رویکردها و چالش‌های مدل جعبهٔ سفید در پیاده‌سازی الگوریتم‌های رمزنگاری قالبی

هادی سلیمانی[1]* و محمدرضا صادقی[2]

[1] استادیار پژوهشکده فضای مجازی، دانشگاه شهید بهشتی، تهران، ایران
h_soleimany@sbu.ac.ir

[2] پژوهشکده فضای مجازی، دانشگاه شهید بهشتی، تهران، ایران
Mohamm.sadeghi@mail.sbu.ac.ir

چکیده

با رشد فناوری و گسترش ابزارهای نرم‌افزاری و سخت‌افزاری، حملات علیه رمزهای قالبی جنبه‌های جدیدی یافته است. در بسیاری از موارد، مهاجمان به جای تلاش برای انجام حملات تحلیل نظری و محاسباتی، از نقاط ضعف موجود در نحوه پیاده‌سازی رمزهای قالبی استفاده می‌کنند. هر چقدر میزان دسترسی مهاجمان به جزئیات پیاده‌سازی رمزهای قالبی بیشتر باشد، شانس حملات موفق افزایش می‌یابد. بنابراین، طراحی و ارائه روش‌هایی برای پیاده‌سازی امن رمزهای قالبی، که حتی در صورت فاش‌شدن تمام جزئیات پیاده‌سازی، کماکان از شکسته‌شدن توسط مهاجمان ایمن بمانند، اهمیت زیادی یافته است. در این مقاله ابتدا مدل‌های مختلف پیاده‌سازی رمزهای قالبی را توضیح خواهیم داد؛ سپس مفاهیم اساسی رمزنگاری جعبهٔ سفید را بیان خواهیم کرد. در ادامه با بیان کاربردهای رمزنگاری جعبهٔ سفید، چندین طرح ارائه‌شده برای پیاده‌سازی جعبهٔ سفید رمزهای قالبی را شرح خواهیم داد.

واژگان کلیدی: رمزهای قالبی، پیاده‌سازی امن، رمزنگاری جعبهٔ سفید

۱- مقدمه

طی چند دهه اخیر، طرح‌های متنوعی برای رمزهای قالبی پیشنهاد شده است که از آن جمله می‌توان به طرح‌های DES[1]، AES[2]، CLEFIA[3] و طرح‌های جدیدتر ZORRO[4] و SKINNY[5] اشاره کرد. همه طرح‌های ارائه‌شده، از بدو معرفی تاکنون، مورد بررسی و تحلیل نظری قرار گرفته و آسیب‌پذیری آن‌ها در برابر انواع حملات خطی، تفاضلی و ...، مشخص شده است. با گسترش فناوری‌های نوین و توسعه شبکه‌های رایانه‌ای، ساختار و روش حملات علیه رمزهای قالبی تغییرات مهمی کرده است [۶, ۷]. برای مثال، نفوذ به دستگاه رمزنگار/رمزگشا از طریق بدافزار و سرقت اطلاعات الگوریتم رمزنگاری از حافظه دستگاه مورد حمله [۶-۸]، تحلیل نرم‌افزارهای رمزنگار/رمزگشا با استفاده از ابزارهای رفع اشکال کد[1] [۷, ۹]، و تحلیل مصرف توان [۱۰] یا حافظه نهان[2] [۱۱] یک دستگاه رمزنگار/رمزگشا، روش‌هایی هستند که با استفاده از نقاط ضعف برنامه‌نویسی و نحوه پیاده‌سازی الگوریتم‌های رمز قالبی، آن‌ها را مورد حمله قرار می‌دهند.

یک عامل مهم در میزان موفقیت حملات مبتنی بر ضعف پیاده‌سازی رمزهای قالبی، سطح دسترسی حمله‌کننده به جزئیات پیاده‌سازی است. میزان مصرف توان، نحوه تخصیص و مدیریت حافظه و نحوه برنامه‌نویسی الگوریتم رمزنگار/رمزگشا، از جمله جزئیاتی هستند که حمله‌کنندگان میل زیادی به استفاده از آن‌ها دارند و حملات شناخته‌شدهٔ بسیاری –نظیر حملات تحلیل توان- بر مبنای آن‌ها ارائه شده‌اند. از نظر سطح دسترسی مهاجم به سامانه‌های رمزنگاری، حملات به سه دسته کلی تقسیم می‌شوند: حملات جعبهٔ سیاه، حملات جعبهٔ خاکستری و حملات جعبهٔ سفید.

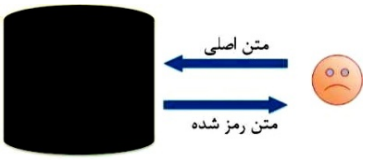
۱-۱- حملات مدل جعبهٔ سیاه

مهاجم از جزئیات داخلی سیستم (نحوه پیاده‌سازی، مقادیر میانی داده‌ها، اطلاعات نشت‌شده از حافظه و ...) بی‌اطلاع است و تنها می‌تواند متن ورودی و متن رمزشدهٔ خروجی متناظر با

[1] Code debugging
[2] Cache memory

* نویسندهٔ مسؤول

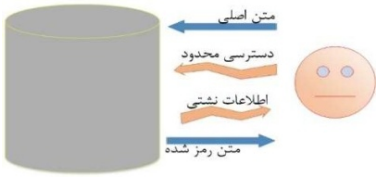
آن را مشاهده کند، همانند آن‌چه در شکل (۱) نمایش داده شده است. در این حملات، هدف مهاجم استخراج کلید از سامانهٔ رمزنگاری است. با فرض مشخص‌بودن نوع الگوریتم رمز قالبی برای مهاجم، حملاتی چون جستجوی کامل کلید[1] یا حملات متن منتخب[2] در این دسته جای می‌گیرند [۷].



(شکل-۱): مدل حملات جعبهٔ سیاه

۱-۲- حملات مدل جعبه خاکستری

در مدل جعبه خاکستری، مطابق شکل (۲)، مهاجم علاوه‌بر مقادیر متن اصلی و متن رمزشده، اطلاعات محدودی راجع به جزئیات داخلی سیستم (مصرف توان، اطلاعات نشت‌شده از حافظه و ...) دراختیار دارد. همچنین مهاجم می‌تواند به‌صورت محدود دسترسی‌هایی به سیستم داشته باشد (به‌عنوان مثال خطاهایی را به سامانه القا کند). در این حملات، هدف مهاجم استخراج کلید از سامانهٔ رمزنگاری است. حملات کانال جانبی در این دسته جای دارند [۷].



(شکل-۲): مدل حملات جعبه خاکستری

۱-۳- حملات مدل جعبهٔ سفید

در این مدل، همان‌گونه که در شکل (۳) نمایش داده شده است، مهاجم از تمام جزئیات داخلی سامانهٔ رمزنگاری نظیر مقادیر میانی داده‌ها، منابع حافظه و مقادیر درون آن‌ها، نحوه اجرای الگوریتم و ... اطلاع دارد و می‌تواند تغییرات دلخواه خود را در سامانه رمزنگاری اعمال کند.

سامانهٔ رمزنگاری که در محیط جعبهٔ سفید پیاده‌سازی می‌شود، در معرض تهدیدات گسترده‌تری قرار دارد. اگر سامانهٔ رمزنگاری بدون ملاحظات امنیتی در برابر حملات جعبهٔ سفید پیاده‌سازی شود، مهاجم جعبهٔ سفید به‌راحتی می‌تواند کلید رمزنگاری را به‌دست آورد، و یا تمام الگوریتم رمزنگاری/رمزگشایی را برداشت کرده، عملیات رمزگشایی/رمزنگاری را بازسازی و اصل یا معکوس الگوریتم را در محل دیگری اجرایی کند [۶-۸, ۱۲-۱۴].

(شکل-۳): مدل حملات جعبهٔ سفید

۱-۴- مثال‌هایی از حملات مدل جعبهٔ سفید و مدل جعبهٔ خاکستری

بدافزارهایی که محیط (سرور، رایانه، دستگاه‌های تلفن همراه هوشمند و ...) را آلوده می‌کنند، می‌توانند بر روند اجرای نرم‌افزارهای رمزنگاری نظارت کنند. همچنین می‌توانند کلمات عبور واردشده توسط کاربران را رصد کرده و تمام این اطلاعات را به محلی دیگر ارسال کنند[۷, ۸, ۱۵, ۱۶].

سامانه‌هایی که به کاربران متعددی خدمت‌رسانی می‌کنند (مثل خدمات مبتنی بر رایانش ابری)، در معرض سوء استفاده کاربران نامطمئن خود هستند. یک کاربر نامطمئن، قادر است با استفاده از نرم‌افزارهای رفع اشکال کد، کدهای اجراشده در سامانه را تشریح و از آن‌ها سوء استفاده کند [۷, ۹].

بردهای رایانه‌ای که امروزه در بسیاری از محیط‌های صنعتی و پژوهشی برای پیاده‌سازی سامانه‌های الکترونیکی، شبکه‌های مانیتورینگ صنعتی، پردازش سیگنال و تصویر و ... مورد استفاده قرار می‌گیرند، در معرض تهدیداتی هستند که ناشی از ضعف پیاده‌سازی الگوریتم‌های رمزنگاری و ضعف امنیتی معماری پردازنده‌های این بردها است. از جمله حملات علیه بردهای رایانه‌ای، حملات کانال جانبی زمان هستند [۱۷] که در مدل جعبهٔ خاکستری جای دارند.

برای برنامه‌ریزی برخی از FPGA[3]ها، جهت جلوگیری از فاش‌شدن نحوهٔ پیکره‌بندی سخت‌افزار، قابلیت رمزنگاری رشته‌بیت[4] برنامه با استفاده از الگوریتم‌هایی چون AES وجود دارد؛ لذا در سخت‌افزار این دسته از FPGAها، بلوک‌هایی برای رمزگشایی از رشته‌بیت برنامه وجود دارند. در برخی از انواع FPGAهای تجاری، این بلوک‌ها آسیب‌پذیر نشان داده و تحت

[1] Exhaustive Key Search
[2] Chosen-Plaintext Attacks
[3] Field Programmable Gate Array (FPGA)
[4] Bitstream

حملات کانال جانبی توان، کلید رمزگشایی به‌صورت کامل از آن‌ها استخراج شده است [۱۸]. به‌علاوه، پیاده‌سازی سخت‌افزاری الگوریتم‌های رمزنگاری بر روی FPGAها، حتی با وجود سازوکارهای امنیتی چون پوشش[1] اطلاعات، در مدل جعبهٔ سفید و جعبهٔ خاکستری می‌تواند هدف حملاتی موفق باشد [۱۹].

در سال‌های اخیر، روش‌هایی که در ابتدا برای پیاده‌سازی رمزهای قالبی به‌صورت مقاوم در برابر حملات مدل جعبهٔ سفید ابداع شدند، بر خلاف انتظارات توسط حملات سخت‌افزاری [۲۰] و نرم‌افزاری [۹] در مدل جعبه خاکستری نیز شکسته شده و مشخص شد حتی وجود دسترسی‌های محدود مهاجمان نیز می‌تواند به انجام حملات موفق منجر شود [۲۱].

با توجه به توضیحات و مثال‌های یادشده، می‌توان گفت پیاده‌سازی رمزهای قالبی در شرایطی که مهاجمان دارای سطح دسترسی فراتر از مدل جعبهٔ سیاه هستند، از چند جهت می‌تواند آسیب‌پذیر باشد:

۱- اگر ساختار و معماری پردازنده‌های سامانهٔ رمزنگار/رمزگشا دارای نقاط ضعفی چون نشت اطلاعات از کانال جانبی توان، زمان یا حافظه باشد، مهاجمان در مدل جعبهٔ خاکستری و جعبهٔ سفید قادر به استخراج کلید از الگوریتم رمزنگاری/رمزگشایی هستند [۱۱, ۱۷, ۱۸].

۲- اگر الگوریتم رمزنگاری/رمزگشایی بدون درنظرگرفتن ملاحظات امنیتی مدل‌های جعبه خاکستری و جعبهٔ سفید پیاده شوند، اطلاعات نشت‌شده از آن‌ها توسط مهاجمان مورد استفاده قرار گرفته و الگوریتم مورد نظر شکسته خواهد شد [۱۹].

۳- الگوریتم‌های رمز قالبی در محیط‌هایی که گمان نمی‌رود مهاجمان دارای فرصت کافی برای انجام حملات موفق باشند نیز در معرض آسیب هستند. سرقت همه یا بخشی از اطلاعات و کدهای الگوریتم رمزنگار/رمزگشا توسط بدافزارها یا دیگر مهاجمان، فرصت کافی را برای نفوذ به آن‌ها در محیطی دیگر فراهم می‌سازد [۶-۸, ۱۲].

