چهارمین کنفرانس انجمن رمز ایران
۲۴ الی ۲۶ مهر ماه ۱۳۸۶ دانشگاه علم و صنعت ایران
انجمن رمز ایران
Iranian Society of Cryptology

بهبود الگوریتم‌های درهم‌ساز خانوادهٔ *MDx* به کمک کدهای خطی اصلاح خطا

نصور باقری
آزمایشگاه رمز و سیستمهای امن، دانشگاه علم و صنعت ایران
n_bagheri@iust.ac.ir

مجید نادری
دانشکده مهندسی برق، دانشگاه علم و صنعت ایران
m_naderi@iust.ac.ir

چکیده: در این مقاله یک روش برای بهبود پایداری توابع درهم‌ساز در مقابل حملات تلاقی ارائه می شود. در اینجا روشی برای بسط پیام ارائه میشود که در صورت به‌کارگیری آن در ساختار الگوریتمهای درهم ساز گروه *MDx*، این الگوریتمها در مقابل حملاتی که تا به امروز برای آنها ارائه شده است با حاشیهٔ امنیت بالایی ایمن می شوند. در این روش از یک کد انتقال خطی مناسب برای بسط پیام استفاده می شود. با توجه به اینکه اساس کار تمامی حملات معرفی شده برای توابع درهم ساز بر کنترل میزان اختلاف موجود بین دو متن مختلف که منجر به تلاقی می شوند است، نشان داده می شود که درصورت استفاده از یک بردار کد با وزن مناسب می توان این الگوریتمها را در مقابل حملات معرفی شده برای توابع فشرده‌ساز ایمن کرد.

واژه های کلیدی: کدهای اصلاح خطا، بسط پیام، توابع درهم‌ساز، تابع فشرده‌ساز، حمله تلاقی، خانواده *MDx*.

۱. مقدمه

یکی از توابع پایهٔ مورد استفاده در رمزنگاری تابع درهم‌ساز است. تابع درهم‌ساز تابعی است که یک پیام با طول تصادفی را بعنوان ورودی دریافت کند و یک نتیجهٔ درهم‌سازی با طول ثابت از آن تولید کند. در یک تابع درهم‌ساز شرط لازم برای اینکه خروجی آن بتواند یک اثر منحصربفرد از پیام ارئه کند این است که پیدا کردن زوجهای دارای تلاقی، بعنوان نمونه پیامهایی که به یک خروجی یکسان نگاشت شوند، عملی نباشد.

معروفترین و پرکاربردترین توابع درهم ساز یک گروه خاص از توابع درهم ساز موسوم به خانوادهٔ *MDx* هستند. ضعفهای اساسی برای نسخهٔ اولیهٔ این الگوریتم پیدا شدند ولی این ضعفها در نسخه‌های بعدی این الگوریتم مد نظر قرار گرفته‌اند. با این وجود حملاتی که اخیراً به این توابع اعمال شده است، امنیت آنها را با تهدید جدی روبرو کرده است و به‌همین دلیل توجه جامعهٔ جهانی به این بخش از علم رمزنگاری بیش از پیش معطوف شده است.

به‌دنبال حملات جدید معرفی شده برای گروه *MDx* تلاشهایی برای جایگزینی این توابع صورت گرفته است و الگوریتمهای جدیدی نیز معرفی شده است. اگر چه در ساختار این الگوریتمها سعی شده است که ضعفهای موجود در گروه *MDx* برطرف شود، ولی تحلیل‌های انجام شده بر روی این توابع جدید نیز مؤید وجود صعف‌های اساسی در طراحی آنها می

باشد. به‌عنوان نمونه‌ای از توابع جدید می توان از الگوریتم *FORk-256* نام برد[۱].

اساس تمامی حملات موفقیت آمیزی که برای این توابع درهم ساز ارائه شده است، کنترل تفاضل ورودی به مراحل مختلف تابع فشرده ساز آنها از طریق کنترل متن ورودی است. تابع بسط پیام در این گروه از کارایی لازم برخوردار نیست و به همین دلیل حمله گر قادر است از قدرت بالایی در اعمال تفاضل ورودی مورد نظر به هر دور از تابع فشرده ساز برخوردار باشد. در این مقاله سعی می شود با استفاده از مشخصهٔ کدهای اصلاح خطا راه کاری برای بهبود عملکرد این الگوریتمها و افزایش حاشیهٔ امنیت آنها در مقابل حملات معرفی شده ارائه گردد.

