حمله خطای تفاضلی به MARX-2، یک الگوریتم رمز مبتنی بر ساختار ARX

محمدرضا عادلی[۱]، رحیم اصغری[۲]، نصور باقری[۳]، سید داوود منصوری[۴]

[۱] کارشناسی ارشد مهندسی برق مخابرات امن و رمزنگاری، دانشگاه صنعتی مالک اشتر، adeli.reza@chmail.ir

[۲] استادیار دانشکده امنیت اطلاعات، دانشگاه صنعتی مالک اشتر meisam.mathhome@gmail.com

[۳] دانشیار دانشکده مهندسی برق ، دانشگاه تربیت دبیر شهید رجایی nbagheri@sru.ac.ir

[۴] پژوهشیار دانشکده امنیت اطلاعات دانشگاه صنعتی مالک اشتر davodmansuri@gmail.com

چکیده

الگوریتمهای رمزنگاری جزء غیر قابل انکار در فراهم کردن امنیت ارتباطات هستند. از طرفی، برای اطمینان از امنیت یک الگوریتم، باید میزان پایداری آن در برابر حملات مختلف ارزیابی شود. در کنار ارزیابی امنیت سامانه های رمز نگاری در برابر حملات آماری نظیر حمله خطی، حمله تفاضلی و امثال آن، امنیت سامانه های رمزنگاری در برابر حملات کانال جانبی همواره مورد توجه بوده است. در این میان حملات مبتنی بر القای خطا اخیرا توجه زیادی را به خود جلب کرده است. از جمله حملات القای خطا، می توان به القای خطای تفاضلی اشاره کرد که در این تحقیق مد نظر قرار می گیرد. اگرچه حمله القای خطای تفاضلی به رمزهای قالبی زیادی اعمال شده است اما در این میان امنیت دسته ای از رمزهای قالبی موسوم به رمزهای ARX، در برابر حملات القای خطا، کمتر مورد توجه قرار گرفته است. در این راستا، در این مقاله امنیت یک رمز قالبی ARX موسوم به MARX-2 در برابر حمله القای خطای تفاضلی مد نظر قرار می گیرد. نتیجه این حمله، بازیابی کلید n بیتی رمز با n بیت اعمال خطا خواهد بود. حمله ارائه شده در این مقاله اولین ارزیابی امنیت رمز MARX-2 در برابر حمله القای خطای تفاضلی ارائه می شود، تا آنجایی که ما اطلاع داریم.

کلیدواژه‌ها: *حمله خطای تفاضلی، ساختار ARX، الگوریتم سبک‌وزن، MARX-2، تحلیل رمز.*

۱– مقدمه

در یک شبکه رایانه‌ای، امنیت مقوله مهمی است که مورد توجه قرار می‌گیرد. برای فراهم کردن امنیت لازم، با وجود محدودیت‌هایی چون سرعت انتقال داده‌ها، لزوم استفاده از الگوریتم‌های رمزنگاری سبک‌وزن برای رسیدن به محرمانگی بدون کاهش قابل توجه سرعت تبادل اطلاعات را دوچندان می‌کند.

الگوریتم‌های رمز مبتنی بر ساختار ARX نوع خاصی از الگوریتم‌های رمز سبک‌وزن قالبی هستند که می توانند در محیتهای با منابع محدود مد نظر قرار گیرند. در طراحی این نوع الگوریتم‌ها از عملگرهای ساده‌ای استفاده می‌شود که موجب سبک شدن آنها می‌گردد و در عین حال با توجه به اینکه در این ساختارها به‌طور هم‌زمان از عناصر خطی و غیرخطی استفاده می‌شود از امنیت قابل قبولی برخوردار هستند. در ساختار رمزهای ARX تنها از عملگرهای جمع پیمانه‌ای، چرخش بیتی و جمع انحصاری استفاده می‌شود. معماری ARX در مقایسه با طرح‌های مبتنی بر جعبه جانشینی که فقط از عناصر غیرخطی هستند، تنها به جمع پیمانه‌ای به عنوان منبع غیرخطی تکیه می‌کند. نمایندگان معروف کلاس ARX عبارتند از: Salsa20 [۱]، ChaCha20 [۲]، Skein [۳] ، BLAKE [۴] ، TEA، XTEA [۵]، Chaskey [۶]، SPECK [۷]، LEA [۸] و SPARX [۹] هستند.

برای ارزیابی امنیتی این ساختارها ناگزیر از حمله و سنجش مقاومت و میزان آسیب‌پذیری آنها هستیم. از جمله تحلیل‌های رمز پرکاربرد، حملات کانال‌جانبی هستند که از اطلاعات نشتی قابل دسترس، نظیر رفتار ورودی‌ها و خروجی‌های سامانه‌های رمزنگاری در شرایط پیرامونی خاص، میزان توان مصرفی در زمان‌های مختلف یا زمان اجرای عملیات، برای تحلیل رمز استفاده می‌کنند. حمله خطای تفاضلی نوع خاصی از این نوع

تاریخ دریافت: ۱۳۹۸/۰۵/۰۲ تاریخ پذیرش: ۱۳۹۹/۰۱/۲۷

حملات است که با تزریق خطا به مقادیر میانی یک الگوریتم رمز و بررسی تأثیر آن در خروجی، اقدام به یافتن کلید مخفی رمز می‌کند.

با مروری بر پژوهش‌های انجام شده درخصوص حمله خطای تفاضلی به برخی از انواع الگوریتم رمز مبتنی بر ساختار ARX می‌توان به حمله یادشده به رمز SPECK اشاره کرد. این رمز در سال ۲۰۱۳ توسط آژانس امنیت ملی آمریکا[۱] طراحی و ارائه شد. اولین حمله خطای تفاضلی با نام بیت وارونه[۲] در سال ۲۰۱۴ توسط آقای توپسامودره[۳] و همکاران بوده است [۱۰]. این گروه با وارون کردن یک بیت از یک مقدار میانی از تابع دور آخر، یک حمله خطا تفاضلی به ورودی دور آخر رمز SPECK اعمال کردند. بر مبنای نشت اطلاعات اساسی عملگر جمع پیمانه‌ای، این حمله، n بیت از کلید دور آخر را با حداقل $\frac{n}{3}$ جفت متن رمزشده سالم و معیوب بازیابی می‌کند. آقای هو[۴] و همکاران در سال ۲۰۱۵ حمله‌ای دیگر را به رمز SPECK اعمال کردند[۱۱]. در این حمله با الهام از مراجع [۱۲] و [۱۳]، جمع در پیمانه 2^n به درون یک سیستم جبری از معادلات حداکثر درجه ۲ بدون هیچ تبدیل متغیری منتقل شده است. سپس سیستم جبری محاسباتی ارائه شده در [۱۴] را جهت محاسبه یک پایه گروبنر از سیستم معادلاتی به‌کارگیری نموده و کلید دور آخر را بازیابی می‌کنند.

