دوفصل‌نامه علمی منادی امنیت فضای تولید و تبادل اطلاعات (افتا)
سال ۱۴۰۰، جلد ۱۹، شماره ۱ (۷۲-۷۹)
https://monadi.isc.org.ir

ارائه‌شده در شانزدهمین کنفرانس بین المللی انجمن رمز ایران در دانشگاه فردوسی مشهد، ۶ و ۷ شهریور ۱۳۹۸

ارائه و پیاده‌سازی حمله‌ی زمانی برنشتاین بهبود یافته بدون داشتن دسترسی ریشه★

وحید معراجی* و هادی سلیمانی

پژوهشکده فضای مجازی، دانشگاه شهید بهشتی ،تهران، ایران

اطلاعات مقاله

کلمات کلیدی:
حافظه‌ی نهان
حمله‌ی زمانی برنشتاین
مشخصه‌ی زمانی
مقدار همبستگی
دسته

dor: 10.0000/000000000

نوع مقاله: پژوهشی

چکیده

دسته‌ی مهمی از حملات کانال‌جانبی با بهره‌بری از اختلاف زمانی موجود بین دسترسی پردازنده به اطلاعات حافظه‌ی اصلی و حافظه‌ی نهان، سعی بر استخراج مقادیر کلید سیستم رمزنگاری دارند. حمله‌ی زمانی برنشتاین یکی از حملات مهم مبتنی بر حافظه‌ی نهان محسوب می‌شود که قابل اعمال به پیاده‌سازی نرم‌افزاری AES است. حمله‌ی برنشتاین را می‌توان به دو مرحله‌ی مهم جمع‌آوری اطلاعات زمانی و استخراج مقادیر کلید از اطلاعات به دست آمده در مرحله‌ی اول تقسیم نمود. تا کنون روش‌های مختلفی برای بهبود حمله برنشتاین با تمرکز بر روی هر یک از مراحل این حمله جهت استخراج مقادیر بیش‌تر از کلید و یا اجرای حمله به ازای نمونه‌های اندازه‌گیری کم‌تر صورت گرفته است. فرض تمامی حملات ارائه شده، دسترسی مهاجم به آدرس‌های مجازی و در نتیجه داشتن دسترسی ریشه (access root) است. در این مقاله، روشی به منظور کاهش تعداد نمونه‌های اندازه‌گیری لازم برای اجرای حمله برنشتاین ارائه می‌کنیم که ویژگی مهم آن، عدم دسترسی به آدرس‌های مجازی بلوک‌های جداول مراجعه‌ی سیستم رمزنگاری است. به عبارت دیگر، ویژگی مهم روش ارائه شده این است که مهاجم نیازی به داشتن دسترسی ریشه در حین اجرای حمله ندارد. بر همین اساس در این مقاله نشان می‌دهیم، جلوگیری از دسترسی ریشه نمی‌تواند روش مناسبی به منظور جلوگیری از حمله برنشتاین باشد. روش ارائه شده به صورت عملی پیاده‌سازی و بر روی پردازنده‌ی Intel Core i5-4200U اجرا شده است. نتایج عملی نشان می‌دهد نه تنها روش ارائه شده نیاز به دسترسی ریشه ندارد، بلکه تعداد نمونه‌های اندازه‌گیری موردنیاز برای اجرای این حمله را از 2^{25} نمونه در مقاله اصلی به 2^{19} کاهش می‌یابد.

۱ مقدمه

متفاوت بودن زمان دسترسی پردازنده به اطلاعات حافظه‌ی اصلی و حافظه‌ی نهان موجب آسیب‌پذیری پردازنده‌های مدرن در مقابل حملاتی موسوم به حملات مبتنی بر حافظه‌ی نهان شده است. این حملات را می‌توان به سه دسته‌ی زیر تقسیم‌بندی نمود.

۱.۱ حملات Time-Driven

در حملات Time-Driven مهاجم با آگاهی از ظرفیت خط‌های[1] حافظه‌ی نهان اقدام به اندازه‌گیری زمان اجرای سیستم رمزنگاری نموده و با انجام یک سری محاسبات آماری بر روی اطلاعات و پروفایل‌های زمانی به

★از کمیته علمی شانزدهمین کنفرانس بین‌المللی انجمن رمز ایران برای داوری این مقاله تشکر می‌شود.

*نویسنده مسئول
آدرس‌های رایانامه: v.meraji@mail.sbu.ac.ir (وحید معراجی)، h_soleimany@sbu.ac.ir (هادی سلیمانی)

[1] line

دست آمده، اقدام به استخراج بیت‌هایی مشخص از کلید سیستم رمزنگاری می‌نماید. مقدار بیت‌های قابل استخراج در این حملات وابسته به ظرفیت خطوط حافظه‌ی نهان است. [۱-۶] از جمله مقالاتی هستند که در حوزه‌ی حملات Time-Driven منتشر شده‌اند.

۲.۱ حملات Trace-Driven

در حملات Trace-Driven نیز مهاجم با بهره‌مندی از پروفایل مربوط به مشخصه‌ی توان مصرفی سیستم رمزنگاری، بین موارد بروز و عدم بروز تصادم تمایز قایل شده و اقدام به استخراج بیت‌هایی مشخص از کلید سیستم رمزنگاری می‌نماید. بدیهی است که در پروفایل‌های مذکور، توان مصرفی زمان‌های بروز عدم تصادم از زمان‌های بروز تصادم بیش‌تر خواهد بود. تعداد بیت‌های استخراجی کلید در این حملات نیز وابسته به ظرفیت خطوط حافظه‌ی نهان است. [۷]، نیز از جمله مقالات منتشر شده‌ای است که حمله‌ی موجود در آن زیرگروه حملات Trace-Driven است.