۴- حتی در طراحی و پیاده‌سازی روش‌هایی که برای محیط‌های جعبهٔ سفید در نظر گرفته شده‌اند نیز هرگونه سهل‌انگاری یا محدودپنداشتن توانایی‌های مهاجمان، می‌تواند منجر به بروز حملات موفق شود [۲۱, ۲۲].

مطابق آن‌چه گفته شد، هرگونه دسترسی محدود و جزئی برای مهاجمان می‌تواند منجر به بروز حملات موفق باشد، لذا طراحی، ارزیابی و پیاده‌سازی روش‌هایی که در برابر حملات مدل جعبهٔ سفید مقاوم باشند، اهمیت زیادی یافته است.

در ادامهٔ مقاله، مطالب به این صورت سازماندهی شده است: در بخش دوم، مفاهیم اساسی در رمزنگاری جعبهٔ سفید توضیح داده می‌شود. در بخش سوم، کاربردها و ضرورت استفاده از رمزنگاری جعبهٔ سفید بیان و در بخش چهارم، روش‌های مهم ارائه‌شده برای رمزنگاری جعبهٔ سفید معرفی خواهند شد. در بخش پنجم، تعدادی از حملات مهم را علیه روش‌های جعبهٔ سفید توضیح خواهیم داد و درنهایت، در بخش ششم، جمع‌بندی و نتیجه‌گیری مطالب ارائه می‌شود.

۲- مفاهیم اساسی در رمزنگاری جعبهٔ سفید

رمزنگاری جعبهٔ سفید، رویکردی است که به‌طور عمده بر روی طراحی و پیاده‌سازی رمزهای قالبی متمرکز شده است؛ به‌گونه‌ای که در برابر حملات مدل جعبهٔ سفید، مقاوم باشند. در ادامه، مفاهیم اساسی و تعاریفی که در طراحی روش‌های جعبهٔ سفید اهمیت ویژه‌ای دارند، بیان خواهد شد.

- **حملهٔ استخراج کلید**

حمله‌کننده تلاش می‌کند با استفاده از نقاط ضعف موجود در نحوهٔ پیاده‌سازی الگوریتم‌های رمزنگاری، کلید یا کلیدهای مخفی را استخراج کند [۶, ۷].

- **حملهٔ برداشت کد[2]**

از آن‌جا که بسیاری از طراحان الگوریتم‌های رمزهای قالبی در مدل جعبهٔ سفید، تلاش می‌کنند تا امنیت طرح خود را با امنیت رمز قالبی استفاده‌شده در طرحشان، در حالت جعبهٔ سیاه معادل سازند [۸, ۱۲, ۱۳, ۲۳, ۲۴]، دشواری استخراج کلید در مدل جعبهٔ سفید با دشواری استخراج کلید در مدل جعبهٔ سیاه یکسان خواهد بود؛ لذا مهاجمان، سرقت کل کدهای الگوریتم جعبهٔ سفید را به‌عنوان راه حلی جایگزین در نظر خواهند داشت. با دراختیارداشتن همه یا بخشی از کدهای الگوریتم رمزنگاری، امکان بازسازی الگوریتم پیاده‌شده در محل دیگری میسر خواهد بود و نیازی به کشف کلید رمز قالبی استفاده‌شده وجود نخواهد داشت؛ چون حمله‌کننده می‌تواند به جای استخراج کلید، کدهای مربوط به عملیات رمزنگاری/رمزگشایی را برداشت کرده و در محل دیگری به سوء استفاده از آن‌ها بپردازد [۷, ۸].

[1] Masking

[2] Code Lifting Attack

• **امنیت ضعیف رمزنگاری جعبهٔ سفید**[1]

این معیار میزان مقاومت یک الگوریتم رمزنگاری جعبهٔ سفید در برابر حملات برداشت کد را از نظر محاسباتی بیان می‌کند و به‌صورت زیر تعریف می‌شود:

تابع F که رمز قالبی E_K را به‌صورت جعبهٔ سفید پیاده کرده است ($F(E_K)$) دارای امنیت ضعیف از مرتبه T است، اگر یافتن بخشی از F با اندازه کوچک‌تر از T که امکان بازسازی F را به‌صورت کامل میسر می‌سازد، از نظر محاسباتی دشوار است. به‌عبارتی مهاجم باید کدی دست‌کم به اندازه T را از کل کدهای تابع F در اختیار داشته باشد تا بتواند عملکرد الگوریتم جعبهٔ سفید موجود را بازسازی کند و امکان یافتن یک تابع فشرده با عملکردی مشابه با F وجود ندارد [۸, ۱۲].

درصورتی‌که مقدار T به اندازه کافی بزرگ باشد، حمله برداشت کد دشوار خواهد بود؛ زیرا مهاجم با دراختیارداشتن بخشی از کد با اندازهٔ کمتر از T، قادر به بازسازی تابع F نیست.

در صورت تحقق این معیار با T به‌اندازهٔ کافی بزرگ، بازسازی تابع F برای مهاجم دشوار است، اما درصورتی‌که بتواند F را بازسازی یا به‌طور کامل استخراج کند، می‌تواند عکس الگوریتم رمزنگاری/رمزگشایی را از روی آن پیاده کند؛ لذا این معیار را به‌صورت امنیت ضعیف رمزنگاری جعبهٔ سفید معرفی کرده‌اند [۸, ۱۲].

• **امنیت قوی رمزنگاری جعبهٔ سفید**[2]

این معیار، دربرگیرنده معیار "امنیت ضعیف رمزنگاری جعبهٔ سفید" و "بازگشت ناپذیری الگوریتم رمزنگاری جعبهٔ سفید" به‌صورت توأم است. بازگشت ناپذیری الگوریتم رمزنگاری جعبهٔ سفید به صورت زیر تعریف می‌شود:

فرض شود زوج (E,D) الگوریتم‌های رمزنگاری و رمزگشایی تحت کلید K باشند و $F(E_K)$ تابعی باشد که E_K را محاسبه می‌کند؛ اگر با دراختیارداشتن تابع اجرای D_K یا هر تابع معادل دیگری، بازسازی $F(E_K)$ از نظر محاسباتی دشوار باشد، $F(E_K)$ بازگشت ناپذیر است [۱۲].

در صورت تحقق معیار "امنیت قوی رمزنگاری جعبهٔ سفید"، علاوه‌بر دشواربودن فشرده‌سازی F، بازسازی الگوریتم F از روی الگوریتم معکوس آن یعنی D نیز دشوار است [۸, ۱۲].

با برآورده‌ساختن معیار امنیت قوی رمزنگاری جعبهٔ سفید، می‌توان از یک الگوریتم جعبهٔ سفید مبتنی بر رمزهای قالبی، به‌عنوان یک رمز نامتقارن نیز بهره برد؛ چون با دراختیارداشتن یکی از توابع رمزنگاری یا رمزگشایی، تابع دیگر قابل بازسازی نخواهد بود [۱۲, ۲۵].

• **(M,Z)-Space Hardness**

این معیار تعریف دقیق‌تری برای ارزیابی مقاومت سامانه در برابر حملات برداشت کد ارائه می‌دهد. بر اساس این معیار، اگر حمله‌کننده، حجمی معادل با مقدار M از کل کد اجرای عملیات رمزنگاری را دراختیار داشته باشد، احتمال محاسبه متن رمزشده متناظر با یک متن اصلی تصادفی به‌صورت صحیح، 2^{-z} است. به‌عبارتی، هرچه مقدار M (یعنی حجم کد برداشت‌شده از عملیات رمزنگاری) بیشتر باشد، مقدار z کوچک‌تر خواهد بود (یعنی مهاجم شانس بیش‌تری برای افشای کلید یا بازسازی کل الگوریتم داشته و با احتمال بیشتری قادر به محاسبه متن رمزشدهٔ صحیح متناظر با یک متن تصادفی است) [۸].

۳- کاربردها و ضرورت‌های استفاده از رمزنگاری جعبهٔ سفید

پیاده‌سازی سامانه‌هایی که در برابر حملات مدل جعبهٔ سفید مقاوم باشند، در موارد زیر کاربرد دارد:

۳-۱- امنیت کاربران و ارائه‌دهندگان خدمات الکترونیکی بر بستر شبکه

با گسترش خدمات مبتنی بر شبکه و رایانش ابری، بسیاری از تولیدکنندگان خدمات و محتوای الکترونیکی (نظیر خدمات بانکی، نرم‌افزار، کتاب، موسیقی، بازی، شبکه‌های اجتماعی، فیلم، سریال و ...)، ترجیح می‌دهند محصولات خود را در قالب خدمات مبتنی بر شبکه به کاربران خود عرضه کنند. در ارائهٔ خدمات مبتنی بر شبکه، کاربران با توجه به جایگاه و سطح دسترسی‌شان، قادر به استفاده از خدمات مشخص و یا بارگیری یا بارگذاری محتوای الکترونیکی در سرور هستند. شناخته‌شده‌ترین راه برای تعیین سطح دسترسی کاربران به محتوای الکترونیکی، ایجاد حساب‌های کاربری برای مشتریان است. جایگاه و سطح دسترسی هرکاربر با توجه به اطلاعات ذخیره‌شده در حساب کاربری او (مثل کلیدعبور، پرداخت‌ها، امتیازات کاربری و ...) مشخص می‌شود. در شکل (۴)، معماری کلی خدمات تحت شبکه نمایش داده شده است.

[1] Weak White-Box Security (WWBS)
[2] Strong White-Box Security(SWBS)

(شکل-۴): معماری کلی خدمات تحت شبکه

مشتریان خوش‌رفتار این خدمات به‌طورمعمول از طریق رایانه یا تلفن همراه خود به حساب کاربریشان وارد شده و از خدمات مجاز متناسب با سطح دسترسی خود بهره‌مند می‌شوند. در این فرآیند، کاربران اطلاعات حساسی چون رمزعبور حساب کاربری، رمزکارت‌های بانکی، فایل‌ها و ... را از طریق شبکه با سرور مورد نظر خود تبادل می‌کنند و خدمات و داده‌های دیجیتال مورد نظر خود را از سرور دریافت می‌کنند.

در فرآیند تبادل اطلاعات میان مشتریان و سرور، برنامه نرم‌افزاری سمت مشتری درعمل در یک محیط جعبهٔ سفید اجرا می‌شود، چون رایانه و گوشی‌های هوشمند شخصی در معرض تهدیدات امنیتی گسترده با سطح دسترسی بالا برای مهاجمان قرار دارند. از جمله این تهدیدات می‌توان به موارد زیر اشاره کرد:

- امکان آلودگی رایانه یا تلفن همراه به بدافزارهایی که اطلاعات شخصی و حساس کاربران را از طریق برنامه سمت مشتری سرقت کرده و برای مهاجم ارسال می‌کنند. این بدافزارها می‌توانند از طریق شبکه منتشر و یا از طریق نرم‌افزارهای غیر قابل اعتماد و مخرب وارد سامانه شوند [۶, ۷, ۱۵, ۱۶].
- شنود بسته‌های تبادل‌شده میان برنامه سمت مشتری و سرور و استخراج اطلاعات کلیدی و حساس از آن [۲۶]
- تکثیر و تحلیل نرم‌افزار یا داده‌های حساس موجود در سیستم با سرقت یا دسترسی به سیستم بدون اطلاع مالک آن [۹]

لذا الگوریتم‌های رمزنگاری به کار رفته در سامانه‌ها و برنامه‌های سمت مشتری باید در برابر حملات مدل جعبهٔ سفید مقاوم باشند تا کاربران در بستری امن به استفاده از خدمات مبتنی بر شبکه بپردازند.

از جمله پیامدهای حملات جعبهٔ سفید و سرقت اطلاعات از دستگاه تحت کاربری افراد، می‌توان به افشای رمزهای عبور، اطلاعات مالی و بانکی، اطلاعات و فایل‌های مشمول حریم خصوصی اشاره کرد. این موارد ممکن است زیان‌های مالی، اجتماعی، حقوقی و... را به کاربران تحمیل کند [۶, ۷, ۱۳, ۲۶, ۲۷].

در ادامه، به مواردی از کاربردهای رمزنگاری جعبهٔ سفید برای حفظ امنیت در سمت مشتری اشاره خواهد شد.

۳-۱-۱- حمایت از حقوق مولفین و مصنّفین

مشتریان بدرفتار و مهاجمان (از جمله اشخاص، بدافزارها و ...)، با نفوذ به برنامه سمت مشتری که در اختیار کاربران مجاز است، قادر به دسترسی غیرمجاز به رمزهای عبور و جزئیات حساب‌های کاربری و سوء استفاده از آن هستند. به این ترتیب، کسانی که مجاز به استفاده از همه یا بخشی از محتوای الکترونیکی موجود در سرور نبوده‌اند، قادر خواهند بود از آن‌ها به‌صورت غیرمجاز بهره‌برداری کرده و به تکثیر غیر قانونی آن بپردازند. به این ترتیب پدیدآورندگان محتوای دیجیتال با خسارات مالی مواجه خواهند شد [۶, ۷, ۱۲, ۱۳].

