علوم و مهندسی کامپیوتر
نشریه علمی پژوهشی انجمن کامپیوتر ایران
مجلد ۸، شماره ۱ و ۳ (الف)، ۱۳۸۹
صفحات ۵۳-۶۲
مقاله کوتاه

یک روش کارآمد برای محافظت از پیاده‌سازی سخت‌افزاری الگوریتم رمزنگاری AES در مقابل حمله تحلیل تفاضلی توان

مسعود معصومی[۱] سید مجتبی دهنوی[۲]

[۱]دانشکده فنی، دانشگاه آزاد اسلامی واحد اسلامشهر، تهران، ایران
[۲]دانشکده علوم ریاضی و کامپیوتر، دانشگاه تربیت معلم، تهران، ایران

چکیده

مدل متداول و سنتی ارزیابی سیستم‌های رمزنگاری، امنیت را از منظر توابع ریاضی بکار رفته در آن مورد بررسی قرار می‌دهد. این روش آثار فیزیکی جانبی پیاده‌سازی و استفاده از این توابع در دنیای واقعی را در نظر نمی‌گیرد. یک مدل واقعی‌تر، امنیت ابزار رمزنگاری را از دید حملات کانال جانبی یا حملاتی که از اطلاعات مرتبط با پیاده سازی فیزیکی توابع رمزنگاری استفاده می‌کنند نیز مورد توجه قرار می‌دهد. حمله تحلیل تفاضلی توان نوع قدرتمند و منحصر بفردی از حملات کانال جانبی است که از توان مصرفی تراشه در حال رمز کردن اطلاعات برای شکستن الگوریتم رمز و بدست آوردن کلید آن استفاده می‌کند. در این مقاله روش جدید و کارآمدی برای محافظت از الگوریتم رمزنگاری AES پیشنهاد شده که قادر به افزایش قابل توجه مقاومت الگوریتم در مقابل این حمله با هزینه سخت افزاری بسیار پایین در مقایسه با سایر روش‌های گزارش شده تاکنون است. روش جدید مبتنی بر استفاده از ریاضیات میدان‌های مرکب است در حالیکه تنها باعث ۷٪ افزایش مساحت اشغالی روی تراشه خواهد شد بدون‌آنکه باعث کاهش فرکانس کاری یا تغییر در الگوریتم شده و خدشه‌ای به استاندارد بودن آن وارد شود. موثر بودن روش پیشنهادی با استفاده از نتایج عملی حاصل از پیاده سازی بر روی تراشه Xilinx Spartan-II FPGA تایید شده است.

کلمات کلیدی: سیستم رمزنگاری AES، حملات کانال جانبی، حمله تحلیل توان، پیاده‌سازی سخت‌افزاری، FPGA.

۱- مقدمه

در مدل متداول ارزیابی سیستم‌های رمزنگاری سیستم امن به سیستمی اطلاق می‌شود که حتی در صورت دسترسی کامل به اطلاعات ردوبدل شده بین رمزنگار و رمزگشا نتوان کلید رمز را بازیابی کرد. در اواسط دهه نود نوع جدیدی از حملات رمزشکنی موسوم به حملات کانال جانبی[۱] معرفی شد که از اطلاعات نشتی از یک سخت‌افزار در حال پردازش اطلاعات نظیر تشعشات الکترومغناطیسی یا توان مصرفی آن برای شکستن رمز و استخراج کلید استفاده می‌کند. این گونه حملات بخاطر سادگی و موثر بودن بسرعت گسترش پیدا کردند تا آنجا که اکنون مهمترین خطر برای سیستم‌های رمزنگاری معاصر از ناحیه هدف قرار دادن نقاط ضعف پیاده‌سازی و استفاده از اطلات جانبی نشتی است و حملات آماری و الگوریتمی در درجه بعدی اهمیت قرار می‌گیرند. در نتیجه پیاده‌سازی صحیح یک الگوریتم رمز الزاما بمعنای امن بودن آن نیست [۱]. پس از انتشار مقاله P. Kocher در سال ۱۹۹۶ با عنوان "حمله تحلیل تفاضلی توان" انواع مختلفی از این گونه حملات پیشنهاد شده و توسعه یافته است [۴-۲] حملات تحلیل توان نوع خاصی از حملات کانال جانبی هستند که از وابستگی توان مصرفی ابزار یا تراشه در حال رمزکردن اطلاعات به اطلاعات در حال پردازش و/یا عملیات در حال انجام برای شکستن الگوریتم و بازیابی کلید رمز استفاده می‌کنند. تحلیل توان خانواده‌ای

قدرتمند و منحصر بفرد از حملات رمزشکنی سخت‌افزاری است. بدلیل ماهیت این نوع حملات جلوگیری از آنها بسادگی میسر نیست ضمن آنکه نیاز به تجهیزات پیچیده و گران قیمت برای پیاده سازی حمله نداشته و بسادگی و در زمان کوتاه قادر به شکستن رمز هستند. حملات تحلیل توان عمدتا به دو دسته حملات ساده و تفاضلی تقسیم‌بندی می‌شوند. حمله آنالیز توان ساده[۲] مستقیماً از اندازه‌گیری‌های توان مصرفی وسیله حین انجام عملیات رمزنگاری استفاده می‌کند. الگوی توان یا انرژی مصرفی سخت افزار می‌تواند اطلاعات مهمی در مورد دستورالعمل‌ها، توالی اجرای آنها و حتی عملوند در اختیار هکر قرار بدهد. حمله تحلیل تفاضلی توان[۳] (DPA) بمراتب موثرتر از حمله تحلیل توان ساده است و در واقع نوعی آزمایش آماری است که کلید رمز با انجام آزمایش های آماری بر روی صدها و شاید هزاران داده نمونه و اندازه گیری توان مصرفی متناظر با آنها حدس زده می‌شود. مقاومت در برابر این حمله بسادگی میسر نیست زیرا حتی در صورت افزودن نویز به سیستم، نویز اضافه شده به مصرف توان با افزایش تعداد نمونه‌ها و متوسط گیری از آنها فیلتر می‌شود. علاوه بر آن نوع قدرتمند دیگری از این حملات موسوم به تحلیل همبستگی توان[۴] نیز بطور گسترده‌ای مورد مطالعه و استفاده قرار گرفته است [۱، ۳]. در سال‌های اخیر امنیت الگوریتم استاندارد پیشرفته رمزنگاری (AES) در مقابل حمله تحلیل توان مورد توجه ویژه قرار گرفته است زیرا این الگوریتم از جمله مهمترین و پرکاربرد ترین الگوریتم‌های رمزنگاری معاصر است که در کاربردهای متنوعی از جمله کارت‌های هوشمند وب سرورها تلفن‌های سلولی شبکه‌های ATM و ... مورد استفاده قرار می‌گیرد [۵، ۶]. در نتیجه تحقیقات و مطالعات انجام شده انواع مختلفی از این حملات و نیز روش‌های مقابله با آنها پیشنهاد شده است. از جمله مهمترین روش‌های مقابله می‌توان به روش‌های تصادفی سازی مانند پالس ساعت تصادفی شده [۷]، رمزکردن حافظه، تصادفی کردن توان مصرفی و از بین بردن همبستگی بین توان مصرفی تراشه و منبع تغذیه خارجی اشاره نمود. علاوه بر آن روش‌های دیگری از قبیل پیاده‌سازی با منطق مکمل[۵]، منطق SABL[۶]، و منطق غیر همزمان[۷] نیز می‌توان نام برد. [۸، ۹، ۱۰]. متاسفانه غالب این تکنیک ها غیر کارآمد و غیر موثر و یا غیر مقاوم در مقابل حملات تحلیل توان از مرتبه بالاتر هستند. بعنوان مثال تکنیک های پیشنهاد شده در [۱۱] تقریبا دو برابر مساحت یک پیاده سازی محافظت نشده سطح اشغال می کنند و یا دو برابر آن توان مصرف می کنند و تکنیک پیشنهادی در [۱۰] سه برابر مساحت اشغال کرده و بازدهی را تا یک چهارم تقلیل می‌دهد. روش شناخته شده دیگر مقابله ماسک گذاری بمفهوم استفاده از اعداد تصادفی برای ماسک کردن داده های ورودی و میانی برای شخص کنجکاو است. مشکل اساسی این روش نیز آن است که نیاز به یک مسیر داده موازی جداگانه برای ماسک گذاری و ماسک برداری است که باعث افزایش قابل ملاحظه مساحت اشغالی روی تراشه و کاهش بازدهی خواهد شد ضمن آنکه تضمین کننده امنیت کامل پیاده سازی در مقابل حمله تحلیل توان تفاضلی نیز نیست [۱۲، ۱۳، ۱۴]. در این مقاله روش ساده جدید و کارآمدی برای خنثی کردن تحلیل توان تفاضلی و نیز تحلیل همبستگی توان پیشنهاد شده که از خصوصیات ریاضی الگوریتم AES استفاده می کند بدون آنکه تغییری در الگوریتم ایجاد کرده یا خدشه ای به استاندارد بودن آن وارد سازد. روش مزبور مبتنی بر نوعی تصادفی سازی در ریاضیات میدانهای مرکب[۸] است و نیازی به استفاده از عامل خارجی برای مقابله با حمله ندارد. بمنظور ارزیابی میزان موثر بودن روش پیشنهادی دو نسخه محافظت نشده و محافظت شده با روش جدید AES را بر روی تراشه Xilinx Spartan-II FPGA پیاده سازی کرده و نتایج را از حیث میزان مقاوم بودن در مقابل حمله و نیز کارآیی پیاده سازی مورد ارزیابی قرار دادیم. نتایج بدست آمده نشان داد که نسخه محافظت شده علاوه بر فراهم آوردن امنیت بالا در مقابل حمله تحلیل توان تنها ۷٪ بیشتر از نسخه محافظت نشده مساحت اشغال می کند ضمن آنکه باعث کاهش سرعت عملیات رمز نیز نخواهد شد. علاوه بر آن، بر خلاف بسیاری از روش‌های گزارش شده تاکنون، روش پیشنهادی در این مقاله منحصر به پیاده سازی سخت افزاری یا FPGA نبوده و قابل بکارگیری در پیاده سازی های نرم افزاری و نیز بسترهایی نظیر کارتهای هوشمند، پردازشگرهای سیگنال دیجیتال و ... می‌باشد. مهمتر اینکه نتایج بدست آمده در این مقاله بروشنی نشان داد که بر خلاف روش‌های پیشنهادی تاکنون که عمدتا بدنبال مقابله با حمله با وارد کردن یک عامل تصادفی خارجی یا استفاده از یک سخت افزار جانبی هستند می توان الگوریتم هایی طراحی کرد که بطور ذاتی در مقابل حمله تحلیل توان مقاوم باشند. قابل ذکر است که عمده مقالات ارائه شده دراین زمینه بخاطر سادگی پیاده سازی حمله، کارت‌های هوشمند و ریزپردازنده ها را هدف قرار گرفته و مقالات کمی در مورد ارزیابی امنیتی FPGA در مقابل حمله تحلیل توان وجود دارد [۱، ۱۵]. در ادامه مقاله ابتدا در بخش دوم کاربرد ریاضیات میدانهای مرکب در پیاده سازی موثر الگوریتم AES را تشریح می‌کنیم. در بخش سوم به تشریح نحوه پیاده سازی عملیات پایه ریاضی در میدان های مرکب خواهیم پرداخت. در بخش چهارم ایده جدید را ارائه می دهیم. بخش پنجم مقاله به نحوه پیاده سازی عملی و آزمایشگاهی حمله، تنظیمات دستگاههای بکار رفته و نتایج بدست آمده اختصاص داده شده است. در نهایت به بررسی برخی چالشهای موجود در مقابله با حملات تحلیل توان و جمع بندی کلی موضوع پرداخته و نتایج نهایی را ارائه می‌دهیم.

