون خود مدار ساخته می‌شود و به همین دلیل افزایش طول آن، سربار کمی ایجاد خواهد کرد اما در مقابل باعث افزایش چشم‌گیر امنیت خواهد شد. روش این مقاله دارای سربار کمتر از ۱٪ است که نسبت به روش‌های پیشین سربار قابل قبولی است. در صورتی که کلید نادرست وارد شود، خروجی تصادفی به مهاجم نشان داده می‌شود که می‌تواند از هر دو دسته حملات تفاضلی و مبتنی بر امضا جلوگیری کند. با افزایش طول کلید، تشخیص حمله بهتر صورت می‌گیرد و همچنین با افزایش طول LFSR و تغییر در تابع آن می‌توان اعداد تصادفی بهتری تولید کرد. از مزیت‌های این روش می‌توان به مستقل‌بودن طول کلید از طول LFSR اشاره کرد که سبب می‌شود افزایش طول کلید سربار کمتری را تولید کند. از طرف دیگر در صورت صحیح‌بودن کلید، زنجیره پویش عملکرد عادی خواهد داشت و به همین دلیل طرح پیشنهادی با ابزارهای صنعتی تولید خودکار الگوی آزمون سازگار است.

روش‌های متنوعی برای تولید اعداد تصادفی به شیوه سخت‌افزاری ارائه شده‌اند که در این مقاله ما صرفاً از روش LFSR برای تولید کلید به صورت تصادفی استفاده کردیم. روش‌های MT[1] [۲۵]، الگوریتم زیگورات [۲۶]، الگوریتم CORDIC [۲۶] و نوسان‌ساز حلقوی آشوب [۲۷] از جمله دیگر روش‌هایی هستند که به صورت سخت‌افزاری می‌توانند پیاده‌سازی شوند و کار تولید کلید تصادفی را بر عهده بگیرند. در حقیقت، در مدار پیشنهادی ما هر کدام از روش‌های ذکرشده می‌تواند جایگزین دو واحد LFSR شود. اما به هر حال ساختار مدار (واحد کنترل خروجی و بخش کنترل ورودی) متناسب با روش به کار گرفته شده، تغییرات چشم‌گیری خواهد داشت. از طرفی دیگر، هرچه کلید تصادفی تولیدشده، بیشتر به سمت اعداد تصادفی واقعی نزدیک باشد، بهتر حمله‌کننده را گمراه می‌کند. بنابراین یکی از کارهای آتی که می‌توان انجام داد جایگزین‌کردن روش‌های ذکرشده با دو واحد LFSR است تا اثر آن را در گمراه‌کردن حمله‌کننده دید و مقایسه نمود.

مراجع

[1] Y. Bo, W. Kaijie, and R. Karri, "Scan-based side-channel attack on dedicated hardware implementations of data encryption standard," in *Proc. IEEE Int. Test Conf., ITC'04,* pp. 339-344, Washington DC, USA, 26-28 Oct. 2004.

[2] Y. Bo, W. Kaijie, and R. Karri, "Secure scan: a design-for-test architecture for crypto chips," *IEEE Trans. CAD Integr. Cir. Syst.*, vol. 25, no. 10, pp. 2287-2293, Oct. 2006.

[3] H. Kodera, M. Yanagisawa, and N. Togawa, "Scan-based attack against DES cryptosystems using scan signatures," in *Proc. Asia Pacific Conf. Cir. Syst., APCCAS'12,* pp. 2-5, Kaohsiung, Taiwan, 2-5 Dec. 2012.

[4] R. Nara, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A scan-based attack based on discriminators for AES cryptosystems," *IEICE Trans. on Fundamentals Electronics, Communications and Computer Sciences*, vol. E92-A, no. 12, pp. 3229-3237, Dec. 2009.

[5] R. Nara, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Scan-based attack against elliptic curve cryptosystems," in *Proc. of Asia and South Pacific Des. Autom. Conf., ASP-DAC'10,* pp. 407-412, Taipei, Taiwan, 18-21 Jan. 2010.

