نشریه علمی «پدافند الکترونیکی و سایبری»
سال هفتم، شماره ۱، بهار ۱۳۹۸؛ ص ۴۸-۳۹

طراحی و پیاده‌سازی عملی حمله تحلیل توان الگو بر روی الگوریتم رمز پیشرفته استاندارد بر بستر پردازنده ARM

علی دهقان‌منشادی[۱]، شهرام وفا[۲]، مسعود معصومی[۳*]

کارشناس ارشد مخابرات، دانشگاه تربیت مدرس ۲- دانشجوی دکتری الکترونیک دانشگاه آزاد اسلامی واحد تهران غرب، ۳- استادیار دانشکده فنی دانشگاه آزاد اسلامی واحد اسلامشهر

(دریافت: ۹۶/۰۹/۲۶، پذیرش: ۹۷/۰۳/۰۶)

چکیده

امروزه تامین امنیت از جمله مهم ترین مباحث اساسی و ضروری در سامانه‌های مخابراتی و الکترونیکی به‌شمار می‌رود. حمله الگو از جمله حملات کانال جانبی غیر تهاجمی از نوع تحلیل توان ساده است که در آن مهاجم با تشکیل الگوهایی از سیگنال‌های توان جمع‌آوری شده از پردازنده در حال انجام عملیات رمز و مقایسه آن با نمونه سیگنال‌های توان پردازنده قربانی قادر به شناسایی دستورالعمل‌های پردازنده و وزن همینگ عملوندهای آنها می‌باشد. در این پژوهش، نحوه پیاده‌سازی عملی این حمله برای شکستن الگوریتم پیشرفته رمز استاندارد پیاده‌سازی شده بر روی بستر ARM-LPC گزارش می‌شود. برای تحقق این مهم، ابتدا نمونه‌های سیگنال‌های توان پردازنده ARM-۱۷۶۸ LPC در حال پردازش الگوریتم رمز پیشرفته استاندارد ذخیره و سپس نمونه‌ها به الگوریتم تحلیل مولفه اصلی جهت کاهش ویژگی اعمال و در نهایت داده‌های با ابعاد تقلیل‌یافته توسط الگوریتم هوش ماشین طبقه‌بندی شد. پردازنده ARM به‌دلیل مصرف توان کم، تعداد خطوط لوله بیشتر در مقایسه با سایر پردازنده‌های مشابه و نیز معماری پیچیده‌تر، کمتردر مقالات مرتبط با حمله الگو مورد بررسی قرار گرفته است. نوآوری این پژوهش در استفاده موثر از هوش ماشین در حمله تحلیل توان الگو برای مهندسی معکوس دستورالعمل‌های پردازنده ARM و دستاورد مهم آن کسب درصد شناسایی صحیح ۷۷٪ برای تشخیص وزن همینگ بایت خروجی تبدیل جانشینی بایت‌های اولین دور رمزنگاری الگوریتم رمز پیشرفته استاندارد و به‌طور متوسط ۵۵٪ تشخیص صحیح دستورالعمل‌های ریزپردازنده است.

کلمات کلیدی: حمله کانال جانبی الگو، الگوریتم رمز پیشرفته استاندارد، پردازنده ARM، هوش ماشین

۱- مقدمه

امروزه روند پیشرفت سریع میکروالکترونیک امکان مجتمع‌سازی در ابعاد بسیار بالا را فراهم ساخته است تقریباً تمام یا مهم‌ترین قسمت‌های یک ابزار دیجیتال یا الکترونیکی مانند تلفن همراه یا کارت هوشمند بر روی تراشه آن پیاده‌سازی می‌شود. این تراشه‌ها (که بعضاً دارای اطلاعات بسیار مهمی هستند) یا سخت‌افزارهای مرتبط با آن‌ها از گزینه‌های بسیار جذاب برای مهاجمین و هکرها هستند. یکی از مهم‌ترین تهدیداتی که راه را برای مهاجم و حمله به تراشه‌های الکترونیکی باز می‌نماید استفاده از اطلاعات فیزیکی نشتی از سخت‌افزار در حال پردازش اطلاعات واجرای الگوریتم رمز موسوم به اطلاعات کانال جانبی مانند توان مصرفی یا تشعشعات الکترومغناطیس ساطع‌شده از آن می‌باشد.

بنابر گزارش‌های متعدد می‌توان از توان مصرفی سخت‌افزار یا تشعشعات الکترومغناطیسی هنگام اجرای الگوریتم استفاده کرد و با کمک تحلیل‌های آماری کلید رمز یا سایر اطلاعات حساس را استخراج کرد. حملات تحلیل توان از جمله قدرتمندترین انواع این حملات محسوب شده که از ضعف ذاتی ترانزیستور CMOS در مصرف متفاوت توان در پردازش بیت‌های صفر و یک استفاده کرده و به دو دسته عمده حملات ساده و تفاضلی تقسیم می‌شوند. عمدتا هر دو نوع حمله نیاز به دانستن الگوریتم قربانی و تا حدی جزئیات پیاده‌سازی دارند و درصد موفقیت هر دو حمله در صورت عدم آگاهی از نوع الگوریتم یا جزئیات پیاده‌سازی بسیار پایین خواهد آمد. ضمن آن‌که در صورت استفاده از روش‌های مقابله مانند نقاب‌گذاری، حملات تحلیل توان با مشکلات زیادی مواجه خواهند شد [۴-۱].

*رایانامه نویسنده پاسخگو: m_masoumi@iiau.ac.ir

حمله کانال جانبی الگو اولین بار در سال ۲۰۰۲ توسط کاری[1] و همکارانش معرفی شد [۶-۵]. این حمله نوع قدرتمندی از حملات تحلیل توان ساده است که قادر به مهندسی معکوس دستورالعمل‌های ریزپردازنده هدف به‌منظور کشف عملیات در حال اجرا و بازیابی اطلاعات حساس و نیز شکستن نقاب‌ها در پیاده‌سازی‌های نقاب‌گذاری شده است اما اجرا و پیاده‌سازی این حمله در عمل به‌مراتب پیچیده‌تر و دشوارتر از سایر حملات تحلیل توان ساده و تفاضلی است. برای پیاده‌سازی عملی این حمله معمولاً یک بورد با پردازنده‌ای مشابه پردازنده هدف ساخته شده و قالب‌هایی از توان مصرفی مجموعه دستورالعمل‌های ریزپردازنده تشکیل و ذخیره می‌شود. پس از آن از توان مصرفی پردازنده قربانی نمونه‌برداری شده و توسط تحلیل‌های آماری این نمونه ها با قالب‌های تشکیل داده شده مطابقت داده می‌شود. اما مشکل اینجاست که تحلیل‌های آماری معمولاً در پردازنده‌های نسل جدید و بخصوص پردازنده‌های با تعداد خط لوله نسبتاً بالا در ساختار داخلی خود پاسخ چندان مناسبی نمی‌دهند و خیلی قابل اطمینان نیستند.

در این پژوهش یک پیاده‌سازی عملی حمله الگو علیه پیاده‌سازی الگوریتم پیشرفته رمز استاندارد بر روی پردازنده ARM گزارش شده است. برای غلبه بر ضعف تحلیل‌های آماری از روش‌های مبتنی‌بر هوش ماشین برای تشخیص صحیح دستورالعمل‌ها و منطبق کردن قالب‌ها استفاده شده است. پردازنده ARM به‌دلیل مصرف توان کم، تعداد خطوط لوله بیشتر در مقایسه با سایر پردازنده‌های مشابه و نیز معماری پیچیده‌تر، کمتردر مقالات مرتبط با حمله الگو مورد بررسی قرار گرفته است. لذا پیاده‌سازی موثر این حمله بر روی این پردازنده دارای پیچیدگی‌های عملی و فنی زیادی است که سایر روش‌ها از جمله روش کواریانس که روش متداول پیاده‌سازی حمله الگو هستند قادر به کشف و استخراج دستورالعمل‌ها نیستند. تمرکز عمده این مقاله بر غلبه بر مشکلات فنی پیاده‌سازی حمله تحلیل توان بر روی معماری ARM بوده که روش هوش ماشین توانسته است تا حد قابل قبولی نقاط ضعف روش‌های متداول را پوشش دهد.