تعداد خطاهای تزریق شده در این حمله حدود ۵ تا ۸خطا بر روی خانواده SPECK است، که خیلی کمتر از $\frac{n}{3}$ ارائه شده در مرجع [۱۵] می‌باشد. علاوه بر این، مقاله مذکور یک مدل تحلیل مبتنی بر مقدار خطای انتخابی را نیز بررسی می‌کند، با این فرض که مهاجم بتواند خطاهای مشخص شده را تزریق کند. در این مدل حمله، نشان داده می‌شود که ۴ خطا برای بازیابی کلید دور آخر، مستقل از طول قالب SPECK ، کافی است. حمله خطای بایت تصادفی به رمز SPECK حمله دیگری است که فنگ[۵] و همکاران در سال ۲۰۱۷ انجام داده‌اند[۱۶]. این حمله فقط به اعمال خطا بر یک دور میانی برای بازیابی کل کلید اصلی نیاز دارد. در این حمله، خواص انتشار خطای SPECK به طور کامل مورد بهره‌برداری قرار گرفته است و نه‌تنها مقدار و موقعیت خطا حدس زده می‌شود بلکه کاندیداهای نادرست کلید نیز حذف می‌شوند. در این حمله، در مقایسه با حملات قبلی مشخصه‌های بیشتری از عملگر جمع پیمانه‌ای و روابط میان جفت متن رمز شده مختلف بهره‌برداری می‌شود که بازده بازیابی کلید را تقویت می‌کند.

برای حدس موقعیت و مقدار خطا، خواص انتشار خطا را تحت شرایطی که پراکنش کامل خطا تا دور آخر رمز نرسیده است بررسی کرده، سپس با توجه به الگوی گسترش خطا، مقدار میانی معیوب و کلید کاندیدای متناظر را استخراج می‌کند. در بازیابی اطلاعات و در مقایسه با نتایج قبلی، این حمله مشخصه‌های بیشتری از عملگر جمع پیمانه‌ای را مورد بهره‌برداری قرار می‌دهد که در مجموع منجر به بازیابی کلید با تعداد خطای تزریق شده کمتر می‌شود.

با بررسی پژوهش‌های گذشته در زمینه حملات، تاکنون حمله خطایی به الگوریتم رمز MARX-2 انجام نشده است، اگرچه تحلیل خطای تفاضلی به رمزهای ARX در قیاس با رمزهای مبتنی بر جعبه جانشینی، در کل کمتر مد نظر قرار گرفته است. ضعفی که در ساختار رمز SPECK وجود داشت و باعث نشت اطلاعات آن می شد ودر حمله خطای تفاضلی بیت وارونه به آن استفاده شد، به نظر در رمز MARX-2 کمتر شده است. اما، حمله‌ای که در این مقاله بر روی رمز MARX-2 انجام می‌شود علی‌رغم ارتقای استحکام این رمز، موجب گردید با هر جفت بیت تزریق خطا به ورودی‌های دور آخر، دو بیت از کلید رمز بازیابی شود و در ادامه نیز با تکرار این اعمال خطاها، تمام بیت‌های کلید بازیابی گردند.

لازم به ذکر است در حمله خطای تفاضلی بایت تصادفی(چند بیت[۶]) که به رمز SPECK اعمال شده است، از خاصیت انتشار خطای تزریقی به یک دور میانی الگوریتم و سپس بررسی آثار آن در دورهای بعدی استفاده کرده و اقدام به بازیابی بیت‌های بیشتری از کلید دور آخر می‌کنند. این حمله نسبت به دیگر حمله‌هایی که تاکنون بر روی رمز SPECK انجام شده بود دارای قدرت بیشتری است، به‌طوری که تعداد کلیدهای بیشتری را بازیابی می‌کند، البته باید متذکر شد که به لحاظ پیاده‌سازی، تزریق خطا به یک بیت به مراتب مشکل‌تر از اعمال آن به یک بایت است؛ لذا با توجه به این‌که رمز MARX-2 در واقع نسخه ایمن‌تر SPECK است، احتمال این‌که حمله مبتنی بر بایت برای رمز MARX-2 نیز کارایی بیشتری داشته باشد متصور خواهد بود. با توجه به پیچیدگی کمتر حمله بیت وارونه و همچنین ایده جدیدی که در این مقاله مکمل این حمله شده است و موجب موفقیت اجرای آن گردید، حمله یادشده به رمز MARX-2 برای این پژوهش انتخاب شده است.

---

۱ NSA
۲ Bit Flip
۳ Tupsamudre
۴ Hou
۵ Feng
۶ Multi Bit

در ادامه، در بخش ۲، ابتدا مروری خواهیم داشت بر بعضی رمزهای ARX. سپس، در بخش ۳، حمله خطای تقاضلی بیت وارونه به رمز MARX-2 اعمال می شود. در نهایت نیز در بخش ۴، به جمع بندی مقاله می پردازیم.

۲- معرفی ساختار چند رمز ARX

برای ایجاد ویژگی غیرخطی بودن، علاوه بر استفاده از جعبه جانشینی می‌توان از روش‌های ساده‌تری نیز استفاده کرد. یک نمونه از این روش‌ها استفاده از توابع جمع پیمانه‌ای، چرخش و جمع انحصاری برای ساخت تابع دور است که اصطلاحاً به این رمزها ARX می‌گویند. در این میان تنها جمع پیمانه‌ای عملکردی غیرخطی داشته و دو تابع دیگر به صورت خطی کار می‌کنند. هرچند جعبه‌های جانشینی معمولاً غیرخطی‌تر از جمع پیمانه‌ای هستند اما هزینه پیاده‌سازی بالاتری دارند، مخصوصاً در پیاده سازی نرم افزاری. لذا معمولاً فضای ورودی‌ها و خروجی جمع پیمانه‌ای، بزرگ‌تر از فضای معمول انتخابی برای ورودی و خروجی جعبه‌های جانشینی انتخاب می‌شود. همین امر گاهی موجب می‌شود تا نتوان تمامی مشخصه‌های خطی و تفاضلی جمع پیمانه‌ای را به‌سادگی ارزیابی کرد.

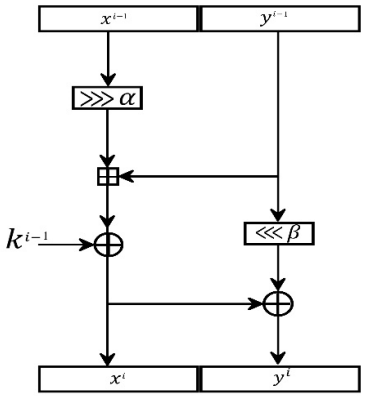
ازجمله ویژگی‌های مناسب رمزهای ARX می‌توان به سرعت بالای عملکرد بر روی رایانه شخصی، پیاده‌سازی فشرده، سادگی الگوریتم و عدم امکان اعمال حملات زمانی به دلیل ثابت بودن همیشگی زمان محاسبه آن‌ها اشاره کرد. از طرف دیگر، آسیب‌پذیری احتمالی در برابر حملات خطی و تفاضلی و نیز برخی حملات کانال‌جانبی ازجمله ضعف‌های شناخته‌شده این دسته رمزها به شمار می‌روند. این ویژگی‌ها که موجب ساده‌تر شدن و کارایی سریع‌تر و درنتیجه سبک شدن یک الگوریتم رمز می‌گردند، دلیل به‌کارگیری این نوع ساختار در طراحی و ساخت الگوریتم‌های سبک‌وزن شده است.

در ادامه به معرفی ساختار چند الگوریتم رمز مبتنی بر ARX پرداخته و در نهایت یک رمز نسبتاً جدید را که از نظر استحکام در برابر حملات به طور قابل ملاحظه‌ای مقاوم است را معرفی می‌نماییم.

۲-۱- ساختار رمز SPECK

SPECK یک خانواده از رمز قالبی سبک‌وزن مبتنی بر ARX است که توسط آژانس امنیت ملی آمریکا در سال ۲۰۱۳ ارائه شد و شامل ۱۳ نوع رمز با طول قالب و کلید متفاوت است[۱۷]. در این رمز، طول قالب‌ها به‌صورت 2n بیت(که در آن n اندازه هر کلمه است) و اندازه هر کلید mn بیت می باشد. از این‌رو رمزهای این خانواده با نماد SPECK2n/mn نشان داده می‌شوند.