[6] R. Nara, K. Satoh, M. Yanagisawa, and N. Togawa, "Scan-based sidechannel attack against RSA cryptosystems using scan signatures," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E93-A, no. 12, pp. 2481-2489, Dec. 2010.

[7] K. S. Kumar, K. Lodha, S. R. Sahoo, and K. K. Mahapatra, "On-chip comparison based secure output response compactor for scan-based attack resistance," in *Proc. Int. Conf. on VLSI Systems, Architecture, Technology and Applications, VLSI-SATA'15,* 6 pp., Bangalore, India, 8-10 Jan. 2015.

[8] A. Mehta, D. Saif, and R. Rashidzadeh, "A hardware security solution against scan-based attacks," *IEEE Int. Symp. on Circuits and Systems, ISCAS'16*, pp. 1698-1701, May 2016.

[9] J. Da Rolt, A. Das, G. Di Natale, M. L. Flottes, B. Rouzeyre, and I. Verbauwhede, "Test versus security: past and present," *IEEE Trans. on Emerging Topics in Computing*, vol. 2, no. 1, pp. 50-62, Mar. 2014.

[10] J. D. Rolt, G. D. Natale, M. L. Flottes, and B. Rouzeyre, "Scan attacks and countermeasures in presence of scan response compactors," in *Proc. 16th IEEE European Test Symp., ETS'11,* Annecy, France, pp. 19-24, May 2011.

[11] J. D. Rolt, G. D. Natale, M. Flottes, and B. Rouzeyre, "Are advanced DFT structures sufficient for preventing scan-attacks?" in *Proc. of the IEEE VLSI Test Symp., VTS'12,* pp. 246-251, Maui, HI, USA, Apr. 2012.

[12] J. D. Rolt, G. D. Natale, M. Flottes, and B. Rouzeyre, "Are advanced DFT structures sufficient for preventing scan-attacks?" in *Proc. IEEE VLSI Test Symp., VTS'12,* pp. 246-251, Maui, HI, USA, 23-25 Apr. 2012.

[13] A. Cui, Y. Luo, and C. H. Chang, "Static and dynamic obfuscations of scan data against scan-based side-channel attacks," *IEEE Trans. on Information Forensics and Security*, vol. 12, no. 2, pp. 363-376, Feb. 2017.

[14] G. Sengar, D. Mukhopadhyay, and D. R. Chowdhury, "Secured flipped scan-chain model for crypto-architecture," *IEEE Trans. on Computer Aided Design of Integrated Circuits and Systems*, vol. 26, no. 11, pp. 2080-2084, Nov. 2007.

[15] Y. Shi, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Robust secure scan design against scan-based differential cryptanalysis," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 176-181, Jan. 2012.

[16] M. Agrawal, S. Karmakar, D. Saha, and D. Mukhopadhyay, "Scan based side channel attacks on stream ciphers and their counter-measures," in *Proc. 9th Annual Int. Conf. on Cryptology in India,* Kharagpu, India, pp. 226-238, Dec. 2009.

[17] Y. Atobe, Y. Shi, M. Yanagisawa, and N. Togawa, "Dynamically changeable secure scan architecture against scan_based side channel attack," in *Proc. Int. SoC Design Conf., ISOCC'12,* pp. 155-158, Jeju Island, Korea, 4-7 Nov. 2012.

[18] Y. Atobe, Y. Shi, M. Yanagisawa, and N. Togawa, "Secure scan design with dynamically configurable connection," in *Proc. 19th IEEE Pacific Rim Int. Sym. Dependable Computing, PRDC'13,* pp. 256-262, Vancouver, Canada, 2-4 Dec. 2013.

[19] D. Hly, *et al*., "Scan design and secure chip," in *Proc. 10th IEEE Int. On-Line Testing Symp.*, pp. 219-226, 14-14 Jul. 2004.

