

ارائه یک حمله SPA موفق به پیاده سازی AES

روی میکروکنترلر PIC

حامد یوسفی^۱، محمود گردشی^۲ و محمد سبزی نژاد فراش^۳
^۱دانشکده فوای دانشگاه جامع امام حسین (ع)، تهران، ایران
^۲دانشکده علوم ریاضی و کامپیوتر دانشگاه تربیت معلم، تهران، ایران

چکیده

در این مقاله یک حمله تحلیل توانی ساده (SPA) به پیاده سازی AES روی میکروکنترلر PIC را معرفی می کنیم. این حمله تبدیل مخلوطساز ستونی (MixColumns) را از الگوریتم AES هدف قرار می دهد. با استفاده از اطلاعات به دست آمده از سیگنال توان مصرفی حین پردازش داده ها در این تبدیل می توان فضای کلید را محدود کرد. این حمله می تواند فضای جستجوی کلید را بسته به تکرار حمله، به طور دقیق تا یک کلید محدود کند. اساس حمله بر مبنای تشخیص نتیجه یک عملگر شرطی در تبدیل مخلوطساز ستونی با استفاده از اثرهای توان است.

واژگان کلیدی: حملات کانال جانبی، حمله تحلیل توانی ساده، الگوریتم AES، میکروکنترلر PIC.

۱- مقدمه

تا اواسط دهه ۱۹۹۰، ابزارهای رمزنگاری، به صورت یک جعبه سیاه در نظر گرفته می شدند. این ابزارها با گرفتن داده ورودی و با استفاده از کلید مخفی خود، خروجی را تولید می کردند. بنابراین حملات با استفاده از متن خوانای شناخته شده، متن رمزی شناخته شده و یا یک زوج متن خوانا و رمزی شناخته شده انجام می شد و به نظر نمی آمد که هیچ گونه اطلاعات اضافه دیگری در دسترس باشد. امروزه مشخص شده که این فرض صحیح نیست. در پیاده سازی های فیزیکی، همواره خروجی های دیگری وجود دارند که باعث نشت اطلاعات از سامانه رمزنگاری می شوند که به آنها اطلاعات کانال جانبی گویند. مهاجم با داشتن دسترسی به رمزکننده و با جمع آوری و تحلیل این داده ها می تواند سامانه رمزنگاری را مورد حمله قرار دهد. به این نوع تحلیل، تحلیل کانال جانبی یا حمله کانال جانبی گویند (Goldack, 2008).

انواع مختلفی از حملات کانال جانبی به سامانه های رمزنگاری ارائه شده اند که هدف همه این حملات به دست آوردن کلید رمزنگاری است. حملات به سامانه های رمزنگاری از نظر هزینه، زمان، تجهیزات و تخصص لازم

متفاوت اند و روش های گوناگونی برای تقسیم بندی آنها وجود دارد. در یک تقسیم بندی، آنها را به صورت فعال و غیر فعال در نظر می گیرند (Mangard et al, 2007, Mangard, 2004). حملات فعال سعی در به دست آوردن کلید با استفاده از برهم زدن عملکرد طبیعی سامانه دارند. در یک حمله فعال، ابزار رمزنگاری یا ورودی ها و یا محیط، طوری دست کاری می شود که سامانه دچار رفتاری غیر معمول شود. سپس با تحلیل این رفتار غیر معمول، سعی در بازیابی کلید می شود. برای مثال، می توان حمله القای خطا^۱ را نام برد که در آن با القای عمدی خطا در محاسبات و استفاده از نتایج آن به سامانه رمزنگاری حمله می کنند. نمونه ای از حمله القای خطا روی AES در (Chen and Yen, 2003) ارائه شده است. در مقابل حملات فعال، حملات غیر فعال قرار می گیرند که عملکرد سامانه را در حالت عادی زیر نظر قرار می دهند و در عملکرد متعارف آن اختلالی ایجاد نمی کنند.

حملات را از جنبه دیگر می توان به تهاجمی^۲، نیمه تهاجمی^۳ و غیر تهاجمی^۱ تقسیم کرد که خود می توانند

^۱Fault Induction Attacks

^۲Invasive

^۳Semi-Invasive

فعال یا غیر فعال باشند (Mangard et al, 2007, Mangard, 2004). در حملات تهاجمی با برداشتن درپوش یا پوشش تراشه، سعی در به دست آوردن اطلاعات از رمزکننده می شود. در این نوع حملات به طور اساسی هیچ گونه محدودیتی در برخورد با رمزکننده برای بازیابی کلید وجود ندارد. به عنوان مثال با وصل کردن یک سیم به گذرگاه داده^۱، می توان اطلاعات مبادله شده را آشکار کرد. از این نوع حمله می توان حمله پروب گذاری^۳ را نام برد که در ایستگاه پروب گذاری و با استفاده از تجهیزات دقیق و میکروسکوپ های با دقت بالا و پروب های بسیار نازک انجام می شود. با این تجهیزات می توان به بازیابی داده ها در نقاط مختلف تراشه و یا مشاهده سلول های حافظه پرداخت. چنین سامانه ای به طور گسترده در صنایع نیمه هادی مورد استفاده قرار می گیرد. این حملات بسیار قدرتمند بوده اما نیازمند تجهیزات با قیمت بالا هستند. در نتیجه کارهای اندکی در این زمینه ارائه شده (Mangard et al, 2007) و همچنین این حملات باعث آسیب فیزیکی به سخت افزار رمزنگاری خواهند شد. یک مدل حمله پروب گذاری روی AES در (Schmidt and Kim, 2008) ارائه شده است.

در حملات نیمه تهاجمی، پوشش تراشه برداشته می شود اما هیچ گونه اتصال الکتریکی مستقیمی به سطح تراشه انجام نمی شود. به طور معمول در حملات نیمه تهاجمی غیرفعال، هدف خواندن محتوای سلول های حافظه بدون استفاده از پروب گذاری است. یک حمله موفق در این مورد در (Samyde et al, 2002) آمده است. در حملات نیمه تهاجمی فعال، هدف القای خطا به سامانه است.

این کار با اشعه X، میدان های الکتریکی و مغناطیسی و یا نور انجام می شود. برای مثال، یک حمله القای خطا به کمک نور را می توان در (Skorobogatov and Anderson, 2003) دید. حملات نیمه تهاجمی به تجهیزات گران قیمت مانند تجهیزات حملات تهاجمی نیاز ندارند؛ اما هنوز برای انجام یک حمله نیمه تهاجمی موفق تلاش بالایی نیاز است. برای مثال فرآیند مشخص کردن مکان دقیق حمله، روی سطح یک تراشه مدرن، به تجربه و زمان کافی نیاز دارد (Mangard et al, 2007). در این زمینه (Skorobogatov, 2005) منبع کاملی است.