عملیات‌های استفاده‌شده در این رمز با توجه به ARX بودن آن، شامل جمع انحصاری بیتی (⊕)، جمع پیمانه‌ای (⊞) و چرخش بیتی هستند. شکل ۱ نشان دهنده یک دور از عملیات‌های انجام‌شده در این رمز است.



شکل ۱. یک دور از رمز SPECK [۱۷].

در پایان هر دور این رمز، خروجی تابع دور به‌صورت زیر است:

$$x_i(mod2^n) = ((x_{i-1} \ggg \alpha) + y_{i-1}) \oplus k_{i-1}$$

$$y_i = (y_{i-1} \lll \beta) \oplus x_i \qquad (۱)$$

در حالی که m=2 است، طرح کلید[۷] رمز SPECK نیز کلید اصلی را به دو قسمت k_1 و k_2 تقسیم کرده و مطابق شکل ۲ و رابطه ۲ برای دورهای مختلف کلی-دور تولید می‌کند:

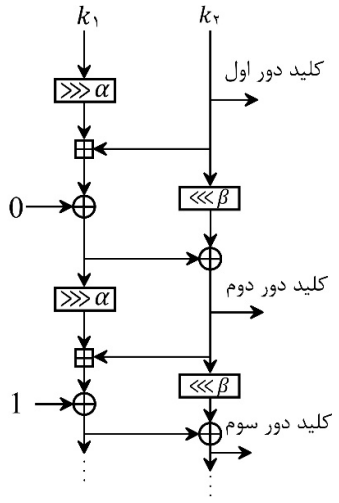
$$K_{(\text{دور اول})} = k_2$$

$$K_{(\text{دور دوم})} = (((k_1 \ggg \alpha) + k_2) mod\ 2^n) \oplus (k_2 \lll \beta)$$

[۷] Key Schedule

$$K_{(\text{دور سوم})} = (((((((k_1 \ggg \alpha) + k_2) mod\ 2^n) \ggg \alpha) + K_{(\text{دور دوم})}) mod\ 2^n) \oplus 1) \oplus (K_{(\text{دور دوم})} \lll \beta)$$

... (۲)



شکل۲. طرح کلید رمز SPECK [۱۷].

اگر $\beta \lll$ چرخش بیتی چپ به اندازه β بیت، $\alpha \ggg$ چرخش بیتی راست به اندازه α بیت، i شماره دور و m تعداد کلمات باشد، به‌طورکلی رابطه ۳ برای طرح کلید رمز SPECK برقرار است:

$$l_{i+m-1} = (k_i + (l_i \ggg \alpha)) \oplus i$$
$$k_{i+1} = (k_i \lll \beta) \oplus l_{i+m-1} \quad (۳)$$

در این مقاله تحلیلی مورد نظرمان که در مرجع [۱۰] روی رمز SPECK انجام شده است را بررسی کرده و سپس بر روی رمز MARX-2 اعمال می‌کنیم.

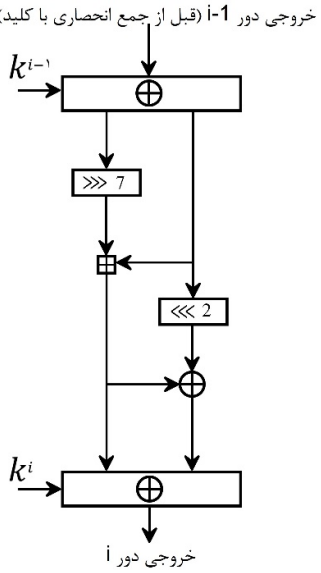
در رمز SPECK نشت اطلاعات از طریق جمع پیمانه‌ای تابع دور به‌دست می‌آید و اختفای کلید دور آخر کاملاً بر اختفای ورودی سمت چپ آخرین دور یعنی x^{i-1} تکیه دارد.

۲-۲- ساختار رمز SPECKEY

رمز SPECKEY در سال ۲۰۱۶ توسط بیریوکوف[۸] و همکاران طراحی شد[۱۸]. همانگونه که در شکل ۳ نیز مشاهده می شود، تابع دور آن مشابه تابع دور SPECK است یعنی جمع پیمانه‌ای، چرخش‌ها و جمع انحصاری مانند قبل هستند با این تفاوت که افزودن کلید که در تابع SPECK به نیمه سمت چپ مقادیر اضافه می‌شد، در اینجا در پایان تابع و به کل مقدار 2n بیت و قبل از خروجی افزوده می‌شود. این نوع طراحی سبب می‌گردد نشت اطلاعات که در رمز SPECK از مسیر y^i رخ می‌داد و از آن طریق بیت‌های y^{i-1} کشف می‌شد محدود شود.

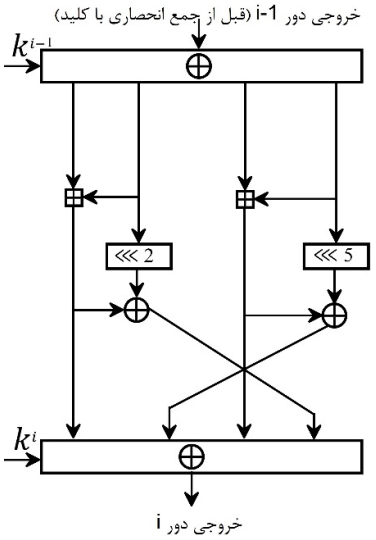
۲-۳- ساختار رمزهای MARX و MARX-2

هر دو ساختار MARX و MARX-2 در سال ۲۰۱۶ توسط بیریوکوف و همکاران طراحی شد[۱۸].مطابق با شکل ۴، تابع دور MARX در واقع از دو تابع دور SPECKEY ساخته شده با این تفاوت که در هر تابع تنها از یک چرخش بیتی استفاده شده است.



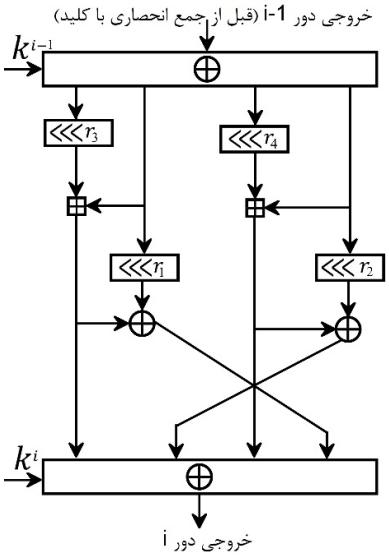
شکل ۳. ساختار یک دور از رمز SPECKEY [۱۸].

[۱] Biryukov



شکل ۴. یک دور از رمز MARX [۱۸].

برای رسیدن به پراکنش کامل، با افزوده شدن دو چرخش بیتی r_3 و r_4 به تابع دور، MARX به صورت شکل ۵ تغییر یافت و MARX-2 نام گرفت:



شکل ۵. یک دور از رمز MARX-2 [۱۸].

همان‌طور که مشاهده می‌شود MARX-2 نیز مانند SPECKEY ضعف رمز SPECK را نداشته، ضمن اینکه به ادعای طراحان، پس از ۱۰ دور به پراکنش کامل دست می‌یابد.