در ادامهٔ این مقاله در بخش ۲ ساختار کلی توابع درهم ساز گروه *MDx* بررسی می شود. در بخش ۳ مشخصات کدهای اصلاح خطا با ویژگی مورد نظر مورد بررسی قرار می گیرند. در بخش ٤ تأثیر به‌کارگیری کدهای اصلاح خطا در بهبود امنیت توابع درهم ساز مورد نظر مورد بررسی قرار می گیرد. در بخش ۵ نیز نتیجه گیری و جمع بندی ارائه می شود.

۲. بررسی مشخصات توابع درهم‌ساز گروه *MDx*

توابع درهم‌ساز گروه *MDx* جزو ساختارهای تکرار شونده هستند. این به این معنی است که درهم‌سازی مبتنی بر تکرار یک تابع فشرده‌ساز است که در هر مرحله، یک قطعه از پیام و یک متغیر زنجیره را بعنوان ورودی دریافت می کند و یک متغیر زنجیره‌ای برای تکرار بعدی بعنوان خروجی تولید می کند. در این سیستم یک مقدار اولیه برای استفاده بعنوان اولین مقدار زنجیره ارائه می شود وپیام نیز قبل از اینکه وارد فرایند شود مورد عملیات پیش پردازش قرار می گیرد و بعضی مقادیر در آن لایی گذاری می شوند و به قطعاتی با طول مساوی (b) تقسیم می شود.

۳. کدهای اصلاح خطا

کدهای اصلاح خطا وسیله‌ای برای یافتن و اصلاح خطاهای حادث شده در بیتهای داده در اثر انتقال از طریق یک کانال مخابراتی می باشند. از طرفی دیگر، رمز نگاری این امکان را فراهم کند که پیام، به جر برای گیرندهٔ مجاز، غیر قابل خواندن باقی بماند. با این حال کدهای اصلاح خطا به دلیل مشخصهٔ انتشار مناسبی که می توانند فراهم کنند در ساختار سیستمهای رمزنگاری قطعه‌ای استفاده فراوانی دارند. یک گروه جالب از این کدها که یک نگاشت خطی است کدهای با حداکثر فاصلهٔ جداسازی یا کدهای *MDS*[1] می باشند[۲]. از این کدها در ساختار الگوریتمهای رمز قطعه‌ای نظیر *SHARK* [۳] و *AES* [۴] استفاده شده است. این گروه از نگاشتهای خطی دارای این مزیت هستند که تعداد جعبه‌های جانشینی که درهر دور از یک تقریب خطی و یا در هر دو دور مشخصهٔ تفاضلی درگیر می شوند برابر با $M+1$ است، که این از نظر تئوری حداکثر مقدار ممکن است. در این مقاله سعی می شود از این مشخصهٔ کدهای اصلاح خطا برای بهبود توابع درهم ساز استفاده شود. در واقع در اینجا نیز سعی می شود با به‌کارگیری این مشخصهٔ مناسب کدهای اصلاح خطا تعداد مراحلی از تابع فشرده ساز که در هر حملهٔ تلاقی درگیر می شوند حداکثر شده و از این راه امکان رسیدن به تلاقی در تابع درهم‌ساز به حداقل برسد. در ادامه یکسری تعاریف مرتبط با موضوع کدهای خطی را بیان می کنیم. برای اطلاعات بیشتر، می توان به مراجع مناسبی که در این زمینه موجود هستند مراجعه کرد[۲].

فاصلهٔ همینگ بین دو بردار v و u از یک فضای برداری n بعدی $GF(2^p)^n$ عبارت است از تعداد درایه‌هایی که u و v برابر نیستند. به همین ترتیب وزن همینگ $W_h(a)$ مربوط به یک عضو $a \in GF(2^p)^n$ عبارت است از فاصلهٔ همینگ بین این عضو و بردار صفر متعلق به $GF(2^p)^n$ که متناظر با اعضای غیر صفر a است.

[1] Maximum Distance Separable

یک کد خطی $[n,k,d]$ بر روی $GF(2^p)^n$ یک زیر فضای k بعدی از فضای برداری $GF(2^p)^n$ است که فاصلهٔ همینگ هر دو زیر بردار مجزای آن حداقل d است و d بزرگترین عددی است که این ویژگی را دارد.