۲- استفاده از ریاضیات میدان‌های مرکب برای پیاده‌سازی AES

در پیاده‌سازی AES Rijndael تقریباً نیمی از تاخیر هر دور هنگام انجام تبدیل جانشینی بایت‌ها و مربوط به تاخیر SBoxهایی است که بصورت LUT[۹] پیاده‌سازی شده‌اند. تاخیر غیر قابل کاهش LUTها عاملی است که مانع از افزایش سرعت مدار خواهد شد [۱۶]. راه‌های دیگری نیز برای پیاده‌سازی SBoxها بصورت سخت‌افزاری و بدون نیاز به LUT وجود دارد که در صورت استفاده از آنها می‌توان هر دور را با استفاده از Subpipelinig به تعدادی زیر دور تقسیم کرده و سرعت مدار را افزایش داد. از آنجا که SBox الگوریتم Rijndael شامل معکوس ضربی عناصر روی میدان $GF(2^8)$ است راه‌های مختلفی برای پیاده‌سازی معکوس ضربی بدون نیاز به LUT و افزایش سرعت مدار مانند استفاده از الگوریتم گسترش یافته اقلیدس[۱۰] یا استفاده از قضیه فرمت[۱۱] پیشنهاد شده است. اما یکی از موثرترین این راه‌ها استفاده از میدان های مرکب[۱۲] برای محاسبه و پیاده‌سازی معکوس ضربی عناصر روی میدان محدود یا پیاده‌سازی مدار معادل SBox است [۱۷، ۲۱]. با استفاده از میدان‌های مرکب می‌توان پیچیدگی سخت‌افزاری پیاده‌سازی را کاهش داد. دو زوج

$$\{GF(2^n),\ Q(y) = y^n + \sum_{i=0}^{n-1} q_i y^i,\ q_i \in GF(2)\}$$ و

$$\{GF((2^n)^m),\ P(X) = X^m + \sum_{i=0}^{m-1} P_i X^i,\ P_i \in GF(2^n)\}$$

را یک میدان مرکب می‌گوییم اگر:

۱- $GF(2^n)$ از $GF(2)$ بوسیله $Q(y)$ ساخته شود.

۲- $GF((2^n)^m)$ از $GF(2^n)$ بوسیله $P(x)$ ساخته شود.

یک میدان مرکب با $GF((2^n)^m)$ نشان داده می‌شود و با میدان $GF(2^k)$ بازای $k = mn$ یکریخت[۱۳] است. لذا یک میدان مرکب را می‌توان بصورت تکراری[۱۴] از

میدان‌های مرتبه پایین‌تر ساخت و محاسبات روی میدان اصلی را روی میدان‌های مرتبه پایین‌تر انجام داده و هزینه سخت‌افزاری را کاهش داد. برای مثال می‌توان میدان مرکب $GF(2^8)$ را از میدان $GF(2)$ با استفاده از چند جمله‌ای‌های زیرساخت.

$$\begin{cases} GF(2) \Rightarrow GF(2^2): & P_0(X) = X^2 + X + 1 \\ GF(2^2) \Rightarrow GF((2^2)^2): & P_1(X) = X^2 + X + \varphi \\ GF((2^2)^2) \Rightarrow GF(((2^2)^2)^2): & P_2(X) = X^2 + X + \lambda \end{cases} \quad (1)$$

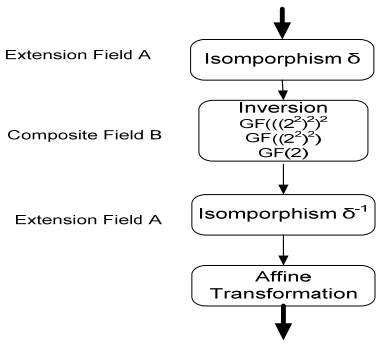
که $\Phi = \{10\}_2$ و $\lambda = \{1100\}_2$ البته برای آنکه عناصر $GF(2^8)$ دقیقاً بر روی عناصر متناظر خود در میدان مرکب تصویر شوند یک تبدیل $x \times f(x) = \delta$ و یک تبدیل خطی affine نیز لازم است. δ یک ماتریس باینری 8×8 متناظر با $m(x) = x^8+x^4+x^3+x+1$ است و می‌تواند بفرم زیر باشد.

$$\delta = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

برای بازگشت از میدان مرکب به میدان $GF(2^8)$ به عکس تبدیل فوق و δ^{-1} نیاز داریم. محاسبه معکوس عناصر در $GF(2^8)$ که عملیات اصلی در تبدیل جانشینی بایت‌ها و نیز معکوس آن است بیشترین سخت‌افزار را در پیاده‌سازی هر دور الگوریتم AES مصرف می‌کند. ماتریس δ^{-1} متناظر با ماتریس δ فوق می‌تواند بشکل زیر باشد.

$$\delta^{-1} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

شکل ۱ نشان دهنده انجام عملیات محاسبه عنصر معکوس بر روی میدان $GF(2^8)$ با استفاده از ریاضیات میدان‌های مرکب است.



شکل ۱- انجام عملیات محاسبه عنصر معکوس بر روی میدان $GF(2^8)$ با استفاده از ریاضیات میدان‌های مرکب

اگر q یک عنصر از میدان $GF(2^8)$ باشد آنگاه نگاشت یکریخت[15] و معکوس آن بفرم ضرب ماتریسی δ^*q و $\delta^{-1}*q$ خواهد بود. اگر q_7 با ارزش‌ترین بیت و q_0 کم‌ارزش‌ترین بیت باشند این دو ضرب ماتریسی را می‌توان با استفاده از گیت‌های XOR بشکل زیر پیاده‌سازی نمود.