حملات غیرتهاجمی دسته دیگر حملات بودند. در این گونه حملات بدون آسیب فیزیکی و برداشتن پوشش تراشه، سعی می شود سامانه شکسته شود. بیشتر حملات غیر

تهاجمی با تجهیزات به نسبت ارزان تری قابل اجرا بوده و بنابراین یک تهدید جدی برای امنیت ابزارهای رمزنگاری هستند. در سال های اخیر حملات غیرتهاجمی غیرفعال بسیار مورد توجه قرار گرفته اند. مهم ترین این حملات از اطلاعاتی مانند زمان، توان مصرفی ابزار و یا تشعشعات الکترومغناطیسی استفاده می کنند که به ترتیب حملات زمانی^۴ (Kocher, 1996)، حملات تحلیل توان^۵ (Kocher et al, 1999) و حملات الکترومغناطیسی (EM)^۶ (Gandolfi et al, 2001, Quisquater and Samyde, 2001) را شامل می شوند. حملات غیرتهاجمی فعال نیز وجود دارند که هدف این حملات تزریق خطا به سامانه بدون برداشتن پوشش آن است. برای نمونه این خطا می تواند به وسیله اختلال^۷ در پالس ساعت، اختلال در توان مصرفی و یا تغییر دمای محیط باشد. در (Whelan et al, 2006) یک جمع بندی از این نوع حملات آمده است.

در حملات زمانی از زمان اجرای الگوریتم و مدتی که طول می کشد تا سامانه به یک ورودی پاسخ دهد، برای انجام حمله استفاده می شود. زمان اجرای یک الگوریتم می تواند حاوی اطلاعاتی درباره مقادیر مخفی در رمزنگاری باشد. به طور معمول برنامه نویسان و طراحان الگوریتم سعی در کاهش زمان اجرا و افزایش سرعت دارند. زمان اجرای الگوریتم، به ازای ورودی های مختلف ممکن است اندکی متفاوت باشد. اگر چنین باشد و زمان اجرای الگوریتم برای ورودی های مختلف به دقت ثبت شود، آنگاه با تحلیل آماری می توان به جستجوی همبستگی بین اندازه گیری های مختلف زمان و بعضی بیت های کلید پرداخت. ایده استفاده از زمان اجرای الگوریتم ابتدا توسط کوچر^۸ (Kocher, 1996) مطرح شد. در مقاله کوچر روشی توضیح داده شد که در آن از اطلاعات زمانی برای به دست آوردن کلید مخفی در یک پیاده سازی RSA استفاده شده بود. در (Koeune and Quisquater, 1999) روشی برای شکستن AES با تحلیل زمان آمده است. این حمله با سه هزار بار اندازه گیری زمان انجام شده است و از اختلاف زمان ایجاد شده در تابع مخلوط ساز ستونی^۹ از الگوریتم AES به ازای ورودی های مختلف استفاده شده است.

ما در این مقاله رمزکننده را با استفاده از تحلیل توان مصرفی مورد حمله قرار می دهیم. در حملات تحلیل توان، به دنبال یافتن وابستگی در سیگنال توان مصرفی سخت افزار

⁴ Timing Attacks

⁵ Power Analysis Attacks

⁶ Electromagnetic (EM) Attacks

⁷ Glitch

⁸ Paul C. Kocher

⁹ Mix-Columns

¹ Non-Invasive

² Data Bus

³ Probing Attack

کلید استفاده شده است. اساس حمله بر مبنای تشخیص وزن همینگ نتایج میانی حین اجرای تابع توسیع کلید در الگوریتم پیاده‌سازی شده برای کارت هوشمند است. موفقیت آن مشروط به تشخیص وزن همینگ داده‌های میانی است و از آنجا که در تشخیص وزن همینگ داده‌ها از روی اثر توان عوامل زیادی تأثیر گذارند (از جمله مصرف توان عملگر و عمل‌گرهای قبل و بعد از آن، وضعیت گذرگاه داده قبل و بعد از اجرای عملگر و همچنین ساختار داخلی سخت‌افزار و ارتباط اجزای مختلف آن)، این تشخیص مشکل شده و باعث ناکارآمدی و پرهزینه شدن حمله می‌شود.

در این مقاله یک حمله SPA موفق روی AES ارائه می‌کنیم و نشان می‌دهیم که در یک پیاده‌سازی ناامن از AES، با استفاده از تحلیل سیگنال توان حاصل از پردازش داده‌های مختلف حین اجرای تبدیل مخلوط‌ساز ستونی، چگونه می‌توان یک حمله تحلیل ساده‌توان موفق و کارآمد داشت. در اینجا از تفاوت ایجاد شده در اثر توان مصرفی رمزکننده به ازای نتایج مختلف یک عمل‌گر شرطی در تبدیل مخلوط‌ساز ستونی استفاده خواهیم کرد.

در ادامه این مقاله، الگوریتم AES و چگونگی اجرای تبدیل مخلوط‌ساز ستونی در بخش دو خواهند آمد. در بخش سه به بیان نقطه آسیب‌پذیر آن برای اعمال حمله می‌پردازیم و در ادامه روش نظری اعمال ایده حمله را در بخش چهار ارائه می‌کنیم. پیاده‌سازی عملی حمله را در بخش پنج خواهیم داشت و در بخش شش نتایج عملی حمله را می‌آوریم. در انتها نیز پس از پیشنهاد راه‌کاری برای مقابله با این تهدید به جمع‌بندی و نتیجه‌گیری می‌پردازیم.

۲- الگوریتم AES

الگوریتم رمز راین‌دال^۴ (Daemen and Rijmen, 2000) توسط NIST در نوامبر ۲۰۰۱ به‌عنوان استاندارد پیشرفته رمزنگاری (AES) معرفی و سپس تحت استاندارد FIPS197 (NIST, 2001) ارائه شد. AES یک الگوریتم رمز قالبی بایت‌گرا با کلید متقارن و طول بلوک ۱۲۸ بیت است. طول کلید می‌تواند ۱۲۸، ۱۹۲ و ۲۵۶ بیت باشد و متناسب با طول کلید، تعداد دورهای الگوریتم به ترتیب برابر ۱۰، ۱۲، و ۱۴ دور است. برای مشخص کردن طول کلید الگوریتم مورد بررسی، AES را به‌صورت AES-128، AES-192 و AES-256 نمایش می‌دهند. هر دور از AES شامل چهار تابع است: جانشینی بیتی^۵، شیفت سطری^۶، مخلوط‌ساز ستونی و جمع