۳- حمله خطای تفاضلی بیت وارونه به رمز MARX-2

در دهه‌های اخیر یک دسته حمله جدید موسوم به حمله‌های کانال‌جانبی برای ادوات رمزنگاری ابداع شده است. این گروه از حمله‌ها از اطلاعات نشتی قابل دسترس مثل توان مصرفی زمان اجرا و رفتار ورودی خروجی در سامانه رمزنگاری برای به‌دست آوردن اطلاعاتی درباره سامانه استفاده می‌کنند. حملات کانال-جانبی از روش‌های آماری برای به‌دست آوردن اطلاعات نشتی استفاده می‌کنند و اغلب بسیار مؤثرتر از حملات آماری مانند حملات خطی و تفاضلی هستند. حمله‌های کانال جانبی بر اساس روند تحلیل به دسته هایی نظیر تحلیل زمانی، تحلیل ساده توان، تحلیل تفاضلی توان، حمله القای خطا، حمله القای تفاضلی خطا و تحلیل تشعشعات الکترومغناطیسی تقسیم می‌شود.

حمله خطا نوعی حمله پیاده‌سازی سخت‌افزاری به‌حساب می‌آید. فرض بر این است که حین اجرای عملیات رمزنگاری دستگاه‌های رمزنگار به‌طور مؤثر و بدون خطا کار می‌کنند. با وجود این نشان داده شده است که خرابی‌های سخت‌افزاری و خطاهایی که در مدت اجرای عملیات رمزنگاری و یا رمزگشایی سخت‌افزار سامانه رخ می‌دهد، تأثیر جدی بر روی امنیت سامانه می‌گذارد. عملکرد خطادار سامانه و خروجی خطادار ممکن است باعث انجام موفقیت‌آمیز حمله خطا گردد. این روش رمز شکنی، حمله خطا (یا القای خطا[۹]) نامیده می‌شود. چنانچه در اثر این حمله در سامانه رمزنگاری خرابی یا خطایی به‌وجود آید، می‌توان با کمک این خطا سامانه را شکسته، اطلاعات مفیدی در مورد آن به‌دست آورد. امکان‌پذیر بودن این حمله، به قابلیت‌های رمز شکن و خرابی‌هایی که می‌تواند در سامانه رمز ایجاد نماید بستگی دارد[۱۹].

در سپتامبر ۱۹۹۶ بونه[۱۰] و همکاران نوع جدید و ابتکاری از حمله کانال جانبی که توجه بسیاری را به خود جلب نمود معرفی نمودند. این حمله از خطاهای محاسباتی به منظور کشف کلیدهای رمزنگاری بهره می‌برد. پس از مدت کوتاهی، بیهام و

[۱] Fault Induction Attack

[۲] Boneh

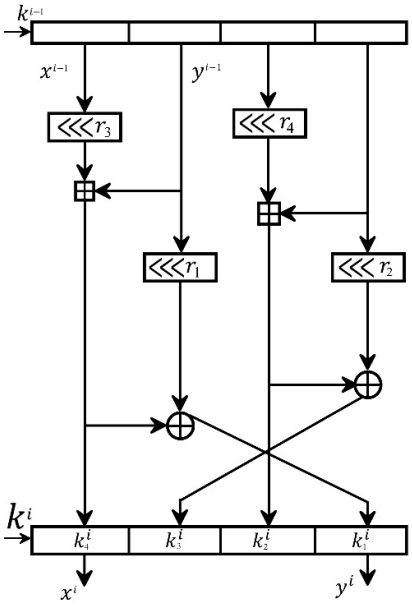
شامیر نشان دادند که چنین حملاتی بر روی رمزهای قالبی نیز به خوبی قابل اجرا است و آن را تحلیل خطای تفاضلی[۱۱] نامیدند. در این مقاله امنیت رمز MARX-2 در برابر حملات القای خطای تفاضلی مد نظر قرار می‌گیرد.

در شکل ۵ یک دور از تابع دور رمز MARX-2 را مشاهده می نمایید. برای انجام تحلیل رمز، بر روی یک نیمه آن(نیمه چپ)، حمله مد نظر را انجام می‌دهیم.

از جمله مهمترین تفاوت های رمز MARX-2 با SPECK این است که در رمز MARX-2 کلید در انتهای تابع دور و به تمام بیت‌های خروجی افزوده می‌شود. MARX-2 از دو شاخه موازی از تابع دور SPECK با کلمه ۸ بیتی تشکیل شده است. مانند SPECK32، به واسطه عملگرهای جمع پیمانه‌ای با تعداد دورهای مشابه (مثلاً ۱۰ تایی) به پراکنش کامل می‌رسد. بهترین مشخصه تفاضلی[۱۲] و همبستگی خطی[۱۳] برای چرخش‌ها به‌صورت (۷و۱و۳و۲)=(r_1, r_2, r_3, r_4) پیشنهاد شده است[۱۸].

ثابت‌های چرخش MARX-2 به‌وسیله جستجوی فراگیر روی همه مقادیر ۴ چرخش (در کل ۴۰۹۵ مقدار، شامل انتخاب تمام صفر) انتخاب شده‌اند و برای هر مجموعه تعداد دورهایی که برای رسیدن به پراکنش کامل نیاز است را ثبت کرده‌اند. نتایج نشان می‌دهد که هیچ مجموعه‌ای از ثابت‌های چرخش وجود ندارد که به ازای آن‌ها، در کمتر از ۱۰ دور بتوان به پراکنش کامل رسید[۱۸].

در این بخش پس از بیان معادلات مربوط به نیمه چپ دور آخر رمز MARX-2، حمله خطای تفاضلی بیت وارونه، به چند روش به ورودی‌های دور آخر این رمز اعمال و سعی می‌شود از طریق تحلیل آن، کلید دور آخر بازیابی گردد.



شکل۶. دور آخر از تابع دور رمز MARX-2 [۱۸].

۳-۱- معادلات تابع دور رمز MARX-2

ما در اینجا معادلات مربوط به دور آخر از تابع دور رمز MARX-2 را با توجه به شکل ۶ محاسبه می‌کنیم:

$$x^i = k_4^i \oplus \left(x^{i-1} + y^{i-1}\right) \longrightarrow \quad (۴)$$
$$x_j^i = k_{4_j}^i \oplus (x_j^{i-1} \oplus y_j^{i-1} \oplus c_j^{i-1})$$

که با جابجایی طرفین خواهیم داشت:

$$k_4^i = x^i \oplus \left(x^{i-1} + y^{i-1}\right) \longrightarrow \quad (۵)$$
$$k_{4_j}^i = x_j^i \oplus (x_j^{i-1} \oplus y_j^{i-1} \oplus c_j^{i-1})$$

و همچنین برای خروجی y^i داریم:

$$y^i = y^{i-1} \oplus x^i \oplus k_4^i \oplus k_1^i \longrightarrow \quad (۶)$$
$$y_j^i = y_j^{i-1} \oplus x_j^i \oplus k_{4_j}^i \oplus k_{1_j}^i$$

و خواهیم داشت:

$$k_1^i = y^i \oplus y^{i-1} \oplus x^i \oplus k_4^i \longrightarrow \quad (۷)$$
$$k_{1_j}^i = y_j^i \oplus y_j^{i-1} \oplus x_j^i \oplus k_{4_j}^i$$

[۱] Diferential Fault Analysis (DFA)
[۲] Differential Probability
[۳] Linear Correlation

۳-۲- حالت های مختلف تزریق خطا به رمز Marx-2

در این بخش چند حالت مختلف تزریق خطا به رمز MARX-2 را بررسی و نتایج حاصله را با توجه به شاخص تعداد بیت‌های قابل بازیابی کلید مقایسه می‌کنیم.