با توجه به تعداد زیاد نمونه‌ها و حجم بالای محاسبات در روش هوش ماشین نیاز است تا در ابتدا از الگوریتم‌های کاهش ابعاد جهت کاهش تعداد نمونه‌های مورد نیاز استفاده شود. برای این منظور ابتدا از الگوریتم تحلیل مولفه اصلی[2] جهت کاهش ابعاد داده استفاده گردید [۸-۷]. سپس داده‌های کاهش بعد داده شده جهت فاز آموزش به الگوریتم هوش ماشین بردار پشتیبان[3] جهت کلاس‌بندی داده شد. نتایج به‌دست‌آمده از پیاده‌سازی حمله بر روی پردازنده ARM نشان داد که با احتمال ۷۷٪ قادر به تخمین صحیح وزن همینگ بایت خروجی تبدیل جانشینی بایت‌ها در دور اول الگوریتم و به‌طور متوسط ۵۵٪ قادر به تشخیص صحیح دستورالعمل‌های انجام شده توسط پردازنده هستیم.

در ادامه مقاله در بخش ۲، کارکرد حمله الگو را به‌طور مختصر تشریح می کنیم. سپس در بخش ۳، نحوه نمونه‌گیری از سیگنال توان پردازنده هدف را توضیح می‌دهیم. در ادامه در بخش ۴، حمله قالب بر مبنای ماتریس کوواریانس و در بخش ۵، ماشین بردار پشتیبان را ارائه می‌دهیم. سپس در بخش ۶، تفاوت‌های معماری پردازنده‌های ARM و PIC به‌عنوان بستر پیاده‌سازی حمله و در انتها نتایج به‌دست‌آمده را تشریح خواهیم کرد.

۲- حمله الگو

حمله الگو اولین بار در سال ۲۰۰۲ توسط فردی به نام کاری معرفی شد [۶-۵]. در این روش فرض بر آن است که مهاجم به دستگاهی مشابه با نمونه حمله شونده برای ساخت الگوها دسترسی دارد. این حمله دارای دو فاز اصلی است.

فاز اول موسوم به مشخصه‌سازی[4] است که به آن ساخت الگو[5] هم اطلاق می‌شود. در این فاز مشاهدات مصرف توان با یک تابع توزیع احتمال[6]، که با یک بردار میانگین و یک ماتریس کوواریانس $(\boldsymbol{m}, \boldsymbol{C})$ تعریف می‌شود مشخص می‌گردد که در این حالت به هر زوج $(\boldsymbol{m}, \boldsymbol{C})$ متناسب با هر مشاهده مصرف توان یک الگو[7] اطلاق می‌شود [۹].

هدف فاز دوم که به انطباق الگوها معروف است پیدا کردن الگویی است که بیشترین شباهت را به الگوی مصرف توان در ابزار تحت حمله دارد. در حملات الگو دو راه کار اصلی جهت انجام فاز دوم یا همان منطبق کردن الگوها وجود دارد به این ترتیب که در روش اول پس از تشکیل الگوها برای انجام این مرحله از روش های تشکیل ماتریس کوواریانس استفاده می‌شود و در روش دوم به جای استفاده از ماتریس کوواریانس از روش های موثرتر مانند هوش ماشین به دلیل بالاتر بودن کارایی آن استفاده می‌شود که در این مقاله نیز روش مذکور به‌کارگیری شده است.

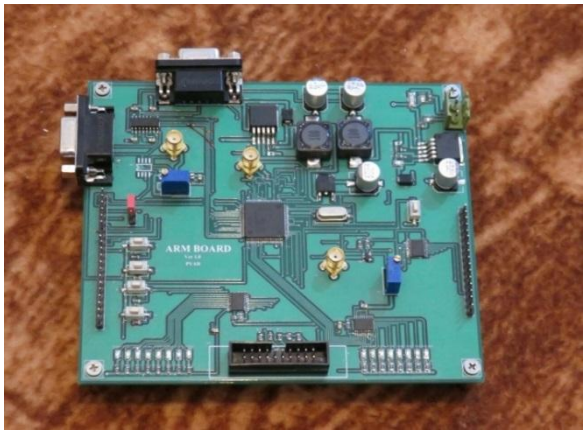
1- Chari
2- Principal Component Analysis (PCA)
3- Support Vector Machine (SVM)
4- Characterization
5- Building Template
6- Probability Distribution Function (PDF)
7- Template

حملات الگو نسبت به حملات کانال جانبی تفاضلی و همبستگی از پیچیدگی پیاده سازی و محاسباتی بمراتب بالاتری برخوردار است. ضمن آن‌که لازم است تا مهاجم به یک کپی از ابزار هدف دسترسی داشته باشد. به‌عبارت دیگر بتواند متن و کلید رمزنگاری را به دلخواه خود تغییر داده و متناسب با آن نمونه مورد نظر را ذخیره نماید. پس از آن بایستی یک سری عملیات پیش پردازش بر روی نمونه‌ها به‌منظور ایجاد الگو انجام گیرد.

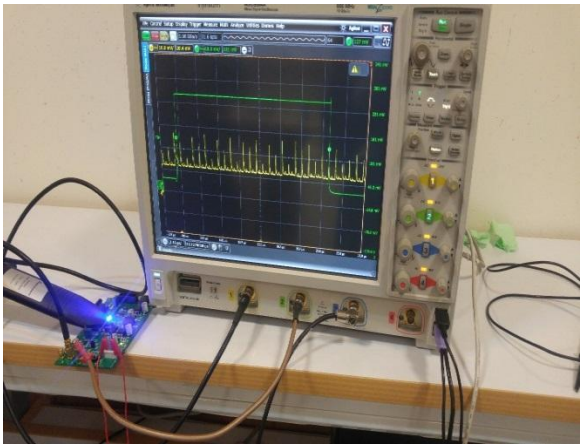
الگوریتم انتخاب شده برای پیاده‌سازی حمله الگو، الگوریتم پیشرفته رمز استاندارد موسوم به AES می‌باشد. از دلایل انتخاب این الگوریتم رمزمی‌توان به امنیت الگوریتم از حیث مقاوم بودن در برابر انواع حملات سخت‌افزاری و نرم‌افزاری، هزینه پیاده‌سازی و اجرای الگوریتم شامل حجم الگوریتم، مقدار حافظه مورد نیاز در پیاده‌سازی نرم‌افزاری، مساحت مورد نیاز برای پیاده‌سازی به‌صورت مدار مجتمع و راندمان محاسباتی بعد از پیاده‌سازی بر روی انواع بسترهای مختلف شامل میکروپروسسورهای ۸، ۱۶ و ۳۲ بیتی، DSPها، FPGA، ASIC، کارت‌های هوشمند و سادگی و انعطاف‌پذیری الگوریتم در پیاده‌سازی بر روی انواع بسترهای مختلف اشاره نمود [۵].