۳.۱ حملات Access-Driven

حملات Access-Driven را می‌توان به طور عمده به دو دسته‌ی سنکرون و آسنکرون تقسیم‌بندی نمود. حملات Evict+Time و Prime+Probe ارائه شده در مقاله [۸] و حمله‌ی Flush+Reload ارائه شده در مقاله [۹] از جمله حملاتی هستند که زیر مجموعه‌ی حملات Access-Driven محسوب می‌شوند. از حمله‌ی ارائه‌شده در مقاله‌ی [۱۰] نیز می‌توان به عنوان یکی از مطرح‌ترین حمله‌های Access-Driven بر روی پردازنده‌های ARM نام برد. حملات Access-Driven نسبت به حملات پیشین از تعداد نمونه‌های اندازه‌گیری کم‌تری استفاده می‌نمایند، اما نیازمند آگاهی مهاجم از آدرس‌های مجازی و یا فیزیکی عناصر موجود در حافظه‌ی نهان می‌باشد. این در حالی است که با اجرای راهکار مقابله ارائه شده در [۹] امکان دسترسی به آدرس‌های مجازی ممکن نبوده و دسترسی به آدرس فیزیکی عناصر مذکور در پردازنده‌های اینتل مجری سیستم عامل لینوکس نیز نیازمند ریشه‌ای بودن دسترسی مهاجم است. [۱۱] اقدام به ارائه‌ی یک روش جهت فائق آمدن بر مانع اول نموده است. روش ارائه شده در این مقاله دارای یک فاز پیش حمله با سربار محاسباتی نسبتا بالایی است. این مقاله با الهام از الگوی ابتدایی این مقاله و حذف فاز مذکور، اقدام به بهبود حمله‌ی زمانی برنشتاین نموده است.

تا به حال اقدامات مختلفی جهت ارتقای سطح کیفی هر یک از حملات مبتنی بر حافظه‌ی نهان ذکر شده صورت گرفته است. به طور مثال استفاده از کوچک‌ترین زمان اختصاص یافته به اجرای سیستم رمزنگاری در کنار بهره‌مندی از میانگین زمانی اجراهای سیستم رمزنگاری در [۱۲] و استفاده از دامنه‌های متنوع با اندازه‌های مختلف در مشخصات زمانی این حمله در [۱۳]، از جمله اقدامات صورت گرفته بر روی مرحله‌ی اول از حمله‌ی برنشتاین جهت ایجاد تمایز مناسب بین کلیدهای صحیح و غلط بوده‌است. از اجرای شمارش کلیدی بهینه‌ی موجود در [۱۴] توسط مقاله‌ی [۱۳] نیز می‌توان به عنوان یکی از مهم‌ترین اقدامات صورت گرفته بر مرحله‌ی دوم حمله‌ی برنشتاین جهت بهبود آن نام برد. روش پاک کردن هشت بلوک متوالی مربوط به اطلاعات جداول مراجعه، قبل از هر بار اجرای سیستم رمزنگاری بدون آگاهی از آدرس مجازی و فیزیکی این بلوک‌ها نیز روش ارائه شده در این مقاله برای کاهش تعداد نمونه‌های اندازه‌گیری لازم برای اجرای این حمله محسوب می‌شود. حمله‌ی ارائه شده در این مقاله نسبت به حملات دیگر دارای دو مزیت زیر می‌باشد:

(۱) برای اولین بار تعداد نمونه‌های اندازه‌گیری لازم برای اجرای حمله‌ی برنشتاین بر روی یک پردازنده‌ی اینتل را به 2^{19} کاهش می‌دهد. این در حالی است که در مقالات دیگر با ترکیب حداقل دو روش به طور همزمان تعداد نمونه‌های اندازه‌گیری لازم برای اجرای حمله به 2^{25} مورد کاهش یافته است.

(۲) برخلاف دیگر حملات، این حمله تنها یک گروه شامل کلید صحیح را به عنوان کلید سیستم رمزنگاری انتخاب می‌نماید. حملات قبلی، چندین گروه شامل کلید صحیح را به عنوان کاندید برای کلید سیستم رمزنگاری معرفی می‌نمایند.

در بخش ۲ از این مقاله پس از بیان چند مورد از پیش‌نیازهای اجرای حملات مبتنی بر ابزار حافظه‌ی نهان، به توصیف عملکرد حمله‌ی برنشتاین پرداخته خواهد شد. بخش ۳ از این مقاله نیز پس از تشریح ایده‌ی اصلی این مقاله برای کاهش تعداد نمونه‌های اندازه‌گیری لازم برای اجرای حمله، به نحوه‌ی اجرایی کردن ایده‌ی مذکور بدون استفاده از هیچ نوع آدرسی از بلوک‌های مربوط به جدول‌های مراجعه‌ی سیستم رمزنگاری AES توسط یک مهاجم بدون هیچ دسترسی ریشه‌ای می‌پردازد. قسمت ابتدایی از بخش ۴ نیز به بررسی نتایج به دست‌آمده از عملیات‌های انجام شده برای دسترسی به آدرس مجازی بلوک‌های جداول مراجعه بدون هیچ نوع دسترسی ریشه‌ای اختصاص یافته است. قسمت آخر ار این بخش نیز به نتایج به دست آمده از اجرای حمله‌ی برنشتاین بابهره‌مندی از ایده‌ی این مقاله می‌پردازد.

۲ پیش‌زمینه‌ها

۱.۲ حافظه‌ی نهان

حافظه‌ی نهان، یک حافظه‌ی واسط بین حافظه‌ی اصلی و پردازنده است. این حافظه وظیفه‌ی کاهش زمان دسترسی پردازنده به اطلاعات مورد نیازش را دارد. عمدتا پردازنده‌های امروزی از چندین دسته[1] تشکیل شده‌اند. هر یک از این دسته‌ها نیز از چندین خط ساخته شده‌اند. با فراخوانی یک داده‌ی مشخص در یک آدرس معین توسط پردازنده در صورت عدم وجود اطلاعات مورد نظر در حافظه‌ی نهان، علاوه بر مقدار فراخوانی شده، داده‌های آدرس بعدی نیز متناسب با ظرفیت خطوط این حافظه به یکی از دسته‌ها منتقل می‌شوند. انتخاب دسته و خط مورد نظر جهت انتقال بلوک اطلاعات، متناسب با شماره‌ی آدرس داده‌ی فراخوانی‌شده صورت می‌گیرد. فیزیکی یا مجازی بودن هریک از آدرس‌ها