۳-۲-۱- اعمال خطا به y^{i-1}:

در صورت تزریق خطا به یک بیت از y^{i-1}، اثر آن بر بیت‌های نقلی بعدی(c^{i-1} ها) و همچنین خروجی‌های تابع دور آخر این رمز(x^i و y^i) باقی خواهد ماند، به‌طوری که باعث تغییر در بیت‌های آن‌ها می‌شود (البته لزوماً تمام بیت‌ها عوض نخواهند شد و این امر بستگی به متغیرها و مقادیر بیت‌های آنان دارد). از تحلیل این تغییرات و با بهره‌گیری از قواعد تحلیل تفاضلی، ما اقدام به حدس‌زدن بیت‌ها کرده و در نهایت تلاش می‌کنیم متغیرهای ورودی از جمله کلید دور آخر را بازیابی نماییم.

در این روش از حمله خطای بیت وارونه، ما صرفاً خطا را به y^{i-1} اعمال می‌کنیم؛ لکن در روش‌های بعدی که در ادامه ارائه خواهد شد خطا به هر دو مقدار y^{i-1} و x^{i-1} تزریق می‌شود.

الف) اجرای حمله: برای حالات مختلف c_j^{i-1} و y_j^{i-1} و با در نظر گرفتن تعداد تغییرات به وجود آمده در تفاضل خروجی x^i و تعداد یک های آن، مقادیر مختلفی برای x_j^{i-1} مطابق جدول ۱ حاصل می‌شود که منجر به رابطه‌ای میان x_j^{i-1} و c_j^{i-1} می‌گردد. ما از این رابطه استفاده کرده، رابطه‌ای جدید برای کلید دور آخر مورد نظر(مجموع دو کلید k_1^i و k_4^i) می‌یابیم که در آن عناصر متغیر کمتر شده و در نتیجه دایره جستجوی تخمین مقدار کلید کاهش پیدا می‌کند.

جدول ۱. محاسبه مقدار x_j^{i-1} در حمله تک خطا

c_j^{i-1}	y_j^{i-1}	x_j^{i-1}	c_{j+1}^{i-1}	$\neq \mathbf{1}(x_j^i \oplus x_j^{*i})$
0	0	0	0	=1
0	1	0	0	=1
1	0	1	1	=1
1	1	1	1	=1
0	0	1	0	>1
0	1	1	1	>1
1	0	0	0	>1
1	1	0	1	>1

اگر وارون کردن بیت در y_j^{i-1} رخ دهد دو حالت متصور است:

۱) تعداد بیت‌های یک در تفاضل x^i و معیوب آن در خروجی متن رمزشده $(x_j^i \oplus x_j^{*i})$ یک مورد باشد، یعنی x^i پس از آن دیگر تغییر نمی‌کند لذا خواهیم داشت:

$$\Delta\, c_{j+1}c_{j+1} \oplus c_{j+1}^* = 0 \quad \text{یا} \quad c_{j+1} = c_{j+1}^*$$

یعنی برای حالت‌های مختلف y_j و c_j مقدار x_j این‌گونه حاصل خواهد شد:

$$c_{j+1}(x_j + y_j + c_j) = c_{j+1}(x_j^* + y_j^* + c_j^*)$$

$$c_{j+1}(x_j + 0 + 0) = c_{j+1}(x_j + 1 + 0)$$

در سمت چپ معادله، بیت نقلی به ازای هر مقدار از x_j برابر صفر خواهد بود. لذا مقدار x_j در معادله سمت راست باید برابر صفر باشد $(x = 0)$.

به همین ترتیب سایر مقادیر برای x به‌دست می‌آید:

$$c_{j+1}(x_j + 1 + 0) = c_{j+1}(x_j + 0 + 0)$$
$$\longrightarrow \quad x_j = 0$$

$$c_{j+1}(x_j + 0 + 1) = c_{j+1}(x_j + 1 + 1)$$
$$\longrightarrow \quad x_j = 1$$

$$c_{j+1}(x_j + 1 + 1) = c_{j+1}(x_j + 0 + 1)$$
$$\longrightarrow \quad x_j = 1$$

۲) تعداد بیت‌های تفاضل x^i و معیوب آن $(x_j^i \oplus x_j^{*i})$ که برابر یک می‌شوند بیش از یک مورد باشد، یعنی x^i بیش از یک تغییر می‌کند:

$$c_{j+1} \neq c_{j+1}^* \text{ یا } \Delta\, c_{j+1} = c_{j+1} \oplus c_{j+1}^* = 1 \longrightarrow$$
$$c_{j+1} = \neg c_{j+1}^*$$

$$c_{j+1}(x_j + y_j + c_j) = \neg c_{j+1}(x_j^* + y_j^* + c_j^*)$$

$$c_{j+1}(x_j + 0 + 0) = \neg c_{j+1}(x_j + 1 + 0)$$
$$\longrightarrow \quad x_j = 1$$

$$c_{j+1}(x_j + 1 + 0) = \neg c_{j+1}(x_j + 0 + 0)$$
$$\longrightarrow \quad x_j = 1$$

$$c_{j+1}(x_j + 0 + 1) = \neg c_{j+1}(x_j + 1 + 1) \longrightarrow \quad x_j = 0$$

$$c_{j+1}(x_j + 1 + 1) = \neg c_{j+1}(x_j + 0 + 1) \longrightarrow \quad x_j = 0$$

در نهایت با توجه به مقادیر حاصله در جدول ۱ رابطه زیر را خواهیم داشت:

$$x_j^{i-1} = \begin{cases} c_j^{i-1} & \neq 1(x_j^i \oplus x_j^{*i}) = 1 \\ \neg c_j^{i-1} & \neq 1(x_j^{\boldsymbol{i}} \oplus x_j^{*\boldsymbol{i}}) > 1 \end{cases} \quad (۸)$$

و همچنین خواهیم داشت:

$$c_{j+1}^{i-1} = \begin{cases} c_j^{i-1} & \neq 1(x_j^i \oplus x_j^{*i}) = 1 \\ y_j^{i-1} & \neq 1(x_j^{\boldsymbol{i}} \oplus x_j^{*\boldsymbol{i}}) > 1 \end{cases} \quad (۹)$$

با توجه به اینکه در رابطه (۴) برای کلید $k_{4_j}^i$ ، مقدار x_j^i را داشته و مقدار x_j^{i-1} نیز از رابطه (۵) به‌دست آمد، خواهیم داشت:

$$k_{4_j}^i = \quad (۱۰)$$
$$\begin{cases} x_j^i \oplus y_j^{i-1} & \neq 1(x_j^i \oplus x_j^{*i}) = 1 \\ x_j^i \oplus y_j^{i-1} \oplus 1 & \neq 1(x_j^i \oplus x_j^{*i}) > 1 \end{cases}$$

ب) پیچیدگی حمله با یک خطا: همان‌گونه که گفته شد ما مقدار x_j^i را داشته و صرفاً مقدار y_j^{i-1} نامعلوم است و این بدان معناست که کلید k_4^i (با توجه به رابطه ۵) بر اساس 2^n حالت به‌دست می‌آید (چون مقادیر x^{i-1} و y^{i-1} ، n بیتی هستند). به همین ترتیب با توجه به رابطه (۷)، k_1^i نیز بر اساس 2^n حالت یافت می‌شود. در واقع با انجام این حمله، پیچیدگی حمله برای مجموع دو کلید k_1^i و k_4^i از 2^{2n} به 2^n کاهش می‌یابد.