اگر برنامه سمت سرور برای انجام رویه‌های جاری و تبادل اطلاعات با تمام مشتریان از الگوریتم‌ها و کلیدهای مشابه استفاده کند، شرایط بدتر نیز خواهد شد، زیرا به‌دست‌آوردن اطلاعات و کلیدهایی که در اختیار یک کاربر است، تمام رویه‌های سمت سرور، اطلاعات داخل آن و سایر مشتریان را تهدید خواهد کرد.

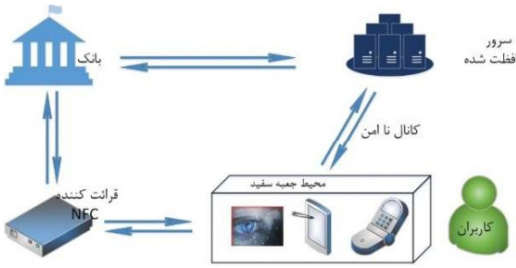
استفاده از الگوریتم‌های رمز قالبی طراحی‌شده برای محافظت از داده‌ها در محیط جعبهٔ سفید، می‌تواند به تقویت امنیت برنامه‌های سمت مشتری و اطلاعات تبادل‌شده از طریق آن منجر شده و به جلوگیری از بروز خسارات مالی برای پدیدآورندگان محتوای دیجیتال کمک کند.

۳-۱-۲- تأمین امنیت تراکنش‌های مالی

فناوری نرم‌افزاری نوین HCE[1]، امکان یک‌پارچه‌سازی اعتبارات و حساب‌های مالی در دستگاه‌های تلفن همراه کاربران را فراهم ساخته است. کاربرانی که تلفن همراه آنان مجهز به سخت‌افزار NFC[2] باشد، قادرند از تلفن همراه خود به‌عنوان کارت اعتباری استفاده کنند. در شکل (۵) طرح کلی معماری و نحوه کارکرد فناوری HCE نمایش داده شده است. مزیت این فناوری، امکان یک‌پارچه‌سازی حساب‌های مالی و پرداخت‌ها است. از جمله کاربردهای فناوری HCE می‌توان به

[1] Host Card Emulation
[2] Near Field Communication

پرداخت بهای بلیط در سامانه‌های حمل و نقل عمومی، پرداخت در فروشگاه‌ها و ... اشاره کرد. با استفاده از این فناوری، پولی میان طرفین مبادله نمی‌شود، بلکه اطلاعات و اعتبارات کاربران در یک سرور امن، میان حساب کاربران انتقال می‌یابد. فناوری HCE از زمان انتشار سیستم‌عامل اندروید KitKat نسخه ۴,۴ و بالاتر، قابل دسترسی است. همچنین سیستم‌عامل‌های مطرح دیگر ازجمله iOS نیز برای کاربران خود امکان استفاده از این فناوری را میسر ساخته‌اند. شرکت‌های بزرگی چون Google، Master Card و Visa ، از فناوری HCE پشتیبانی می‌کنند.



(شکل-۵): طرح کلی معماری و نحوه کارکرد فناوری HCE

واردکردن اطلاعات مالی و بانکی تهدیداتی در محیط نرم‌افزاری سمت مشتری، مشابه آن‌چه در بخش‌های ۱-۳ و ۱-۱-۳ بیان شد، به همراه دارد؛ لذا حملات جعبهٔ سفید علیه تلفن‌های همراه کاربران، می‌تواند موجب خسارات سنگین مالی شود. به همین علت، استفاده از سامانه‌های رمزنگاری مقاوم در برابر حملات جعبهٔ سفید در نرم‌افزارهای سمت مشتری، می‌تواند از اطلاعات حساس مالی کاربران محافظت کند [۱۳].

۳-۱-۳- محافظت کلی سیستم‌ها در برابر بدافزارها

جاسوس‌افزارها، ویروس‌ها، اسب‌های تروجان و دیگر بدافزارها، می‌توانند از طریق شبکه یا برنامه‌های نامطمئن منتشر شده و دستگاه‌های دارای سیستم‌عامل را آلوده کنند. از فعالیت‌های مخرب بدافزارها، مواردی چون تحلیل پردازنده و حافظه‌های تحت اختیار آن در حین اجرای یک برنامه، به‌دست‌آوردن اطلاعات حساسی چون بایت‌های کلید موجود در یک برنامه، سرقت اطلاعات از سیستم قربانی و ارسال آن به محل دیگر را می‌توان برشمرد [۱۵, ۱۶].

اگر نرم‌افزارهای موجود در یک سیستم برای انجام عملیات رمزنگاری از الگوریتم‌های امن در برابر تهدیدات محیط جعبهٔ سفید استفاده کنند، آسیب‌پذیری سیستم در برابر بدافزارهایی که قصد استخراج و سرقت اطلاعات حساس را (از جمله کلیدهای استفاده شده برای رمزنگاری داده‌ها) از قربانی دارند، کاهش خواهد یافت.

به‌عنوان مثال، درصورت استفاده از یک الگوریتم رمزنگاری جعبهٔ سفید که ویژگی (M,Z)-Space Hardness را با مقادیر قابل قبول محقق سازد، سرقت کدهای نرم‌افزارهای رمزنگار/رمزگشا دشوار خواهد بود، چون برای بازسازی سریع الگوریتم رمزنگار/رمزگشای مورد حمله در محلی دیگر، به حجم زیادی از کدهای آن نیاز است و احتمال ارسال موفق حجم زیادی از داده‌ها توسط جاسوس‌افزارها کاهش خواهد یافت [۸]. به عبارتی سرقت بخشی از اطلاعات الگوریتم رمزنگاری از رایانه یا تلفن همراه توسط جاسوس‌افزارها یا ... ، کمکی به استخراج کلید یا بازسازی الگوریتم در زمانی کوتاه نخواهد کرد [۶, ۷].

۳-۱-۴- محافظت از اطلاعات شهروندان در برابر سرویس‌های جاسوسی بیگانه

بسیاری از شهروندان در زندگی روزمره خود از خدمات نرم‌افزاری مبتنی بر شبکه‌های اینترنتی استفاده می‌کنند. برخی از اطلاعات خصوصی شهروندان (از جمله عادات شخصی و اجتماعی، نحوه برخورد با موضوعات اجتماعی و سیاسی، مستندات مشمول حریم خصوصی و ...) برای سرویس‌های جاسوس بیگانه حائز اهمیت هستند. ازآن جا که بسیاری از شهروندان از طریق رایانه شخصی یا تلفن همراه از اینترنت استفاده می‌کنند (که مشمول حملات جعبهٔ سفید است)، درصورتی‌که بیم آن وجود داشته باشد که اطلاعات آنان در معرض حملات بیگانگان قرار گیرد، استفاده از طرح‌ها و روش‌های رمزنگاری جعبهٔ سفید که حجم کد زیادی دارند (دارای ویژگی (M,Z)-Space Hardness با مقادیر M و Z بسیار بزرگ هستند) استخراج اطلاعات و کلیدهای خصوصی تمام کاربران را برای مهاجمان دشوار می‌سازد؛ زیرا در حمله استخراج کلید پیچیدگی محاسباتی برای آنان کمتر از جستجوی کامل نخواهد بود و برای حمله برداشت کد نیز قادر به سرقت حجم بسیار بزرگ اطلاعات الگوریتم رمزنگاری تمام شهروندان نیستند [۸].

۳-۲- استفاده از رمزهای متقارن به جای رمزهای نامتقارن

مزیت رمزهای نامتقارن نسبت به رمزهای متقارن، امکان انجام یکی از فرآیندهای رمزنگاری یا رمزگشایی، فقط برای مالک کلید خصوصی است. از طرفی، سرعت طرح‌هایی که برای

رمزنگاری (رمزگشایی) نامتقارن ارائه‌شده، بسیار پایین تر از سرعت رمزهای قالبی است. به همین دلیل رمزنگاری/ رمزگشایی پیام‌های طولانی توسط رمزهای نامتقارن، نیازمند زمان زیادی است.

در صورت تحقق ویژگی "امنیت قوی" برای یک طرح رمزنگاری جعبهٔ سفید، می‌توان از سرعت رمزهای قالبی و یک‌طرفه‌بودن رمزهای نامتقارن، توأمان بهره برد [۲۵]. همان‌طور که پیشتر اشاره شد، بر اساس ویژگی "امنیت قوی"، درصورت فاش‌شدن کدهای رمزنگاری/رمزگشایی یک طرح جعبهٔ سفید، بازسازی معکوس آن برای حمله‌کننده امکان‌پذیر نیست. به عبارتی مشابه آن چه در رمزهای نامتقارن رخ می‌دهد، یکی از عمل‌های رمزنگاری یا رمزگشایی، تنها برای یک شخص خاص میسر است [۱۲, ۲۵]. برای مثال، فرض کنید عمل رمزنگاری اطلاعات در تلفن همراه کاربران (محیط جعبهٔ سفید با امکان حمله برداشت کد) و عمل رمزگشایی در سرور انجام شود. درصورت فاش‌شدن الگوریتم رمزنگاری، به‌دلیل تحقق ویژگی "امنیت قوی"، حمله‌کننده قادر نخواهد بود الگوریتم رمزگشایی را نیز به‌دست آورد. به‌عبارتی این اطمینان وجود دارد که فقط سرور قادر به رمزگشایی اطلاعات است؛ لذا با پیاده‌سازی جعبهٔ سفید یک رمز قالبی، ویژگی مهم رمزهای نامتقارن نیز حاصل خواهد شد.

دوگان فرض بالا نیز صادق است، یعنی می‌توان اطمینان حاصل کرد فقط یک شخص خاص قادر به رمزنگاری اطلاعات است. در این حالت، استفاده از کدهای احراز اصالت[1] مبتنی بر رمزهای قالبی، به جای امضاهای دیجیتال مبتنی بر رمزهای نامتقارن، میسر خواهد بود [۱۲, ۲۵].

۳-۳- تقویت امنیت تجهیزات رمزنگاری نظامی

اگر در تجهیزات رادیویی و سایر تجهیزاتی که از سامانه‌های رمزنگاری استفاده می‌کنند، از روش‌های جعبهٔ سفید استفاده شود، به‌دلیل ویژگی‌هایی که در بندهای پیشین برشمرده شد، در هنگام ازدست‌رفتن و به‌جاماندن آنان در مناطق مورد تهدید دشمن، اطمینان بیشتری از عدم آسیب‌پذیری کل سامانهٔ مخابراتی وجود دارد. به عبارت دیگر، زمان بازیابی کلید از تجهیزات کمتر از جستجوی کامل نخواهد بود، بازسازی الگوریتم رمزگشایی و شنود کل سامانه امکان‌پذیر نیست و فرصت کافی برای تعویض کلید و اقدامات لازم دیگر وجود خواهد داشت.

۴-۳- تأمین امنیت شبکه‌های هوشمند صنعتی

در سامانه‌های اتوماسیون سنتی، تعدادی حس‌گر در نقاط مختلف یک واحد صنعتی جایگذاری شده و اطلاعات خود را به کنترل‌کننده مرکزی ارسال می‌کنند. کنترل‌کنندهٔ مرکزی با استفاده از این اطلاعات، نسبت به کنترل تجهیزات موجود اقدام خواهد کرد.

در شبکه‌های نوین اتوماسیون، از مفاهیمی چون شبکه حس‌گرها و اینترنت اشیا استفاده می‌شود. تعداد زیادی حس‌گر در یک واحد صنعتی نصب شده و اطلاعات خود را برای یک سرور محلی یا فضای ابری ارسال می‌کنند. پردازش اطلاعات (که می‌تواند مبتنی بر هوش مصنوعی، منطق فازی یا ... باشد) و تصمیم‌گیری‌های لازم در سمت سرور انجام شده و تجهیزات با اتصال به سرور، برنامه عملکرد خود را مطابق نتیجه پردازش‌ها تنظیم می‌کنند. فناوری‌هایی نظیر شبکه هوشمند توزیع انرژی[2]، شهر هوشمند[3]، کارخانه هوشمند[4]، گلخانه هوشمند[5] و ... از این معماری بهره می‌برند [۲۸, ۲۹].