رمزنگاری هستیم. در حالت کلی دو نوع وابستگی در سیگنال توان وجود دارد: وابستگی به عمل‌وند (داده) و وابستگی به عمل‌گر. حملات تحلیل توان از این حقیقت نتیجه می‌شوند که توان مصرفی ابزار رمزنگاری وابسته به داده‌هایی است که پردازش می‌کند و عمل‌گرهایی که اجرا می‌کند. ایده استفاده از این حقیقت در تحلیل ابزارهای رمزنگاری نیز ابتدا توسط کوچر (Kocher et al, 1999) مطرح شد. حملات تحلیل توان را می‌توان به دو دسته ساده^۱ (SPA) و تفاضلی^۲ (DPA) تقسیم کرد. تحلیل توان ساده تکنیکی است که شامل مشاهده و بررسی مستقیم اثرهای توان جمع‌آوری شده حین رمزنگاری است. به عبارت دیگر مهاجم سعی دارد با تعداد محدودی اثر توان، کم و بیش به کلید و یا اطلاعاتی درباره الگوریتم رمزنگاری دست یابد. در SPA اغلب نیازمند داشتن اطلاعاتی از جزئیات پیاده‌سازی ابزار مورد حمله هستیم. در عمل SPA زمانی مفید و مورد استفاده قرار می‌گیرد که یک یا تعداد اندکی اثر توان برای یک مجموعه ورودی داشته باشیم. اما در تحلیل تفاضلی توان، هدف یافتن کلید با استفاده از تعداد زیادی اثر توان است. این سیگنال‌های توان در حین رمزنگاری و یا رمزگشایی بلوک‌های داده مختلف ذخیره شده‌اند و برای تحلیل آنها از روش‌های آماری استفاده می‌شود. در DPA نیازمند داشتن اطلاعات دقیق درخصوص ابزار مورد حمله نیستیم. به‌طور معمول تنها کافی است بدانیم که الگوریتم در حال اجرا روی سخت‌افزار رمزنگار چیست. همچنین در این حمله مهاجم می‌تواند کلید را در شرایط به‌شدت نوفه‌ای نیز کشف کند. در مقایسه با SPA، در DPA نیازمند داشتن اثرهای توان بسیار زیادی هستیم. بنابراین برای اعمال حمله لازم است که ابزار مورد حمله، مدتی در دسترس فیزیکی باشد. نتایج زیادی از اعمال DPA روی الگوریتم‌های مختلف به‌خصوص AES گزارش شده است. در (Masoomi et al, 2010) نمونه‌ای از این حمله آمده است.

ما در اینجا به اعمال یک حمله SPA موفق روی AES می‌پردازیم. حملات SPA با تعداد اندکی اثر توان سعی در شکستن سامانه دارند. در زمینه اعمال SPA روی AES کارهای اندکی صورت گرفته است. در (Mangard, 2003) یک نمونه از این حمله به AES آمده است که پیاده‌سازی تابع توسیع کلید^۳ از AES را هدف قرار داده‌است. در این حمله از نشت اطلاعات و وجود رابطه بازگشتی در محاسبه کلیدهای دور AES در تابع توسیع کلید جهت کاهش فضای

⁴ Rijndael

⁵ SubBytes

⁶ ShiftRows

¹ Simple Power Analysis

² Differential Power Analysis

³ Key Expansion

ضرب کردن '02' در بایت داده، معادل ضرب کردن x در $m(x)$ است و عبارت است از:

$$x \cdot b(x) = b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x \quad (3)$$

که برای رسیدن به جواب باید به پیمانه $m(x)$ کاهش یابد. اگر $b_7 = 0$ باشد پاسخ به خودی خود کاهش یافته است. اما اگر $b_7 = 1$ باشد، آنگاه کاهش با استفاده از کم کردن (XOR کردن) چندجمله‌ای $m(x)$ حاصل می‌شود. عملیات ضرب x در $b(x)$ به پیمانه $m(x)$ را $xTime$ گویند. در واقع ضرب در '02' را می‌توان این چنین سرعت بخشید:

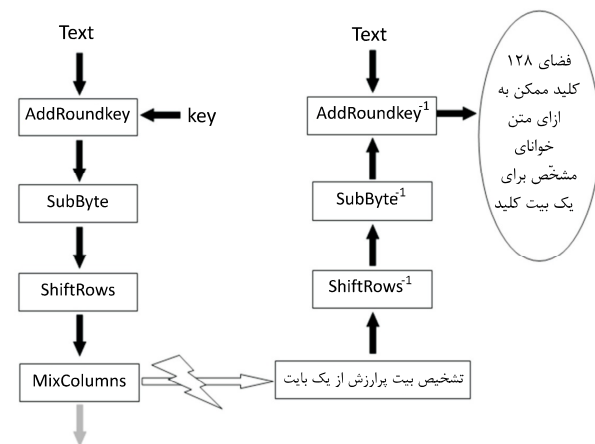
(۱) بایت مورد نظر را یک واحد به چپ شیفت می‌دهیم.

(۲) اگر بیت نقلی^۲ ایجاد شد، نتیجه با '1B' XOR می‌شود.

اگر پیاده‌سازی با دقت انجام نشود، عملیات بالا که شامل یک عمل شرطی است، می‌تواند در حالات مختلف اثر متفاوتی روی نمودار توان بگذارد و باعث نشت اطلاعات شده و مقدار بیت پرارزش بایت ضرب شده را آشکار کند. در این مقاله از این نقطه‌ضعف برای پی‌ریزی حمله استفاده می‌کنیم.

۴- روش اعمال نظری حمله

در قسمت قبل نقطه‌ضعف موجود در تابع مخلوط‌ساز ستونی را بیان کردیم. در این قسمت روش استفاده از این نقطه‌ضعف برای اعمال حمله و بازیابی کلید را بیان می‌کنیم. اساس حمله عبارت است از تشخیص بیت پرارزش بایت‌های ماتریس حالت در تبدیل مخلوط‌ساز ستونی دور اول با استفاده از اثرهای توان.



(شکل ۱): الگوریتم AES تا تبدیل مخلوط‌ساز ستونی دور اول و روند بازگشت الگوریتم با داشتن بیت پرارزش یک بایت در تبدیل مخلوط‌ساز ستونی

² Carry

با زیرکلید دور^۱، که به ترتیب در هر دور اجرا می‌شوند. پیش از دور اول به منظور سفیدسازی، یک مرحله جمع با زیرکلید وجود دارد. همچنین تابع مخلوط‌ساز ستونی در آخرین دور وجود ندارد. طول بلوک ۱۲۸ بیت است و در آغاز عملیات رمزنگاری، پیام به بلوک‌های ۱۲۸ بیتی (۱۶ بایتی) تقسیم می‌شود و به صورت ماتریسی ۴×۴ نمایش داده می‌شود که به آن ماتریس حالت گویند. جمع با زیرکلید دور عبارت است از یک عملیات ساده XOR که در آن عناصر ماتریس حالت با زیرکلید هر دور ($RoundKey$) بایت به بایت XOR می‌شوند. جانشینی بایتی یک تابع غیر خطی است که هر بایت از ماتریس حالت را با بایت متناظر آن در جدول $S-Box$ جایگزین می‌کند. جدول $S-Box$ شامل مقادیر معکوس ضربی همه حالت‌های ممکن یک بایت (۲۵۶ حالت) در $GF(2^8)$ با یک تبدیل آفینی است. تبدیل شیفت سطری یک تابع خطی است که سطر اول را تغییر نمی‌دهد و سطرها دوم، سوم و چهارم را به ترتیب یک بایت، دو بایت و سه بایت به صورت چرخشی به سمت چپ انتقال می‌دهد. در تبدیل مخلوط‌ساز ستونی، ضرب ماتریسی زیر روی ستون‌های ماتریس حالت اعمال می‌گردد.