۳-۲-۲- اعمال هم‌زمان خطا به y^{i-1} و x^{i-1}

الف) اجرای هم‌زمان خطا:

مشابه حالات قبل، برای حالات مختلف c_j^{i-1} و y_j^{i-1} و با در نظر گرفتن تعداد تغییرات به‌وجود آمده در تفاضل خروجی x^i و تعداد یک‌های آن، مقادیر مختلفی برای x_j^{i-1} مطابق جدول ۲ حاصل می‌شود که منجر به رابطه‌ای برای x_j^{i-1} مبتنی بر y_j^{i-1} می‌گردد.

جدول ۲. محاسبه مقدار x_j^{i-1} در حمله با دو خطای همزمان

c_j^{i-1}	y_j^{i-1}	x_j^{i-1}	$\neq 1(x_j^i \oplus x_j^{*i})$
0	0	1	=1
0	1	0	=1
1	0	1	=1
1	1	0	=1
0	0	0	>1
0	1	1	>1
1	0	0	>1
1	1	1	>1

نحوه به‌دست آمدن رابطه برای x_j^{i-1} به شرح ذیل است:

- اگر وارون‌کردن بیت در y_j^{i-1} رخ دهد دو حالت متصور است:

۱- تعداد بیت‌های تفاضل x^i و معیوب آن در خروجی متن رمز-شده $(x_j^i \oplus x_j^{*i})$ که برابر یک می‌شوند، یک مورد باشد، یعنی x^i پس از آن دیگر تغییر نمی‌کند. لذا خواهیم داشت:

$$c_{j+1} = c_{j+1}^* \quad \text{یا} \quad \Delta\, c_{j+1} = c_{j+1} \oplus c_{j+1}^* = 0$$

یعنی برای حالت‌های مختلف y_j و c_j مقدار x_j این‌گونه حاصل خواهد شد:

$$c_{j+1}(x_j + y_j + c_j) = c_{j+1}(x_j^* + y_j^* + c_j^*)$$

$$c_{j+1}(x_j + 0 + 0) = c_{j+1}(x_j^* + 1 + 0)$$

در سمت چپ معادله ، بیت نقلی به ازای هر مقدار از x_j برابر صفر خواهد بود. لذا مقدار x_j معیوب (x_j^*) در معادله سمت راست باید برابر صفر باشد. درنتیجه x_j برابر یک خواهد بود $(x_j = 1)$.

به همین ترتیب سایر مقادیر برای x به‌دست می‌آید:

$$c_{j+1}(x_j + 1 + 0) = c_{j+1}(x_j^* + 0 + 0) \longrightarrow \quad x_j = 0$$

$$c_{j+1}(x_j + 0 + 1) = c_{j+1}(x_j^* + 1 + 1) \longrightarrow \quad x_j = 1$$

$$c_{j+1}(x_j+1+1)=c_{j+1}(x_j^*+0+1)$$
$$\longrightarrow\quad {x_j}^*=1\quad\longrightarrow\quad x_j=0$$

۲- تعداد بیت‌های تفاضل x^i و معیوب آن $(x_j^i\oplus x_j^{*i})$ که برابر یک می‌شوند بیش از یک مورد باشد، یعنی x^i بیش از یک تغییر می‌کند:

$$c_{j+1}\neq c_{j+1}^*\ \text{یا}\ \Delta\, c_{j+1}=c_{j+1}\oplus c_{j+1}^*=1\quad\longrightarrow$$
$$c_{j+1}=\neg c_{j+1}^*$$

$$c_{j+1}(x_j+y_j+c_j)=\neg c_{j+1}(x_j^*+y_j^*+c_j^*)$$

$$c_{j+1}(x_j+0+0)=\neg c_{j+1}(x_j+1+0)$$
$$\longrightarrow\quad {x_j}^*=1\quad\longrightarrow\quad x_j=0$$

$$c_{j+1}(x_j+1+0)=\neg c_{j+1}(x_j+0+0)$$
$$\longrightarrow\quad x_j=1$$

$$c_{j+1}(x_j+0+1)=\neg c_{j+1}(x_j+1+1)$$
$$\longrightarrow\quad x_j=0$$

$$c_{j+1}(x_j+1+1)=\neg c_{j+1}(x_j+0+1)$$
$$\longrightarrow\quad {x_j}^*=0\quad\longrightarrow\quad x_j=1$$

در نهایت با توجه به مقادیر حاصله در جدول ۲ رابطه زیر را خواهیم داشت:

$$x_j^{i-1}=\begin{cases}\neg y_j^{i-1} & \neq 1(x_j^i\oplus x_j^{*i})=1\\ y_j^{i-1} & \neq 1(x_j^i\oplus x_j^{*i})>1\end{cases}\qquad(۱۱)$$

که از این طریق مقدار کلید به‌دست می‌آید:

$$k_{4_j}^i=x_j^i\oplus x_j^{i-1}\oplus y_j^{i-1}\oplus c_j^{i-1}$$

$$k_{4_j}^i=\begin{cases}x_j^i\oplus c_j^{i-1}\oplus 1 & \neq 1(x_j^i\oplus x_j^{*i})=1\\ x_j^i\oplus c_j^{i-1} & \neq 1(x_j^i\oplus x_j^{*i})>1\end{cases}$$

حالا سعی می‌کنیم برای بیت‌های مختلف مقدار بیت‌های کلید را حدس بزنیم:

$$j=0\ \longrightarrow\ k_{4_j}^i=\begin{cases}x_0^i\oplus c_0^{i-1}\oplus 1\\ x_0^i\oplus c_0^{i-1}\end{cases}$$
$$=\begin{cases}x_0^i\oplus 1 & \neq 1(x_j^i\oplus x_j^{*i})=1\\ x_0^i & \neq 1(x_j^i\oplus x_j^{*i})>1\end{cases}$$

و با عنایت به این‌که x_0^i را داریم لذا اولین بیت کلید بازیابی می‌شود.

همچنین برای بیت نقلی c_1 داریم:

$$c_{j+1}=c(x_j^{i-1}+y_j^{i-1}+c_j^{i-1})\longrightarrow c_1$$
$$=c(x_0^{i-1}+y_0^{i-1}+c_0^{i-1})$$

و با توجه به x_0^{i-1} که معادل عبارت (۴-۹) است خواهیم داشت:

$$x_0^{i-1}=\begin{cases}\neg y_0^{i-1} & \neq 1(x_j^i\oplus x_j^{*i})=1\\ y_0^{i-1} & \neq 1(x_j^i\oplus x_j^{*i})>1\end{cases}\qquad(۱۲)$$

و همچنین داریم:

$$c_1^{i-1}=\qquad(۱۳)$$
$$\begin{cases}c(\neg y_0^{i-1}+y_0^{i-1}+c_0^{i-1}) & \neq 1(x_j^i\oplus x_j^{*i})=1\\ c(y_0^{i-1}+y_0^{i-1}+c_0^{i-1}) & \neq 1(x_j^i\oplus x_j^{*i})>1\end{cases}$$

$$=\begin{cases}c(1+0+0)=c(0+1+0)=0\\ c(0+0+0)\neq c(1+1+0)\end{cases}$$

یعنی برای حالتی که تعداد یک‌های تفاضل $(x_j^i\oplus x_j^{*i})$ برابر یک باشد مقدار بیت نقلی c_1 برابر صفر است، اما زمانی که تعداد یک‌های مذکور بیش از یک مورد باشد c_1 با احتمال $\frac{1}{2}$ برابر یک و با احتمال $\frac{1}{2}$ برابر صفر خواهد بود.