به‌دلیل پراکندگی حس‌گرها و تجهیزات، همچنین حضور عوامل انسانی در محیط‌های صنعتی، امکان دسترسی به حس‌گرها و سامانه‌های الکترونیکی موجود در شبکه وجود دارد. بنابراین در هر دو حالت یادشده، امنیت تبادل اطلاعات میان حس‌گرها، تجهیزات و سرور امری ضروری است؛ زیرا نفوذ به شبکه و دست‌کاری اطلاعات، موجب ایراد خسارت و اختلال در عملکرد سامانه می‌شود [۳۰]. برای جلوگیری از چنین مواردی، پیاده‌سازی روش‌های جعبهٔ سفید بر روی حس‌گرها، تجهیزات و سرور حائز اهمیت است؛ چون درصورت دسترسی نفوذگر به یک گره[6] از شبکه، امکان شکستن الگوریتم رمزنگاری موجود در آن به‌سادگی میسر نیست و امنیت اطلاعات در کل شبکه تهدید نخواهد شد.

۴- روش‌های شاخص طراحی و پیاده‌سازی رمزنگاری جعبهٔ سفید

ایده اصلی طرح‌ها و روش‌های ارائه‌شده برای ساخت رمزهای قالبی امن در محیط جعبهٔ سفید و پیاده‌سازی امن آن‌ها، استفاده از جداول مراجعهٔ حاوی اطلاعات کلید مخفی رمزنگاری است. برای تشکیل این جداول مراجعه، داده‌هایی که وابسته به مقدار کلید هستند، در محلی امن با کلید مورد نظر محاسبه و در جداول مراجعه ذخیره می‌شوند. با این کار،

[1] Massage Authentication Codes (MAC)

[2] Smart grid
[3] Smart city
[4] Smart factory
[5] Smart greenhouse
[6] Node

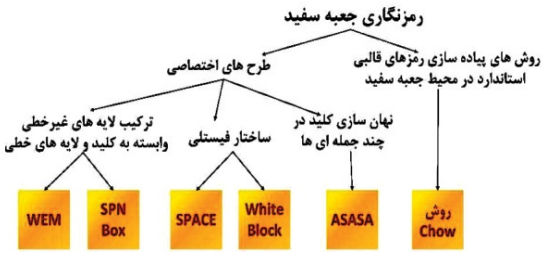
در هنگام اجرای عملیات رمزنگاری/رمزگشایی، بدون این‌که نیازی به مقدار کلید مخفی باشد، داده‌های مراحل میانی رمزهای قالبی در جداول مراجعه در دسترس هستند.

به‌رغم وجود این ایده مشترک در طرح‌ها و روش‌های رمزنگاری جعبهٔ سفید، از سال ۲۰۰۲، زمانی که نخستین روش کاربردی برای رمزنگاری جعبهٔ سفید ارائه شد، طراحی و پیاده‌سازی رمزهای قالبی مقاوم در برابر حملات جعبهٔ سفید، دو رویکرد متفاوت داشته است.

رویکرد نخست طراحان، تلاش برای ارائه روش‌هایی جهت پیاده‌سازی امن رمزهای قالبی استاندارد چون AES و DES در محیط جعبهٔ سفید بوده است. این تلاش‌ها در حدود یک دهه ادامه یافت و روش‌هایی چون روش "Chow" (نخستین روش کاربردی برای رمزنگاری جعبهٔ سفید) [۶, ۷]، روش "پیاده‌سازی آشوبناک" [۳۱]، روش "Xiao" [۳۲] و روش "رمزهای دوتایی" [۳۳] ارائه شدند. حملات تحلیل تفاضلی [۳۴, ۳۵] و حملات تحلیل جبری [۳۶-۳۸] موفقی علیه روش Chow ارائه شد. روش Xiao نیز در برابر حملات تحلیل جبری آسیب‌پذیر بود [۳۹]. علیه روش "رمزهای آشوبناک" نیز حملات موفقی ارائه شده است [۴۰].

با توجه به عدم کارآیی و شکسته‌شدن بیش‌تر طرح‌هایی که تحت رویکرد نخست ارائه شدند، طراحان رویکرد جدیدی را اتخاذ کردند. در رویکرد جدید، طراحان به جای تلاش برای ارائه روش‌هایی که رمزهای قالبی استاندارد را در محیط جعبهٔ سفید به‌صورت امن پیاده می‌کنند، به طراحی ساختارها و الگوریتم‌هایی روی آوردند که به‌صورت ذاتی، در محیط جعبهٔ سفید نیز امنیتی معادل با محیط جعبهٔ سیاه داشته باشند. نخستین روش شاخص در این حوزه در سال ۲۰۱۴ ارائه شد [۱۲] و پس از آن طرح‌های متعدد دیگری معرفی شدند که این مسئله نشان‌دهندهٔ افزایش توجهات به رمزنگاری جعبهٔ سفید در سال‌های اخیر است. برخی از این طرح‌ها از رمزهای قالبی با امنیت اثبات‌شده در مدل جعبهٔ سیاه در طراحی خود سود می‌برند، و امنیت طرح خود را در مدل جعبهٔ سفید، با امنیت رمز قالبی استفاده‌شده، در مدل جعبهٔ سیاه، معادل می‌سازند [۸, ۱۳, ۲۳]. برخی دیگر، امنیت جداول مراجعه حاوی اطلاعات کلید را با استفاده از روش‌های محاسباتی و آماری تأمین می‌کنند؛ به‌گونه‌ای که استخراج کلید از جداول مراجعه، مستلزم جستجوی کامل یا انجام محاسبات دشوار ریاضی باشد [۱۲, ۲۴].

در شکل (۶)، دسته‌بندی روش‌های شاخص طراحی و پیاده‌سازی رمزنگاری جعبهٔ سفید، نمایش داده شده است.



(شکل-۶): نمودار درختی روش‌های شاخص طراحی و پیاده‌سازی رمزنگاری جعبهٔ سفید [۷, ۸, ۱۲, ۱۳, ۲۳, ۲۴]

علاوه‌بر روش‌هایی که در شکل (۶) مشخص شده‌اند، روش‌های دیگری نیز وجود دارند که ایدهٔ اصلی آن‌ها، تقویت و تکمیل این چند روش اصلی است و روش شاخصی به‌شمار نمی‌روند. در ادامه، به بررسی طرح‌های ارائه‌شده پرداخته می‌شود.

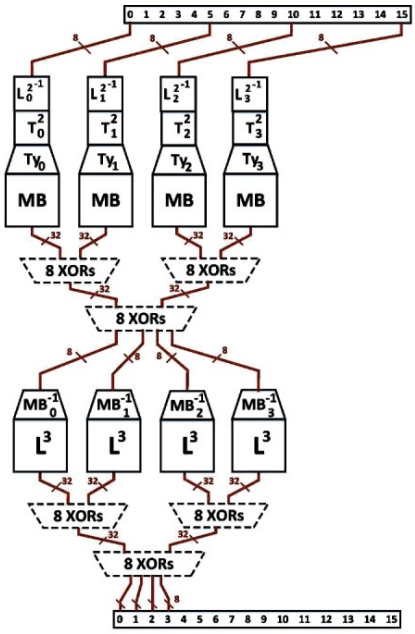
۴-۱- روش Chow

در سال ۲۰۰۲، روشی توسط Chow و همکارانش ارائه شد که هدف آن، پیاده‌سازی رمزهای قالبی استاندارد AES و DES به‌صورت جعبهٔ سفید بود [۶, ۷]. این روش بر مبنای استفاده از ترکیب جداولی با مقادیر از پیش تعیین شده است؛ به‌گونه‌ای که زمان و نحوه اعمال کلید به الگوریتم رمزنگاری برای مهاجم نامشخص باشد. کاربرد این روش در سامانه‌هایی است که کلید برای مدت طولانی تغییر نمی‌کند [۷].

از آن‌جا که رمزنگاری جعبهٔ سفید با این روش متولد شده، لازم است، علاوه‌بر معرفی این روش، ویژگی‌های آن نیز مورد بررسی قرار گیرند تا جهت‌گیری طرح‌های پس از آن روشن شود؛ لذا پس از توضیح روش Chow ، نقاط ضعف، حملات علیه این روش و سایر ویژگی‌های آن بیان می‌شود.

نمودار بلوکی $\frac{1}{4}$ دور از روش Chow در شکل (۷) قابل مشاهده است. در شکل (۷)، جداولی که با نام T_i^r مشخص شده‌اند، از تلفیق عملکرد لایه‌های "اضافه کردن کلید دور"[1] و "جعبه‌های جانشانی"[2] در الگوریتم AES ساخته می‌شوند. رابطه (۱) و رابطه (۲)، نحوه محاسبه مقادیر درون جداول مراجعه T_i^r را بیان می‌کنند. خروجی این جداول، به‌عنوان نشانی ورودی در اختیار جداولی با نام T_{y_i} قرار می‌گیرد که وظیفه‌شان پیاده‌کردن نتیجه عملیات ضرب داده‌ها در "ماتریس‌های درهم ساز" است. عملیات XOR بر روی بایت‌های داده نیز توسط جداول مراجعه انجام می‌شود.

[1] Add Round Key
[2] S-Boxes



(شکل-۷): ۱/۴ دور از روش Chow [۷, ۴۱]

$$T_i^r(x) = S\left(x \oplus \hat{k}_{r-1}[i]\right) \quad \text{for } i = 0 \dots 15 \text{ and } r = 1 \dots 9 \qquad (۱)$$

$$T_i^{10}(x) = S\left(x \oplus \hat{k}_9[i]\right) \oplus k_{10}[i] \quad \text{for } i = 0 \dots 15 \qquad (۲)$$

برای جلوگیری از دسترسی مهاجم به مقادیر میانی، در روند انتقال اطلاعات میان جدول‌ها، از کدهای خطی و غیرخطی استفاده می‌شود. جداولی که با نام L و MB در شکل (۷) مشخص هستند، برای اعمال کدهای خطی بازگشت‌پذیر (کدهایی مقادیر وارون متناظر با آن‌ها وجود دارد) بر روی داده‌های مراحل میانی طراحی شده‌اند. همچنین در ورودی و خروجی جداول XOR کدهایی با اندازه ورودی/خروجی چهار بیت اعمال می‌شود [۷, ۴۱].

مطابق شکل (۷)، هر ربع دور از طرح Chow معادل با یک نگاشت ۳۲بیت به ۳۲بیت است. در ادامه ویژگی‌های این طرح برشمرده می‌شود [۶, ۷, ۴۱].

۴-۱-۱- ویژگی‌های روش Chow

این روش در تلاش است، مراحل میانی یک رمزقالبی استاندارد را با استفاده از جداول مراجعه پیاده کند. این روش پیاده‌سازی در مواردی که کلید برای مدت طولانی تغییر نمی‌کند، قابل استفاده است؛ زیرا با تغییر کلید، مقادیر جداول مراجعه نیز باید دوباره محاسبه شوند [۷].

با توجه به درنظرگرفتن مدل جعبهٔ سفید در فرایض طراحی روش Chow، داده‌های ورودی و خروجی هرکدام از جدول‌ها برای مهاجم آشکار است؛ لذا برای محافظت از کلید اصلی و کلید دور آخر، یک کد خطی ۱۲۸بیتی در ورودی و خروجی الگوریتم AES استاندارد اضافه می‌شود تا خروجی لایه "اضافه‌کردن کلید" در دور آخر (و ورودی این لایه در دور نخست) به‌صورت خام برای مهاجم آشکار نباشد [۷, ۴۱].

پیاده‌سازی این روش به حدود ۵۱۰ کیلوبایت حافظه نیاز دارد. طراحان این روش باور داشتند که پیچیدگی حمله جستجوی کامل در این طرح بسیار بیشتر از پیچیدگی حمله جستجوی کامل علیه خود رمزقالبی است؛ زیرا مهاجم مجبور است، محتوای جداول حاوی تبدیلات خطی و غیرخطی را هم به‌دست آورد، اما این ادعا در حمله‌ای که در سال ۲۰۰۴ به این روش صورت گرفت، رد شد [۳۶].

۴-۱-۲- نقاط ضعف روش Chow

از جمله نقاط ضعف این روش، می‌توان موارد زیر را برشمرد :

۱. زمان انجام عملیات رمزنگاری/رمزگشایی بسیار طولانی‌تر از روش‌های عادی است [۷, ۴۱]؛
۲. این روش با حالت استاندارد یک رمز قالبی متفاوت است؛ زیرا تعدادی عنصر خطی و غیر خطی در مراحل میانی به آن اضافه می‌شود. همچنین کدگذاری‌های ابتدایی و انتهایی، حاصل کل عملیات را از حاصل پیاده‌سازی یک رمزقالبی به صورت استاندارد متفاوت می‌کند [۸]؛
۳. در این روش، نقاط ضعف امنیتی بسیاری وجود دارد؛ لذا پیچیدگی استخراج کلید بسیار کمتر از جستجوی کامل خواهد بود [۳۶].