۳- نمونه‌گیری از سیگنال توان پردازنده

همان‌طور که گفته شد اولین قدم در انجام حمله الگو نمونه‌برداری از سیگنال توان مصرفی پردازنده هدف و تشکیل قالب‌هایی از توان دستورالعمل‌هاست. برای این منظور ابتدا دو بورد آزمایشگاهی مبتنی‌بر پردازنده‌های ARM LPC1768 و PIC16F690 با تمهیدات خاص برای اندازه‌گیری بهینه توان مصرفی پردازنده مشابه با بورد استاندارد ارزیابی حملات کانال جانبی موسوم به بورد ساسبو[1] طراحی و ساخته شد. بر این اساس در بوردهای طراحی‌شده یک تثبیت‌کننده ولتاژ اختصاصی برای تغذیه پردازنده و نیز پایه تغذیه پردازنده از سایر قسمت‌ها جدا در نظر گرفته شد تا توان مصرفی سایر قطعات روی بورد، تاثیری بر روی توان مصرفی پردازنده نداشته باشد و اندازه‌گیری ها با حداکثر دقت به انجام برسد. شکل‌های (۲-۱) شمایی از بوردهای طراحی‌شده همراه با مجموعه ابزارهای آزمایشگاهی را نشان می‌دهد. عمل ذخیره نمونه‌ها توسط اسیلوسکوپ دیجیتال Infinum Keysight MSO9064A با حداکثر نرخ نمونه‌برداری 10GS/sec و پهنای باند ۶۰۰ مگاهرتز بانجام رسید. فرکانس نمونه‌برداری در آزمایشات انجام‌شده 1GS/sec و مدت زمان نمونه‌برداری برای ۲۵۰۰ نمونه ۲/۵ میکروثانیه بوده است.



شکل (۱): بورد مبتنی‌بر پردازنده ARM طراحی‌شده در این پروژه.



شکل (۲): شمای مجموعه ابزارهای مورد استفاده در پیاده‌سازی حمله الگو شامل بورد آزمایشگاهی و اسیلوسکوپ دیجیتال.

۴- حمله قالب مبتنی‌بر تشکیل ماتریس کوواریانس

حملات الگو از این مساله که مصرف توان به دستورالعمل‌های پردازنده و نیز داده پردازش‌شده بستگی دارد استفاده می‌کنند. عمدتاً درحملات توان، مشاهدات توان با یک توزیع نرمال چند متغیری[2] مشخص و توصیف می‌شوند. چنانچه در قسمت قبل اشاره شد این حمله بر خلاف سایر حملات شامل دو فاز است: فاز اول که شامل مشخصه‌سازی[3] اتفاق می‌افتد و فاز دوم که از این مشخصه‌سازی برای حمله استفاده می‌شود. در فاز اول مشاهدات مصرف توان توسط یک توزیع نرمال چندمتغیری که معمولاً توسط یک بردار میانگین و یک ماتریس کوواریانس (m, C) تعریف می‌شوند مشخص می‌شوند که زوج (m, C) الگو یا قالب نامیده می‌شود. در این حمله، برای مجموعه مشخصی از

1- Side-Channel Attack Standard Evaluation Board (SASEBO)

2- Multivariate Normal Distribution
3- Characterization

دستورالعمل‌ها الگو یا قالب درست می‌کنیم. یک الگو مجموعه‌ای است از توزیع‌های احتمالی که بیان‌کننده این هستند که توان مصرفی به توان مصرفی کدام یک از کلیدها شباهت دارد. به‌طور کلی یک الگو بیان می‌نماید که: اگر قصد بر استفاده از کلید k را دارید آن‌گاه توان مصرفی شما به توزیع $f_k(X)$ شبیه خواهد بود. با استفاده از این اطلاعات می‌توانیم تفاوت‌های جزیی بین نمونه‌های توان را پیدا نماییم و بنابراین، قادر خواهیم بود تا حدس‌های صحیحی را از کلید صحیح بر روی یک تک توان مصرفی ایجاد نماییم.

به‌عنوان مثال، چنانچه بتوانیم قالب مجموعه مشخصی از دستورالعمل‌های یک میکروکنترولر را تهیه کنیم آن‌گاه با تغییر داده‌ها d_i و کلیدها K_j، همین مجموعه دستورالعمل‌ها را اجرا کرده و مصرف توان متناظر را ثبت می‌کنیم. مجموعه مشاهدات که متناظر با یک مجموعه (d_i, K_j) هستند را گروه‌بندی کرده و سعی می‌کنیم تا بردار میانگین و ماتریس کوواریانس توزیع نرمال چند متغیره را تشکیل دهیم. در نتیجه، برای هر زوج داده و کلید (d_i, K_j) یک الگو یا قالب به‌دست می‌آوریم که شامل یک بردار میانگین m و یک ماتریس کوواریانس C است و آن را با $h_{d_i,k_j} = (m, C)_{d_i,k_j}$ نشان می‌دهیم. متعاقب آن، از این مشخصه‌سازی همراه با یک مشاهده مصرف توان استفاده می‌کنیم تا کلید را به‌دست آوریم. این بدان مفهوم است که تابع توزیع احتمال توزیع نرمال چندمتغیری را با $(m, C)_{d_i,k_j}$ و مشاهده مصرف توان ابزار تحت حمله ارزیابی می‌کنیم. به بیان دیگر، با داشتن یک مشاهده توان t از ابزار قربانی و یک قالب یا الگوی $h_{d_i,k_j} = (m, C)_{d_i,k_j}$ می‌توانیم احتمال زیر را از رابطه (۱) محاسبه کنیم.

$$p\left(t; (m, C)_{d_i,k_j}\right) = \frac{\exp\left(\frac{-1}{2} . (t-m)' . C^{-1} . (t-m)\right)}{\sqrt{(2.\pi)^T . \det(C)}} \quad (۱)$$

ازاین‌رو، برای پیاده‌سازی حمله پس از تشکیل قالب‌ها احتمالات $p(t; (m, C)_{d_1,k_1})$ و $p(t; (m, C)_{d_2,k_2})$ و ... $p(t; (m, C)_{d_D,k_K})$ را به‌دست می‌آوریم. این احتمالات نشان‌دهنده آن است که چقدر یک قالب با یک مشاهده توان مشخص مطابقت دارد. مشخص است که بالاترین احتمال نشان‌دهنده قالب صحیح است. از آنجا که هر قالب متناظر با یک کلید است می‌توانیم نشانه‌هایی از کلید را به‌دست آوریم. این مساله همچنین توسط ادبیات آماری نیز حمایت شده است. اگر همه کلیدها به‌طور یکسان محتمل باشند، قاعده تصمیم‌گیری که احتمال یک تصمیم غلط را به حداقل می‌رساند به‌صورت قاعده (۲) که به قاعده بیشترین شباهت موسوم[1] است بیان می‌شود.

$$p\left(t; h_{d_i,k_j}\right) > p\left(t; h_{d_i,k_l}\right) \quad \text{for } \forall k \neq l \quad (۲)$$

به‌منظور کاهش مشکلات به توان‌رسانی در معادله (۱) از طرفین این معادله لگاریتم می‌گیریم. بنابراین، قالبی که منجر به کوچکترین مقدار قدر مطلق لگاریتم احتمال بشود بیانگر کلید صحیح خواهد بود [۳].

$$\ln p\left(t; (m, C)\right) = -\frac{1}{2}\left(\ln\left((2\pi)^{N_{IP}} . \det(C)\right) + (t-m)' . C^{-1} (t-m)\right) \quad (۳)$$

$$\left|p\left(t; h_{d_i,k_j}\right)\right| < \left|p\left(t; h_{d_i,k_l}\right)\right| \text{ for } \forall\, k \neq l \quad (۴)$$

چنانچه توضیح داده شد برای مشخصه‌سازی از یک ابزار، مجموعه مشخصی از دستورالعمل‌ها را برای زوج‌های مختلف (d_i, k_j) اجرا می‌کنیم و توان مصرف‌شده متناظر را ثبت می‌کنیم. سپس مشاهدات متناظر با برخی از زوج‌ها را گروه‌بندی کرده و بردار میانگین و ماتریس کوواریانس متناظر توزیع نرمال چند متغیری را تخمین می‌زنیم. باید خاطر نشان کرد که ابعاد ماتریس کوواریانس با افزایش تعداد نقاط مشاهده به‌صورت درجه دوم افزایش می‌یابد و لازم است تا نقاطی که حاوی اطلاعات مهم و مفیدی نیستند از فرآیند پردازش کنار گذاشته شوند. به‌طور روشن، لازم است تا شخص یک راهبرد برای یافتن نقاط مورد علاقه[2] تعیین کند. ما تعداد نقاط مورد علاقه را با N_{IP} نشان می‌دهیم. نقاط مورد علاقه نقاطی هستند که حاوی بیشترین اطلاعات در مورد دستورالعمل‌های مشخص شده[3] هستند.