$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

منظور از عمل گر• ضرب در فضای $GF(2^8)$ است.

این ضرب‌ها به صورت ضرب چندجمله‌ای‌های باینری به پیمانه چند جمله‌ای تحویل‌ناپذیر $m(x) = x^8 + x^4 + x^3 + x + 1$ روی $GF(2)$ تعریف می‌شوند.

۳- نقطه آسیب‌پذیر

در هنگام پیاده‌سازی AES، تابع مخلوط‌ساز ستونی را با توجه به ماتریس بالا و چندجمله‌ای تحویل‌ناپذیر $m(x)$ می‌توان بسیار کارآمد پیاده‌سازی کرد (NIST, 2001). ماتریس ضربی فقط شامل عناصر '01'، '02' و '03' است و از آنجا که '01' = '02' + '03' است، پس تنها ضربی که باید در عمل پیاده‌سازی شود همان ضرب در '02' است. همچنین یک بایت داده را می‌توان به صورت چند جمله‌ای زیر در نظر گرفت:

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 : \{b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0\} \quad (1)$$

و برای $m(x)$ داریم:

$$m(x) = x^8 + x^4 + x^3 + x + 1 : \{100011011\} : \{01\}\{1B\} \quad (2)$$

¹ AddRoundKey

مراحل بالا را تکرار می‌کنیم تا فضای کلید به تعداد مورد نظر کاهش یابد. اگر تعداد تکرار کافی باشد، آنگاه فقط یک کلید به‌عنوان اشتراک همه فضاهاى کلید باقی می‌ماند که همان بایت کلید صحیح است.

۵- پیاده‌سازی عملی حمله

برای عملی کردن حمله خود ابتدا الگوریتم *AES* را مطابق استاندارد *FIPS197* روی یک میکروکنترلر هشت‌بیتی *PIC* پیاده‌سازی کردیم و بدین ترتیب سخت‌افزار رمزکننده خود را آماده ساختیم. میکروکنترلر *PIC* محصول شرکت *Microchip* است و سهم به‌سزایی در بازار میکروکنترلرها و کارتهای هوشمند دارد.

رمزکننده ساخته‌شده از طریق درگاه *USB* به رایانه متصل می‌شود. یک رابط کاربری^۱ روی رایانه با استفاده از محیط *C#* از مجموعه *Visual Studio* تهیه کردیم که با رمزکننده ارتباط برقرار می‌کند و قالب‌های^۲ ۱۲۸ بیتی داده را برای آن ارسال می‌کند. سخت‌افزار رمزکننده با دریافت یک قالب داده، آن را رمز می‌کند و نتیجه را به رایانه باز می‌گرداند. حال با داشتن سخت‌افزار به عملی‌سازی حمله خود می‌پردازیم.

پیاده‌سازی حمله شامل دو مرحله است، یکی نمونه‌برداری و ذخیره اثرهای توان مصرفی رمزکننده حین رمزنگاری و سپس تشخیص بیت پر ارزش بایتهای ماتریس حالت در تبدیل مخلوط‌ساز ستونی و دیگری استفاده از بیت تشخیص داده شده برای یافتن یک فضای کلید متناظر با داده ورودی.

برای نمونه‌برداری از اثر توان باید از یک اسلوسکوپ دیجیتال حافظه‌دار استفاده کنیم. در اینجا ما از اسلوسکوپ *RIGOL DS1202CA* استفاده کرده‌ایم. این اسلوسکوپ دارای پهنای باند *200MHz* و نرخ نمونه‌برداری *2GSa/S* است. همچنین دارای قابلیت اتصال به رایانه از طریق *USB* است. همچنین می‌تواند سیگنال‌های گرفته شده را روی *USB Flash Memory* ذخیره کند. از دیگر امکانات مفید این اسلوسکوپ قابلیت برنامه‌پذیری و کنترل از طریق رایانه است. کنترل این ابزار با استفاده از توابع موجود در *VISA API* امکان‌پذیر است. این توابع زیرمجموعه‌ای از *NI-VISA* هستند که به‌صورت استاندارد برای کار با وسایل اندازه‌گیری و آزمایشگاهی قابل اتصال به رایانه تدوین شده

در سمت چپ از (شکل ۱) الگوریتم *AES* تا رسیدن به خروجی اولین تبدیل مخلوط‌ساز ستونی آمده است. حال اگر بیت پرارزش بایتهای ماتریس حالت را در تبدیل مخلوط‌ساز ستونی تشخیص داده باشیم، می‌توانیم در جهت عکس نمودار به سمت بالا حرکت کنیم تا به یک فضای کلید کاهش‌یافته برسیم.

با داشتن بیت پرارزش بایتهای ماتریس حالت، وقتی از *SubBytes⁻¹* عبور کنیم یا نیمه بالایی جدول معکوس *S-Box* انتخاب می‌شود و یا نیمه پایینی آن که ۱۲۸ عضو ممکن از جدول برای هریک از حالات صفر یا یک بودن بیت پرارزش انتخاب می‌شوند. حال با *XOR* کردن فضای انتخابی با بایت متناظر از متن خوانا، یک فضای کلید با ۱۲۸ عضو برای یک بایت از کلید به‌دست می‌آید.

اگر عمل بالا را برای ورودی دیگری تکرار کنیم و بیت پرارزش از همان بایت ماتریس حالت را تشخیص دهیم، این‌بار نیز یا نیمه بالایی جدول معکوس *S-Box* انتخاب می‌شود و یا نیمه پایینی آن که این‌بار نیز ۱۲۸ عضو ممکن انتخاب می‌شوند. حتی اگر قسمت انتخاب شده از جدول معکوس *S-Box* تکراری باشد، از آنجا که داده ورودی متفاوت بوده، با *XOR* کردن فضای انتخابی با بایت متناظر از متن خوانا، یک فضای کلید ۱۲۸ عضوی جدید برای همان بایت از کلید به‌دست می‌آید که با فضای به‌دست آمده قبلی حداقل یک اشتراک دارد (کلید صحیح).

با تکرار روند بالا برای ورودی‌های مختلف، هر بار یک فضای کلید ۱۲۸ عضوی برای کلید به‌دست می‌آید که فضای کلید نهایی عبارت است از اشتراک همه فضاهاى کلید به‌دست آمده.