۴-۲- طرح‌های اختصاصی

بر خلاف روش Chow ، که در تلاش است تا الگوریتم AES استاندارد را به‌صورت امن در مدل جعبهٔ سفید پیاده کند، طرح‌های دیگری ارائه شده‌اند که به جای تمرکز بر روی پیاده‌سازی امن رمزهای قالبی استاندارد در محیط جعبهٔ سفید، یک ساختار یا الگوریتم منحصربه‌فرد را که به‌ذاته در برابر حملات مدل جعبهٔ سفید امن است ارائه می‌دهند. چنین طرح‌هایی را طرح‌های اختصاصی می‌گویند. در ادامه، شاخص‌ترین طرح‌های اختصاصی ارائه‌شده برای رمزنگاری جعبهٔ سفید، معرفی خواهند شد.

۴-۲-۱- طرح اختصاصی ASASA

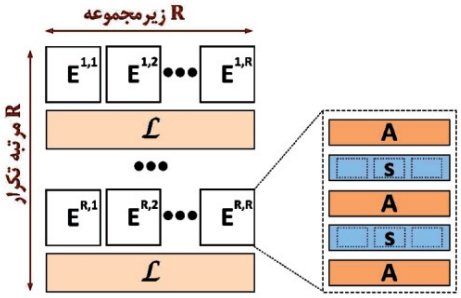
بررسی‌های ابداع کنندگان این روش نشان داد هر دور از روش Chow ، معادل ترکیب دو لایهٔ آفینی (A) و یک لایه غیرخطی

(S) است که به‌صورت ASA نمایش داده می‌شود. آن‌ها تصمیم گرفتند، ساختار جدیدی را ارائه دهند تا ضمن افزایش لایه‌های غیرخطی و آفینی، با تعریف چندجمله‌ای‌های درجه دوم روی لایه‌های غیرخطی، امنیت طرح Chow را بهبود بخشند. آن‌ها ادعا کردند که طرح آن‌ها دارای ویژگی "امنیت قوی" است و از ایده آن می‌توان برای پیاده‌سازی رمزهای نامتقارن و کلید عمومی نیز بهره برد و روش خود را در سال ۲۰۱۴ ارائه دادند [۱۲].

در ساختار پیشنهادشده در طرح ASASA، از سه لایهٔ آفینی (A) و دو لایهٔ غیرخطی (S) به‌صورت متوالی استفاده می‌شود. کلید، در لایه‌های غیرخطی پنهان شده است. پنهان‌سازی کلید، در توان چند جمله‌ای‌های مرتبه دوم صورت می‌گیرد؛ لذا استخراج کلید از جداول مراجعه، مستلزم حل‌کردن مسئلهٔ دشوار لگاریتم گسسته است. رابطه (۳) مثالی از استفاده از چندجمله‌ای‌ها در نهان‌سازی کلید را بیان می‌کند [۱۲].

$$S(X.k) = \underset{over\ F_{2^n}}{(X^{2^k}+X+a)^{-l}+X} \qquad (۳)$$

ساختار این روش در شکل (۸) نمایش داده شده است.

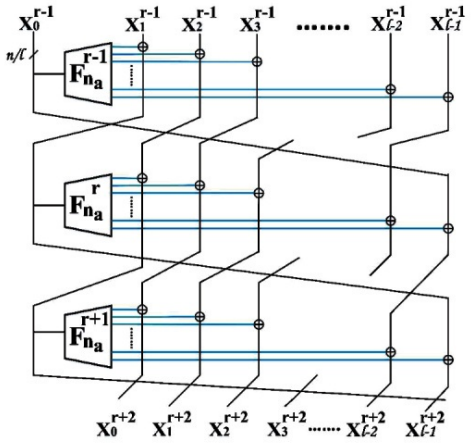


(شکل-۸): ساختار روش ASASA در رمزنگاری جعبهٔ سفید [۱۲]

در پیاده‌سازی ساختار طرح ASASA، برای افزایش سرعت اجرای رمزنگاری، از دستورهای پردازش برداری پردازنده‌های جدید استفاده می‌شود. همچنین اگر کل طرح با استفاده از جداول مراجعه پیاده شود، به حافظه‌ای از محدوده بیست مگابایت تا بیست گیگابایت نیاز است [۱۲]. پس از ارائه این روش، حملاتی علیه آن صورت گرفت [۴۲, ۴۳] و نقاط ضعف آن آشکار شد.

۴-۲-۲- طرح SPACE

این طرح در سال ۲۰۱۵ ارائه شد و برای تأمین امنیت در محیط جعبهٔ سفید، از یک ساختار فیستلی حاوی رمز قالبی AES به‌صورت شکل (۹) استفاده می‌کند [۸].

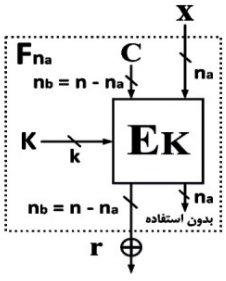


(شکل-۹): ساختار طرح SPACE [۸]

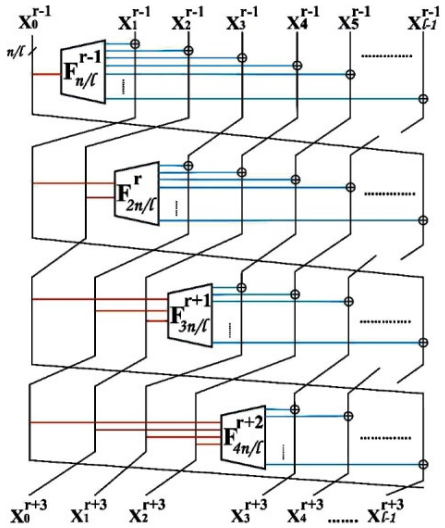
در طرح SPACE، اندازه بردار ورودی به تابع F می‌تواند متغیر باشد. در این صورت، آن را N-SPACE گویند، که N نشان‌دهندهٔ تعداد بردارهای $\mathbf{n_a}$ بیتی واردشده به تابع F است. نحوهٔ تشکیل تابع F در شکل (۱۰) نمایش داده شده است. همچنین، طرح N-SPACE، به ازای $N = 4$ در شکل (۱۱) قابل مشاهده است [۸].

مطابق شکل (۹) هر قالب داده ورودی، به L بردار موازی تقسیم می‌شود. اگر اندازه قالب داده ورودی n بیت باشد، اندازه هر بردار $n_a = n/L$ بیت است. نخستین بردار، وارد تابع F شده و خروجی تابع F (که شامل $L-1$ بردار n_a بیتی است) با بردارهای دیگر جمع دودویی می‌شود. این روند، R دور تکرار می‌شود. نام‌گذاری SPACE به مقدار n_a بستگی دارد، طرح‌های موجود عبارتند از : SPACE-8,16,24,32. تابع **F**، یک رمز قالبی استاندارد است که حاصل آن به‌ازای حالات مختلف ورودی، از قبل محاسبه شده و در جداول مراجعه ذخیره می‌شود [۸].

برخلاف روش Chow که در تلاش است، مراحل میانی یک رمزقالبی را با استفاده از جداول مراجعه مختلف پیاده کند، این روش حاصل کل عملیات در یک رمز قالبی را در یک جدول ذخیره کرده و پیوسته از آن استفاده می‌کند.



(شکل-۱۰): نحوه تشکیل تابع F در طرح SPACE [۸]



(شکل-۱۱): طرح N-SPACE برای N = 4 [۸]

۴-۲-۲-۱- مزایای طرح SPACE

دستاوردهای این طرح نسبت به روش Chow عبارتند از:

۱. سرعت اجرای روند رمزنگاری بسیار افزایش می‌یابد؛
۲. درصورت تغییر کلید، کافی است تنها مقادیر یک جدول تغییر کنند؛
۳. با تغییر پارامتر L، می‌توان اندازهٔ جدول مراجعه F را (با توجه به نوع کاربرد SPACE) تنظیم کرد؛
۴. تنها اطلاعاتی که حمله‌کننده از رمز قالبی تشکیل‌دهندهٔ جدول مراجعه F در اختیار دارد، ورودی و خروجی آن است؛ لذا توانایی‌های حمله‌کننده جعبهٔ سفید، به توانایی‌های حمله‌کننده جعبهٔ سیاه کاهش می‌یابد؛
۵. ارزیابی امنیت طرح بسیار ساده‌تر است؛ زیرا امنیت آن برابر با امنیت رمز قالبی موردنظر در حالت جعبهٔ سیاه است؛ لذا امکان بررسی مقاومت طرح در برابر حملات شناخته‌شده تحلیل نظری وجود دارد.

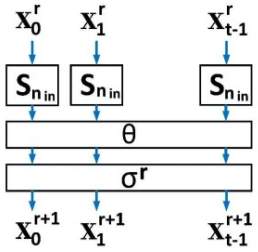
۴-۲-۲-۲- نقاط ضعف

در طرح SPACE، به‌دلیل استفاده از ساختار فیستلی، امکان موازی‌سازی و افزایش سرعت عملیات رمزنگاری وجود ندارد.

۴-۲-۳- طرح SPN-Box

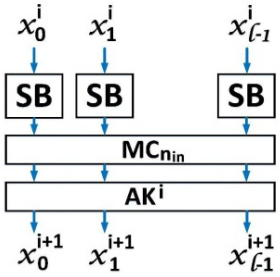
طراحان SPACE، یک سال پس از معرفی آن، طرح جدیدی به نام SPN-BOX را در سال ۲۰۱۶ ارائه کردند [۱۳]. ادعای آن‌ها، ارتقای ویژگی‌های امنیتی طرح جدید نسبت به طرح قبلی است. امنیت طرح SPN-BOX، هم‌چون طرح SPACE، مبتنی بر امنیت رمز قالبی مورد نظر در حالت جعبهٔ سیاه است. تفاوت این طرح با طرح قبلی، استفاده از S-BOX های وابسته به کلید و تلفیق آن با لایه‌های خطی است. ورودی الگوریتم، به بردارهای موازی n_{in} بیتی تقسیم می‌شود. تعداد این بردارها را با t نمایش می‌دهند و بردارها را از 0 تا $t-1$ شماره‌گذاری می‌کنند. نام‌گذاری SPN-BOX به مقدار n_{in} بستگی دارد، طرح‌های موجود عبارتند از : SPN-Box 8,16,24,32 [۱۳].

هر دور از SPN-Box، از ساختار شکل (۱۲) بهره می‌برد. مطابق شکل (۱۲)، پیاده سازی SPN-Box، قابلیت اجرای عملیات به‌صورت موازی و افزایش سرعت رمزنگاری را به تبع آن فراهم می‌سازد.



(شکل-۱۲): ساختار هر دور از SPN-Box [۱۳]

در شکل (۱۳)، $S_{n_{in}}$، لایهٔ غیرخطی وابسته به کلید است که برای تشکیل آن از رمزهای قالبی استاندارد با کلید دلخواه استفاده می‌شود (به عبارتی کلید رمزنگاری در این لایه پنهان شده است).



(شکل-۱۳): نحوهٔ تشکیل لایه غیرخطی Sn_{in} در SPN-Box[۱۳]

همان‌طور که در شکل (۱۳) مشخص شده است، در لایه Sn_{in}، از S-Box و ماتریس درهم‌ساز مربوط به AES استفاده می‌شود (اندازه ماتریس درهم‌ساز متناسب با اندازهٔ بردارهای ورودی تغییر می‌کند). کلید رمزنگاری نیز در این لایه غیرخطی به داده‌ها اضافه می‌شود [۱۳]. حاصل عملیات در این لایهٔ غیرخطی، در جداول مراجعه ذخیره شده و در طول عملیات رمزنگاری/رمزگشایی مورد استفاده قرار می‌گیرد.

در شکل (۱۲) لایه‌های σ و θ لایه‌های خطی هستند. عملکرد لایه‌های خطی در روابط (۴) تا (۶) بیان شده است.

$$\theta : GF(2^{n_{in}})^t \rightarrow GF(2^{n_{in}})^t$$
$$(X_0, \dots, X_{t-1}) \xrightarrow{\theta} (X_0, \dots, X_{t-1})\, M_{n_{in}} \quad (۴)$$

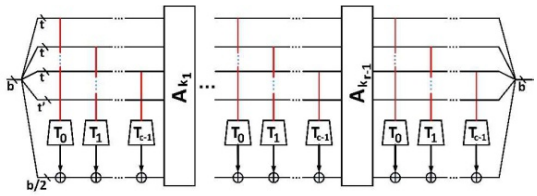
$$\sigma^r : GF(2^{n_{in}})^t \rightarrow GF(2^{n_{in}})^t$$
$$(X_0, \dots, X_{t-1}) \xrightarrow{\sigma} (X_0 \oplus C_0^r, \dots, X_{t-1} \oplus C_{t-1}^r) \quad (۵)$$

$$C_i^r = (r-1)t + i + 1 \quad for\ 0 \le i \le t-1 \quad (۶)$$

در رابطهٔ (۴)، ماتریس‌های $M_{n_{in}}$ در لایهٔ θ، ماتریس‌های MDS بازگشت‌پذیر هستند. در رابطه (۵)، r بیان‌گر شماره دور و لایهٔ **σ**، وابسته به شماره دور است و از یک ثابت **C** وابسته به دور استفاده می‌کند [۱۳].