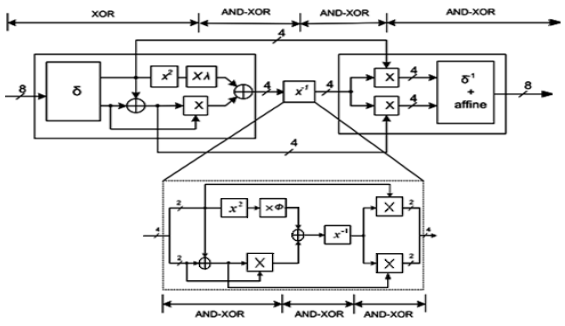
$$\delta \times q = \begin{bmatrix} q_7 \oplus q_5 \\ q_7 \oplus q_6 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \\ q_7 \oplus q_5 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_6 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_6 \oplus q_4 \oplus q_1 \\ q_6 \oplus q_1 \oplus q_0 \end{bmatrix}$$

$$\delta^{-1} \times q = \begin{bmatrix} q_7 \oplus q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_2 \\ q_6 \oplus q_5 \oplus q_1 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_7 \oplus q_4 \oplus q_3 \oplus q_2 \oplus q_1 \\ q_5 \oplus q_4 \\ q_6 \oplus q_5 \oplus q_4 \oplus q_2 \oplus q_0 \end{bmatrix}$$

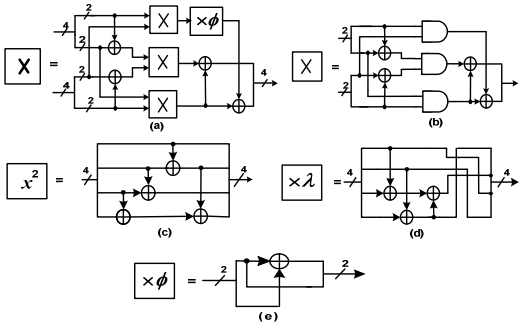
در میدان مرکب $GF((2^4)^2)$ هر عنصر را می‌توان بصورت $s_h\, x + s_l$ نشان داد جائیکه s_h و s_l عضو $GF(2^4)$ هستند و x نیز ریشه $P_2(x)$ است. طبق الگوریتم توسعه یافته اقلیدس معکوس عنصر $s_h\, x + s_l$ در هنگ $P_2(x)$ را می‌توان بصورت زیر محاسبه نمود:

$$(s_h X + s_l)^{-1} = s_h\, \theta\, X + (s_h + s_l)\, \theta \quad (2)$$

که در آن $\theta = (s_h^2 \lambda + s_h s_l + s_l^2)^{-1}$ طبق رابطه فوق معکوس ضربی هر عنصر روی $GF(2^8)$ را می توان با استفاده از معماری نشان داده شده در شکل ۲ روی میدان مرکب $GF((2^4)^2)$ بدست آورد. ضرب‌کننده‌های در $GF(2^4)$ را می‌توان به ضرب‌کننده‌های روی $GF((2^2)^2)$ و سپس $GF(2)$ تجزیه[16] نمود که عمل ضرب در $GF(2)$ براحتی و با گیت AND قابل انجام است. شکل ۳ نحوه تجزیه ضرب‌کننده روی $GF(2^4)$ را نشان می‌دهد و چنانچه از شکل پیداست این ضرب‌کننده را می‌توان با ۲۱ گیت XOR و ۹ گیت AND پیاده‌سازی نمود در حالیکه ۴ گیت XOR و یک گیت AND در مسیر بحرانی آن وجود دارد، اما معکوس کننده ضربی روی $GF(2^4)$ را نشان نمی‌دهد [۱۶].



شکل ۲- پیاده‌سازی تبدیل جانشینی بایت‌ها با استفاده از میدان مرکب [۱۶]



شکل ۳- پیاده‌سازی بلوک‌های شکل ۴- (a) ضرب‌کننده در $GF(2^4)$، (b) ضرب‌کننده در $GF(2^2)$، (c) مربع کننده در $GF(2^4)$; (d) و (e) ضرب‌کننده‌های ثابت [۱۶].

۳- پیاده‌سازی سخت‌افزاری عملیات ریاضی در میدان مرکب

از آنجا که برای درک و پیاده سازی. ایده پیشنهادی نیاز به درک نحوه پیاده‌سازی برخی از عملیات ریاضی مانند محاسبه معکوس، جمع، ضرب و مربع در میدان‌های مرکب داریم لذا در این بخش بطور مختصر نحوه تحقق چنین عملیاتی را شرح می‌دهیم. هر چند جمله‌ای دلخواه را می‌توان بصورت $bx + c$ نشان داد جائیکه b نیمه بالا و c نیمه پایین را نشان می‌دهد. بنابراین هر عدد باینری q در میدان گالوا را می‌توان به دو جز تقسیم نمود و به شکل $q_Hx + q_L$ نشان داد [۲۱]. برای مثال $q = \{1011\}_2$ را می‌توان بصورت $\{10\}_2x + \{11\}_2$ نشان داد جائیکه $q_H = \{10\}_2$ و $q_L = \{11\}_2$ است. متعاقبا می توان q_H و q_L را بترتیب بصورت $\{1\}_2x + \{0\}_2$ و $\{1\}_2x + \{1\}_2$ تجزیه کرد. با استفاده از این ایده می‌توان عملیات منطقی برای پیاده‌سازی بلوک‌های شکل ۲ را تعیین کرده و بدست آورد. تشریح جزئیات پیاده سازی تمام بلوک‌های شکل ۲ خارج از موضوع این مقاله است ولی برای درک بهتر مساله و روشن شدن موضوع به ذکر نحوه پیاده‌سازی دو بلوک ضرب در مقادیر ثابت (λ×) و(φ×) می‌پردازیم. سایر بلوک‌ها نیز بطریق مشابه قابل پیاده سازی هستند. برای سادگی، مقداری از عمق ریاضی مساله کاسته شده تا خواننده بهتر با عملیات انجام شده در درون تبدیل جانشینی بایت‌ها آشنا شود.

فرض کنیم $k = q\lambda$ باشد جائیکه $k = \{k_3\ k_2\ k_1\ k_0\}_2$ و $q = \{q_3\ q_2\ q_1\ q_0\}_2$ و $\lambda = \{1100\}_2$ عناصر $GF(2^4)$ هستند.

$$k = \{k_3 k_2 k_1 k_0\} = k_H x + k_L = \{q_3 q_2 q_1 q_0\}(1100)_2$$

جائیکه $\lambda_H = \{11\}_2$ و. $\lambda_L = \{00\}_2$

$$k = (q_H x + q_L)(\lambda_H x + \lambda_L)$$

از آنجا که $\lambda_L = \{00\}_2$ است می‌توان از آن صرفنظر کرد.

$$k = q_H \lambda_H x^2 + q_l \lambda_H x$$

با جایگزینی $x^2 = x + \varphi$ از معادله (۲) خواهیم داشت.

$$k = q_H \lambda_H(x+ \varphi) + q_L \lambda_H x$$
$$k = (q_H \lambda_H + q_l \lambda_H)x + (q_H \lambda_H)\ \varphi \in GF(2^2)$$

k_H و k_L را می‌توان به $GF(2)$ برد.

$$k_H = q_H \lambda_H + q_L \lambda_H$$
$$k_H = (q_3q_2)(11)_2 + (q_1q_0)(11)_2$$
$$k_H = (q_3x+q_2)(x+1)+(q_1x+q_0)(x+1)$$
$$k_H = q_3x^2+(q_3+q_2)x+q_2+q_1x^2+(q_1+q_0)x+q_0 \qquad (۳)$$

جایگزینی $x^2 = x + 1$ منجر به نتیجه زیر خواهد شد.

$$k_H = q_3(x+1) + (q_3+q_2)x + q_2+q_1(x+1)+(q_1+q_0)x+q_0$$
$$k_H = (q_3+q_3+q_2+q_1+q_1+q_0)x+(q_3+q_2+q_1+q_0)$$
$$k_3x+k_2 = (q_2+q_0)x + (q_3+q_2+q_1+q_0) \in GF(2)$$

همین روال را می‌توان برای تجزیه k_L روی میدان $GF(2)$ بکار برد.

$$k_L = q_H \lambda_H \varphi$$
$$k_L = (q_3q_2)(11)_2\ (10)_2$$
$$k_L = (q_3x+q_2)(x+1)x$$
$$k_L = q_3x^3+q_2x^2+q_1x+q_0$$

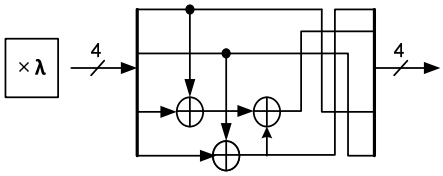
مجددا x^2 را می‌توان طبق معادله $x^2 = x + 1$ جایگزین کرده و بطریق مشابه نیز به این نتیجه می‌رسیم که $x^3 = 1$ لذا خواهیم داشت.

$$k_L = q_3(1)+q_2(x+1)+ q_3(x+1)+q_2x$$
$$k_L = (q_3+q_2+q_1)x+(q_3+q_2+q_1)$$
$$k_1x+k_0 = (q_3)x + (q_2) \in GF(2) \qquad (۴)$$

با ترکیب دو رابطه (۳) و (۴) می‌توان به منطق لازم برای پیاده‌سازی ضرب در مقدار ثابت λ رسید.