[20] J. D. Rolt, G. Di Natale, M. L. Flottes, and B. Rouzeyre, "Thwarting scan-based attacks on secure-ICs with on-chip comparison," *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 4, pp. 947-951, Apr. 2014.

[21] B. Yang, K. Wu, and R. Karri, "Secure scan: a design-for-test architecture for crypto chips," in *Proc. of 42nd Annual Conf. on Design Automation*, pp. 135-140, Anaheim, CA, USA, 13-17 Jun. 2005.

[22] S. Paul, R. S. Chakraborty, and S. Bhunia, "VIm-scan: a low overhead scan design approach for protection of secret key in scan-based secure chips," in *Proc. the 25th IEEE VLSI Test Symp.*, pp. 455-460, Berkeley, CA, USA, 6-10 May 2007.

[23] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing designs against scan-based side-channel attacks, " *IEEE Trans. on Dependable and Secure Computing*, vol. 4, no. 4, pp. 325-336, Oct. 2007.

[24] J. Lee, M. Tehranipoor, C. Patel, and J. Plusquellic, "Securing designs against scan-based side-channel attacks," *IEEE Trans. on Dependable and Secure Computing*, vol. 4, no. 4, pp. 325-336, Oct. 2007.

[25] J. Li, J. Jiang, H. Cheng, M. Zhang, and S. Wei, "An efficient hardware random number generator based on the MT method," in *12th IEEE Int. Conf. on Computer and Information Technology*, pp. 1011-1015, Chengdu, China, 27-29 Oct. 2012.

[26] B. Sileshi, C. Ferrer, and J. Oliver, "Accelerating hardware Gaussian random number generation using Ziggurat and CORDIC algorithms," in *Proc. IEEE Int. Conf. on Sensors*, pp. 2122-2125, Valencia, Spain, 2-5 Nov. 2014.

[27] I. Cicek and G. Dundar, "A hardware efficient chaotic ring oscillator based true random number generator," in *Proc.18th IEEE Int. Conf. on Electronics, Circuits, and Systems*, pp. 430-433, Beirut, Lebanon, 11-14 Dec. 2011.

1. Mersenne Twister

فاطمه جمالی زواره تحصیلات خود را در مقطع کارشناسی ارشد مهندسی کامپیوتر در سال ۱۳۹۷ از دانشگاه علم و صنعت ایران به پایان رسانده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: امنیت سخت‌افزار، مدارات قابل بازپیکربندی و شبکه بر روی تراشه.

حاکم بیت‌الهی در سال ۱۳۸۱ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه تهران و در سال ۱۳۸۴ مدرک کارشناسی ارشد مهندسی کامپیوتر خود را از دانشگاه صنعتی شریف دریافت نمود. ایشان در سال ۱۳۹۱ مدرک دکترای مهندسی کامپیوتر خود را از دانشگاه کاتولیک لوون کشور بلژیک دریافت نمود. سپس تا سال ۱۳۹۲ در دانشگاه مذکور به‌عنوان پسا دکترا به فعالیت خود ادامه داد. دکتر بیت‌الهی در سال ۱۳۹۲ به دانشگاه صنعتی امیرکبیر پیوست اما یکسال بعد به خارج از کشور برگشت. سرانجام از سال ۱۳۹۵ به جمع اعضای هیأت علمی دانشکده مهندسی کامپیوتر دانشگاه علم و صنعت ایران پیوست. از سال ۱۳۹۶ ایشان مدیرگروه معماری سیستم‌های کامپیوتری در دانشکده مذکور است. زمینه‌های علمی مورد علاقه ایشان متنوع بوده و شامل موضوعاتی مانند امنیت سخت‌افزار، سیستم‌های قابل بازپیکربندی، شتاب‌دهنده‌های سخت‌افزاری، سیستم‌های بی‌درنگ، طراحی سخت‌افزار برای هوش مصنوعی و امنیت شبکه.