۵- ماشین بردار پشتیبان

۵-۱- تشریح عمومی روش

اگر یک مسئله طبقه‌بندی دو کلاسه در فضای B بعدی $\mathbb{R}^B$، با m نمونه آموزشی $x_i \in \mathbb{R}^B$ و برچسب‌های متناظر با آنها $\{(x_i, y_i) \mid i \in [1, m], y_i \in \{-1, +1\}\}$ را در نظر بگیریم. طبقه‌بندی ماشین بردار پشتیبان شامل یافتن ابرصفحه جدا کننده‌ای است که حاشیه آن (یعنی فاصله از نزدیکترین نمونه‌های آموزشی هر دو کلاس) بیشنه باشد. به نزدیکترین نمونه‌ها به این صفحه، بردارهای پشتیبان گفته می‌شود.

1- Maximum Likelihood (ML)
2- Interesting Points
3- Characterized Instructions

شکل (۳)، ابرصفحه جداکننده بین دو کلاس مختلف با فرض تفکیک‌پذیر بودن خطی دو مجموعه را نشان می‌دهد.

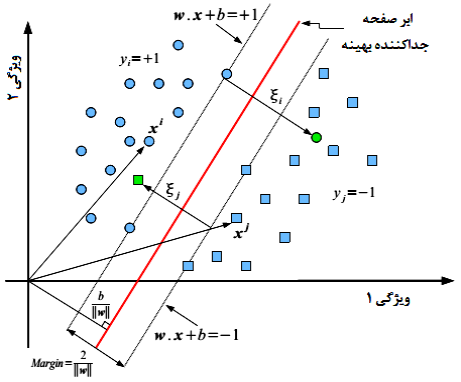
ابرصفحه H_p به‌صورت رابطه (۵) تعریف می‌شود که در آن، $w \in \mathbb{R}^B$ بردار نرمال ابرصفحه و $b \in \mathbb{R}$ بیانگر بایاس آن است.

$$w \cdot x + b = 0 \ , \ \forall x \in H_p \qquad (۵)$$

به‌ازای هر $x \notin H_p$ آن‌گاه:

$$f(x) = \frac{|w \cdot x + b|}{\|w\|} \qquad (۶)$$

بیانگر فاصله x از ابرصفحه H_p است.



شکل (۳): ابر صفحه جداکننده بین دو کلاس با فرض تفکیک‌پذیری خطی

با توجه به روابط بیان شده در بالا، ابرصفحه جداساز حداکثر حاشیه در طبقه‌بندی ماشین بردار پشتیبان از حل رابطه بهینه‌سازی (۷) به‌دست می‌آید.

$$\begin{cases} \min \ \dfrac{\|w\|^2}{2} \\ y_i \left(w \cdot x_i + b \right) > 1 \ , \ \forall i \in [1, m] \end{cases} \qquad (۷)$$

به‌منظور جلوگیری از بیش برازش و نیز جداسازی کلاس‌های که به دلیل همپوشانی، جدایی‌پذیر خطی نیستند، رابطه (۷) را می‌توان با در نظر گرفتن ترم رگولاریزاسیون به‌صورت رابطه (۸) تعمیم داد.

$$\begin{cases} \min \ \dfrac{\|w\|^2}{2} + C \sum_{i=1}^{m} \xi_i \\ y_i \left(w \cdot x_i + b \right) > 1 - \xi_i \ , \ \xi_i > 0 \ \ \forall i \in [1, m] \end{cases} \qquad (۸)$$

که در آن، ξ_i ها بیانگر موقعیت نمونه i-ام نسبت به حاشیه‌ها هستند و C نیز پارامتر رگولاریزاسیون است.

با استفاده از پارامترهای (w, b) به‌دست‌آمده از حل مسئله بهینه‌سازی رابطه (۸) تابع جداسازی ماشین بردار پشتیبان به‌صورت رابطه (۹) خواهد بود که در نهایت به‌صورت رابطه (۱۰) ساده می‌شود.

$$g(x) = \text{sgn}\left(w \cdot x + b \right) \qquad (۹)$$

$$g(x) = \text{sgn}\left(\sum_{j=1}^{m_s} \lambda_j y_j x_j x + b \right) \qquad (۱۰)$$

در رابطه (۱۰)، x_j ها بیانگر بردارهای پشتیبان هستند و λ_j ها ضرایب لاگرانژ متناظر با x_j ها می‌باشند که از حل مسئله بهینه‌سازی رابطه (۸) با روش لاگرانژ به‌دست آمده‌اند.

اگر کرنل‌هایی که شرایط Mercer در مورد آنها برقرار باشد را به‌جای عبارت ضرب داخلی در رابطه (۱۰) جاگذاری شود، می‌توان طبقه‌بند ماشین بردار پشتیبان خطی را مطابق رابطه (۱۱) به طبقه‌بند ماشین بردار پشتیبان غیرخطی تعمیم داد.

$$g(x) = \text{sgn}\left(\sum_{j=1}^{m_s} \lambda_j y_j K(x_j, x) + b \right) \qquad (۱۱)$$

روابطی که در بالا بیان شدند، مربوط به طبقه‌بندی دو کلاسه هستند. به‌منظور حل مسائل طبقه‌بندی k کلاسه $(k > 2)$ با استفاده طبقه‌بند باینری ماشین بردار پشتیبان، روش‌های مختلفی پیشنهاد شده‌اند که دو مورد از مهمترین آنها به شرح زیر است:

الف) یک در مقابل همه: در این روش k طبقه‌بند باینری به‌ازای هر کلاس در مقابل سایر کلاس‌ها اعمال می‌شود و در نهایت هر پیکسل x به کلاسی که به‌ازای آن $f(x)$ بهینه باشد اختصاص می‌یابد.

ب) یک در مقابل یک: در این روش $\frac{k(k-1)}{2}$ طبقه‌بند باینری به‌ازای هر جفت کلاس اعمال می‌شود و در نهایت هر پیکسل به کلاسی که بیشترین آراء را کسب کند، تعلق می‌یابد.

ماشین بردار پشتیبان استاندارد یک طبقه‌بند غیر پارامتری و غیر احتمالاتی است که هیچ تخمینی از احتمال کلاس‌ها را مهیا نمی‌کند. اگرچه در صورت نیاز می‌توان از طبقه‌بندهای احتمالاتی دیگر استفاده نمود ولی به دلیل عملکرد خوب طبقه‌بندهای ماشین بردار پشتیبان در ابعاد بالا، تخمین احتمال کلاس‌ها از نتایج این طبقه‌بند می‌تواند مفید باشد.