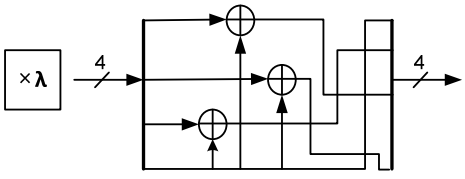
$$\boldsymbol{k_3 = q_2 \oplus q_0}$$
$$\boldsymbol{k_2 = q_3 \oplus q_2 \oplus q_1 \oplus q_0}$$
$$\boldsymbol{k_1 = q_3}$$
$$\boldsymbol{k_0 = q_2} \qquad (۵)$$

با استفاده از معادله (۵) می‌توان مدار شکل ۴ را برای پیاده‌سازی سخت‌افزاری بلوک ضرب در مقدار ثابت λ در نظر گرفت که در این شکل مقدار $\lambda = \{1100\}_2$ است.



شکل ۴- پیاده‌سازی سخت‌افزاری بلوک ضرب در مقدار ثابت $\lambda = 12$

همین روال را می‌توان برای پیاده‌سازی ضرب در مقدار ثابت $\lambda = \{1111\}_2$ در نظر گرفت و به مدار شکل ۵ رسید.



شکل ۵- پیاده‌سازی سخت‌افزاری بلوک ضرب در مقدار ثابت $\lambda = 15$

یا بعنوان مثال دیگر ضرب در مقدار ثابت $(\times\varphi)$ روی میدان $GF(2^2)$ را در نظر می‌گیریم جائیکه $\varphi = \{10\}_2$. فرض کنیم $k=q\varphi$ باشد جائیکه $q = k=\{k_1k_0\}_2$، $\{q_1q_0\}_2$ و $\varphi = \{10\}_2$ عناصری از $GF(2^2)$ هستند.

$$k = k_1x + k_0 = (q_1q_0)(10)_2 = (q_1x + q_0)(x)$$
$$k = q_1x^2 + q_0x$$

جایگزینی x^2 با $x+1$ به نتیجه زیر منجر خواهد شد.

$$k = q_1(x+1)+q_0x$$
$$k = (q_1 + q_0)x + (q_1) \in GF(2)$$

بنابراین منطق مورد نیاز برای پیاده سازی ضرب در ثابت $\varphi = \{10\}_2$ برابر خواهد بود با:

$$k_1 = q_1 + q_0$$
$$k_0 = q_1 \qquad (۶)$$

بنابراین مدار پیاده سازی ضرب در ثابت $\varphi = \{10\}_2$ بشکل ۶ خواهد بود.

چنانچه $\varphi = \{11\}_2$ باشد سخت‌افزار مورد نیاز برای ضرب در مقدار ثابت φ را می‌توان بصورت زیر بدست آورد.

$$k = k_1x + k_0 = (q_1q_0)(11)_2 = (q_1x + q_0)(x+1)$$
$$k = q_1x^2 + (q_0+ q_1)x + q_0$$

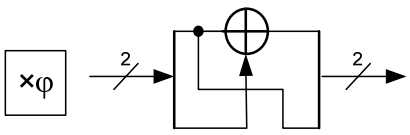
جایگزینی x^2 با $x+1$ به نتیجه زیر منجر خواهد شد.

$$k = q_1(x+1)+q_0x$$
$$k = (q_1 + q_0)x + (q_1) \in GF(2)$$

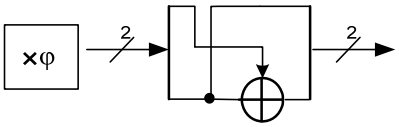
بنابراین مدار پیاده سازی ضرب در ثابت $\varphi = \{11\}_2$ در میدان $GF(2^2)$ برابر خواهد بود با:

$$k_1 = q_0$$
$$k_0 = q_0+q_1 \qquad (۷)$$

و مدار منطقی لازم برای تحقق این پیاده‌سازی بصورت شکل ۷ خواهد بود.



شکل ۶- پیاده‌سازی ضرب در مقدار ثابت $\varphi = \{10\}_2$ در میدان $GF(2^2)$



شکل ۷- پیاده‌سازی ضرب در مقدار ثابت $\varphi = \{11\}_2$ در میدان $GF(2^2)$

با جمع‌بندی مطالب گفته شده در بخش ۲ و ۳ و نیز با توجه به معماری شکل ۲ می‌توان چنین نتیجه گرفت که هر نوع تغییر در مقادیر مجموعه $\{\Phi, \lambda, \delta, \delta^{-1}\}$ یا بعبارات دیگر هر نوع تغییر بر این مبنا در معماری ساختار تبدیل جانشینی بایت‌ها قابل ترجمه به گیت های منطقی است و هزینه سخت افزاری چندانی را به پیاده ساز تحمیل نمی‌کند.

۴- ایده جدید

تحلیل توان برای بازیابی کلید از وابستگی توان مصرفی ابزار رمزنگاری به مقادیر میانی در حین اجرای الگوریتم استفاده می کند. بنابراین برای جلوگیری و ناکام گذاشتن حمله این وابستگی باید تا حد ممکن از بین برود. همانگونه که گفته شد چالش اصلی در مقابله با حمله تحلیل توان در ماسک کردن تبدیل جانشینی بایت هاست جائیکه هر عنصر با معکوس ضربی خود در میدان $GF(2^8)$ جایگزین می‌شود. با در نظر گرفتن این حقیقت و با مطالعه ساختار ریاضی الگوریتم Rijndael می توان دریافت که سه پارامتر $\{\Phi, \lambda, \delta\}$ منحصر بفرد نیستند زیرا تعداد بسیار بیشتری از یک یکریختی روی میدان $GF(2^8)$ وجود دارد. دو تبدیل یکریخت نامیده می‌شوند چنانچه هر دو بازای یک ورودی مشخص خروجی یکسانی تولید کنند. چند جمله ای $P_1(x) = x^2 + x + \Phi$ روی میدان $GF(2^2)$ تحویل ناپذیراست اگر و فقط اگر $1< \Phi <4$. $P_2(x) = x^2 + x + \lambda$ نیز روی $GF(2^4)$ تحویل‌ناپذیر است اگر و فقط اگر $7< \lambda <16$ باشد. همچنین تنها یک چند جمله‌ای تحویل‌ناپذیر روی $GF(2)$ وجود دارد که همان $P_3(x) = x^2 + x + 1$ است.

بنابراین شانزده ترکیب مختلف برای $\{\Phi , \lambda\}$ وجود دارد که همگی منجر به جواب یکسانی خواهند شد. همچنین با جستجو در مجموعه تبدیلات خطی از $GF(2^8)$ بر روی خودش یا تمام خود یکریختی ها[۱۷] بر روی $GF(2^8)$ می‌توان اثبات کرد که تعداد $\prod_{i=0}^{7}(2^8 - 2^i)$ از چنین تبدیلاتی و به همین تعداد ماتریس تبدیل δ/δ^{-1} وجود دارد [۲۲، ۲۳]. طبیعتا تعداد قابل ملاحظه‌ای یکریختی بر مبنای مجموعه‌های $\{\Phi, \lambda, \delta, \delta^{-1}\}$ برای Rijndael وجود خواهد داشت که می‌توان برخی از آنها را انتخاب کرد بدون آنکه باعث تغییر در ساختار ریاضی یا خروجی تبدیل جانشینی بایت‌ها بشود. لازم به ذکر است که تمام انتخاب‌های ممکن الزاما منجر به یکریختی میدانی مناسب برای Rijndael نخواهد شد. ما برای پیاده‌سازی ایده پیشنهادی ۳۲ مجموعه مختلف از $\{\Phi, \lambda, \delta, \delta^{-1}\}$ را که قابل پیاده‌سازی با حداقل گیت‌های منطقی هستند را انتخاب کردیم. بعنوان مثال سه مجموعه زیر چنین ویژگی دارند. برای سادگی مقادیر ماتریس‌های δ و δ^{-1} و نیز Φ و λ بصورت دسیمال نشان داده شده‌اند.