برای بیت بعدی $j=1$ داریم:

$$k_{4_1}^i=\begin{cases}x_1^i\oplus c_1^{i-1}\oplus 1\\ x_1^i\oplus c_1^{i-1}\end{cases}=\qquad(۱۴)$$
$$\begin{cases}x_1^i\oplus 1 & \neq 1(x_j^i\oplus x_j^{*i})=1\\ x_1^i\oplus c_1^{i-1} & \neq 1(x_j^i\oplus x_j^{*i})>1\end{cases}$$

یعنی برای حالتی که تعداد یک‌های تفاضل $(x_j^i\oplus x_j^{*i})$ برابر یک باشد مقدار $k_{4_1}^i$ بازیابی می‌شود، اما زمانی که تعداد یک‌های مذکور بیش از یک مورد باشد مقدار $k_{4_1}^i$ به مقدار c_1 (همان طور که در بالا گفته شد) بستگی دارد.

پیچیدگی حمله: در حمله خطای توام (با توجه به اینکه اولین بیت کلید بازیابی می‌شود)، پیچیدگی حمله نسبت به حمله تک خطا کاهش یافته و به 2^{n-1} می‌رسد. البته با توجه به اینکه در اعمال هم‌زمان خطا به x^{i-1} و y^{i-1} ، چون مجموع دو مقدار خطادار شده در خروجی جمع پیمانه‌ای، اثر یکدیگر را خنثی می‌کنند؛ لذا نتیجه مدنظر ما که بازیابی بیت‌های کلید است حاصل نمی‌شود.

ب) اعمال غیرهم‌زمان و گام به گام خطا:

گام اول: اعمال خطا به y^{i-1} و یافتن x^{i-1}

با توجه به محاسبات قسمت‌های قبل، مطابق شکل ۷ با اعمال خطا به بیت اول در y^{i-1} (در $j=0$)، y_0^{i-1} وارون شده و موجب می‌گردد x_0^i تغییر کند یعنی $\left(x_0^i \oplus x_0^{*i}\right)=1$.

اگر $\left(x_1^i \oplus x_1^{*i}\right)=1$ بود، این بدان معناست که $x_0=1$ بوده و باعث شده c_1 وارون شود ($c_1 \neq c_1^*$) و اثرش را روی x_1^i نشان داده و آن را به x_1^{*i} که متفاوت از حالت پیش از اعمال خطا است تغییر دهد. اما اگر $\left(x_1^i \oplus x_1^{*i}\right)=0$ بود یعنی x_1^i تغییر نکند، بدان معناست که c_1 تغییر نکرده ($c_1 = c_1^*$) و در این حالت نمی توان x_0^i را حدس زد.

$$\begin{array}{rr} \ldots\ c_2\ c_1\ c_0 & \ldots\ c_2\ c_1\ c_0 \\ + \quad \ldots\ x_2\ x_1\ x_0 & + \quad \ldots\ x_2\ x_1\ x_0 \\ \hline \ldots\ y_2\ y_1\ y_0 & \ldots\ y_2\ y_1\ y_0 \\ \hline \ldots\ x_2^{*i}\ x_1^{*i}\ x_0^{*i} & \ldots\ x_2^i\ x_1^i\ x_0^i \end{array}$$

شکل۷. تحلیل تفاضلی اثر خطای غیرهم‌زمان بر خروجی جمع پیمانه‌ای دور آخر رمز MARX-2

پیچیدگی حمله: با اعمال خطا به y_0^{i-1} ، با احتمال $\frac{1}{2}$ بیت نقلی c_1 وارون شده لذا احتمال به‌دست آمدن x_0^{i-1} برابر $\frac{1}{2}$ است. برای این‌که l-امین بیت از c^{i-1} (c_l) وارون شود باید $(l-1)$ بیت نقلی قبلی وارون شود که احتمال وارون شدن هر کدام از آنها برابر است. لذا برای x^{i-1} تعداد بیت‌هایی که با اعمال یک بیت خطا در y^{i-1} بازیابی می‌شوند برابر است با :

$$\sum_{z=1}^{l} z * \Pr[z] \approx 2$$

یعنی با اعمال یک بیت خطا در y^{i-1} ، حدود دو بیت از x^{i-1} بازیابی می‌شود.

گام دوم: اعمال خطا به x^{i-1} و یافتن y^{i-1}:

مشابه گام اول خطا را این‌بار به x^{i-1} اعمال کرده و نتیجه‌ای مشابه می‌گیریم و با اعمال یک بیت خطا در x^{i-1} نیز حدود دو بیت از y^{i-1} بازیابی می‌شود.

در نتیجه با اعمال یک جفت خطا بر روی x^{i-1} و y^{i-1} ، عملاً حدود دو بیت از کلید $k_{4_j}^i$ بازیابی می‌شود.

پیچیدگی حمله: با توجه به اینکه x^{i-1} و y^{i-1} هر کدام n بیتی هستند، با به‌دست آمدن همه بیت‌های آن‌ها که ماحصل اعمال خطا (به‌ترتیب) بر روی y_j^{i-1} و x_j^{i-1} هستند می‌توان حدوداً تمام بیت‌های کلید k_4^i را با $\frac{n}{2}$ خطا بازیابی کرد.

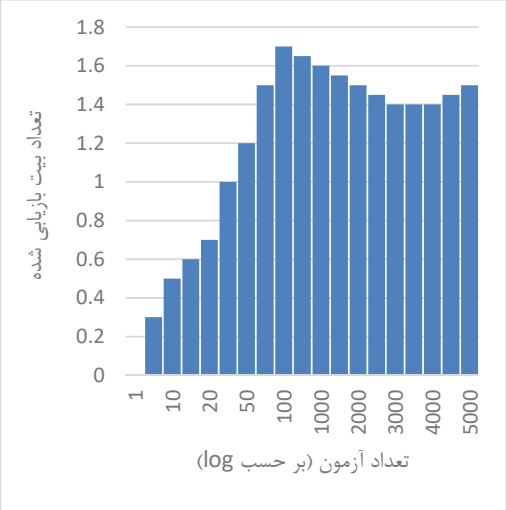
به همین ترتیب با توجه به رابطه (۷)، k_1^i نیز با داشتن k_4^i یافت می‌شود. یعنی با اعمال $\frac{n}{2}$ بیت خطای غیرهم‌زمان به x^{i-1} و $\frac{n}{4}$ خطا به y^{i-1} ($\frac{n}{4}$ خطا به x^{i-1} و y^{i-1} تقریباً تمام بیت‌های کلید k_1^i و k_4^i به‌دست می‌آیند. به همین ترتیب، برای نیمه سمت راست تابع دور نیز با اعمال $\frac{n}{2}$ بیت خطای غیرهم‌زمان به دو ورودی دور ماقبل نهایی، تمام بیت‌های کلید k_2^i و k_3^i نیز حاصل خواهند شد.

یعنی در مجموع با اعمال n بیت خطا در دور پایانی از تابع دور MARX-2 ، تمام بیت‌های کلید K^i بازیابی می‌شود.