[1] set

جهت انتخاب دسته‌ها وخطوط در پردازنده‌های مختلف، متفاوت است. به طور مثال در پردازنده‌های اینتل از آدرس‌های فیزیکی جهت انتقال داده‌ها به حافظه‌ی نهان استفاده می‌شود. به طور مثال جهت انتقال داده‌هایی تک‌بایتی با آدرس‌هایی Nبیتی به یک لایه از حافظه‌ی نهان که دارای ۲^n دسته و خطوطی با ظرفیت $۲^{\&}$ بایت است، بیت‌های $\&$ تا $n+\&-۱$ دسته‌ی میزبان داده‌ی مورد نظرش و بلوک همراه را مشخص می‌نماید. از طرفی دیگر بیت‌های $n+\&$ تا N نیز مقدار برچسب [1] این بلوک را مشخص می‌نمایند.

با پر شدن هر یک از دسته‌های حافظه‌ی نهان از سیاست‌های مختلفی جهت جایگزینی اطلاعات جدید در خطوط هر دسته استفاده می‌شود. پردازنده‌های اینتل از سیاست LRU در هریک از لایه‌های خود برای جایگزینی اطلاعات جدید در دسته‌ها استفاده می‌کنند. مطابق با این سیاست خطی از دسته که زودتر پرشده‌است زودتر نیز خالی‌می‌شود. با این اوصاف در صورت انتقال w بلوک تصادفی به یک دسته‌ی w خطی تمام اطلاعات قبلی آن دسته پاک خواهد شد.

پردازنده‌ی اینتل مورد حمله‌ی این مقاله شامل دو لایه حافظه‌ی نهان اختصاصی برای هر هسته و یک لایه‌ی مشترک بین تمام هسته‌ها است. لایه‌ی مشترک آخرین و لایه‌های اختصاصی اولین لایه‌های حافظه‌ی نهان این پردازنده هستند. در این پردازنده‌ها اطلاعات موجود در لایه‌های ابتدایی حافظه‌ی نهان در لایه‌ی آخر این حافظه نیز وجود دارد، به این نوع از حافظه‌ها، حافظه‌های نهان شامل [2] گفته می‌شود. با این اوصاف در صورتی که پردازش‌های مورد استفاده در این مقاله اقدام به پاک کردن دسته‌های لایه‌ی آخر نمایند، در صورت وجود این اطلاعات در لایه‌های بالایی آن‌ها نیز پاک خواهند شد.

۲.۲ حافظه‌ی مجازی

جهت عدم تداخل اطلاعات موجود در پردازش‌های مختلف یک پردازنده، سیستم‌عامل‌ها به داده‌های موجود در هریک ازپردازش‌ها، آدرس‌هایی با نام آدرس‌های مجازی اختصاص می‌دهند. به فضای نگهدارنده‌ی این آدرس‌ها حافظه‌ی مجازی گفته می‌شود. این حافظه در پردازنده‌های مختلف به صفحاتی با ظرفیت مشخص تقسیم می‌شود. سایز این صفحات در پردازنده‌های مختلف متفاوت است. سایز این صفحات تعداد بیت‌های کوچک‌تر یکسان بین آدرس‌های مجازی و فیزیکی را مشخص می‌نماید. کوچک‌ترین سایز مورد استفاده در حافظه‌ی مجازی پردازنده‌های اینتل 4kB است. 4kB بودن سایز صفحات موجود در پردازنده‌های اینتل به معنای برابری ۱۲ بیت کوچک‌تر آدرس مجازی داده‌های موجود در این پردازنده‌ها با ۱۲ بیت کوچک‌تر آدرس فیزیکی آن‌ها است. با اجرا کردن حمله‌ی این مقاله بر روی پردازنده‌های با سایز صفحات 4kB، امکان اجرا کردن آن بر روی صفحات بزرگ‌تر نیز واضح خواهد بود.

[1] tag [2] inclusive

۳.۲ پیاده‌سازی نرم‌افزاری سیستم رمزنگاری AES

سیستم رمزنگاری AES شامل ده دور متوالی است. هر دور از این سیستم رمزنگاری ۱۶ مقدار $x^i \oplus k^i$ را به عنوان ورودی دریافت کرده و با اجرای عملیات‌هایی جبری شامل جمع بیتی با کلید هر دور، و عبور از بخش‌های sub-byte و shift row و mix-column، ۱۶ مقدار را به عنوان خروجی تحویل می‌دهد. در نه دور اول سیستم رمزنگاری چهار عملیات مذکور بر روی هریک از ورودی‌های هر دور صورت می‌گیرد. در دور آخر فقط به اعمال سه عملیات اول جمع بیتی کلید و sub-byte و shift row بر روی ورودی‌ها اقدام می‌شود. با مورد استفاده قرار دادن کتابخانه‌های رایجی مانند openssl در جهت پیاده‌سازی این سیستم رمزنگاری، می‌توان ستون‌های خروجی دورهای اول تا نهم از این سیستم رمزنگاری را با استفاده از جدول‌های مراجعه‌ی موجود در (۱) به صورت (۲) نشان داد. در (۲)، $Z^{۰,i}$ به صورت $Z^{۰,i} = x^{۰,i} \oplus k^{۰,i}$ محاسبه می‌شود.