با توجه به بار محاسباتی سنگینی که الگوریتم ماشین بردار پشتیبان دارد از روش (ب) و کتابخانه‌ای که به زبان C نوشته شده و با نام libsvm شناخته می‌شود استفاده شده است.

۵-۲- الگوریتم PCA

این روش بهترین شیوه برای کاهش ابعاد داده به صورت خطی است به این صورت که با حذف ضرایب کم اهمیت به‌دست‌آمده از این تبدیل، اطلاعات از دست‌رفته نسبت به روش‌های دیگر کمتر است. در این روش محورهای داده جدید با توجه به مقدار واریانس داده‌ها چیدمان می‌شود بدین مفهوم که اولین محور محوری است که واریانس داده‌ها در جهت آن محور بیشترین مقدار را دارد و محور دوم عمود بر محور اول و به همین ترتیب تا انتها، با توجه به ابعاد داده‌ای که وجود دارد.

الگوریتم PCA به‌صورت مختصر شامل مراحل زیر است:

مرحله ۱- انتخاب داده

داده‌های مورد علاقه جمع‌آوری و با توجه به ابعاد داده در یک ماتریس ذخیره می‌شوند.

مرحله ۲- کم کردن میانگین از داده‌ها

در این مرحله، میانگین هر بُعد از مقادیر آن بُعد کم می‌شود تا میانگین داده‌ها در هر بُعد صفر شود.

مرحله ۳- محاسبه ماتریس کواریانس

کواریانس بین تمامی ابعاد داده‌ها را می‌توان دو به دو محاسبه کرد و در یک ماتریس ذخیره نمود. ماتریس کواریانس یک ماتریس مربعی متقارن است مثلاً اگر سه بعد به‌نام‌های x، y و z داشته باشیم، ماتریس کواریانس آنها از رابطه (۱۲) به‌دست می‌آید.

$$C=\begin{pmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{pmatrix} \qquad (۱۲)$$

ابعاد ماتریس کواریانس یک ماتریس مربعی برابر با تعداد نقاط موجود در هر نمونه است.

مرحله ۴- محاسبه بردارهای ویژه و مقادیر ویژه

در این مرحله بردارهای ویژه و مقادیر ویژه ماتریس کواریانس محاسبه می‌شود. ماتریس کواریانس، یک ماتریس نیمه قطعی مثبت متقارن[1] است. طبق قضایای جبر خطی، یک ماتریس متقارن $n \times n$ دارای n بردار ویژه مستقل و n مقدار ویژه می‌باشد. همچنین یک ماتریس نیمه قطعی مثبت، دارای مقادیر ویژه غیرمنفی است.

مرحله ۵- کاهش ویژگی

بردار ویژه‌ای که مقدار ویژه متناظر با آن از بقیه مقادیر بزرگتر باشد حاوی بیشترین اطلاعات بوده و به‌عنوان مؤلفه اصلی[2] داده‌های موجود شناخته می‌شود. در نتیجه می‌توان مقادیر ویژه را از بیشترین مقدار به کمترین مقدار در یک ماتریس مرتب کرد. سپس با انتخاب مولفه‌های ابتدایی ماتریس به‌دست‌آمده از یک طرف می‌توان به بیشترین اطلاعات دسترسی داشت و از طرف دیگر می‌توان حجم بالای داده‌ها را کاهش داد.

۶- تفاوت معماری پردازنده‌های ARM و PIC

پردازنده‌های ARM از خط لوله برای پردازش استفاده می‌کنند. پردازنده‌های ARM مبتنی بر خط لوله دارای سه وضعیت کاری برای اجرای یک دستورالعمل هستند: واکشی، کدگشایی و اجرا. به عبارت ساده‌تر، در چرخه اول، دستور اول واکشی می‌شود، در چرخه دوم، دستور اول کدگشایی و دستور دوم واکشی می‌شود و در چرخه سوم، دستور اول اجرا و دستور دوم رمزگشایی و دستور سوم واکشی می‌شود. به این معماری سیستم خط لوله سه مرحله‌ای گفته می‌شود. خط لوله‌ها از اولین پردازنده ARM تا هسته ARM7TDMI سه مرحله‌ای هستند و در نسخه‌های بالاتر تعداد مراحل خط لوله‌ها افزایش پیداکرده است. به‌طور مثال، پردازنده‌های ARM9 دارای خط لوله‌های پنج مرحله‌ای هستند که عملیات خواندن و نوشتن از حافظه‌ها نیز جزء این عملیات قرارگرفته است. خط لوله‌ها در ARM10 شش مرحله‌ای و در ARM11 هشت مرحله‌ای هستند. در واقع هرچه تعداد مراحل خط لوله‌ها افزایش پیدا کند قدرت پردازش پردازنده و درنتیجه کارایی آن افزایش می‌یابد که با توجه به اجرای تودرتوی دستورالعمل‌ها و درهم آمیخته شدن توان مصرفی آنها با یکدیگر اجرای حملات تحلیل توان و به‌خصوص حمله الگو بر روی آنها مشکل‌تر خواهد شد.

میکروکنترلر PIC16F690 یک میکروکنترلر ۸ بیتی با پردازشگر RISC است که با معماری هاروارد[3] طراحی شده است. بر این اساس، حافظه برنامه و داده‌ها جدا از یکدیگر می‌باشند. حافظه برنامه می‌تواند با ۱۴بیت گذرگاه برنامه قابل دسترسی باشد که به‌عنوان Flash-ROM ساخته شده است که می‌تواند تا ۲۰۴۸ دستورالعمل را ذخیره کند. در مقابل آن، ۱۲۸ بایت SRAM، ۲۵۶ بایت حافظه EEPROM به ۸ بیت گذرگاه داده متصل می‌باشند که علاوه‌بر آن، به اجزاهای جانبی و محیطی نظیر (درگاه[4]، تبدیل آنالوگ به دیجیتال،USART و...) می‌تواند متصل شود.

چرخه دستورالعمل در رجیستر دستورالعمل شروع می‌شود که شامل دستورالعمل‌هایی است که اجرا می‌شوند. این رجیستر به یک واحد کنترل دستورالعمل و کدگشایی و مالتی پلکسر آدرس‌دهی SRAM و مالتی پلکسر واحد محاسبات ریاضی

1- Symmetric Positive Semi Definite Matrix
2- Principle Component
3- Harvard Architecture
4- Port

منطقی متصل شده است. واحد کنترل و کدگشایی چرخه دستورالعمل را کنترل می‌کند. به‌عنوان مثال مشخص می‌کند که کدام عملوند به واحد محاسبات ریاضی منطقی ارسال می‌شود، کدام فایل رجیستر آدرس‌دهی شده و کدام اجزا نتیجه عملیات را ذخیره می‌کند و همه دستورالعمل‌ها توسط این اجزاها کنترل شده است. با قادر بودن به تغییر و خواندن فایل رجیستر از SRAM با استفاده از یک دستورالعمل، ۷ بیت از رجیستر دستورالعمل به یک مالتی پلکسر آدرس‌دهی به‌عنوان آدرس مستقیم متصل شده است.

این پردازنده در معماری خود دارای دو خط لوله می باشد که از این حیث از معماری ساده‌تری در مقایسه با پردازنده ARM برخوردار است و لذا اجرای تحلیل‌های توان و به‌طور خاص تحلیل الگو بر روی آن به‌مراتب ساده‌تر از پردازنده ARM می‌باشد.