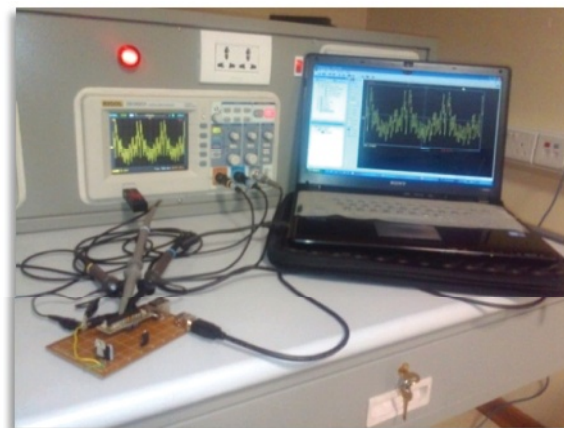
پس می‌توانیم روش حمله خود را به‌صورت زیر مرتب کنیم:

۱. انتخاب یک متن خوانا
۲. اعمال متن به رمزنگار و نمونه‌برداری از توان مصرفی
۳. تشخیص بیت پرارزش یک بایت از ماتریس حالت با استفاده از شکل موج اثر توان در تبدیل مخلوط‌ساز ستونی دور اول
۴. استفاده از بیت تشخیص داده شده طبق (شکل ۱) برای یافتن فضای کلید
۵. اشتراک‌گیری بین فضای کلید به‌دست آمده و اشتراک فضاهاى کلید قبلی

¹ Application Interface

² Blocks

است. رایانه دستورات را به صورت متوالی از طریق *USB* و یا *RS-232* به ابزار ارسال و ابزار تقاضای داده شده را اجرا می کند (RIGOL, 2009). از آنجا که در حملات تحلیل توان باید اثرهای توان زیادی را ذخیره کنیم، قابلیت برنامه پذیری اسیلوسکوپ برای سرعت بخشیدن به عملیات حمله بسیار مؤثر خواهد بود. به منظور کنترل کردن اسیلوسکوپ از طریق رایانه نیز یک رابط کاربری با استفاده از *Visual C#* تهیه کردیم که با اسیلوسکوپ ارتباط برقرار کرده و در هنگام رمزنگاری، با ارسال دنباله دستورات مناسب به اسیلوسکوپ، اثر توان مصرفی را روی حافظه فلش ذخیره می کند. در (شکل ۲) تجهیزات حمله را مشاهده می کنید.



(شکل ۲): تجهیزات حمله شامل اسیلوسکوپ و رایانه

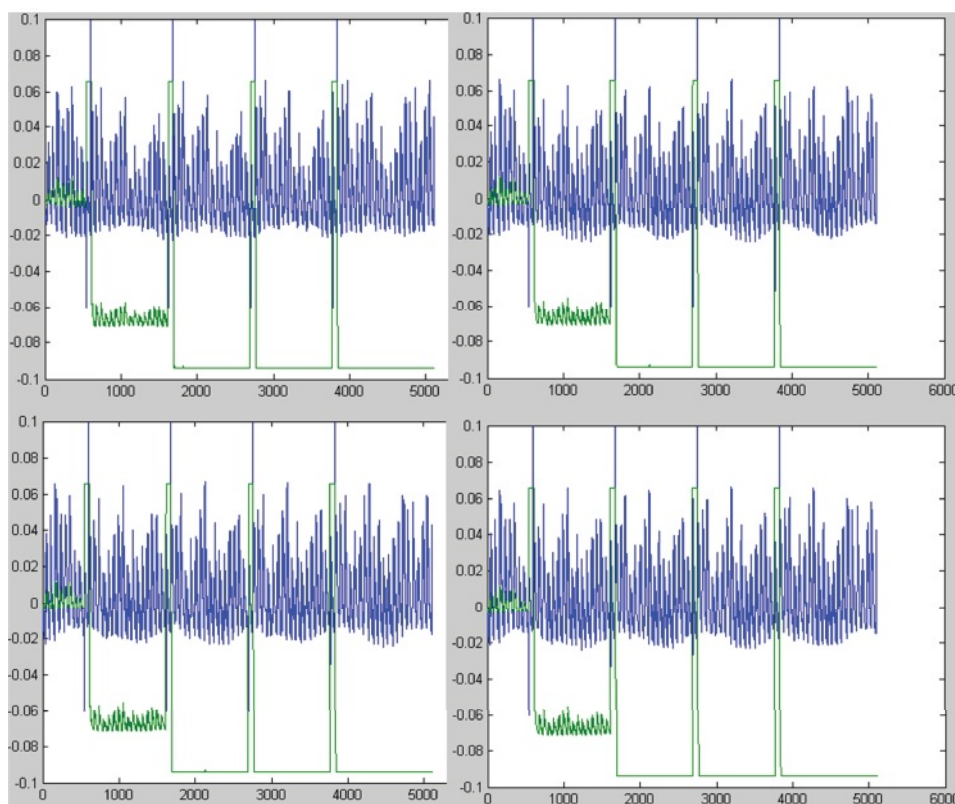
روش کلی کنترل و همزمان سازی اسیلوسکوپ و سخت افزار رمزکننده برای اعمال حمله این چنین است: برای شروع یک نمونه برداری از سیگنال توان، ابتدا رایانه اسیلوسکوپ را (که حافظه *USB Flash* به آن وصل است) آماده می کند و اسیلوسکوپ منتظر دریافت سیگنال چکانه^۱ برای شروع نمونه برداری می ماند. این چکانه به صورت خارجی و از طریق رمزنگار تولید خواهد شد. علت این امر آن است که به محض شروع یک عملیات خاص، اسیلوسکوپ شروع به نمونه برداری کند و داده ای از دست نرود. حال رایانه داده مورد نظر برای رمزنگاری را به رمزنگار می فرستد. سپس به رمزنگار فرمان شروع می دهد. رمزنگار با دریافت این دستور عملیات خود را آغاز می کند و در ابتدای عملیات سیگنال، چکانه را نیز تولید می کند. با این کار اسیلوسکوپ شروع به نمونه برداری کرده و یک نمونه از اثر توان مصرف شامل ۵۱۲۰ نقطه برداشته می شود. وقتی رمزنگار عملیات خود را تمام کرد، اتمام کار خود را به رایانه اعلام می کند. در این لحظه رایانه با ارسال دنباله دستورات مناسب به اسیلوسکوپ، نمونه توان برداشته شده را روی حافظه فلش

ذخیره و سپس اسیلوسکوپ را برای نمونه برداری بعدی آماده می کند. نمونه های توان لازم در این مقاله به ترتیب بالا و درحالی که فرکانس کاری سخت افزار رمزنگار 16MHz بوده است تهیه شده اند. در حملات تحلیل توان برای داشتن یک نتیجه خوب باید اثر نوبه را حذف کرد. برای حذف نوبه، عملیات رمزنگاری را برای یک داده مشخص چندین بار تکرار کرده و اثرات توان را ذخیره می کنیم. سپس با انتقال داده ها به رایانه، از این اثرها میانگین گیری می کنیم. لازم به ذکر است که اسیلوسکوپ اطلاعات خود را با قالب تک ستونی *CSV*^۲ شامل ۵۱۲۰ نقطه از شکل موج نمونه برداری شده ذخیره می کند، میانگین گیری و رسم نمودارهای توان در محیط نرم افزار *MATLAB* انجام شده است. در نرم افزار *MATLAB* با استفاده از دستور *'importdata'* داده ها را به محیط نرم افزار فراخوانی می کنیم و پردازش های خود را روی آنها انجام می دهیم. در (شکل ۳) اثرات توان مصرفی سخت افزار رمزکننده حین اجرای تابع مخلوط ساز ستونی برای چهار بار تکرار عملیات رمزنگاری به ازای ورودی ثابت *'0x00'* آمده است. باتوجه به شکل، می توان تفاوت هایی در اثرات ذخیره شده، به خصوص در قله ها، دید و این به دلیل تأثیر نوبه است.