$$T^{۰}(Z) = \begin{pmatrix} ۲*S(z) \\ S(z) \\ S(z) \\ ۳*S(z)) \end{pmatrix} \quad T^{۱}(Z) = \begin{pmatrix} ۳*S(z) \\ ۲*S(z) \\ S(z) \\ S(z)) \end{pmatrix}$$
$$T^{۲}(Z) = \begin{pmatrix} S(z) \\ ۳*S(z) \\ ۲*S(z) \\ S(z)) \end{pmatrix} \quad T^{۳}(Z) = \begin{pmatrix} S(z) \\ S(z) \\ ۳*S(z) \\ ۲*S(z)) \end{pmatrix} \quad (۱)$$

$$\begin{aligned} W^{۰,i} &= T^{۰}(Z^{۰,i}) \oplus T^{۱}(Z^{۵,i}) \oplus T^{۲}(Z^{۱۰,i}) \oplus T^{۳}(Z^{۱۵,i}) \\ &\quad \oplus [k^{۰,i} \cdot k^{۱,i} \cdot k^{۲,i} \cdot k^{۳,i}]^t \\ W^{۱,i} &= T^{۰}(Z^{۴,i}) \oplus T^{۱}(Z^{۹,i}) \oplus T^{۲}(Z^{۱۴,i}) \oplus T^{۳}(Z^{۳,i}) \\ &\quad \oplus [k^{۴,i} \cdot k^{۵,i} \cdot k^{۶,i} \cdot k^{۷,i}]^t \\ W^{۲,i} &= T^{۰}(Z^{۸,i}) \oplus T^{۱}(Z^{۱۳,i}) \oplus T^{۲}(Z^{۲,i}) \oplus T^{۳}(Z^{۷,i}) \\ &\quad \oplus [k^{۸,i} \cdot k^{۹,i} \cdot k^{۱۰,i} \cdot k^{۱۱,i}]^t \\ W^{۳,i} &= T^{۰}(Z^{۱۲,i}) \oplus T^{۱}(Z^{۱,i}) \oplus T^{۲}(Z^{۶,i}) \oplus T^{۳}(Z^{۱۱,i}) \\ &\quad \oplus [k^{۱۲,i} \cdot k^{۱۳,i} \cdot k^{۱۴,i} \cdot k^{۱۵,i}]^t \end{aligned} \quad (۲)$$

با نصب کتابخانه‌های openssl جهت پیاده‌سازی رمزنگاری AES، آفست آدرس‌های هر یک از عناصر جدول‌های مراجعه‌ی $T^{۰}$ تا $T^{۳}$ به صورت متوالی درون فایل libcrypto.so قرار می‌گیرند.

۴.۲ حمله‌ی زمانی برنشتاین

در حمله‌ی برنشتاین ارائه شده در [۴] مهاجم با محاسبه‌ی مقدار همبستگی [3]، بین مشخصه‌های زمانی به دست آمده از اجرای سیستم رمزنگاری با یک کلید معلوم در ماشین خود و با یک کلید اصلی بر روی ماشین قربانی، سعی در استخراج کلید سیستم رمزنگاری موردحمله دارد. می‌توان گفت این حمله شامل پنج فاز مهم زیر است:

فاز ۱: تشکیل ماتریس $t[i][j]$، با اجرای سیستم رمزنگاری AES به ازای ورودی‌های مختلف و کلید معلوم k. ماتریس $t[i][j]$ نشان‌دهنده‌ی

[3] correlation

جمع کل زمان اجرای سیستم رمزنگاری AES به ازای $p^i = j$ می‌باشد. در این حالت p^i نیز iامین بایت از ورودی سیستم رمزنگاری AES است.

فاز ۲: تشکیل ماتریس $t'[i][j]$ با اجرای سیستم رمزنگاری مورد حمله و به ازای کلید نامعلوم k'.

فاز ۳: تشکیل مشخصات زمانی با استفاده از اطلاعات به دست آمده و به وسیله‌ی (۳) و (۴).

$$D[i][j] = \frac{t[i][j]}{n[i][j]} - \frac{\sum_i \sum_j t[i][j]}{\sum_i \sum_j n[i][j]} \quad (۳)$$

$$D'[i][j] = \frac{t'[i][j]}{n'[i][j]} - \frac{\sum_i \sum_j t'[i][j]}{\sum_i \sum_j n'[i][j]} \quad (۴)$$

ماتریس‌های $n[i][j]$ و $n'[i][j]$ نیز به ترتیب تعداد تکرارهای ماتریس‌های $t[i][j]$ و $t'[i][j]$ در حین تشکیل آن‌ها در دو فاز اول از حمله هستند.

فاز ۴: محاسبه‌ی مقدار همبستگی با استفاده از (۵). مهاجم در این مرحله برای کلیدهای مربوط به هر بایت یک مقدار حدس زده و همبستگی آن با پروفایل‌های مشخصه‌ی زمانی خود را محاسبه می‌نماید.

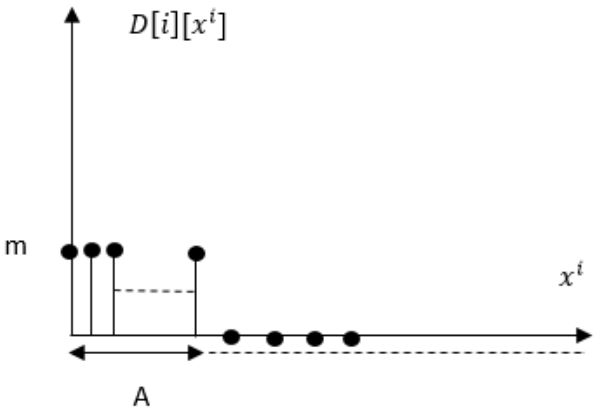
$$c[i][k^{\text{guess}}] = \sum_{x=0}^{255} D[i][x \oplus k^i] * D'[i][x \oplus k^{\text{guess}}] \quad (۵)$$

فاز ۵: مشخص کردن مقادیری از $c[i][k^{\text{guess}}]$ که نسبت به یک مقدار آستانه دارای اختلاف از میانگین بیش‌تری هستند. مقادیر مشخص‌شده در مرحله‌ی ۵، کاندیدهای مناسب برای کلید $k^{i\prime}$ هستند.