۴-۲-۴- طرح White Block

این طرح نیز در سال ۲۰۱۶ ارائه شد [۲۳] و برمبنای استفاده از یک ساختار فیستلی و تعدای جدول مراجعه استوار است؛ به‌گونه‌ای که اگر مهاجم قسمت قابل توجهی از جداول مراجعه را در اختیار نداشته باشد، امکان بازسازی و معکوس‌سازی عملیات رمزنگاری/رمزگشایی را نخواهد داشت. ساختار پیشنهادشده در طرح White Block، در شکل (۱۴) قابل مشاهده است.



(شکل-۱۴): ساختار پیشنهادی در روش White Block [۲۳]

در شکل (۱۴)، b اندازهٔ بلوک دادهٔ ورودی است که برای آن، مقدار ۱۲۸ بیت پیشنهاد شده است. همچنین جداولی که برای نهان‌سازی کلید از آن‌ها استفاده شده، با نام T در شکل (۱۴) نمایش داده شده‌اند. برای درهم‌ریختن و تصادفی‌سازی هرچه بیشتر داده‌ها در مراحل میانی، از یک رمز قالبی استاندارد (AES-128) تحت کلید K استفاده شده که با نام A_K در شکل (۱۴) قابل ملاحظه است. در رابطه (۷)، تابع دور طرح White Block توصیف شده است.

$$F : \{0.1\}^{64} \rightarrow \{0.1\}^{64} \quad (۷)$$
$$x_{63} \dots x_0 \rightarrow T_3(x_{63} \dots x_{48}) \oplus T_2(x_{47} \dots x_{32}) \oplus T_1(x_{31} \dots x_{16}) \oplus T_0(x_{15} \dots x_0)$$

در رابطه (۸)، عملکرد یک دور از ساختار فیستلی طرح White Block توصیف شده است. A_K، نماد فراخوانی رمز قالبی استاندارد A تحت کلید K است.

$$R_K : \{0.1\}^{128} \rightarrow \{0.1\}^{128} \quad (۸)$$
$$x_{63} \dots x_0 \rightarrow A_k\left(\left((x_{127} \dots x_{64}) \oplus F(x_{63} \dots x_0)\right) \parallel x_{63} \dots x_0\right)$$

رابطه (۹)، عملکرد کلی طرح White Block را توصیف می‌کند.

$$White\ Block_{K_0 \dots K_r} : \{0.1\}^{128} \rightarrow \{0.1\}^{128} \quad (۹)$$
$$x \rightarrow A_{K_r} \circ R_{K_{r-1}} \circ \dots \circ R_{K_0}(x)$$

رابطه (۱۰) نیز نحوه ساخت جداول T را بیان می‌کند. این جداول، هسته اصلی و مهم‌ترین پارامتر جعبهٔ سفیدبودن طرح White Block هستند.

$$T_i : \{0.1\}^{16} \rightarrow \{0.1\}^{64} \quad (۱۰)$$
$$T(x) \triangleq \perp A_K(c \parallel x) \perp_{64}$$
$\perp_{64}$: **بیت کم‌ارزش** 64
$$K \in \{0.1\}^{128} \qquad c \in \{0.1\}^{128}$$

برای نام‌گذاری طرح White Block، از اندازهٔ ورودی جداول T استفاده می‌شود. برای مثال، اگر داده ورودی جداول T، ۱۶ بیتی باشد (مطابق رابطه ۱۰)، نام‌گذاری به‌صورت White Block 16 خواهد بود. برای انتخاب تعداد دور مناسب در طرح White Block، از اطلاعات جدول (۱) استفاده می‌شود [۲۳].

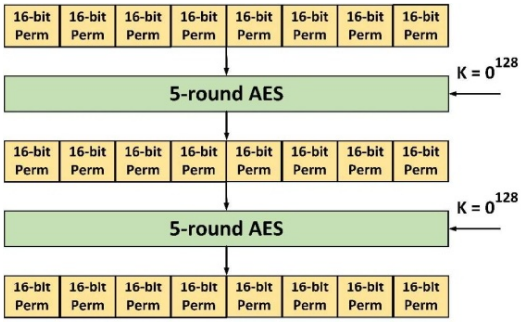
از طرح White Block، با هسته AES-128 برای ساخت دو رمز قالبی استفاده شده است. این دو رمز قالبی، PuppyCipher و Hound نام دارند. رمز قالبی Hound نسبت به PuppyCipher ساختار سبک‌تری دارد، به‌گونه‌ای که:

- در بلوک‌های A_K، به جای AES کامل، از ۵ دور AES استفاده می‌شود.
- کلیدهای هر دور، تحت یک رابطهٔ ساده، از یک کلید اصلی استخراج می‌شوند [۲۳].

۴-۲-۵- طرح WEM

این طرح در سال ۲۰۱۷ ارائه و ساختار آن در شکل (۱۵) نمایش داده شده است [۲۴]. همانند روش‌های پیشین، قالب

داده ورودی به بردارهای موازی هم‌اندازه تقسیم می‌شود تا عملیات به‌صورت موازی و هم‌زمان صورت پذیرد. در این ساختار، از سه لایهٔ S-Box وابسته به کلید و دو لایهٔ رمز قالبی AES-128 (با کلید اصلی صفر) استفاده شده است. استفاده از AES، برای ایجاد درهم ریختگی به‌طورکامل تصادفی داده‌ها است.



(شکل-۱۵): ساختار روش WEM [۲۴]

در شکل (۱۵)، ۲۴ عدد S-Box وجود دارد. این S-Boxها به‌صورت جداول مراجعه پیاده می‌شوند و باید مجزا از هم باشند. برای ساخت لایهٔ S-Box، نخست، با استفاده از کلید مخفی و متن تصادفی، یک دنبالهٔ تصادفی (متن رمزشده) توسط یک رمز قالبی استاندارد تولید و سپس این دنباله تصادفی، با الگوریتم مخلوط‌سازی Fisher-Yates، به دنباله جدید تبدیل و برای ساخت S-Box ها از آن استفاده می‌شود [۲۴].

تعداد لایه‌ها، تعداد عملیات موازی و ... می‌تواند تغییر کند؛ لذا ارزیابی‌های مختلفی از امنیت این طرح وجود دارد. اما در طراحی آن، تلاش شده تا بین سرعت و امنیت، تعادل مناسبی برقرار شود [۲۴].

۵- برخی از حملات شاخص علیه روش Chow

تاکنون حملات متنوعی علیه روش‌های رمزنگاری جعبهٔ سفید ارائه شده است. همان‌طورکه در بخش (۴) به آن اشاره شد، عمده این حملات روش Chow و کارهای تکمیلی پس از آن را هدف قرار داده‌اند. این حملات در دو دسته حملات تحلیل نظری و حملات کانال جانبی جای دارند. در این بخش به تعدادی از حملات شاخص اشاره خواهد شد.

۵-۱- حمله BGE[1]

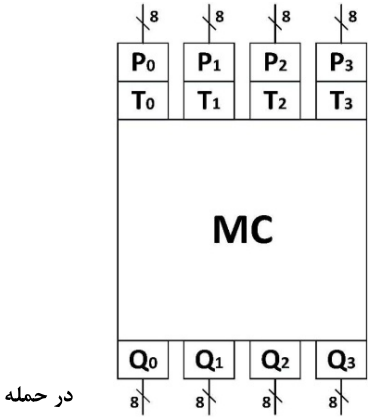
این حمله، یک حمله تحلیل جبری است که در سال ۲۰۰۴ ارائه شد [۳۶] و بر روی مراحل میانی طرح Chow تمرکز کرد، چرا که در دورهای نخست و دهم طرح Chow از کدهای خطی ۱۲۸بیتی استفاده می‌شود [۷, ۴۱]؛ لذا در سناریوی حمله BGE، فرض می‌شود، این کدها برای حمله‌کننده آشکار هستند و تمرکز بر مراحل میانی است.

همان‌طور که در بخش ۱-۴ گفته شد، هر ربع دور از طرح Chow ، معادل با یک نگاشت ۳۲بیت به ۳۲بیت است. مبتکران حمله BGE، یک ربع دور از طرح Chow را به‌صورت شکل (۱۶) بازتعریف کردند. در شکل (۱۶)، جداول P برآیند تمام کدهای خطی ورودی و جداول Q برآیند تمام کدهای خطی خروجی برای یک ربع دور از طرح Chow است. جداول T در روابط (۱ و ۲) تعریف شده‌اند. عملیات MC نیز ضرب در ماتریس‌های درهم‌ساز است.

مبتکران حمله BGE دریافتند که ساختار بالا اطلاعاتی را درخصوص جداول کدهای خروجی (Q) نشت می‌دهد. مطابق رابطه (۱۱)، آن‌ها هر جدول Q را با یک جدول $\tilde{Q}$ تخمین زدند.

$$\tilde{Q}_i = Q_i \, o \, A_i \qquad (11)$$

در رابطهٔ (۱۱)، A_i حاصل XOR یک تبدیل خطی بازگشت‌پذیر و یک عدد ثابت است. آن‌ها نشان دادند، می‌توان A_i را به‌درستی به‌دست آورد و هر یک از جداول Q را به‌طورکامل مشخص کرد. به این ترتیب، خروجی جداول T که حاوی کلید هستند، به‌طورکامل مشخص خواهد شد. ورودی جداول T نیز قابل تشخیص است؛ چون با داشتن جداول Q دور قبل، جداول P نیز مشخص می‌شوند و به این ترتیب ورودی و خروجی جداول T و درنهایت کلید رمزنگاری در دسترس خواهد بود.



(شکل-۱۶): بازتعریف ۱/۴ دور از طرح Chow در حمله BGE [۳۶]

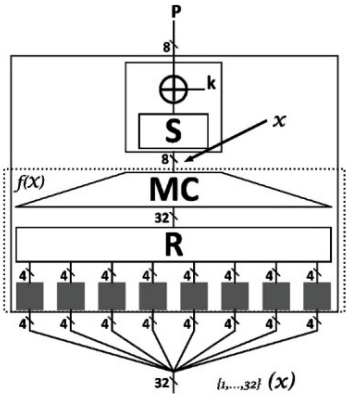
[1] Billet, Gilbert & Ech-Chatbi

از نظر پیچیدگی زمانی، برای به‌دست‌آوردن هر یک از جداول Q، 2^{24} عمل محاسباتی نیاز است. هر ربع دور از طرح Chow، ۴ جدول Q دارد؛ لذا درنهایت $2^{24}\times16=2^{28}$ عمل محاسباتی برای کشف کلید هر دور انجام خواهد شد و برای استخراج کلید سه دور متوالی، $2^{32} < 3\times2^{28}$ عمل محاسباتی صورت خواهد گرفت [۳۶, ۴۱].

۵-۲- حمله کانال جانبی توان بر روی سخت‌افزار رمزنگار/رمزگشا

این حمله در سال ۲۰۱۶ مطرح شده است و در دسته حملات کانال جانبی توان جای می‌گیرد [۲۰]. برای اجرای این حمله، طرح Chow به‌صورت سخت‌افزاری بر روی یک برد SAKURA-X، که از یک تراشه FPGA مدل KYNTEX-7 XC7K160T بهره می‌برد، پیاده شده است. فرکانس کار FPGA بر روی سه مگاهرتز تنظیم و با نرخ پنج میلیون نمونه در ثانیه (پهنای باند بیست مگاهرتز) از مسیر تغذیه اصلی FPGA نمونه‌برداری شده است. متن ورودی سخت‌افزار به‌صورت تصادفی تعیین شده و در کل حدود ده میلیون نمونه از مسیر تغذیه برداشت شده است. پس از نمونه‌برداری، برای تحلیل داده‌ها از روش‌های تحلیل توان چون DPA[1]، CPA[2] و روش‌های تصادم استفاده شده است [۲۰].