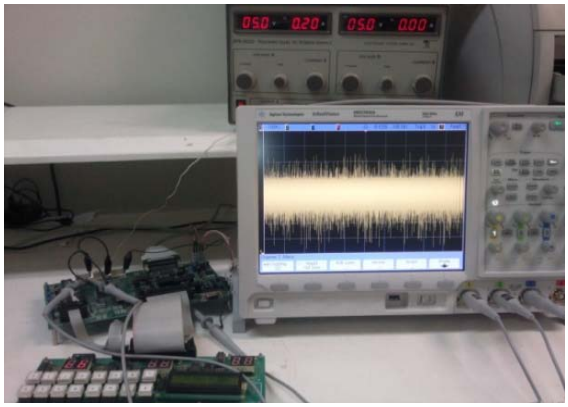
$\Phi= 2, \lambda = 15$
$\delta= \{160, 126, 114, 162, 182, 84, 16, 217\}^T$
$\delta^{-1}=\{46, 28, 174, 2, 122, 26, 144, 75\}^T$
$\Phi= 3, \lambda = 12$
$\delta= \{160, 222, 172, 174, 202, 238, 44, 227\}^T$
$\delta^{-1}=\{102, 212, 230, 162, 10, 234, 176, 233\}^T$
$\Phi= 3, \lambda = 10$
$\delta= \{160, 126, 172, 2, 20, 132, 130, 99\}^T$
$\delta^{-1}= \{190, 132, 62, 106, 98, 2, 112, 141\}^T$

ایده پیشنهادی تصادفی‌سازی در یکریختی تبدیل جانشینی بایت‌هاست به اینصورت که در ابتدا تعداد مشخصی مثلا ۳۲ مجموعه از $\{\Phi, \lambda, \delta, \delta^{-1}\}$ تعین شده و انتخاب می‌شوند. ۳۲ مجموعه SBox بر مبنای این ۳۲ مجموعه مقادیر مطابق با آنچه در شکل ۲ نشان داده شد تولید کرده و ذخیره می‌کنیم. بعبارت دیگر ۳۲ مجموعه SBox با ورودی- خروجی مشابه ولی ساختار درونی متفاوت خواهیم داشت. در هنگام رمز کردن هر قالب ۱۲۸ بیتی داده یا حتی برای هر بایت داده یکی از این ساختارها برای رمزنگاری توسط یک شمارنده اعداد شبه تصادفی

انتخاب می‌شود. بنابراین برای یک ورودی مشخص هر بار مقادیر باینری متفاوتی پردازش شده و الگوی مصرف توان ثابت نخواهد بود در حالیکه خروجی یکسان است. چنانچه در بخش 4 نشان داده شد ایجاد SBox ۳۲ تصادفی بشکلی که ذکر شد چندان هزینه بر نخواهد بود زیرا این تغییرات براحتی قابل ترجمه به گیت‌های منطقی است. ماتریس δ در ورودی SBox مانند بردار ماسک ورودی عمل کرده و در هر مرحله با تغییر آن داده های میانی که در درون تبدیل جانشینی بایت‌ها مورد پردازش قرار می گیرد متفاوت خواهد بود. از اینرو حتی اگر چنانچه بارها یک ورودی چندین بار اجرا شود پروفایل مصرف توان و داده‌هایی که برای حمله تحلیل تفاضلی توان جمع آوری می شوند متفاوت با یکدیگر خواهد بود که این مساله باعث مشکل شدن کار شخص کنجکاو در اجرای موفقیت آمیز حمله خواهد بود. همین امر باعث کاهش نسبت سیگنال به نویز (SNR) خواهد شد و همانطور که می‌دانیم کیفیت حمله بستگی به میزان نسبت سیگنال به نویز دارد. برای جلوگیری از نشت اطلاعات هنگام ذخیره کردن خروجی تبدیل جانشینی بایت‌ها در رجیستر مربوطه، یک شمانده تصادفی دیگر دو مقدار δ^{-1} دیگر را انتخاب کرده و مقدار پیش نهایی بصورت همزمان و موازی در آنها ضرب می شود. این امر باعث غیر قابل تشخیص بودن نحوه مصرف توان در مرحله نهایی خروجی تبدیل جانشینی بایت‌ها و جلوگیری از حمله شخص کنجکاو به این نقطه خواهد شد. پیاده‌سازی ضرب همزمان در FPGA بسادگی قابل تحقق است زیرا قابلیت انجام عملیات همزمان جز ویژگی های این تراشه است. چنانچه به روش پیشنهادی توجه کنیم متوجه این نکته اساسی خواهیم شد که روش مزبور منحصر به پیاده‌سازی سخت‌افزاری نبوده و قابل اعمال به پیاده سازی‌های نرم‌افزاری و سایر بسترها مانند کارت‌های هوشمند، پردازشگرهای سیگنال دیجیتال و سایر توکن های امنیتی نیز هست. هیچ پارامتر دیگری بجز کلید رمز مورد پردازش قرار نمی‌گیرد و چنانچه در بخش بعد نشان خواهیم داد این روش علاوه بر فراهم آوردن امنیت بالا در برابر حمله تحلیل توان هزینه پیاده‌سازی کمی را به کاربر تحمیل کرده و باعث کاهش سرعت سیستم نیز نخواهد شد و لذا تاثیر کمی بر میزان بازدهی سیستم خواهد داشت.

۵ - حمله به یک سیستم واقعی

برای بررسی میزان موثر بودن روش پیشنهادی، دو نسخه از الگوریتم AES، یک نسخه محافظت شده و یک نسخه محافظت نشده در یک معماری حلقه بشکلی که تنها یک قالب از داده‌ها بطور همزمان مورد پردازش قرار می‌گیرند با استفاده از کدهای قابل سنتز زبان توصیف سخت‌افزار Verilog پیاده‌سازی شده و بر روی تراشه Xilinx Spartan-II FPGA سنتز شدند. کلید رمز در هر دو الگوریتم ۱۲۸ بیتی بوده است. پیاده‌سازی هسته رمزنگار در نسخه محافظت نشده بر روی این تراشه، 983 CLB slices یا ۴۱٪ مساحت روی تراشه را اشغال نمود در حالیکه همان پیاده‌سازی برای نسخه محافظت شده با ۳۲ ساختار SBox، ۱۰۵۰ CLB slices مساحت اشغال کرد که تنها ۷٪ بیشتر از مقدار قبلی و بسیار کمتر از مقادیر گزارش شده برای سایر روش‌ها در ادبیات مربوطه تاکنون است. ابزار سنتز سرعت پالس ساعت را در هر دو حالت 42.2 MHz نشان داد. تجهیزات اصلی اندازه‌گیری چنانکه در شکل ۸ نشان داده شده است شامل یک برد FPGA، یک اسیلوسکوپ دیجیتال MSO7034A ساخت کارخانه Agilant با پهنای باند 350 MHz و فرکانس نمونه برداری 2 GS/sec است. برد شامل دو تغذیه جداگانه 3.3 V برای تغذیه I/O و یک تغذیه 2.5 V برای تغذیه هسته تراشه است. در آزمایشات انجام شده تنها تغذیه هسته مورد بررسی و اندازه‌گیری قرار گرفت. یک مقاومت کم اهم (۱۰ اهم) بین تغذیه برد و زمین قرار داده شد تا تغییرات جریان یا تغییرات توان اندازه‌گیری شود. بمنظور کاهش نویز الکترونیک و سوئیچینگ و نیز افزایش دقت حمله فرکانس کار تراشه را تا۱۰ کیلو هرتز کاهش داده، اندازه گیری‌ها را ده بار تکرار کرده و از نمونه‌های بدست آمده متوسط گرفتیم.



شکل ۸- تجهیزات آزمایشگاهی بکار گرفته شده برای پیاده‌سازی حمله

در فرم اولیه حمله تحلیل تفاضلی توان به AES، یک تابع انتخاب D بعنوان تابع محاسبه کننده مقدار یک بیت b که جزئی از یک بردار میانی S_1 است در نظر گرفته می‌شود. می‌توان b را بشکل زیر محاسبه کرد.

$$b = \text{one output bit of } S_1 (P_i \oplus K_1) \qquad (۸)$$

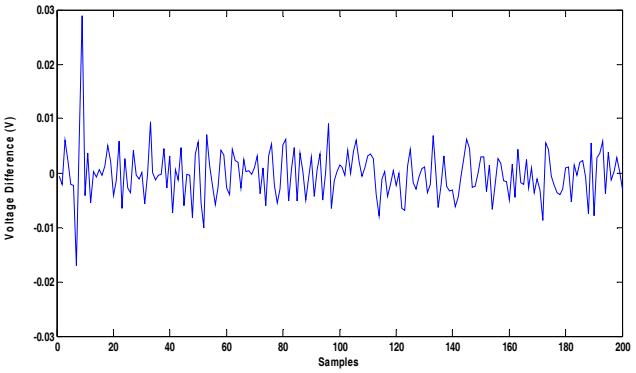
در معادله (۸)، S_1 نشان دهنده تبدیل جانشینی بایت‌ها در دور اول، P_i معرف i امین متن آشکار تصادفی و K_1 اولین زیر کلید است. بمنظور افزایش نسبت سیگنال به نویز و نیز دقت حمله بجای در نظر گرفتن یک بیت از خروجی تبدیل جانشینی بایت‌ها، یک مجموعه چهار بیتی مورد بررسی قرار گرفت بدین مفهوم که تابع انتخاب هنگامی که وزن همینگ خروجی بیش از چهار باشد مقدار یک و در غیر اینصورت مقدار صفر را برمیگرداند. مرجع [۲۴] نشان داده است که به این ترتیب میزان قله‌های ثانویه[۱۸] و قله‌های شبح[۱۹] در منحنی‌های تفاضل توان کاهش می‌یابند.

نتایج آزمایشات انجام شده برای مشاهدات تفاضل توان برای کلیدهای درست و غلط متناظراً در شکل‌های ۹ و ۱۰ نشان داده شده است. آزمایشات بروشنی قابل اندازه‌گیری بودن وزن همینگ خروجی تبدیل جانشینی بایت‌ها را نشان داد. بازیابی بایت اول کلید نیاز به تقریبا ۱۰۰۰ اندازه‌گیری توان بازای ۱۰۰۰ زوج متن آشکار- متن رمز داشت. مقدار زیر کلید صحیح چنانچه در شکل پیداست برابر 0x3C است.