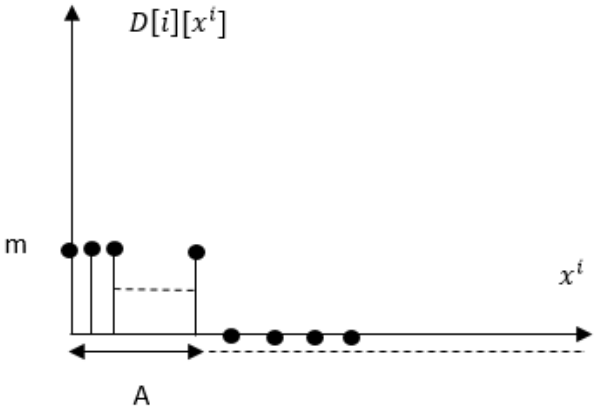
اگر در حین تشکیل ماتریس‌های $t[i][j]$ و $t'[i][j]$، قبل از هر بار اجرای سیستم رمزنگاری AES یک دسته‌ی خاص و ثابت از حافظه‌ی نهان پاک شود مقادیر این ماتریس‌ها و به دنبال آن مقادیر مشخصات زمانی به ازای x^iهایی که منجر به فراخوانی بلوک پاک‌شده از حافظه‌ی نهان می‌شوند، به علت بروز عدم تصادم بزرگ‌تر خواهد بود. به طور مثال اگر در دو فاز اول از حمله‌ی برنشتاین، A عنصر اولِ یکی از جدول‌های مراجعه قبل از هر بار اجرای سیستم رمزنگاری پاک شود، مشخصه‌ی زمانی یکی از بایت‌های ورودی به صورت شکل ۱ خواهد بود. در شکل ۱ از زمان‌های بروز تصادم صرف‌نظر شده است و برابر با صفر قرارگرفته‌اند. مقدار A نیز از با توجه به ۴ بایتی بودن عناصر جداول مراجعه با استفاده از معادله (۶) به دست می‌آید:

$$A = \frac{۲^{۸}}{۴} \quad (۶)$$

با توجه به یکسان بودن معماری پردازنده‌ها در دو فاز اول از حمله، مشخصه‌ی زمانی حاصل‌شده در فاز دوم به دلیل تفاوت کلید با سیستم رمزنگاری اصلی، شیفت‌یافته‌ای از مشخصه‌ی شکل ۱ خواهد بود. با اجرای (۵) بر روی دو مشخصه‌ی مذکور و شیفت $D'[i][x^i]$ بر روی $D[i][x^i]$ به ازای $k^{i\prime}$های مختلف، مقدار همبستگی به ازای A مورد از $k^{i\prime}$ها برابر با Am^2 و به ازای بقیه‌ی $k^{i\prime}$ها برابر با صفر خواهد بود. A



شکل ۱. مشخصه‌ی زمانی با پاک کردن A عنصر متوالی



شکل ۲. مشخصه‌ی زمانی با کاهش تعداد نمونه‌های اندازه‌گیری

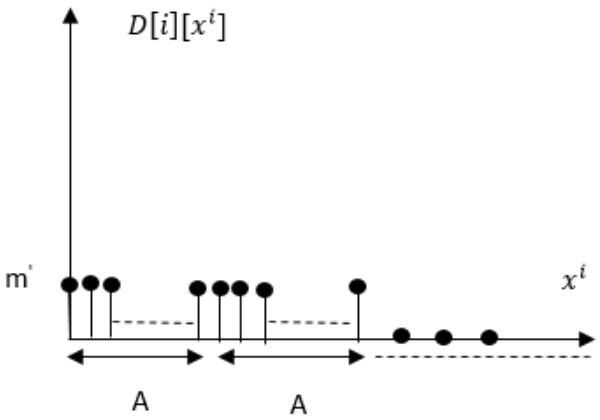
مورد از $k^{i\prime}$های مذکور، کلیدهایی هستند که موجب هم‌پوشانی کامل دو مشخصه‌ی زمانی با یکدیگر می‌شود و در واقع کاندیدهای انتخابی برای $k^{i\prime}$ خواهند بود. این کلیدها دارای z بزرگ‌تر مساوی با کلید اصلی $k^{i\prime}$ خواهند بود. مقدار z از طریق معادله (۷) محاسبه می‌شود.

$$z = ۸ - \log_۲ A \quad (۷)$$

۳ ایده‌ی مقاله

کاهش یافتن مقادیر مشخصات زمانی در پی کاهش تعداد نمونه‌های اندازه‌گیری در فازهای اول و دوم از حمله، امری واضح و بدیهی است. به طور مثال با کاهش تعداد نمونه‌های اندازه‌گیری در دوفاز اولِ حمله بر روی سیستم رمزنگاری مربوط به مشخصه‌ی زمانی شکل ۱، مشخصه‌ی زمانی به صورت تغییر شکل ۲ خواهد کرد. مقدار m' در مشخصه‌ی زمانی جدید از m، کوچک‌تر است.

مقدار همبستگی در مشخصه‌ی زمانی جدید برای کاندیدهای کلید $k^{i\prime}$، برابر با Am'^2 بوده که نسبت مورد قبلی کاهش یافته است. کاهش اختلاف بین مقادیر منسوب به کلیدهای صحیح و غلط، موجب دشوار شدن تمایز بین کلیدهای صحیح و غلط توسط مهاجم می‌شود. در صورتی که به جای افزایش تعداد نمونه‌های اندازه‌گیری در فازهای اول و دوم از



شکل ۳. مشخصه‌ی زمانی با پاک‌کردن $2A$ عنصر متوالی جدول مراجعه

حمله، تعداد دسته‌های متوالی بیش‌تری قبل از اجرای سیستم رمزنگاری در حین تشکیل ماتریس‌های $t[i][j]$ و $t'[i][j]$ پاک شوند، اختلاف بین مقادیر همبستگی منسوب به گروه شامل کلید صحیح نسبت به دیگر گروه‌ها افزایش می‌یابد. به طور مثال با پاک کردن دو دسته‌ی متوالی که به ترتیب میزبان A عنصر اول و A عنصر دوم جدول‌ی مراجعه‌ی مربوط به شکل ۲ هستند، مشخصه‌ی زمانی از شکل ۲ به شکل ۳ تغییر خواهد کرد.