۳-۳ شبیه‌سازی

این حمله برای دور آخر تابع دور رمز MARX-2 توسط زبان برنامه نویسی C شبیه‌سازی گردید. در این شبیه‌سازی که در ابتدا برای ۱۰۰ (و در ادامه ۱۰ هزار) ورودی و همچنین کلید

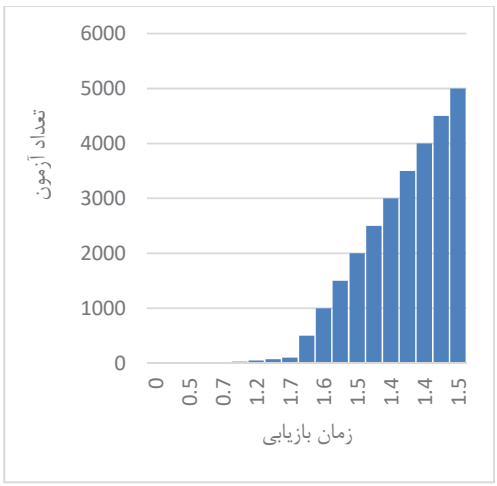
تصادفی اجرا شد، به‌طور میانگین با اعمال ۲ خطا در ورودی، حدود ۲ بیت از کلید بازیابی گردید که این نتیجه مؤید نتیجه مباحث تئوری اثبات‌شده بود.



شکل۷. تعداد بیت های بازیابی شده بر مبنای تعداد آزمون

در شکل ۷ مشاهده می شود که تعداد بیت های بازیابی شده در بهترین حالت برابر ۱,۷ بیت است که تقریباً با میانگین به دست آمده نظری ۲ بیت مطابقت دارد.

در شکل ۸ نیز تعداد آزمون اجرا شده بر حسب زمان بازیابی قابل مشاهده است.



شکل۸. تعداد بیت های بازیابی شده بر مبنای زمان

۴–نتیجه گیری

با توجه به کاربردها و رغبت بیشتر فعالان حوزه امنیت فناوری اطلاعات به رمزهای قالبی، استفاده از الگوریتم‌های قالبی سبک وزن اهمیت روز افزونی یافته‌اند. در این میان، رمزهای ARX با توجه به استفاده از عناصر ساده و سبک منجر به ایجاد رمزهایی قوی از منظر امنیت و سبک‌وزن بودن می شود.

در این مقاله، به‌منظور بررسی امنیتی رمز قالبی سبک‌وزن MARX-2 به‌عنوان یک رمز مبتنی بر ساختار ARX، یک نوع از حمله القای تفاضلی به نام حمله بیت وارونه که تاکنون بر روی این الگوریتم انجام نشده است اعمال شد و نتیجه حاصله که بازیابی بیت‌های کلید این رمز بود را به چند روش مقایسه و ارزیابی گردید.

از طرفی در مورد رمز SPECK، الگوهای دیگر القای خطای تفاضلی، مانند خطای بایتی و خطای کنترل شده منجر به بازیابی کلید با تعداد خطای بسیار کمتر، در قیاس با بیت وارونه، شده است. به همین دلیل در ادامه این پژوهش، ارزیابی رمز MARX-2 بر اساس چنین الگوهای القای خطایی می تواند مد نظر قرار گیرد. البته انتظار می رود به دلیل پیچیده تر بودن ساختار MARX-2 در قیاس با SPECK عیناً نتوان نتایج تحلیل مربوطه را به این رمز گسترش داد، اما ارزیابی اولیه ما نشان از امیدوار کننده بودن این کار و کاهش تعداد خطای مورد نیاز برای بازیابی کلید دارد. لذا این مورد، به عنوان کار آتی این پژوهش مد نظر قرار گرفته است.

تقدیر

بخشی از این پژوهش توسط دانشگاه تربیت دبیر شهید رجایی حمایت مالی شده است.

فهرست مراجع

[۱] D.J. Bernstein,“New Stream Cipher Designs: The eSTREAM Finalists”, Springer Berlin Heidelberg 84–97, 2008.

[۲] D.J. Bernstein, ”ChaCha, a variant of Salsa20”. In: WorkshopRecord of SASC, Volume 8, 2008.

[۳] F. Niels, S. Lucks, , B. Schneier and D. Whiting, M .Bellare, T .Kohno, J .Callas, J .Walker, “The Skein hash function family”, Submission to NIST(round 3) , 2010.

[۴] J.P. Aumasson, L. Henzen, , W .Meier, and R.C.W. Phan, “SHA-3 Proposal BLAKE”. https://131002.net/blake/blake.pdf , 2010.

[۵] R.M. Needham and D.J. Wheeler, “Tea extensions”, Technical report, Cambridge University, Cambridge, UK , 1997.

[۶] N. Mouha, B. Mennink, A. Herrewege, D. Watanabe, B. Preneel and I. Verbauwhede, “Chaskey: An Efficient MAC Algorithm for 32-bit Microcontrollers”, In: Selected Areas in Cryptography – SAC 2014: 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers, Springer International Publishing, Cham 306–323, 2014.

[۷] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks and L. Wingers, “The SIMON and SPECK Families of Lightweight Block Ciphers”, IACR Cryptology ePrint Archive, 2013.

[۸] D. Hong, J.K. Lee, D.C. Kim, D. Kwon, K.H . Ryu and D.G . Lee, “LEA: A 128-bit block cipher for fast encryption on common processors”, In: Information Security Applications. Springer 3–27, 2013.

[۹] D. Dinu , L. Perrin , A. Udovenko , V. Velichkov, J. Großschädl and A. Biryukov, “Design Strategies for ARX with Provable Bounds Sparx and LAX”, Springer , 2016.

[۱۰] H. Tupsamudre, S. Bisht, and D. Mukhopadhyay, “Differential fault analysis on the families of SIMON and SPECK ciphers”, In 2014 Workshop on Fault Diagnosis and Tolerance in Cryptography (pp. 40-48). IEEE, 2014.

[۱۱] Y. Huo, F. Zhang, X. Feng and L. P. Wang, “Improved differential fault attack on the block cipher SPECK”, In 2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) (pp. 28-34), IEEE, 2015.

[۱۲] N. T. Courtois and B. Debraize, “Algebraic Description and Simultaneous Linear Approximations of Addition in Snow 2.0”, In Information and Communications Security, volume 5308 of Lecture Notes in Computer Science, pages 328-344, Springer, 2008.

[۱۳] B. Debraize and I. M. Corbella., “Fault Analysis of the Stream Cipher Snow 3g”, In Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC) , pages 103110, IEEE, 2009.

[۱۴] G.M. Greuel, G. Pfister and H. Schönemann, “A Computer Algebra System for Polynomial Computations, Centre for Computer Algebra”, University of Kaiserslautern, http://www.singular.uni-kl.de, 2001.

[۱۵] H. Tupsamudre, S. Bisht and D. Mukhopadhyay, “Differential Fault Analysis on the Families of Simon and Speck Ciphers”, In Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), pages 40-48, IEEE, 2014.

[۱۶] J. Feng, H. Chen, S .Gao, L. Fan and D. Feng, “Improved Fault Analysis on the Block Cipher SPECK by Injecting Faults in the Same Round”. In International Conference on Information Security and Cryptology, Springer, Cham (pp. 317-332), 2016.

[۱۷] R. Beaulieu, D. Shors, J. Smith, S .Treatman-Clark, B. Weeks and L. Wingers, “The SIMON and SPECK Famillies of Lightweight Block Ciphers”, Cryptology ePrint archive, Report 2013/543 , 2013.

[۱۸] A. Biryukov, V. Velichkov and Y.L. Corre, “Automatic search for the best trails in ARX: application to block cipher speck”, In: FSE 2016. LNCS, Springer , 2016.

[۱۹] J. Blömer and J.P. Seifert, ”Fault based cryptanalysis of the advanced encryption standard (AES)”. In International Conference on Financial Cryptography, Springer, Berlin, Heidelberg (pp. 162-181), 2003.