بعد از تهیه سیگنال ها و میانگین گیری و آماده سازی آنها باید به تشخیص بیت پرارزش بایتهای ماتریس حالت در تابع مخلوط ساز ستونی دور اول پردازیم. برای تشخیص بیت پرارزش باید اطلاعاتی درخصوص پیاده سازی الگوریتم داشته باشیم. همان طور که در قبل گفته شد، اگر عمل *XOR* با *{Ib}* بعد از عمل انتقال به چپ اجرا شود، می توان نتیجه گرفت که مقدار بیت پرارزش بایت داده قبل از انتقال به چپ *'I'* و در غیر این صورت *'0'* بوده است. پس اگر بتوانیم اجرا یا عدم اجرای *XOR* با *{Ib}* را تشخیص دهیم، درواقع وضعیت بیت پرارزش بایت موردنظر را مشخص کرده ایم. ممکن است که الگوریتم طوری پیاده سازی شده باشد که تعداد سیکل دستورالعمل در حالت *'I'* بودن بیت و اجرای *XOR* با تعداد سیکل دستورالعمل در حالت *'0'* بودن بیت و اجرا نشدن *XOR* یکسان نباشد. در این حالت به راحتی می توان با داشتن سیگنال توان مصرفی حین اجرای عمل ضرب بایت در *'02'* و مشاهده تعداد سیکل های دستورالعمل در اثر توان، حالتی را که بیت موردنظر *'I'* بوده و *XOR* اجرا شده را تشخیص داد. اما در یک پیاده سازی امن تر می توان پیاده سازی را طوری انجام داد که در هر دو حالت صفر و یا یک بودن بیت، اجرای برنامه با تعداد سیکل ساعت یکسان

² Comma Separated Value

¹ Trigger



(شکل ۳): اثرهای توان مصرفی حین اجرای تابع مخلوط‌ساز ستونی برای چهار بار تکرار عملیات رمزنگاری به‌ازای ورودی ثابت '0x00'

در (شکل ۴ - الف) بیت مورد نظر از بایت اول، صفر است و به‌جای XOR یک NOP اجرا شده است. اما در (شکل ۴ - ب) بیت مورد نظر یک بوده و XOR انجام شده است. در قسمت‌های مشخص‌شده از دو سیگنال در (شکل ۴)، تفاوت سطح توان مصرفی به‌خاطر این تفاوت تصمیم‌گیری مشخص است.

برای انجام این حمله، مراحل تحلیلی حمله و اشتراک‌گیری فضاهای کلید به‌دست آمده به‌صورت نرم-افزاری انجام می‌شود. این نرم‌افزار در محیط $C++$ تهیه شده است و با گرفتن دنباله بیت‌های به‌دست آمده به‌عنوان بیت پرارزش بایتی از ماتریس حالت در حین اجرای تابع مخلوط-ساز ستونی دور اول در تکرارهای مجدد آزمایش با ورودی متفاوت، به محدود کردن فضای کلید می‌پردازد.

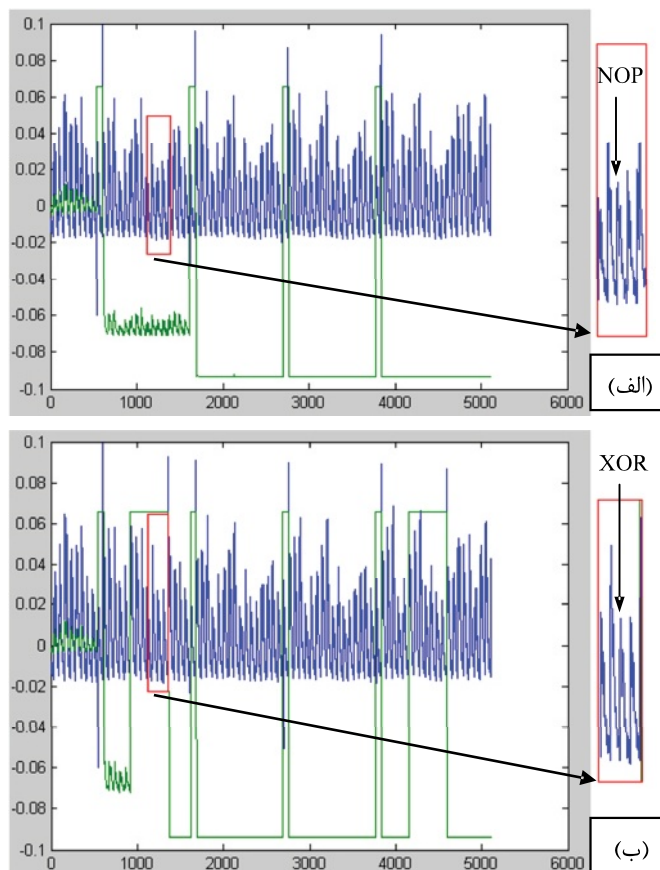
۶- نتایج عملی و تحلیل‌ها

در اینجا نتایج حاصل از اعمال حمله روی پیاده‌سازی و روند محدود شدن مرحله به مرحله فضای کلید را بیان می‌کنیم. این مراحل را برای بایت اول کلید عنوان می‌کنیم و برای دیگر بایت‌های کلید نیز مشابه همین روند وجود دارد. در اولین آزمایش، بایت اول متن خوانا را $0x00$ قرار داده و با شروع رمزنگاری حمله خود را اعمال کردیم که

ادامه یابد. از آنجاکه ما از میکروکنترلر PIC به‌عنوان بستر پیاده‌سازی استفاده کرده‌ایم، پس باتوجه به برگه اطلاعاتی این میکروکنترلر (*Microchip, 2007*) از دنباله دستورات اسمبلی زیر برای اجرای عمل ضرب در '2' به پیمانه $m(x)$ استفاده می‌کنیم که شامل بررسی بیت پرارزش یک بایت و تصمیم‌گیری طبق آن برای اجرا یا عدم اجرای دستور بعد است:

```
RLNCF      WREG,f
BTFSC     WREG,0
XORLW     0x1a
```

دستور $RLNCF$ ثبات W را یک بیت بدون جابه‌جا کردن نقلی به سمت چپ می‌چرخاند. حال دستور $BTFSC$ بیت 0 از ثبات W را بررسی می‌کند. اگر این بیت، که قبل از چرخش بیت پرارزش W بود، صفر باشد، از دستور بعد پرش می‌شود که درواقع یک NOP به‌جای دستور بعد اجرا و بدین ترتیب در هر صورت ('0' و یا '1' بودن بیت مورد نظر از ثبات W) تعداد پالس‌های ساعت لازم مساوی خواهد شد. با وجود این پیاده‌سازی اگر دوباره سیگنال اثر توان مربوط به بررسی شرط یک‌بودن بیت پرارزش ثبات W را ذخیره کنیم شکل‌هایی مانند (شکل ۴) خواهیم داشت که اگر اثر نوفه را به‌خوبی حذف کرده باشیم، دوباره تحلیل‌هایی روی آنها می‌توانیم داشته باشیم.