با محاسبه‌ی مقدار همبستگی مشخصات زمانی مربوط به سیستم موردحمله‌ی شکل (۳)، با استفاده از (۵)، مقدار همبستگی به ازای A کاندید شامل کلید صحیح برابر با $2Am'^2$، به ازای دو گروه دیگر از کلیدها که موجب هم‌پوشانی نیمی از مشخصات زمانی فازهای اول و دوم حمله با یکدیگر می‌شود نیز برابر با Am'^2 و به ازای دیگر کلیدها برابر با صفر خواهد بود. همان‌طور که مشاهده می‌شود، اختلاف همبستگی مربوط به گروه شامل کلید صحیح، نسبت به دیگر کلیدها، در این حالت نسبت به مشخصه‌ی زمانی شکل ۳، افزایش یافته است. در صورتی که در دو فاز اول از حمله‌ی برنشتاین تعداد دسته‌های متوالی میزبان یکی از جداول مراجعه به گونه‌ای پاک شود که دامنه‌ی جهش موجود در مشخصه‌ی زمانی نصف کل دامنه‌ی آن مشخصه ار پوشش بدهد، مقدار همبستگی منسوب به گروه شامل کلید صحیح که بیش‌ترین مقدار ممکن نیز هست، برابر با $128m'^2$ و کم‌ترین مقدار همبستگی نیز برابر با ۰ خواهد بود. در صورت افزایش αA واحدی دامنه‌ی جهش از ۱۲۸ و رسیدن آن به $128+\alpha A$، بیش‌ترین و کم‌ترین مقدار همبستگی نیز به ترتیب برابر با $(128+\alpha A)m'^2$ و $2\alpha m'^2$ خواهد بود. بیش‌ترین اختلاف مقدار همبستگی بین گروه شامل کلید صحیح و غلط در حالتی که جهش نصف دامنه‌ی مشخصه‌ی زمانی را پوشش بدهد برابر با $128-\alpha Am'^2$ و در صورت افزایش دامنه از نصف برابر با $128-\alpha Am'^2$ خواهد بود. بدیهی است که بهترین حالت جهت ایجاد تمایز بین کلیدهای صحیح و غلط، حالتی است که نصف دامنه‌ی مشخصه‌ی زمانی توسط جهش پوشیده شده باشد. لازم به ذکر است که لزوما جهش در نصف دامنه‌ی مشخصه‌ی زمانی به معنای دامنه‌ی ۰ تا ۱۲۸ نبوده بلکه می‌تواند هر ۱۲۸ واحد دیگری مانند دامنه‌ی ۱۶ تا ۱۴۴ را نیز شامل شود.

۱.۳ پاک کردن بلوک‌های متوالی جداول مراجعه

پاک کردن هر بلوک دلخواه از حافظه‌ی نهان، نیازمند آگاهی از آدرس فیزیکی آن بلوک است. متناسب با فرض موجود در حملات زمانی، مبنی بر عدم دسترسی مهاجم به آدرس‌های فیزیکی جدول‌های مراجعه، مهاجم باید به دنبال راهکاری جدید برای پاک کردن هر بلوک دلخواه از حافظه‌ی نهان باشد.

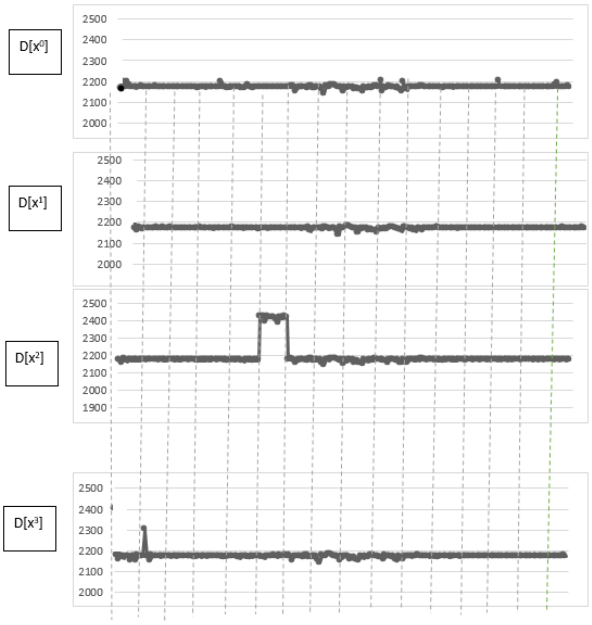
[۱۱] بیان می‌کند که با توجه به متوالی بودن آدرس‌های مجازی عناصر جدول‌های مراجعه، ۴۰۹۶ بایتی بودن مجموع جدول‌های مراجعه و بزرگ‌تر یا مساوی بودن سایز صفحات تقسیم‌کننده‌ی حافظه‌ی مجازی با مقدار 4kB، می‌توان هریک از بلوک‌های دلخواه از جدول‌های مراجعه را با تعداد فراخوانی معقول از عناصری دلخواه که تعداد آن‌ها از طریق (۸) قابل محاسبه است، پاک نمود. الگوریتم ۱ نحوه‌ی شناسایی شماره بلوکی از جدول‌های مراجعه را که با تعداد فراخوانی مشخص از داده‌هایی دلخواه، پاک می‌شوند را نشان می‌دهد. رابطه‌ی (۸) تنها برای پردازنده‌های اینتل که از سیاست جایگزینی LRU پشتیبانی می‌نمایند، صادق است.

$$\frac{\binom{N+2^{n+\&-12}-w-1}{2^{n+\&-12}-1}}{\binom{N+2^{n+\&-12}-1}{2^{n+\&-12}-1}} > \frac{1}{2} \qquad (8)$$

الگوریتم ۱ روال شناسایی ۱۲ بیت کوچک‌تر بلوک‌های جداول مراجعه

```
for (i = 0; i <; i++)
    for (o = 0; o < N; o++)
        maccess(M[o]);
    end for
    t1 = rdtsc();
    AES^k(x);
    t2 = rdtsc();
    c[x^b][j]++;    b = 0.4.8.12   0 ≤ j < 256
    T[x^b][j] = T[x^b][j] + (t2 − t1);
end for
for (b = 0; b < 16; b++)
    for (x^b = 0; x^b < 256; x^b++)
        for (j = 0; j < 256; j++)
            D[x^b][j] = T[x^b][j] / c[x^b][j];
        end for
    end for
end for
```