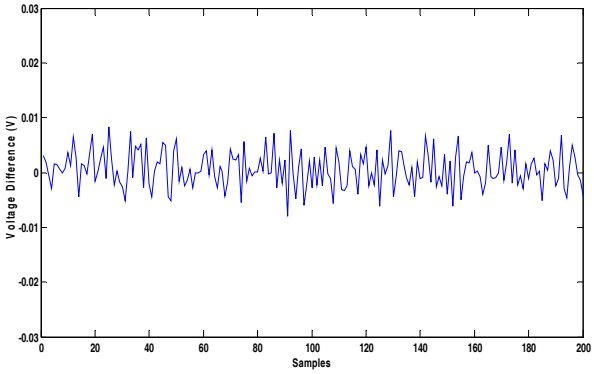
برای پیاده‌سازی حمله تحلیل همبستگی توان در ابتدا با استفاده از FPGA تعداد N متن آشکار را رمز می‌کنیم. سپس مصرف توان در اولین پالس ساعت یا پالس ساعتی که عملیات جانشینی بایت‌ها در آن انجام می شود را اندازه گیری می‌کنیم. اندازه گیری ها ده مرتبه تکرار شده و متوسط گیری می‌شود تا اثر نویز های الکترونیک به حداقل برسد. سپس حداکثر مقدار مصرف توان را بازای هر متن آشکار بدست آورده و در یک ماتریس ذخیره می‌کنیم. از اینرو یک ماتریس ستونی Nتایی *مصرف* توان بازای تمام متون آشکار خواهیم داشت که به آن ماتریس مصرف توان سراسری[۲۰] می گوییم. برای اولین تبدیل جانشینی بایت‌ها یک تابع انتخاب D در نظر می‌گیریم.

تابع انتخاب در اینجا تعداد بیتهایی است در رجیستر خروجی تبدیل جانشینی بایت‌ها بازای هر ورودی تغییر می‌کند. سپس با استفاده از یک ابزار شبیه سازی مقدار D را بازای تمام ۲۵۶ کلید ممکن پیش بینی کرده و ذخیره می کنیم. لذا یک ماتریس پیش بینی بابعاد N×256 خواهیم داشت که به آن ماتریس

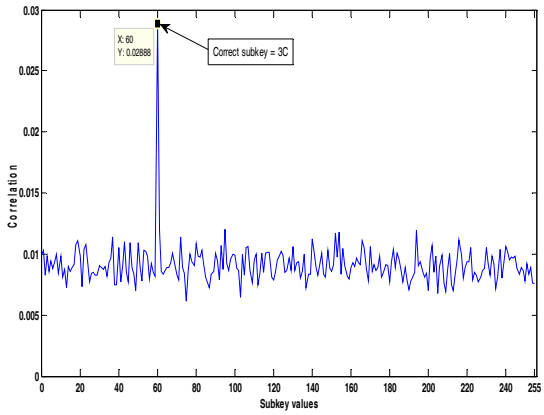
پیش‌بینی سراسری[۲۱] می‌گوییم و عناصر آن اعداد صحیح بین ۰ تا ۷ هستند. سپس ضریب همبستگی بین ماتریس مصرف توان و تمام ستون‌های ماتریس پیش بینی (متناظر با تمام ۲۵۶ حدس کلید) را محاسبه می‌کنیم. اگر حمله موفقیت آمیز باشد انتظار می رود تا تنها یک مقدار که همان حدس صحیح از کلید است مقدار کاملا بزرگتری در مقایسه با دیگران داشته باشد. نتیجه آزمایش حمله همبستگی برای اولین بایت کلید در شکل ۱۱ نشان داده شده است و چنانچه از شکل پیداست مقدار ضریب همبستگی در حدس صحیح یا 0x3C بصورت یک پیک کاملا واضح است. آزمایشات ما نشان داد که پس از تکمیل اندازه گیری ها کشف کامل ۱۲۸ بیت کلید با استفاده از یک کامپیوتر چهار هسته‌ای با فرکانس 2.5 GHz تقریبا در دو ساعت امکان‌پذیر است.



شکل ۹- منحنی تفاضل توان برای یک حدس صحیح از زیر کلید اول

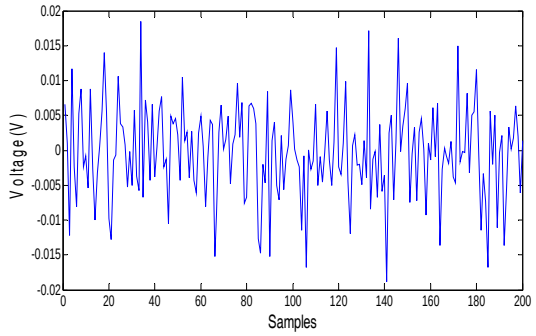


شکل ۱۰- منحنی تفاضل توان برای یک حدس غیر صحیح از زیر کلید اول

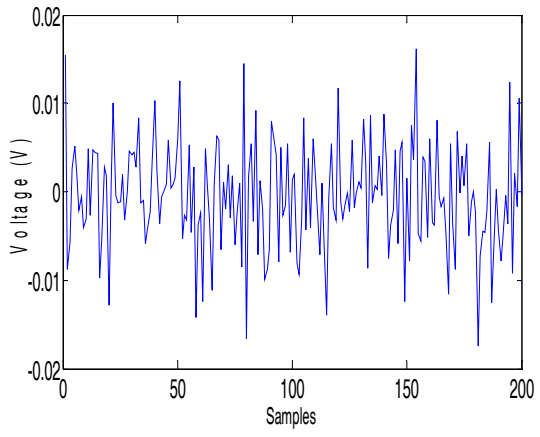


شکل ۱۱- حدس صحیح زیر کلید اول با استفاده از حمله تحلیل همبستگی توان

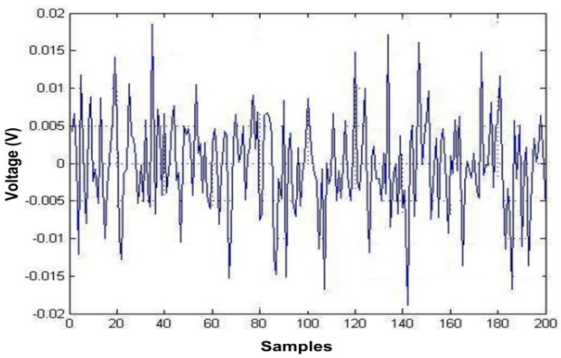
آزمایشات فوق برای نسخه محافظت شده تکرار شد. شکل‌های ۱۲ و ۱۳ نشان دهنده منحنی‌های تفاضل توان برای حدس صحیح و یک حدس غلط از زیر کلید اول هستند. همانگونه که از این دو شکل مشخص است زیر کلید صحیح قابل تشخیص از زیر کلید اشتباه نیست زیرا نویز زیادی در سیستم وجود دارد. شکل‌های ۱۴ و ۱۵ نشان‌دهنده منحنی مشاهده توان در نسخه حفاظت شده بازای دو ورودی یکسان است که همانطور که از شکل پیداست با اینکه در هر دو حالت ورودی برابر است اما تفاضل مصرف توان اصلا شبیه به یکدیگر نیست. شکل ۱۶ نشان‌دهنده آزمایش همبستگی برای نسخه محافظت شده پس از ثبت مصرف توان ۱۰۰۰ متن رمز شده است. همانگونه که از شکل پیداست هیچ پیک واضحی در شکل دیده نمی‌شود و تنها باندی از اعداد تصادفی دیده می شود. آزمایش فوق مجددا برای ۶۰۰۰ متن آشکار و رمز شده معادل آن تکرار گردید و باز هم کلید صحیح قابل تشخیص از کلیدهای اشتباه نبود. قابل ذکر است که در این کار حملات توان از مرتبه بالاتر مورد بررسی قرار نگرفتند و تحقیق در مورد میزان مقاومت روش پیشنهادی در برابر چنین حملاتی از جمله برنامه‌های آتی تحقیقاتی است. برای مقاوم تر کردن هر چه بیشتر پیاده‌سازی در مقابل حمله توان می‌توان روش ارائه شده در این مقاله را با سایر تکنیک‌های تصادفی‌سازی مصرف توان مانند اجرای روال‌های شبه تصادفی یا جابجا کردن ترکیب کرد. همانطور که می‌دانیم ترتیب ۱۶ بایتی که در هر قالب وارد تبدیل جانشینی بایت‌ها می‌شود اهمیتی ندارد و می‌توان ترتیب آنها را جابجا کرد. از اینرو می توان با عوض کردن ترتیب ورود و خروج بایت‌ها به این تبدیل شخص کنجکاو را دچار گمراهی بیشتری کرد. اجرای روال‌های تصادفی باعث کاهش بازدهی خواهد شد اما برهم زدن ترتیب بایت‌ها باعث کاهش بازدهی نخواهد شد.