(شکل ۴): اختلاف در سطح توان عمل XOR و NOP

key = 9, 6a, d5, a5, 38, a3, fb, 39, 2f, 87, 8e, 43, 44, cb, 94, a6, c2, 95, b, 42, fa, c3, 8, 2e, d9, 24, b2, a2, 49, 6d, 8b, d1, 25, f6, 64, 86, d4, a4, 5c, 5d, 65, 6c, 48, b9, a7, 8d, d8, 0, 8c, bc, a, f7, b8, b3, 45, d0, 8f, ca, 2, bd, 3, 1, 8a, 6b.

در تلاش سوم بایت اول متن خوانا را 0x02 انتخاب

می‌کنیم. این بار نیز '0' برای مقدار بیت پرارزش به دست آمد.

در این مرحله اشتراک فضاهای کلید شامل سی عضو است:

key = 9, 6a, d5, a5, a3, 8e, 44, a6, b, 42, fa, c3, 8, 2e, d1, 64, 86, a4, 5c, a7, 8d, d8, 0, 8c, a, 8f, 2, bd, 3, 1.

به همین ترتیب تلاش‌های خود را ادامه دادیم تا

این که در آزمایش سیزدهم تنها به یک کلید رسیدیم. در زیر

نتایج این مراحل به ترتیب آمده است:

tray : 4, text[0]=0x03, bit=0, n=16

key = 9, a5, 8e, a6, b, 8, a4, a7, 8d, 0, 8c, a, 8f, 2, 3, 1.

tray : 5, text[0]=0x04, bit=1, n=8

key = 9, 8, a4, 8d, 0, 8c, a, 3.

tray : 6, text[0]=0x05, bit=0, n=4

key = a4, 0, a, 3.

tray : 7, text[0]=0x06, bit=0, n=3

key = a4, 0, 3.

tray : 8, text[0]=0x07, bit=1, n=2

key = 0, 3.

مقدار '0' برای بیت پرارزش از بایت اول ماتریس حالت در تبدیل مخلوطساز ستونی حاصل شد. با دادن این مقدار به نرم‌افزار، ۱۲۸ حالت زیر به عنوان فضای کلید ممکن معرفی شد:

key = 52, 9, 6a, d5, 30, 36, a5, 38, bf, 40, a3, 9e, 81, f3, d7, fb, 7c, e3, 39, 82, 9b, 2f, ff, 87, 34, 8e, 43, 44, c4, de, e9, cb, 54, 7b, 94, 32, a6, c2, 23, 3d, ee, 4c, 95, b, 42, fa, c3, 4e, 8, 2e, a1, 66, 28, d9, 24, b2, 76, 5b, a2, 49, 6d, 8b, d1, 25, 72, f8, f6, 64, 86, 68, 98, 16, d4, a4, 5c, cc, 5d, 65, b6, 92, c, 70, 48, 50, fd, ed, b9, da, 5e, 15, 46, 57, a7, 8d, 9d, 84, 90, d8, ab, 0, 8c, bc, d3, a, f7, e4, 58, 5, b8, b3, 45, 6, d0, 2c, 1e, 8f, ca, 3f, f, 2, c1, af, bd, 3, 1, 13, 8a, 6b.

در آزمایش دوم، بایت اول متن خوانا را 0x01 قرار

داده و حمله را تکرار کردیم که دوباره مقدار '0' برای بیت

پرارزش از بایت اول ماتریس حالت در تبدیل مخلوطساز

ستونی به دست آمد. این بار نیز نرم‌افزار با گرفتن این مقدار،

یک فضای ۱۲۸ تایی برای کلیدهای ممکن به دست می‌آورد

که اشتراک آن با فضای کلید آزمایش قبل، شامل ۶۴ عضو

زیر است:

فصل دوم



پس تنها تغییر برنامه برای مقاوم‌سازی در برابر این حمله SPA، تغییر دادن تابع 'XTime' به گونه‌ای است که حاصل ضرب از جدول جستجو فراخوانی شود.

۸- جمع‌بندی و نتیجه‌گیری

در این مقاله ضمن ارائه مقدمه‌ای از حمله‌های کانال جانبی به‌خصوص حمله‌های تحلیل توان و مرور ساختار الگوریتم AES به ضعف موجود در تابع مخلوط‌ساز ستونی این الگوریتم در برابر SPA پرداختیم و نشان دادیم که چگونه می‌توان از این ضعف برای اعمال یک حمله تحلیل توان ساده به AES استفاده کرد. سپس روش عملی کردن این حمله را روی یک پیاده‌سازی AES بر روی میکروکنترلر PIC آوردیم. همچنین تجهیزات سخت‌افزاری و نرم‌افزاری لازم به همراه نتایج عملی و نحوه بازیابی کلید بیان شد. حمله‌های تحلیل توان همواره یکی از تهدیدهای مهم برای سامانه‌های رمزنگاری هستند. برای مقابله با حمله SPA در پیاده‌سازی‌ها به این نکته باید دقت کرد که از وجود ساختارهای شرطی در الگوریتم اجتناب شود.

۹- منابع

Chen, C.N., Yen, S.M., 2003. "Differential Fault Analysis on AES Key Schedule and Some countermeasures," ACISP'03 Proceedings of the 8th Australasian conference on Information security and privacy, LNCS2727, pp.118-129, Wollongong, Australia, Jul. 2003, Springer-Verlag.

Daemen, J., Rijmen, V., 2000. Rijndael Block Cipher, AES Proposal, Computer Security Resource Center, National Institute of Standards and Technology.

Gandolfi, K., Naccache, D., Paar, C., Karine, G., Mourtel, C., Olivier, F., 2001. "Electromagnetic Analysis: Concrete Results," CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings, LNCS2162, pages 251-261. Springer.

Goldack, M., 2008. Side-Channel based Reverse Engineering for Microcontrollers, Ruhr-University.

Koeune, F., Quisquater, J.J., 1999. "A Timing Attack against Rijndael," Technical Report CG-1999/1, Universite Catholique de Louvain, Available online at <http://www.di.ceucl.ac.be/cry/pt/o/techreports.html>

Kocher, P., 1996. "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems," CRYPTO'96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings, LNCS1109, pages 104-113. Springer-Verlag London.

tray : 9, text[0] = 0x08, bit=0, n=2
key=0, 3.
tray : 10, text[0] = 0x09, bit=0, n=2
key=0, 3.
tray : 11, text[0] = 0x0a, bit=0, n=2
key = 0, 3.
tray : 12, text[0] = 0x0b, bit=0, n=2
key = 0, 3.
tray : 13, text[0] = 0x0c, bit=1, n=1
key = 0.