۷- اجرای حمله و نتایج به‌دست‌آمده

اولین قدم در ساخت قالب مرحله پروفایلینگ است. برای ساخت قالب برای هر دستورالعمل نیاز به چندین اندازه‌گیری توان مختلف برای هر دستورالعمل است. به این منظور کدهای آموزشی بر روی دستگاه نمونه تولید می‌شود. از آنجا که قالب باید تا جایی که می‌شود مستقل از همه عوامل دیگر به‌جز خود دستورالعمل باشد از این رو باید همه متغیرهای دیگر موثر بر مصرف توان را تقلیل داده و تحت کنترل قرار داده شوند. از این رو کدهای کوچکی تولید می‌گردد که شامل دستورالعمل مورد نظر بوده در حالی که داده در حال پردازش, محل حافظه و دستورالعمل‌های قبل و بعد دستورالعمل اصلی متغیر هستند. این کدها در داخل پردازنده یا میکروکنترلر اجراشده و سپس نمونه‌برداری انجام می‌گیرد. پس از جمع‌آوری نمونه‌ها, نمونه‌برداری از الگوریتم هدف صورت پذیرفته و با مقایسه نمونه‌های مصرف توان و به‌کارگیری الگوریتم‌های کلاسه‌بندی, دستورالعمل‌های الگوریتم هدف مشخص خواهند شد.

همان‌طور که در بخش قبل بیان شد پردازنده، PIC16F690 ۸ بیتی از شرکت میکروچیپ و دارای ۳۵ دستورالعمل مختلف است که بیشتر آنها در یک سیکل دستورالعمل اجرا می‌شوند. برخی دستورالعمل‌ها مانند دستورالعمل‌های شاخه‌ای یا پرشی دو سیکل دستورالعمل طول می‌کشند. هر دستورالعمل در تعداد مشخصی از سیکل ساعت اجرا می‌شود. به‌طور مثال میکروکنترلر PIC16F690 دستورات یک سیکلی را در ۴ سیکل ساعت اجرا می‌نماید. همان‌طور که گفته شد این میکروکنترلر در ساختار خود دارای خط لوله است. بدین‌صورت که در حالی که یک دستورالعمل در حال اجراست همزمان دستورالعمل بعدی در حال پیش واکشی[1] خواهد بود.

عمدتآ نشت اطلاعات توان در ریزپردازنده‌ها از طریق گذرگاه‌ها در حین انتقال داده یا پردازش توسط واحد پردازشگر مرکزی یا واحد ریاضی- منطقی یا حتی رجیسترهایی که داده را نگهداری می‌کنند اتفاق می‌افتد. همان‌طور که می‌دانیم مشخصه واحد هر دستورالعمل یک کد عملیاتی است که در حافظه برنامه ذخیره‌شده و قبل از اجرا به رمزگشای دستورالعمل منتقل می‌شود. به‌علاوه هر دستورالعمل با رفتار مشخصی از واحد ریاضی منطقی، گذرگاه‌ها و احتمالا مولفه‌های دیگر مشخصه‌سازی می‌شود [۹-۱۵].

برای ساخت قالب‌هایی که مستقل از سایر عوامل و با کمترین اثر پذیری از آنها، تنها بر گرفته از خود دستورالعمل باشند می‌توان از راه کار تغییر همه مولفه‌های موثر بر مصرف توان استفاده کرد. از این رو برای مرحله ساخت قالب آزمایشی از برنامه‌های کوچکی شامل چند دستورالعمل استفاده می‌شود که دستورالعمل هدف در وسط قرار گرفته و تمامی مولفه‌هایی مانند داده‌ها، محل حافظه و نوع دستورالعمل‌های قبل و بعد در آن به‌صورت متغیر خواهند بود

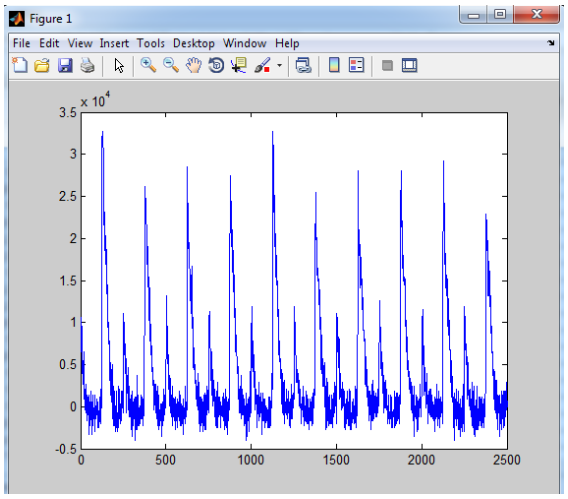
در مرحله ساخت قالب‌ها, قالب هر دستورالعمل با ثابت فرض کردن دستور بعدی به‌دست می‌آید. برای مثال برای تمامی دستورات هدف, دستور بعدی را NOP در نظر گرفته که با توجه به این‌که NOP دارای کد عملیاتی با وزن همینگ صفر است با این روش قالب خود دستورات هدف ساخته شده بدون این‌که دستور بعدی بر آن اثر داشته باشد. به‌طور مثال برای ساخت الگوی یک دستور XORLW از مجموعه دستورالعمل‌های زیر استفاده می‌شود.

```
MOVLW 0x88h;
MOVLW 0b00000010;
MOVWF PORTC;
NOP;
NOP;
XORLW 0xFFh;
NOP;
NOP;
MOVLW 0b00000000
MOVWF PORTC;
```

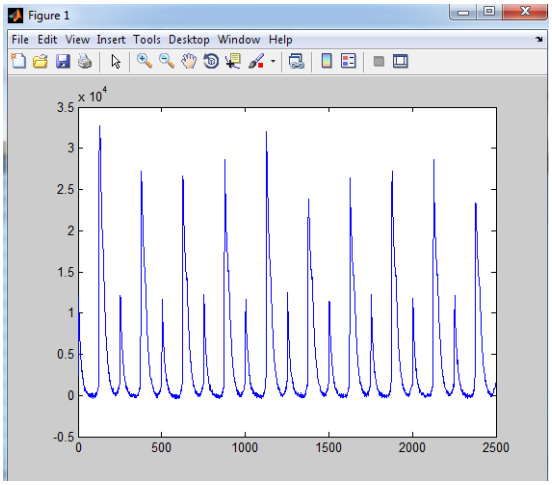
شکل (۴): ساخت الگوی لازم برای تشکیل قالب دستورالعمل XORLW از طریق قرار دادن دستورالعمل NOP در قطعه برنامه.

1- Prefetch

پس از اجرای هر دستور، نمونه‌برداری توسط اسیلوسکوپ انجام می‌شود. به‌طور مثال اگر گراف شکل (۵) به‌عنوان خروجی اسیلوسکوپ چند دستورالعمل نمونه با ۲۵۰۰ نقطه نمونه بر روی محور x و مقادیر توان نمونه‌برداری‌شده بر روی محور y باشد با توجه به این‌که نیاز است تعداد مناسبی نمونه جهت ساخت قالب و سپس انجام تست تطبیق صورت پذیرد، هر مجموعه دستورالعمل ده‌ها بار اجرا شده و نمونه‌ها توسط برنامه متلب جهت حذف نویز به ماتریس میانگین داده‌ها تبدیل شده و سپس مراحل انتخاب نقاط مهم صورت می‌پذیرد به‌منظور حذف نویز نمونه‌گیری‌ها ده بار تکرار و از آنها میانگین گرفته می‌شود که این امر در شکل (۶) نشان داده شده است.