با توجه به این‌که بیت‌های & تا ۱۱ آدرس مجازی عناصر موجود در ماتریس M به طور یقین با بیت‌های & تا ۱۱ آدرس مجازی یکی از بلوک‌های جدول‌های مراجعه برابر است، با هر بار تکرار فراخوانی عناصر این ماتریس، w بلوک تصادفی درون دسته‌های میزبان یکی از این بلوک‌ها که دارای سیاست جایگزینی LRU است قرار گرفته و موجب خالی‌شدن



شکل ۴. مشخصات زمانی خروجی الگوریتم (۱)

آن از حافظه‌ی نهان می‌شود. با تکرار الگوریتم ۱ به ازای تعداد نمونه‌های مناسب، می‌توان بلوکی را که بیت‌های & تا ۱۱ آدرس مجازی آن با بیت‌های & تا ۱۱ آدرس مجازی عناصر موجود در ماتریس M برابر است شناسایی نمود. شکل ۴، خروجی این الگوریتم بر روی پردازنده‌ی Intel Core i5-4200U که لایه‌ی آخر حافظه‌ی نهان آن دارای ۴۰۹۶ دسته‌ی ۱۲ خطی، با خطوطی به ظرفیت ۶۴ بایت است را نشان می‌دهد. با توجه به بروز جهش در $D[x^{۲}]$ به ازای $\langle x^{۲}\rangle = ۵$ و $\langle k^{۲}\rangle = ۲$ می‌توان گفت بیت‌های & تا ۱۱ آدرس مجازی بلوک هفتم از جدول مراجعه‌ی $T^{۲}$ با بیت‌های & تا ۱۱ از عناصر موجود در ماتریس M برابر است. الگوریتم (۱) فاز پیش از اجرای حمله‌ی ارائه شده در [۱۱] است. در واقع مهاجم با اطلاعات به دست آمده از این مرحله اقدام به شناسایی بیت‌های & تا ۱۱ آدرس مجازی بلوک‌های اولیه‌ی جداول مراجعه‌ی $T^{۰}$ تا $T^{۳}$، جهت اجرای حمله‌ی خود می‌نماید.

مهاجم در گام اول از این مقاله نیازی به اجرای این پیش‌حمله نداشته و فقط باید از پاک شدن ۸ بلوک متوالی از جدول‌های مراجعه قبل از اجرای سیستم رمزنگاری در هریک از فازهای حمله‌ی زمانی برنشتاین، اطمینان حاصل نماید. مهاجم برای رسیدن به این مهم باید اقدام به فراخوانی عناصر ماتریس M به صورت (۹) بنماید.

$$\begin{matrix} M[۰], & M[۴۰۹۶], & \cdots, & M[۴۰۹۶ * N] \\ M[۱۶], & M[۱۶ + ۴۰۹۶], & \cdots, & M[۱۶ + (۴۰۹۶ * N)] \\ M[۳۲], & M[۳۲ + ۴۰۹۶], & \cdots, & M[۱۶ + (۴۰۹۶ * N)] \\ \vdots & \vdots & \vdots & \vdots \\ M[۱۲۸], & M[۱۲۸ + ۴۰۹۶], & \cdots, & M[۱۲۸ + (۴۰۹۶ * N)] \end{matrix} \quad (۹)$$

با توجه به این‌که ظرفیت خطوط حافظه‌ی نهان و عناصر جدول‌های

جدول ۱. رتبه‌ی گروه شامل کلید صحیح به ازای نمونه‌ها و بلوک‌های پاک شده‌ی مختلف (NSB تعداد بلوک‌های متوالی پاک شده و NSM تعداد نمونه‌های اندازه‌گیری لازم است).

NSB / NSM	۱	۲	۳	۴	۵	۶	۷	۸
$۲^{۲۵}$	۱	۱	۱	۱	۱	۱	۱	۱
$۲^{۲۴}$	۱۴	۸	۱	۱	۱	۱	۱	۱
$۲^{۲۳}$	۳	۱۰	۷	۵	۱	۱	۱	۱
$۲^{۲۲}$	۱۰	۴	۶	۴	۱	۱	۱	۱
$۲^{۲۱}$	۵	۳	۸	۲	۳	۹	۱	۱
$۲^{۲۰}$	۶	۷	۲	۲	۲	۹	۸	۱
$۲^{۱۹}$	۷	۵	۳	۹	۳	۹	۲	۱

مراجعه، به ترتیب برابر با ۶۴ و ۴ بایت است، با افزایش ۱۶ واحدی عناصر فراخوانی شونده در هر مرحله، نسبت به مرحله‌ی قبل می‌توان بلوک‌های متوالی از جدول مراجعه را پاک نمود. مقدار N نیز با استفاده از رابطه (۸) محاسبه می‌شود. مهاجم با اجرای فراخوانی‌های مذکور قبل از هریک از فازهای حمله، علاوه بر شناسایی یک گروه مناسب شامل کلید صحیح، بیت‌های ۶ تا ۱۱ آدرس مجازی یکی از بلوک‌های جداول مراجعه را نیز شناسایی نموده و بستر را برای تشخیص کلید صحیح دیگر بایت‌ها نیز فراهم می‌آورد. در واقع با اجرای این عملیات بار محاسباتی ناشی از اجرای الگوریتم (۱) خنثی می‌شود.