همان‌طور که مشاهده می‌شود، در نهایت تنها عضو مشترک همه فضاها به دست آمده، عبارت است از '0x00' که مقدار صحیح برای بایت اول کلید استفاده شده در سخت‌افزار در زمان انجام این حمله بود.

این حمله برای مقادیر مختلفی از بایت‌های کلید آزمایش شده و همواره کلید را به درستی نتیجه گرفته است. طبق نتایج حاصل از پیاده‌سازی این حمله به این نتیجه می‌رسیم که با داشتن حداکثر شانزده متن خوانا، در هر صورت اشتراک فضاها کلید تنها شامل یک عضو خواهد شد که همان کلید رمزنگاری است.

۷- راه‌کار پیشنهادی برای مقابله با این نقطه ضعف

به‌طور معمول ضعف سامانه‌ها در برابر SPA، به دلیل وجود عملگر شرطی در ساختار آنها و بی‌دقتی در پیاده‌سازی این عملگرهای شرطی است. در این مقاله نیز ما از وجود این چنین ضعفی در پیاده‌سازی تابع مخلوط‌ساز ستونی استفاده کردیم. در واقع وجود عملگر شرطی در ساختار برنامه می‌تواند باعث ایجاد تفاوت قابل کشف در شکل موج توان توسط SPA شود. به عنوان پیشنهادی برای مقابله با این حمله، می‌توان ساختار شرطی برای محاسبه 'XTime' که در واقع همان عمل ضرب یک بایت در '2' در $GF(2^8)$ است را کنار گذاشت و به جای آن از جدول جستجو^۱ برای یافتن این حاصل ضرب استفاده کنیم. بدین ترتیب در هنگام پیاده‌سازی مخلوط‌ساز ستونی، در مواقعی که لازم است حاصل ضرب یک بایت در '2' را بدانیم، کافی است باتوجه به بایت مورد نظر، مقدار متناظر با آن را از جدول جستجو فراخوانی کرده و ادامه برنامه را پیگیری کنیم. از آنجاکه هر بایت شامل ۲۵۶ حالت است، جدول جستجو نیز شامل ۲۵۶ حاصل ضرب ممکن برای ضرب یک بایت داده در '2' به صورت ضرب چند جمله‌ای‌های باینری به پیمانه چند جمله‌ای تحویل‌ناپذیر $X^8+X^4+X^3+X+1$ روی $GF(2)$ خواهد بود.

^۱Look-up Table



حامد یوسفی در سال ۱۳۸۸ مدرک کارشناسی خود را در رشته مهندسی برق گرایش الکترونیک از دانشگاه شهرکرد دریافت کرد و مدرک کارشناسی ارشد خود را در رشته مهندسی مخابرات گرایش رمز از دانشگاه جامع امام حسین(ع) در سال ۱۳۹۰ دریافت نمود. زمینه‌های مورد علاقه وی امنیت سخت‌افزاری و سامانه‌های تعبیه شده است. نشانی رایانامه ایشان عبارتست از:

h.yusefi@rcisp.com



محمود گردشی کارشناسی خود را از دانشگاه شیراز، کارشناسی ارشد را از دانشگاه تبریز و دانشوری را از دانشگاه امیرکبیر به ترتیب در سال‌های ۱۳۶۸، ۱۳۷۰ و ۱۳۷۸ اخذ نموده‌اند و درحال حاضر عضو هیئت علمی دانشگاه جامع امام حسین(ع) هستند. زمینه مورد علاقه وی پروتکل‌های رمزنگاری و سامانه‌های کلید عمومی است. نشانی رایانامه ایشان عبارتست از:

mgardeshi2000@yahoo.com



محمد سبزی‌نژاد فراش درجه کارشناسی خود را در رشته مهندسی برق گرایش الکترونیک از دانشکده فنی شهید چمران کرمان در سال ۱۳۸۵ دریافت نمود. او درجه کارشناسی ارشد را در رشته مهندسی مخابرات رمز از دانشگاه جامع امام حسین(ع) در سال ۱۳۸۷ اخذ نمود. او از سال ۱۳۸۸ تا کنون دانشجوی دکتری ریاضیات رمز دانشکده علوم ریاضی و کامپیوتر دانشگاه تربیت معلم می‌باشد. پروتکل‌های رمزنگاری، مدل‌های اثبات امنیتی، امنیت شبکه و امنیت سخت‌افزاری از جمله علاقمندی‌های تحقیقاتی وی است. نشانی رایانامه ایشان عبارتست از:

sabzinejad@tmu.ac.ir

Kocher, P., Jaffe, J., Jun, B., 1999. "Differential Power Analysis," CRYPTO'99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, LNCS1666, pages 388-397, Springer.

Mangard, S., 2003. A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion, Springer-Verlag Berlin Heidelberg.

Mangard, S., 2004. Securing Implementations of Block Ciphers against Side-Channel Attacks, Ph.D. Thesis, Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology, Austria.

Mangard, S., Oswald, E., Popp, T., 2007. Power Analysis Attacks-Revealing the Secrets of Smart Cards, Springer Science+Business Media.

Masoomi, M., Masoumi, M., Ahmadian, M., 2010 "A Practical Differential Power Analysis Attack against an FPGA Implementation of AES Cryptosystem," IEEE International Conference on Information Society (I-Society 2010), United Kingdom, London, 28-30 June.

Microchip Technology Inc, 2007. PI C18F 245 5/25 50/4455/4550 Data Sheet.

NIST. 2001. ADVANCED ENCRYPTION STANDARD (AES), Federal Information Processing Standards Publication 197, November 26.

Quisquater, J. J., Samyde, D., 2001. "ElectroMagnetic Analysis (EMA): Measures and Counter-Measures for Smart Cards," International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings, LNC-S2140, pages 200-210. Springer.

RIGOL Technologies Inc, 2009. RIGOL Programming Guide DS1000CA Series Digital Oscilloscope.

Samyde, D., Skorobogatov, S., Anderson, R., Quisquater, J., 2002. On a New Way to Read Data from Memory. In IEEE Security in Storage Workshop (SISW'02), IEEE Computer Society, pp. 65-69.

Schmidt, J., Kim, C.H., 2009. "A Probing Attack on AES", WISA 2008, LNCS 5379, Springer-Verlag Berlin Heidelberg, pp. 256-265.

Skorobogatov, S., Anderson, R., 2003. "Optical Fault Induction Attacks," CHES '02 Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems, LNCS2523, pages 2-12. Springer.

Skorobogatov, S., 2005. Semi-invasive attacks - A new approach to hardware security analysis. PhD thesis, University of Cambridge, Available online at <http://www.cl.cam.ac.uk/TechReports/>.

Whelan, C., Tunstall, M., Choukri, H., Naccache, D., Bar-El, H., 2006. "The Sorcerer's Apprentice Guide to Fault Attacks," Proceedings of the IEEE, Special Issue on Cryptography and Security, Volume 96, Number 2, pages 370-382.

فصلنامه



سال ۱۳۹۱ شماره ۱ پیاپی ۱۷