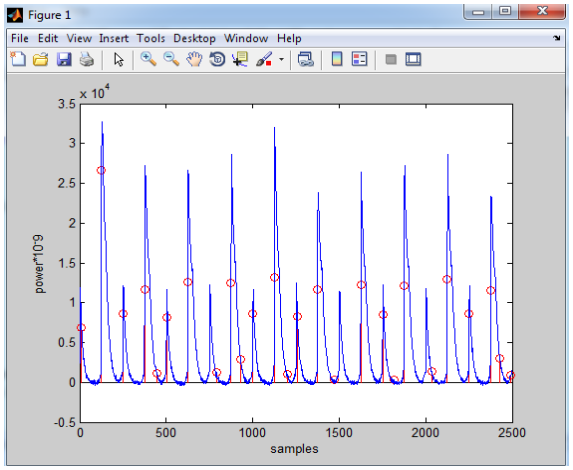


شکل (۵): نمونه خروجی اسیلسکوپ دیجیتال پس از نمونه‌گیری از قطعه کد الگوی یک دستور XORLW و ۲۵۰۰ نقطه.



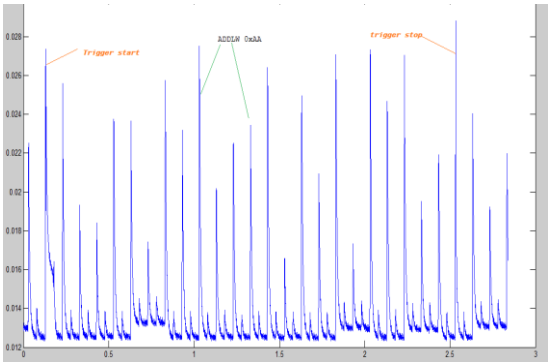
شکل (۶): نمونه خروجی اسیلسکوپ دیجیتال پس از نمونه‌گیری از قطعه کد الگوی یک دستور XORLW و ده بار میانگین‌گیری و حذف نویز با ۲۵۰۰ نقطه.

سپس اختلاف میانگین مقادیر متوسط را محاسبه نموده و بر اساس تعداد نقاط مورد نظر در هر سیکل ساعت نقاط مورد علاقه انتخاب می‌شود. هر چه تعداد نقاط انتخابی در یک سیکل بیشتر باشد تعداد نقاط مورد علاقه بالاتر می‌رود. یک راهبرد این است که تنها یک نقطه در یک سیکل ساعت باید به‌عنوان نقطه مورد علاقه انتخاب شود و نقاط بیشتر در همان سیکل ساعت دارای اطلاعات بیشتری نیستند البته این امر در مواردی قابل نقض بوده و نشان‌داده شده که این تعداد می تواند بیشتر نیز باشد هر چند باید به تعداد نقاط، جهت بزرگ نشدن زیاد از حد ابعاد ماتریس کواریانس هم توجه نمود. شکل (۷) تعداد نقاط مورد علاقه در نمونه‌های ثبت‌شده از الگوی دستورالعمل XORLW را نشان می‌دهد.

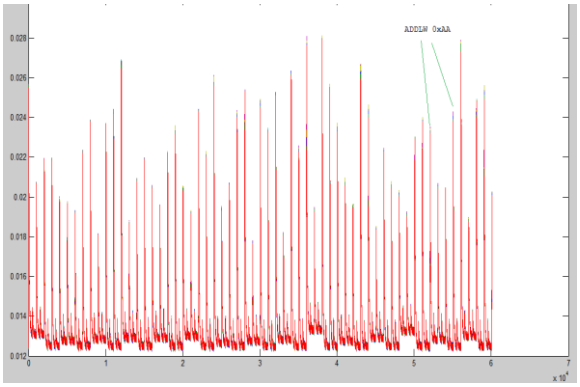


شکل (۷): نقاط مورد علاقه بر حاصل از منحنی نمونه شکل (۵).

آزمایشات مشابهی بر روی پردازنده ARM نیز بانجام رسید. شکل (۸) همان نمونه‌گیری را برای دستورالعمل هدف ADDLW 0Xaa این پردازنده نشان می‌دهد که دستور هدف که همان ADDLW 0xAA است در گراف مشخص شده است. شکل (۹) توان مصرفی پردازنده حین اجرای دستورالعمل ADDLW در توالی دستورالعمل‌ها نشان می‌دهد.



شکل (۸): توان مصرفی پردازنده حین اجرای دستورالعمل ADDLW پردازنده ARM.



شکل (۹): توان مصرفی پردازنده حین اجرای دستورالعمل ADDLW پردازنده ARM در توالی دستورالعمل‌ها.

پس از بررسی نتایج مشخص شد نتایج حاصل از روش کوواریانس چندان مطلوب نبوده با توجه به قدیمی بودن روش کوواریانس در انجام حمله الگو و به وجود آمدن روش‌های به روزتر مثل روش‌های هوش ماشین و این‌که، بدین سبب نتایج ارائه‌شده در جدول‌های (۲-۱) مربوط به روش ماشین بردار پشتیبان است. برطبق جدول‌های (۲-۱)، نرخ تشخیص دستورالعمل‌ها متفاوت و از ۷/۸ درصد تا ۹۸/۸ درصد متغیر بوده و میانگین نرخ برای پردازنده PIC16F690 در حدود ۶۳ درصد و برای پردازنده LPC1768 در حدود ۵۴ درصد برای کدهای مورد آزمایش به‌دست آمد.

جدول (۱): نرخ تشخیص دستورالعمل‌ها در پردازنده PIC16F690.

دستورات	نرخ تشخیص%
SUBLW	۸۸/۱
IORLW	۷۰
CLRWDT	۶۲/۵
BCF	۶۶/۲
BTFSC	۴۲/۵
BSF	۶۰/۶
BTFSS	۵۶/۸۷
SUBWF	۵۶/۸
RRF	۵۹/۳
RLF	۶۰
MOVLW	۳۹/۳
CLRW	۷۶/۲
CLRF	۶۳/۱
ADDLW	۶۶/۲
ADDWF	۸۱/۲

جدول (۲): نرخ تشخیص دستورالعمل‌ها در پردازنده LPC1768.

دستورات	نرخ تشخیص%
MOV	۹۸/۸
AND	۹۳/۶
SUB	۳۹
RSB	۴۶/۸
EOR	۵۷/۲
ORR	۷/۸
BIC	۳۶/۴
MVN	۷۸
CMP	۷۸
TST	۳۱/۲
RSC	۷۵/۴
TEQ	۳۶/۴
ADC	۲۰,۸

۸- نتیجه‌گیری

انجام این پروژه بروشنی نشان داد که تحلیل توان ساده و انواع آن از جمله ابزارهای قدرتمند مهندسی معکوس دستورالعمل‌های ریزپردازنده، رخنه به محتوای آنها و کشف اطلاعات حساس از آنها با هزینه کم و بدون نیاز به تجهیزات گران قیمت از جمله تجهیزات لایه‌برداری هستند. در برخی موارد این حمله حتی بمراتب قدرتمندتر از تحلیل‌های تفاضلی یا همبستگی توان است زیرا این حملات در حمله به پیاده‌سازی‌های نقاب گذاری شده مشکلات فراوانی دارند درحالیکه حمله قالب می‌تواند به‌عنوان یک حمله مکمل برای شکستن نقاب‌ها و کمک به حمله تحلیل تفاضلی توان به‌کار گرفته شود.

برای انجام این حمله لازم است تا به معماری و مجموعه دستورالعمل‌های ریزپردازنده که معمولاً در مستندات منتشر شده از طرف کارخانه سازنده مشخص می‌شوند دسترسی داشته باشیم. با توجه به وجود نویز سوئیچینگ، آزمایشات ما نشان داد که بهتر است از هر دستورالعمل چند قالب تهیه شود. علاوه بر آن، انتخاب روش آماری تحلیل نیز نقش به‌سزایی در میزان موفقیت حمله دارد.