فرض می‌شود مهاجم از کدهای خطی بزرگ در ابتدا و انتهای طرح Chow مطلع باشد، لذا مقدار ورودی به دور نخست طرح Chow برای مهاجم معلوم است. در شکل (۱۷)، یک جدول ۸×۳۲ بیت از دور نخست طرح Chow قابل مشاهده است (به شکل (۷) نیز رجوع شود).



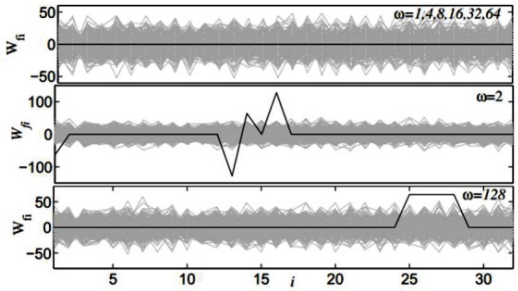
(شکل-۱۷): یک جدول ۸بیت به ۳۲بیت در طرح Chow [۲۰]

هدف این حمله، آشکارکردن کلید با استفاده از مقدار میانی x است. برای معادل‌سازی عملیات ماتریس درهم‌ساز و کدهای خطی پس از آن، از ۳۲ تابع بولین یک بیتی به موازات هم استفاده شده است که رفتار کل جدول مراجعه را در قبال مقادیر مختلف x شبیه‌سازی کند.

با حدس‌زدن هر کلید *k، مقدار میانی x نیز مشخص می شود (زیرا متن ورودی و جعبه‌های جانشانی معلوم هستند). هر تابع بولین با ساختار تعریف‌شده به‌صورت $f_{i\in\{1,\dots,32\}}(x):\{0.1\}^n \rightarrow \{0.1\}$، خروجی یک بیتی (۰ یا ۱) خواهد داشت. برای مشخص‌شدن تمایزگر مناسب در حمله تفاضلی توان، از تابع تبدیل والش[3] استفاده می‌شود. بر اساس نتیجه محاسبات تابع تبدیل والش بر روی توابع f (به‌ازای تمام مقادیر ممکن خروجی توابع)، مشخص خواهد شد که کدام مسیر و کدام بیت از مقدار x برای انجام حمله تفاضلی توان مناسب است. نحوه محاسبه تابع تبدیل والش در رابطه (۱۲) مشخص شده است[۲۰].

$$w_f(\omega) = \sum_{x\in\{0.1\}^n} (-1)^{f(x)\oplus x\cdot\omega} \qquad (۱۲)$$

برای مشخص‌شدن این‌که کدام بیت از x برای مدل‌سازی در حمله تفاضلی توان مناسب است، از متغیر $\boldsymbol{\omega}$ با وزن همینگ "۱" استفاده می‌شود تا حاصل ضرب داخلی x در $\boldsymbol{\omega}$ دقیقا فقط از یک بیت از x تأثیر پذیرد. در شکل (۱۸)، نتایج محاسبهٔ تابع تبدیل والش به ازای کلیدهای مختلف و توابع f مختلف نمایش داده شده است. همان‌طور که در شکل (۱۸) مشخص است، به‌ازای ω=2 بیشترین عدم تعادل ایجاد و تمایزگر مناسب مشخص خواهد شد.



(شکل-۱۸): نتیجه محاسبه تابع تبدیل والش به‌ازای کلیدها و توابع مختلف [۲۰]

یکی از راه‌های جلوگیری از چنین حملاتی، انتخاب درست کدهای خطی و غیرخطی در مراحل میانی طرح Chow

[1] Differential Power Analysis
[2] Correlation Power Analysis
[3] Walsh Transform

است، به‌گونه‌ای که پس از محاسبه تابع تبدیل والش به‌ازای کلیدهای متفاوت، عدم توازن محسوسی در نتایج حاصل نشود [۲۰].

۵-۳- حملهٔ محاسباتی تفاضلی (DCA[1])

این حمله در سال ۲۰۱۶ ارائه شده است [۹]. ایدهٔ اساسی موجود در این روش، اجرای حمله کانال جانبی توان با استفاده از داده‌های حاصل از تحلیل نرم‌افزار الگوریتم رمزنگاری است. نرم‌افزارهایی که برای پیاده‌سازی جعبهٔ سفید رمزهای قالبی طراحی شده‌اند (به‌عنوان مثال کد نرم‌افزاری که طرح Chow را پیاده می‌کند)، هدف این روش حمله هستند. در روند اجرای این حمله، سه گام اصلی وجود دارد:

در گام نخست، مهاجم کد نرم‌افزار مورد نظر خود را در محیط "چارچوب تنظیم نرم‌افزار[2]" اجرا می‌کند. این ابزارهای نرم‌افزاری نظیر PIN و Valgrind، به کاربران خود این امکان را می‌دهند تا تمام جزئیات مربوط به تخصیص حافظه، نحوه ارتباط با حافظه نهان، پیشبینی انشعاب[3] در حافظه نهان و ... را در حین اجرای کد نرم‌افزار هدف، مشاهده و بررسی کنند. مهاجم با اجرای کد نرم‌افزار هدف خود با متن‌های ورودی تصادفی در محیط PIN یا Valgrind، تمامی نشانی‌های فراخوانی حافظه، مقادیر درون حافظه و توزیع داده‌های موجود در حافظه نهان را به‌ازای تمامی متن‌های ورودی، مشاهده و ثبت می‌کند. این اطلاعات بسیار مهم هستند، برای مثال، نشانی‌های فراخوانی حافظه، ورودی جداول مراجعه استفاده شده در کد هستند و حکم مقادیر میانی در یک رمزقالبی را دارند.

در گام دوم، مهاجم نشانی‌های فراخوانی حافظه را که در مرحله قبل ثبت کرده است، به‌صورت یک رشته دودویی مرتب می‌کند، به‌گونه‌ای که مشابه داده‌های ثبت‌شده در یک حملهٔ تحلیل توان باشد (داده‌هایی که با اسیلوسکوپ از مسیر تغذیه سخت‌افزار هدف جمع‌آوری شده‌اند).

در گام سوم، مهاجم کلیدهای مختلف را حدس می‌زند و به ازای متن‌های مختلف رمز شده ورودی، مدل تحلیل توان خود را می‌سازد. سپس با بهره‌گیری از تکنیک‌های محاسباتی مربوط به حمله DPA، از داده‌هایی که به‌صورت نرم‌افزاری جمع شده‌اند، به‌عنوان نمونه‌های توان مصرفی بهره می‌برد و میزان همبستگی آماری آن‌ها را با مدلی که ساخته است، محاسبه می‌کند. همانند یک حمله تحلیل توان، کلیدهای درست، بیش‌ترین همبستگی آماری را ایجاد کرده و مهاجم حمله خود را به پایان می‌رساند [۹, ۴۴].

حُسن این روش از حملات، عدم حضور نوفه و اطلاعات ناخواسته در نمونه‌برداری نرم‌افزاری است [۹]. البته با تصادفی‌سازی تخصیص حافظه و نشانی‌های آن، اجرای این حمله به‌سادگی امکان پذیر نخواهد بود [۴۴].

در پژوهشی که در سال ۲۰۱۸ منتشر شده است [۲۱]، روشی پیشنهاد شد که بر اساس آن، نوعی پوشش[4] بولین بر روی داده‌ها و آدرس‌های دسترسی به حافظه، در مرحله تولید جداول مراجعه صورت می‌گیرد؛ سپس حمله DCA اجرا شده و نتایج اجرای این حمله با استفاده از تابع تبدیل والش بررسی می‌شود. ارزیابی‌ها نشان داد با اعمال پوشش‌های بولین یادشده، تمایزگر مناسبی توسط تابع تبدیل والش یافت نشده و پیچیدگی حمله DCA، در حدود پیچیدگی جستجوی کامل کلید است [۲۱]. همچنین حجم جداول مراجعه در روش پیشنهادشده در حدود ۱/۵ الی ۱۰ برابر حجم جداول در روش Chow خواهد بود [۲۱].

۶- جمع‌بندی، نتیجه‌گیری و پیشنهادها

بسیاری از حملات امنیتی که منجر به شکستن الگوریتم‌های رمزنگاری و سرقت کلیدها، اطلاعات و... می‌شود، ناشی از نقاط ضعف موجود در نحوهٔ پیاده‌سازی الگوریتم‌های رمزنگاری و درنظرنداشتن قابلیت‌های مهاجمان در مدل جعبه خاکستری و جعبهٔ سفید است [۲۱]. هرچند کاربرد رمزنگاری جعبهٔ سفید در حوزه رمزهای نامتقارن به صورت پراکنده مورد توجه برخی از پژوهش‌گران قرار گرفته است [۱۲, ۴۵, ۴۶]، اما بیش‌تر پژوهش‌های منتشرشده درخصوص رمزنگاری جعبهٔ سفید، در زمینه‌های مرتبط با رمزهای قالبی است. با وجود پژوهش‌های انجام‌شده در زمینهٔ رمزنگاری جعبهٔ سفید از سال ۲۰۰۲ تاکنون، این حوزه چالش‌هایی نیز فراروی خود دارد. از جمله:

- تاکنون روش قابل اعتمادی (با اثبات امنیت برپایه محاسبات ریاضی و آماری) برای پیاده‌سازی رمزهای قالبی استاندارد در محیط جعبهٔ سفید ارائه نشده است. عمده طرح‌های ارائه‌شده برای این منظور، شکسته شده‌اند [۱۴, ۲۱, ۲۲]. به‌دلیل اهمیت ماهیت استاندارد و قابل استناد فرم اصلی رمزهای قالبی نظیر AES، ارائهٔ یک روش که الگوریتم‌های رمز قالبی را، بدون تغییر در نتایج اجرای فرم استاندارد آن‌ها، به‌صورت امن بر بستر جعبهٔ سفید پیاده کند، همچنان یک مسئله باز و حل نشده است.

برای دست‌یابی به راه‌کاری کارآمد در خصوص

[1] Differential Computational Analysis
[2] Software Instrumentation Framework
[3] Branch Prediction
[4] Masking

پیاده‌سازی امن رمزهای قالبی استاندارد، مسابقه WhiBox با محوریت پیاده‌سازی امن AES-128 در سال ۲۰۱۷ برگزار شد [۴۷]، اما هیچ یک از روش‌های پیشنهادشده در این مسابقه، از آشکارشدن کلید در برابر حملات موصون نماندند و شکسته شدند [۴۸]. به‌دلیل استقبال جامعه رمزنگاری از این مسابقه، دور دوم آن در سال ۲۰۱۹ برگزار خواهد شد [۴۹] تا شاید روشی کارآمد برای پیاده‌سازی امن رمزهای قالبی استاندارد ارائه شود.

- برخی از طرح‌های ارائه‌شده برای رمزنگاری جعبهٔ سفید (از جمله [۸, ۱۳, ۲۳, ۲۴])، پیچیدگی حمله استخراج کلید را با پیچیدگی این حمله در مدل جعبهٔ سیاه معادل ساخته‌اند، اما برای مقاومت در برابر حمله برداشت کد ایده کارآمدی ارائه نداده‌اند. راه‌کار طرح‌های یادشده، افزایش حجم جداول مراجعه است. این راه‌کار استفاده از برخی از انواع طرح‌های یادشده را در بسیاری از کاربردها (از جمله سیستم‌های توکار[1] که منابع حافظه و پردازش محدودی دارند) ناممکن ساخته است.
- همان‌طور که گفته شد، بیش‌تر توجهات به رمزنگاری جعبهٔ سفید از منظر پیاده‌سازی امن رمزهای قالبی بوده است. کاربرد رمزنگاری جعبهٔ سفید در سایر رمزهای متقارن و رمزهای نامتقارن، از مواردی است که باید مورد توجه پژوهش‌گران قرار گیرد.

این مسئله گفتنی است که درعمل، هیچ یک از روش‌های گفته‌شده در بخش ۴ به‌تنهایی مورد استفاده قرار نخواهد گرفت، بلکه از روش‌هایی چون مبهم‌سازی جریان دستورهای کنترلی[2]، تصادفی‌سازی محل ذخیره‌سازی جداول مراجعه[3] و عملیات ساختگی[4] در کد، به‌صورت مکمل در کنار روش‌های گفته شده در بخش ۴ استفاده خواهد شد تا رمزهای قالبی پیاده‌شده در محیط جعبهٔ سفید، از جنبه‌های مختلف مورد حفاظت قرار گیرند [۴۴].