۴ نتایج پیاده‌سازی عملی

مهاجم با فراخوانی عناصر ۸ ماتریس بیان‌شده در بخش قبل، گروه شامل کلید صحیح یکی از بایت‌ها را شناسایی نموده و با بستر فراهم شده توسط این فراخوانی بیت‌های ۶ تا ۱۱ آدرس مجازی دیگر بلوک‌های جدول‌های مراجعه را مشخص می‌کند. مهاجم با توجه به آگاهی از بیت‌های ۶ تا ۱۱ آدرس‌های مجازی تمام بلوک‌های جداول مراجعه می‌تواند، با بهره‌مندی از (۸)، اقدام به فراخوانی داده‌هایی جهت تخلیه‌ی تعدادی دلخواه از بلوک‌های متوالی یک جدول مراجعه‌ی دلخواه بنماید. جدول ۱ رتبه‌ی مقدار همبستگی منسوب به گروه شامل کلید صحیح نسبت به دیگر کلیدها را نشان می‌دهد. ستون این جدول تعداد بلوک‌های متوالی پاک‌شده و سطر آن تعداد نمونه‌های اندازه‌گیری برای اجرای حمله را نشان می‌دهد. در واقع مقدار موجود در هر خانه از جدول ۱ نشان می‌دهد که با پاک کردن تعداد مشخصی از بلوک‌های متوالی جدول مراجعه، مهاجم کلید صحیح سیستم رمزنگاری را به عنوان چندمین کلید منتخب برای آن سیستم در نظر می‌گیرد.

همان‌طور که در جدول ۱ قابل مشاهده است، با پاک کردن هشت بلوک متوالی از جدول مراجعه، مقدار همبستگی گروه شامل کلید صحیح به ازای تمام نمونه‌ها دارای بیش‌ترین مقدار است. به عبارتی دقیق‌تر با پاک کردن هشت بلوک متوالی از جدول مراجعه، کلید اصلی سیستم رمزنگاری اولین کلید منتخب مهاجم به عنوان کلید صحیح سیستم رمزنگاری می‌باشد.

جدول ۲. مقایسه‌ی حمله‌ی مورد نظر با دیگر حملات

تعدادنمونه‌ی اندازه‌گیری موردنیاز	پردازنده‌ی موردنظر	مقاله
$۲^{۲۷}$	Pentium 3	[۴]
$۲^{۳۰}$	Acer Iconia	[۶]
$۲^{۲۶}$	Pentium Dual Core	[۱۲]
$۲^{۲۱}$	ARM-Cortex-A	[۱۲]
$۲^{۱۹}$	Intel Core i5-4200U	حمله‌ی ما

۵ نتیجه‌گیری

آزمایش‌ها و بررسی‌های صورت گرفته در این مقاله نشان داد که در صورت پاک کردن نصف بلوک‌های هر جدول مراجعه قبل از اجرای سیستم رمزنگاری در فازهای ۱ و ۲ از حمله‌ی زمانی برنشتاین، تعداد نمونه‌های اندازه‌گیری مورد نیاز حمله به تعداد چشم‌گیری کاهش می‌یابد. ما در این مقاله با به چالش کشیدن موانع اصلی اجرای این ایده که ریشه‌ای بودن دسترسی مهاجم و دسترسی مهاجم به آدرس‌های مجازی بلوک‌های جداول مراجعه می‌باشد، ایده‌ی مذکور را عملی نموده و تعداد نمونه‌های اندازه‌گیری را به $۲^{۱۹}$ کاهش داده‌ایم. جدول ۲ اقدام به مقایسه‌ی تعداد نمونه‌های اندازه‌گیری مورداستفاده‌ی حملات زمانی برنشتاین در مقالات مختلف می‌نماید.

مراجع

[1] Bonneau, Joseph and Mironov, Ilya. Cache-collision timing attacks against aes. in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 201–215. Springer, 2006.

[2] Lauradoux, Cédric. Collision attacks on processors with cache and countermeasures. in *WEWoRC 2005–Western European Workshop on Research in Cryptology*. Gesellschaft für Informatik eV, 2005.

[3] Acıiçmez, Onur, Schindler, Werner, and Koç, Çetin K. Cache based remote timing attack on the aes. in *Cryptographers' track at the RSA conference*, pp. 271–286. Springer, 2007.

[4] Bernstein, Daniel J. Cache-timing attacks on aes. 2005.

[5] Neve, Michael, Seifert, Jean-Pierre, and Wang, Zhenghong. A refined look at bernstein's aes side-channel analysis. in *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pp. 369–369, 2006.

[6] Spreitzer, Raphael and Plos, Thomas. On the applicability of time-driven cache attacks on mobile devices. in *International Conference on Network and System Security*, pp. 656–662. Springer, 2013.

[7] Aciçmez, Onur and Koç, Çetin Kaya. Trace-driven cache attacks on aes. 2006.

[8] Osvik, Dag Arne, Shamir, Adi, and Tromer, Eran. Cache attacks and countermeasures: the case of aes. in *Cryptographers' track at the RSA conference*, pp. 1–20. Springer, 2006.

[9] Yarom, Yuval and Falkner, Katrina. {FLUSH+RELOAD}: A high resolution, low noise, l3 cache {Side-Channel} attack. in *23rd USENIX security symposium (USENIX security 14)*, pp. 719–732, 2014.

[10] Lipp, Moritz, Gruss, Daniel, Spreitzer, Raphael, Maurice, Clémentine, and Mangard, Stefan. {ARMageddon}: Cache attacks on mobile devices. in *25th USENIX Security Symposium (USENIX Security 16)*, pp. 549–564, 2016.

[۱۱] معراجی، و و سلیمانی، ه. ارائه‌ی یک حمله‌ی Access-Driven جدید بر روی پردازنده‌های اینتل. در بیست و چهارمین کنفرانس ملي سالانه انجمن کامپیوتر ایران، ۱۳۹۷.

[12] Aly, Hassan and ElGayyar, Mohammed. Attacking aes using bernstein's attack on modern processors. in *International Conference on Cryptology in Africa*, pp. 127–139. Springer, 2013.

[13] Spreitzer, Raphael and Gérard, Benoît. Towards more practical time-driven cache attacks. in *IFIP International Workshop on Information Security Theory and Practice*, pp. 24–39. Springer, 2014.

[14] Veyrat-Charvillon, Nicolas, Gérard, Benoît, Renauld, Mathieu, and Standaert, François-Xavier. An optimal key enumeration algorithm and its application to side-channel attacks. in *International Conference on Selected Areas in Cryptography*, pp. 390–406. Springer, 2012.