در این مقاله، تشکیل قالب توان دستورالعمل‌ها جهت انجام حمله الگو و تشخیص، بازیابی و تمایز آنها از یکدیگر تشریح شد. پس از انجام آزمایش‌ها و تحلیل‌های مربوطه به این نتیجه رسیدیم که روش مبتنی‌بر هوش ماشین به‌مراتب کارآمدتر از روش ماتریس کوواریانس می‌باشد و به درصد موفقیت بالاتری منجر می‌شود.

پس از مطالعه، بررسی و آزمایشات عملی انجام‌شده برای اجرای موثر حمله الگو به این نتیجه رسیدیم که انجام این حمله بر روی تراشه‌های FPGA در عمل امکان‌پذیر نیست و یا این‌که حداقل با دشواری‌های زیادی توام می‌باشد. از اینرو با توجه به منابع در دسترس، تعداد دستورالعمل‌ها، معماری و تعداد خطوط لوله به‌کار رفته در ساختار ریزپردازنده، پردازنده‌های سری PIC16F690 و ARM-LPC1768 برای انجام حمله انتخاب گردید. سپس الگوریتم رمز پیشرفته استاندارد که یکی از رایج ترین، مهم‌ترین و پرکاربردترین الگوریتم‌های رمز بلوکی حال حاضر می‌باشد را بر روی پردازنده‌های مزبور پیاده‌سازی نمودیم.

البته لازم به‌ذکر است که درصد تشخیص موفقیت‌آمیز دستورالعمل‌ها با یکدیگر یکسان نیست و بستگی به معماری پردازنده، تعداد چرخه‌های ماشین لازم برای اجرای هر دستورالعمل و نیز تعداد خطوط لوله به‌کار گرفته شده در واکشی و اجرای دستورالعمل‌ها دارد. آنچه مشخص شد این بود که به دلیل تاثیری که در یک خط لوله پردازنده هر دستورالعمل روی دستورالعمل قبلی یا بعدی خود می‌گذارد، توان گرفته شده از دستورالعمل را نمی‌توان منحصراً مربوط به یک دستورالعمل دانست و این امر تشخیص و تمایز دستورالعمل‌ها را مخصوصاً با افزایش تعداد خط لوله به ۳ عدد و بالاتر بسیار دشوار می‌سازد. با این وجود نتایج به‌دست‌آمده نشان داد که به‌طور متوسط ۶۳٪ قادر به تشخیص صحیح دستورالعمل‌های پردازنده PIC و به‌طور متوسط در حدود ۵۴٪ قادر به تشخیص صحیح پردازنده ARM هستیم. نتایج به‌دست‌آمده نشان داد که می‌توان بر مبنای حملات کانال جانبی یک دیس اسمبلر دستورات پردازنده برای کاربردهای واقعی ساخت به‌شکلی که مکمل یا جایگزین روش‌های مهندسی معکوس و واکشی اطلاعات باشد زیرا اولاً همواره واکشی اطلاعات ریزپردازنده‌ها امکان‌پذیر نیست ثانیاً در برخی موارد رمزگشایی از اطلاعات استخراج شده به‌سادگی امکان‌پذیر نیست.

۹- منابع

[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” proceedings of CRYPTO ‘99, Lecture Notes in Computer Science, vol. 1666, Springer, pp. 388–397, 1999.

[2] S. B. Ors and E. Oswald, “Power Analysis Attacks against FPGA-First Experimental Results,” Advances in Cryptology-CHES2003, LNCS 2779, Springer-Verlag, pp. 35-50, 2003.

[3] S. Mangard, E Oswald, and T. Popp, “Power Analysis Attacks: Revealing the Secrets of Smart Cards (Advances in Information Security),” Springer, may 2007.

[4] P. N. Fahn and Peter K. Pearson, “IPA: A New Class of Power Attacks,” Proceedings of CHES, 1999.

[5] S. Chari, J. R. Rao, and P. Rohatgi, “Template Attacks,” Proceedings of CHES, 2002.

[6] S. Chari, J. Rao, and P. Rohatgi, “Template Attacks,” in the proceedings of CHES 2002, Lecture Notes in Computer Science, vol. 2523, pp. 13-28, CA, USA, August 2002.

[7] T.-F. Wu, C.-J. Lin, and R. C. Weng, “Probability estimates for multi-class classification by pairwise coupling,” The Journal of Machine Learning Research, vol. 5, pp. 975-1005, 2004.

[8] J. C. Platt, “Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods,” in Advances in large margin classifiers, 1999.

[9] M. Golack, “Side-Channel based Reverse Engineering for Microcontrollers,” PhD Thesis, Ruhr-University Bochum, 2009.

[10] PIC16F631/677/685/687/689/690 Data Sheet, Microchip Technology Inc., 2007, http://ww1.microchip.com/downloads/en/DeviceDoc/41262D.pdf

[11] M. O. Choudary and M. G. Kuhn, “Efficient, Portable Template Attacks,” IEEE Transactions on Information Forensics and Security, vol. 13, no. 2, Feb. 2018.

[12] A. Chakraborty, Y. Xie, and A. Srivastava, “Template Attack Based Deobfuscation of Integrated Circuits,” IEEE 35th International Conference on Computer Design, Boston, USA, pp. 41-44, 2017.

[13] A. Chakraborty1, S. Bhattacharya1, T. H. Dixit, C. Rebeiro, and D. Mukhopadhyay, “Template Attack on SPA and FA Resistant Implementation of Montgomery Ladder,” IET Inf. Security, vol. 10, Issue. 5, pp. 245-251, 2016.

[14] Q. Wang, A. Wang, G. Qu, and G. Zhang, “New Methods of Template Attack Based on Fault Sensitivity Analysis, Ieee Transactions on Multi-Scale Computing Systems, vol. 3, Issue 2, pp. 113-123, 2017.

[15] A. Chakraborty and D. Mukhopadhyay, “A Practical Template Attack on MICKEY-128 2.0 Using PSO Generated IVs and LS-SVM,” 2016 29th Int. Conf. on VLSI Design and 2016 15th In. Conf. on Embedded Systems (VLSID), pp. 529-534, 2016.

Journal of Electronical & Cyber Defence
Vol. 7, No. 1, 2019, Serial No. 25

A Practical Template Attack Against the Implementation of the Advanced Encryption Standard on ARM Processor

A. Dehghan Menshadi, S. Vafa, M. Masoumi*

*Islamic Azad University Islamshahr Branch
(Received: 17/12/2017, Accepted: 27/05/2018)

ABSTRACT

In our days, the need for secure protocols and devices seems to be one of the most important issues in the communication systems. Template attacks is a powerful kind of simple power analysis attack that is able to effectively identify and retrieve efficiently the instructions executed by a typical processor and the Hamming weight of their operands. It is usually carried out by using templates that are created from the samples of power consumed by the device on a test platform and statistical analysis of real measurements. This paper describes practical implementation of this attack against the realization of the Advanced Encryption Standard (AES) on ARM-LPC processor. In order to mount the attack, the power samples of the cryptoprocessor processor during the execution of the AES was recorded and exported to the feature extraction and reduction algorithm. Then, the reduced samples were categorized using the machine learning algorithm. Due to more complex architecture, lower power consumption and larger number of pipeline stages compared to other microprocessors which make the attack more difficult, practical implementation of this attack on ARM processor has received less attention in related articles. The main contribution of this paper is efficient use of machine intelligence in improving the attack performance such that the improved attack is able to recover the Hamming weight of the output of the first AES SBox with 77% success rate and correct identification of the instructions of the processor with 55% success rate in average.

Keywords: Template Side-Channel Attack, Advanced Encryption Standard, ARM Microprocessor, Machine Intelligence

* Corresponding Author Email: m_masoumi@iiau.ac.ir