با توجه به مطالب یادشده، موارد زیر به‌عنوان پیشنهادهایی برای پژوهش‌های آینده در زمینه رمزنگاری جعبهٔ سفید ارائه می‌شود:

- ارائه و توسعهٔ روش‌هایی درخصوص پیاده‌سازی امن رمزهای قالبی استاندارد؛ به‌گونه‌ای که امنیت آن‌ها در محیط جعبهٔ سفید از نظر محاسباتی قابل اثبات باشد.
- برطرف‌ساختن مشکل تأمین امنیت ساختارهای موجود در برابر حملات برداشت کد، به‌گونه‌ای که منابع حافظه بسیار بزرگ مورد نیاز نباشد. رمزهای قالبی سبک می‌توانند به‌عنوان ابزاری مناسب در تجهت نیل به این مقصود مورد استفاده قرار گیرند.
- ارائه و توسعهٔ روش‌هایی برای استفاده از مفهوم رمزنگاری جعبهٔ سفید در حوزهٔ رمزهای نامتقارن، به‌خصوص الگوریتم‌های کلید عمومی که در امضاهای دیجیتال و پروتکل‌های امن کاربرد زیادی دارند.

[1] Embedded Systems
[2] Control flow obfuscation
[3] Table location randomization
[4] Dummy operations

۷- مراجع

[1] F. PUB, "Data Encryption Standard (DES)," *FIPS PUB,* pp. 46-3, 1999.

[2] J. Daemen and V. Rijmen, "AES proposal: Rijndael," 1999.

[3] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-bit blockcipher CLEFIA," in *International Workshop on Fast Software Encryption*, 2007, pp. 181-195: Springer.

[4] B. Gérard, V. Grosso, M. Naya-Plasencia, and F.-X. Standaert, "Block ciphers that are easier to mask: How far can we go?," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2013, pp. 383-399: Springer.

[5] C. Beierle *et al.*, "The SKINNY family of block ciphers and its low-latency variant MANTIS," in *Annual Cryptology Conference*, 2016, pp. 123-153: Springer.

[6] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot" ,A white-box DES implementation for DRM applications," in *ACM Workshop on Digital Rights Management*, 2002, pp. 1-15: Springer.

[7] S. Chow, P. Eisen, H. Johnson, and P. C. Van Oorschot, "White-box cryptography and an AES implementation," in *International Workshop on Selected Areas in Cryptography*, 2002, pp. 250-270: Springer.

[8] A. Bogdanov and T. Isobe, "White-box cryptography revisited: Space-hard ciphers," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, 2015 , pp. 1058-1069: ACM.

[9] J. W. Bos, C. Hubain, W. Michiels, and P. Teuwen, "Differential computation analysis: Hiding your white-box designs is not enough," in *International Conference on Cryptographic Hardware and Embedded Systems*, 2016, pp. 215-236: Springer.

[10] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Annual International Cryptology Conference*, 1999, pp. 388-397: Springer.

[11] J. Bonneau and I. Mironov, "Cache-collision timing attacks against AES," in *International*

Conference on Selected Areas in Cryptography, 2013, pp. 247-264: Springer.

[**23**] P.-A. Fouque, P. Karpman, P. Kirchner, and B. Minaud, "Efficient and Provable White-Box Primitives," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2016, pp. 159-188: Springer.

[**24**] J. Cho *et al.*, "WEM: A New Family of White-Box Block Ciphers Based on the Even-Mansour Construction," in *Cryptographers' Track at the RSA Conference* ,2017 ,pp. 293-308: Springer.

[**25**] M. Joye, "On white-box cryptography," *Security of Information and Networks,* pp. 7-12, 2008.

[**26**] N. Hoque, M. H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya, and J. K. Kalita, "Network attacks: Taxonomy, tools and systems **",***Journal of Network and Computer Applications,* vol. 40, pp. 307-324, 2014.

[**27**] S.-V. Ghiţă, V.-V. Patriciu, and I. Bica, "A new DRM architecture based on mobilel code and white-box encryption," in *Communications (COMM), 2012 9th International Conference on*, 2012, pp. 303-306: IEEE.

[**28**] K. Xu, Y. Qu, and K. Yang, "A tutorial on the internet of things: From a heterogeneous network integration perspective," *IEEE Network,* vol. 30, no. 2, pp. 102-108, 2016.

[**29**] D. Lucke, C. Constantinescu, and E. Westkämper, "Smart factory-a step towards the next generation of manufacturing," in *Manufacturing systems and technologies for the new frontier*: Springer, 2008, pp. 115-118.

[**30**] A. Abbasi and M. Hashemi, "Ghost in the PLC Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack," *Black Hat Europe,* pp. 1-35, 2016.

[**31**] J. Bringer, H. Chabanne, and E. Dottax, "White Box Cryptography: Another Attempt," *IACR Cryptology ePrint Archive,* vol. 2006, no. 2006, p. 468, 2006.

[**32**] Y. Xiao and X. Lai, "A secure implementation of white-box AES," in *Computer Science and its Applications, 2009. CSA'09. 2nd International Conference on*, 2009, pp. 1-6: IEEE.

[**33**] M. Karroumi, "Protecting white-box AES with dual ciphers," in *International Conference on Information Security and Cryptology*, 2010, pp. 278-291: Springer.

[**34**] L. Goubin, J.-M. Masereel, and M. Quisquater, "Cryptanalysis of white box DES implementations," in *International Workshop on Selected Areas in Cryptography*, 2007, pp. 278-295 Springer.

[**35**] B. Wyseur, W. Michiels, P. Gorissen, and B. *Workshop on Cryptographic Hardware and Embedded Systems*, 2006, pp. 201-215: Springer.

[**12**] A. Biryukov, C. Bouillaguet, and D. Khovratovich, "Cryptographic schemes based on the ASASA structure: Black-box, white-box, and public-key," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2014, pp. 63-84: Springer.

[**13**] A. Bogdanov, T. Isobe, and E. Tischhauser, "Towards practical whitebox cryptography: Optimizing efficiency and space hardness," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2016, pp. 126-158: Springer.

[**14**] A. Biryukov and A. Udovenko, "Attacks and Countermeasures for White-box Designs," *IACR Cryptology ePrint Archive,* 2018.

[**15**] C. C. Zou, D. Towsley, and W. Gong, "Email virus propagation modeling and analysis," Department of Electrical and Computer Engineering, Univ. MassachusettsTR-CSE-03-04, 2003.

[**16**] M. Erbschloe, *Trojans, worms, and spyware: a computer security professional's guide to malicious code* .Elsevier, 2004.

[**17**] A. Bogdanov, T. Eisenbarth, C. Paar, and M. Wienecke, "Differential cache-collision timing attacks on AES with applications to embedded CPUs," in *Cryptographers' Track at the RSA Conference*, 2010, pp. 235-251: Springer.

[**18**] A. Moradi, D. Oswald, C. Paar, and P. Swierczynski, "Side-channel attacks on the bitstream encryption mechanism of Altera Stratix II: facilitating black-box analysis using software reverse-engineering," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, 2013, pp. 91-100: ACM.

[**19**] E. Peeters, F.-X. Standaert, N. Donckers, and J.-J. Quisquater, "Improved higher-order side-channel attacks with FPGA experiments," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2005, pp. 309-323: Springer.

[**20**] P. Sasdrich, A. Moradi, and T. Güneysu, "White-Box Cryptography in the Gray Box," in *International Conference on Fast Software Encryption*, 2016, pp. 185-203: Springer.

[**21**] S. Lee, T. Kim, and Y. Kang, "A Masked White-box Cryptographic Implementation for Protecting against Differential Computation Analysis," *IEEE Transactions on Information Forensics and Security,* 2018.

[**22**] C. Delerablée, T. Lepoint, P. Paillier, and M. Rivain, "White-box security notions for symmetric encryption schemes," in *International*

Preneel, "Cryptanalysis of white-box DES implementations with arbitrary external encodings," in *International Workshop on Selected Areas in Cryptography*, 2007, pp. 264-277: Springer.

[**36**] O. Billet, H .Gilbert, and C. Ech-Chatbi, "Cryptanalysis of a white box AES implementation," in *International Workshop on Selected Areas in Cryptography*, 2004, pp. 227-240: Springer.

[**37**] W. Michiels, P. Gorissen, and H. D. Hollmann, "Cryptanalysis of a generic class of white-box implementations," in *International Workshop on Selected Areas in Cryptography*, 2008, pp. 414-428: Springer.

[**38**] T. Lepoint, M. Rivain, Y. De Mulder, P. Roelse, and B. Preneel, "Two attacks on a white-box AES implementation," in *International Conference on Selected Areas in Cryptography*, 2013, pp. 265-285: Springer.

[**39**] Y. De Mulder, P. Roelse, and B. Preneel, "Cryptanalysis of the Xiao–Lai white-box AES implementation," in *International Conference on Selected Areas in Cryptography*, 2012, pp .49-34 :Springer.

[**40**] Y. De Mulder, B. Wyseur, and B. Preneel, "Cryptanalysis of a perturbated white-box AES implementation," in *International Conference on Cryptology in India*, 2010, pp. 292-310: Springer.

[**41**] J. A. Muir, "A Tutorial on White-box AES ",in *Advances in Network Analysis and its Applications*: Springer, 2012, pp. 209-229.

[**42**] B. Minaud, P. Derbez, P.-A. Fouque, and P. Karpman, "Key-recovery attacks on ASASA," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2014, pp. 3-27: Springer.

[**43**] H. Gilbert, J. Plût, and J. Treger, "Key-recovery attack on the ASASA cryptosystem with expanding S-boxes," in *Annual Cryptology Conference*, 2015, pp. 475-490: Springer.

[**44**] S. Banik, A. Bogdanov, T. Isobe ,and M. Jepsen, "Analysis of software countermeasures for whitebox encryption," *IACR Transactions on Symmetric Cryptology,* vol. 2017, no. 1, pp. 307-328,2017.

[**45**] A. Jivanyan, A. Oliynyk, and M. Raievskyi, "Efficient Oblivious Transfer Protocols based on White-Box Cryptography," *IACR Cryptology ePrint Archive,* 2016.

[**46**] A. Biryukov and L. Perrin, "Symmetrically and Asymmetrically Hard Cryptography," in *International Conference on the Theory and Application of Cryptology and Information Security*, 2017 ,pp. 417-445: Springer.

[**47**] C.-M. C. Emmanuel Prouff, Bo-Yin Yang, Thomas Baignères, Matthieu Finiasz, Pascal Paillier, Matthieu Rivain. (2017, 10/16/2018). *CHES 2017 Capture the Flag Challeng, The WhibOx Contest, An ECRYPT White-Box Cryptography Competition*. Available: https://whibox-contest.github.io/

[48] ECRYPT-CSA, CryptoExperts, and T. Eindhoven. (2017, 12/8/2018). *CHES 2017 Challenge*.Available:https://ches.2017.rump.cr.yp.to/a905c99d1845f2cf373aad564ac7b5e4.pdf

[**49**] P. P. Chris Brzuska. (2019, 10/16/2018). *WhibOx — White-box cryptography and Obfuscation* Available:https://eurocrypt.iacr.org/2019/affiliatedevents.html

هادی سلیمانی کارشناسی را در رشته مخابرات از دانشگاه علم و صنعت ایران در سال ۱۳۸۷ اخذ کرد و کارشناسی‌ارشد را با گرایش مخابرات رمز در سال ۱۳۸۹ در دانشگاه امام حسین (ع) به پایان رساند؛ سپس مدرک دکترای خود را از دانشکده علوم کامپیوتر دانشگاه آلتوی فنلاند در سال ۱۳۹۴ اخذ کرد. وی همچنین طی یک دوره کوتاه‌مدت پسادکترا در گروه رمزنگاری دانشگاه DTU دانمارک درخصوص تحلیل و طراحی رمزهای قالبی نوین مشغول به پژوهش شد. نامبرده هم‌اکنون، ضمن همکاری با پژوهشکده‌ها و مراکز پژوهشی مختلف در حوزه رمزنگاری و امنیت اطلاعات، به‌عنوان استادیار گروه امنیت شبکه و رمزنگاری پژوهشکده فضای مجازی دانشگاه شهید بهشتی مشغول به کار است. زمینه‌های پژوهشی مورد علاقهٔ ایشان تحلیل و طراحی اولیه‌های رمزنگاری متقارن و همچنین پیاده‌سازی امن است.

محمدرضا صادقی مدرک کارشناسی خود را در رشتهٔ مهندسی برق گرایش الکترونیک، در سال ۱۳۹۵ از دانشگاه اصفهان اخذ کرد و از سال ۱۳۹۵، دانشجوی مقطع کارشناسی ارشد دانشگاه شهید بهشتی در رشته مهندسی برق گرایش مخابرات امن و رمزنگاری است. زمینه‌های پژوهشی مورد علاقه وی عبارتند از: پردازش سیگنال‌های دیجیتال، رمزنگاری جعبه سفید، پیاده‌سازی امن الگوریتم‌های رمزنگاری، امنیت اینترنت اشیا و شبکه‌های حسگری بی‌سیم.