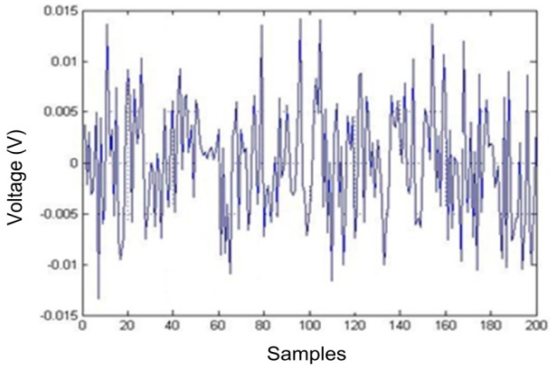
شکل ۱۲- منحنی تفاضل توان برای حدس صحیح از زیر کلید اول در نسخه محافظت شده



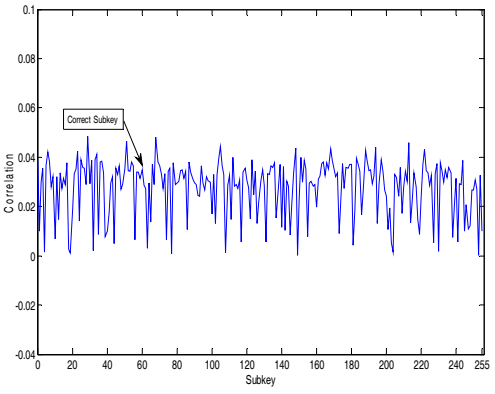
شکل ۱۳- منحنی تفاضل توان برای حدس غیر صحیح از زیر کلید اول در نسخه محافظت شده



شکل ۱۴- منحنی مشاهده توان در نسخه حفاظت شده بازای یک ورودی دلخواه



شکل ۱۵- منحنی مشاهده توان در نسخه حفاظت شده بازای همان ورودی پس از اجرای مجدد الگوریتم



شکل ۱۶- منحنی تحلیل همبستگی توان در نسخه محافظت شده

۶- چالش‌های مقابله با حمله تحلیل توان

در سال‌های اخیر تحقیقات زیادی در مورد جلوگیری از حمله تحلیل توان انجام شده است. بر این مبنا روش‌های مقابله با حمله تحلیل توان را می توان به دو گروه عمده تقسیم کرد: روش‌های نرم افزاری و روش‌های سخت‌افزاری. عمده روش‌های مطرح شده روش‌های نرم‌افزاری بوده و سعی در مقابله با حمله با استفاده از روش‌های مقابله در سطح الگوریتم دارند مانند ماسک کردن دیتا با استفاده از اعداد تصادفی. برخی از این روش‌ها قابل اعمال به سخت‌افزارهای اختصاصی مانند FPGA نیز هستند. روش‌های مقابله سخت‌افزاری عمدتا شامل روش‌های یکنواخت کردن منحنی مصرف توان و یا تغییرات در سطح گیت‌ها و ترانزیستورها هستند. این روش‌ها معمولا بدون هماهنگی با کارخانه سازنده FPGA بسختی قابل پیاده‌سازی هستند اما برخی روش‌های مبتنی بر تغییر در معماری پیاده‌سازی قابل تحقق در FPGA هستند. برخی از روش‌ها که بطور حسی و شهودی روش‌های موثری هستند مانند تصادفی کردن اجرای کدها یا افزودن ماژول‌های مصرف کننده توان بطور تصادفی مانند تاخیر دهنده‌ها یا شیفت دهنده‌های تصادفی و یا فروبرنده‌های جریان بسختی می‌توانند در مقابل این حمله مقاومت کنند. روش‌های جدید سعی در از بین بردن وابستگی مصرف توان ابزار رمزنگاری با مقادیر سیگنال‌ها در گره‌های درونی مدار با استفاده از تصادفی‌سازی یا هموار کردن مصرف توان دارند اما همچنانکه گفته شد هیچ کدام از این روش‌ها نمی‌توانند امنیت کاملی را در مقابل این حملات فراهم آورند و تنها باعث مشکل‌تر شدن کار شخص کنجکاو و نیاز وی به اندازه‌گیری‌های بیشتر خواهند شد. تصادفی کردن مصرف توان عمدتا با ماسک کردن یعنی تصادفی کردن مقادیر در گره‌های میانی مدار بدون آنکه مقدار نهایی تغییر کند یا در سطح گیت استفاده از ماسک بیت‌هایی که باعث متعادل شدن احتمال گذار در خروجی گیت‌ها می شود انجام می‌شود. هموار کردن منحنی مصرف توان در سطح مدار با استفاده از منطق هایی مانند منطق تفاضلی پویا[۲۲] که از آن به منطق دو ریلی با پیش شارژ[۲۳] نیز تعبیر می شود و باعث می شود تا هر گیت منطقی در هر سیکل تنها یک بار شارژ شود. البته افزایش مقاومت در برابر حمله تحلیل توان بدون هزینه بدست نخواهد آمد. ماسک کردن داده‌ها در سطح الگوریتم باعث افزایش سطح اشغالی بر روی تراشه حداقل یک و نیم برابر و ماسک کردن در سطح ترانزیستور یا گیت بین دو تا پنج برابر باعث افزایش مساحت اشغالی روی تراشه خواهد شد. مثلا همان منطق دو ریلی با پیش شارژ که در مورد آن بحث شد باعث افزایش سطح تا سه برابر خواهد شد. استفاده از منطق تفاضلی یعنی استفاده از "10" بجای "0" و "01" بجای "1" باعث کاهش نسبت سیگنال به نویز و ثابت ماندن وزن همینگ اطلاعات رد و بدل شده خواهد شد اما هزینه سخت‌افزاری را تقریبا تا صد درصد افزایش می‌دهد. برخی تکنیک‌ها مانند طراحی سلول‌های اختصاصی تا حدی باعث کاهش هزینه مقابله می‌شود اما همچنان هزینه بالاست. افزودن فیلتر برای ماسک کردن مشخصات طیفی یکی از راه‌های موثر برای مخفی کردن وابستگی توان مصرفی دستورالعمل‌ها به عملوندهاست. باید توجه داشت که شخص باید در ارائه روش مقابله و ادعا در مورد کارایی روش خود کاملا محتاط باشد زیرا بسیاری از روش‌ها که بلحاظ ظاهری یا حتی تئوری در مقابل حملات کانال جانبی مقاوم هستند در عمل در مقابل روش‌های پیچیده حمله چندان تاب نمی آورند و مغلوب می‌شوند. سوال اساسی اینجاست که آیا می‌توان این تکنیک‌ها را بیشتر بهینه‌سازی کرد و اینکه آیا می‌توان تکنولوژی جدیدی برای مقابله با این حملات ابداع نمود؟ سوال مهم دیگر آن است که آیا می‌توان راه دیگری برای سنجش میزان مقاومت در مقابل حمله بدون پیاده‌سازی حمله بر مبنای ابزارهای آماری یا هر ابزار دیگر برای ارزیابی پیاده‌سازی پیدا کرد؟ آیا می‌توان معیار دیگری برای اندازه‌گیری میزان مقاومت بر مبنای پارامترهای طراحی مانند فاکتور فعالیت[۲۴] یا پروفایل مصرف توان پیدا کرد؟ با چه معیار یا معیارهایی می‌توان میزان مقاومت دو پیاده‌سازی مختلف را با یکدیگر مقایسه نمود؟ این سوالات نمونه‌هایی از سوالات مهم و زمینه‌های تحقیقاتی باز در مورد استفاده از FPGA بعنوان ماژول رمز است که هنوز پاسخ چندان روشنی به آنها داده نشده است.

۷- نتیجه‌گیری

حملات کانال جانبی از جمله مهمترین تهدیدات برای امنیت سیستم‌های رمزنگاری معاصر هستند. اهمیت این حملات بگونه‌ای بوده است که بمحض مطرح

شدن مورد توجه ویژه صاحب نظران این رشته، کارخانه‌های سازنده و نیز موسسات مختلف استفاده کننده از این سیستم‌ها قرار گرفته‌اند. الگوریتم‌هایی نظیر الگوریتم رمزنگاری استاندارد از نظر ریاضی باندازه کافی در مقابل حملات مقاوم هستند اما این کار نشان داد هنگامیکه بر روی ماژول‌های سخت‌افزاری مانند FPGA پیاده‌سازی می‌شوند در مقابل تحلیل توان کاملا آسیب‌پذیرند. گرچه تحقق موفقیت‌آمیز این حملات بر روی تراشه‌های FPGA بمراتب سخت‌تر از کارت‌های هوشمند است اما در این تحقیق ضمن پیاده‌سازی صحیح حمله موفق به بازیابی کلید در مدت زمان حدود دو ساعت شدیم. علاوه بر آن روش ساده جدیدی بر مبنای ساختار ریاضی الگوریتم AES برای مقابله با حمله تحلیل توان ارائه دادیم که بدون آنکه خدشه ای به استاندارد بودن الگوریتم وارد شود باعث افزایش قابل توجه مقاومت پیاده‌سازی با هزینه پیاده‌سازی کم خواهد شد. تقریبا تمام الگوریتم‌هایی که در کارت‌های هوشمند تعبیه می‌شوند بگونه ای طراحی می‌شوند که در مقابل انواع حملات توان مقاوم باشند اما تقریبا هیچ کاری در زمینه آنکه این الگوریتم‌ها را بطور ذاتی در برابر حمله مقاوم کنند انجام نشده است. این پژوهش نشان داد که می توان الگوریتم هایی طراحی کرد که بطور ذاتی در مقابل این حملات مقاوم باشند. محتمل است در آینده حملاتی که ترکیبی از چند منبع اطلاعات کانال جانبی مانند ترکیب حمله تحلیل توان و القا خطا هستند همراه با روش‌های قدرتمندتر آماری و مشاهده همزمان چند عملیات بتوانند بطور جدی‌تر سخت‌افزارهای رمزنگاری را مورد تهدید قرار دهند. بررسی تئوری این مساله و نیز روش‌های مقابله با این حملات بالقوه می‌تواند از جمله موضوعات جذاب و مهم تحقیقاتی برای محققین و نیز صاحبان صنایع باشد.

مراجع

[1] M. Masoomi, M. Masoumi, and M. Ahmadian, "A Practical Differential Power Analysis Attack against an FPGA Implementation of AES Cryptosystem," *IEEE I-Society 2010*, 2010.

[2] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Proc. Crypto*, pp. 388-397, 1999.

[3] F. X. Standaert, L. vanOldeneel, D. Samyde, and J. J. Quisquater, "Power Analysis of FPGAs, How Practical is the Attack?," *Proc. the International Conference on Field Programmable Logic and Application*, pp. 701-711, 2003.

[4] L. T. Mc Daniel, *An Investigation of Differential Power Analysis Attacks on FPGA-based Encryption Systems*, Master Thesis, Virginia Polytechnic Institute, 2003.

[5] N. T. Courtois, and L. Goubin, "An Algebraic Masking Method to Protect AES against Power Attacks," *Proc. International Conference on Information Security and Cryptology*, pp. 199–209, 2006.

[6] M. Masoumi, F. Raissi, and M. Ahmadian, "NanoCMOS-Molecular Realization of Rijndael," *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 285-297, 2006.

[7] C. Clavier, J. S. Coron, and N. Dabbous, "Differential Power Analysis in the Presence of Hardware Countermeasures," *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 252-263, 2000.

[8] J. J. A. Fournier, S. Moore, H. Li, R. Mullins, and G. Taylor, "Security Evaluation of Asynchronous Circuits," *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 125-136, 2003.

[9] S. Moore, R. Anderson, R. Mullins, G. Taylor, and J. J. A. Fournier, "Balanced Self-Checking Asynchronous Logic for Smart Card Applications," *Journal of Microprocessors and Microsystems*, vol. 27, no. 9, pp. 421- 430, 2003.

[10] K. Tiri, D. Hwang, A. Hodjat, B. C. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "Prototype IC with WDDL and differential routing-DPA sesistance assessment," *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 354-365, 2005.

[11] K. Tiri, and I. Verbauwhede, "Securing Encryption Algorithms against DPA at the Logic Level: Next Generation Smart Card Technology," *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 125-136, 2003.

[12] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks, Revealing the Secret of Smart Cards*, Springer, 2006.

[13] S. Mangard, N. Pramstaller, and E. Oswald, "Successfully Attacking Masked AES Hardware Implementations," *Proc. International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 157-171, 2005.

[14] N. T. Courtois, and L. Goubin, "An Algebraic Masking Method to Protect AES against Power Attacks," *Proc. International Conference on Information Security and Cryptology*, pp. 199–209, 2006.

[15] T. Wollinger, J. Guajardo, and C. Paar, "Security on FPGAs: State-of-the-Art Implementations and Attacks," *ACM Transactions on Embedded Computing Systems*, Vol. 3, No. 3, pp. 534–574, 2004.

[16] X. Zhang, and K. K. Parhi, "High-Speed VLSI Architectures for the AES Algorithm," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 12, no. 9, pp. 957-967, 2004.

[17] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A Compact Rijndael Hardware Architecture with S-Box Optimization," *Proc. of the Advances in Cryptology*, pp. 239–254, 2001.

[18] J. P. Deschamps, J. L. Imana, and G. D. Sutter, *Hardware Implementation of Finite-Field Arithmetic*, Mc-Graw Hill, 2009.

[19] A. Rudra, P. K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, and P. Rohatgi, "Efficient Rijndael Encryption Implementation with Composite Field Arithmetic," *Proc. of*

the International Workshop on Cryptographic Hardware and Embedded Systems, pp. 171-184, 2001.

[20] A. Morioka, and A. Satoh, "A 10 Gbps Full-AES Crypto Design with a Twisted-BDD SBox Architecture," Proc. International Conference on Computer Design, pp. 98-103, 2002.

[21] V. Rijmen, *Efficient Implementation of the Rijndael S-Box*, Katholieke Universiteit Leuven, Dept. ESAT. Belgium.

[22] L. Xiao, and H. M. Heys, "Hardware Design and Analysis of Block Cipher Components," *Proc. International Conference on Information Security and Cryptology*, pp. 164-181, 2002.

[23] R. Lidl, and H. Niederreiter, *Introduction to Finite Field and Applications*, Cambridge University Press, 1986.

[24] T-Ha Lee, C. Canovas, and J. Cledier, "An Overview of Side-Channel Analysis Attacks," *Proc. of the ACM symposium on Information, computer and communications security* pp. 33-43, 2008.

اطلاعات بررسی مقاله:

تاریخ ارسال: ۸۹/۱۰/۷
تاریخ اصلاح: ۹۰/۲/۱۴
تاریخ قبول شدن: ۹۰/۳/۱۱
نویسنده مرتبط: مسعود معصومی، دانشکده فنی، دانشگاه آزاد اسلامی واحد اسلامشهر، تهران، ایران.

[1] Side-Channel Attacks
[2] Simple Power Analysis
[3] Differential Power Analysis
[4] Correlation Power Analysis
[5] Complementary Logic
[6] Sense Amplifier Based Logic
[7] Asynchronous Logic
[8] Composite Field Arithmetic
[9] Lookup Tables
[10] Extended Euclid's Theorem
[11] Fermat Little Theorem
[12] Composite Field
[13] Isomorphism
[14] Iterative
[15] Isomorphism Mapping
[16] Decompose
[17] Autoisomorphism
[18] Secondary Peaks
[19] Ghost Peaks
[20] Global Consumption Matrix
[21] Global Prediction Matrix
[22] Differential Dynamic Logic
[23] Precharged Dual Rail Logic
[24] Activity Factor

مسعود معصومی کارشناسی خود را در رشته مهندسی الکترونیک درسال ۱۳۷۴ از دانشگاه گیلان با عنوان دانشجوی رتبه اول اخذ نمود و در سال ۱۳۷۹ موفق به اخذ مدرک کارشناسی‌ارشد در همین رشته از دانشگاه صنعتی خواجه نصیر با عنوان دانشجوی رتبه اول گردید. نامبرده در سال ۱۳۸۵ مدرک دکترای خود را از دانشگاه صنعتی خواجه نصیر با ارائه روشی جدید برای پیاده‌سازی الگوریتم‌های رمزنگاری با استفاده از نانوالکترونیک دریافت نمود. مسعود معصومی در سال ۱۳۸۹ پس از طی یک دوره تحقیقاتی یک و نیم ساله در زمینه حملات کانال جانبی به سیستم‌های رمزنگاری دوره فرا دکترا را در دانشگاه صنعتی خواجه نصیر با موفقیت به اتمام رساند. مهمترین زمینه‌های مورد علاقه وی پیاده‌سازی الگوریتم‌های رمزنگاری، پردازش سیگنال و کدینگ بصورت سخت‌افزاری و مجتمع، حملات کانال جانبی به سیستم‌های رمزنگاری و روش‌های مقاوم‌سازی این سیستم‌ها در مقابل چنین حملاتی می‌باشد.
آدرس پست‌الکترونیکی ایشان عبارت است از:
m_masoumi@eetd.kntu.ac.ir

سید مجتبی دهنوی مدرک کارشناسی را در رشته ریاضی کاربردی از دانشگاه علم و صنعت ایران در سال ۱۳۷۹ و کارشناسی‌ارشد را در گرایش جبر و ترکیبیات از دانشگاه صنعتی امیرکبیر در سال ۱۳۸۲ دریافت نموده است. وی در حال حاضر دانشجوی دکترای رشته ریاضیات رمز در دانشگاه تربیت معلم تهران می‌باشد. زمینه‌های مورد علاقه او رمزنگاری متقارن، توابع بولی و عملگرهای پایه در رمزنگاری می‌باشد.
آدرس پست‌الکترونیکی ایشان عبارت است از:
dehnavism@tmu.ac.ir