ماتریس مولد G متعلق به کد خطی $[n,k,d]$ یک ماتریس $k \times n$ است که سطرهای آن تشکیل یک پایه برای کد C می دهد. در حالت استاندارد ماتریس مولد G یک ماتریس به فرم $G=\left[I_{k\times k}, A_{k\times(n-k)}\right]$ است، که در اینجا $I_{k\times k}$ یک ماتریس یکانی به بعد k است. در جایی که ابعاد ماتریس مولد روشن باشند، برای سادگی آن را به فرم $G=[IA]$ می نویسیم.

کدهای خطی $[n, k, d]$ از محدودیت سینگلتن[۲] تبعیت می کنند که می توان آن را به‌این صورت نوشت:

$$d \le n-k+1$$

۴. بررسی کارایی به‌کارگیری کدهای اصلاح خطا در بهبود امنیت توابع درهم ساز

یکی از بخشهای اصلی که در ساختار تمامی توابع درهم‌ساز خانوادهٔ *MDx* لحاظ شده است، فرایند گسترش پیام است. دلیل این امر از آنجا ناشی می شود که همواره طول پیام به‌کار رفته در مراحل میانی تابع فشرده ساز طولانی‌تر از طول قطعه پیام وارد شونده به یک تابع فشرده ساز است. جدول ۱ طول قطعه پیام وارد شونده به هر تابع درهم ساز و نیز طول پیام بسط یافته متناظر با آن را نمایش می دهد. بنابر این می توان تابع بسط پیام را بصورت یک تابع $EP : M \rightarrow EM$، که در اینجا EP تابع بسط پیام، $M=\{0,1\}^m$ همان قطعه پیام وارد شونده به تابع فشرده ساز با طول m و $EM=\{0,1\}^{em}$ پیام بسط یافته با طول em است. در توابع درهم‌ساز گروه *MDx* همواره $m<em$ است. پس می توان از یک کد خطی با مشخصهٔ $[em,n,d]$ برای بسط پیام استفاده کرد. در اینجا em و n کاملاً مشخص هستند و جزو مشخصات الگوریتم هستند. مقدار d می تواند تأثیر زیادی بر پایداری الگوریتم داشته باشد. بسته به مقدار d، تعداد کلمات پیام بسط یافته که وارد زیر مراحل پردازش تابع فشرده ساز می شوند و برای دو قطعه پیام متفاوت با هم تفاوت خواهند داشت، تغییر می کند. با توجه به اینکه هدف ما این است که درصورت تغییر یک بیت از پیام ورودی تقریباً ۵۰٪ از بیتهای پیام وارد شونده به مرحلهٔ فشرده‌سازی تغییر پیدا کند بنابراین d را برابر با $em/2$ درنظر می گیریم. با توجه به ساختار الگوریتمهای گروه *MDx* امکان رسیدن به چنین مشخصه‌ای وجود دارد زیرا با توجه به جدول ۱ تابع *EP* مربوط به تمامی این الگوریتمها برای این مقدار از d، در شرط سینگلتن صدق می کنند. درصورتی که d برابر با $em/2$ درنظر گرفته شود و احتمال تغییر هر بیت از کلمهٔ کد در اثر تغییر کلمهٔ پیام را یکسان در نظر بگیریم، در این صورت برای دو قطعه پیام متفاوت، هر کلمه از پیام بسط یافته با احتمالی برابر با $(1-2^{-32})$ یکسان نخواهند بود، و با احتمالی برابر با ۱ حداقل ۵۰٪ کلمات دو پیام بسط یافته متفاوت هستند.

جدول ۱ طول قطعه پیام ورودی به تابع فشرده ساز و پیام بسط یافته متناظر با آن.

الگوریتم	طول قطعه پیام	طول پیام بسط یافته
MD4	۵۱۲	۱۵۳۶
MD5	۵۱۲	۲۰۴۸
HAVAL	۱۰۲۴	۵۱۲۰
RIPEMD	۵۱۲	۱۵۳۶
RIPEMD-128	۵۱۲	۲۰۴۸
RIPEMD-160	۵۱۲	۲۵۶۰
SHA	۵۱۲	۲۵۶۰
SHA-1	۵۱۲	۲۵۶۰

اگر حملات معرفی شده برای توابع درهمساز را در قالب دو گروه زیر دسته بندی کنیم:

۱. حملات معرفی شده برای ساختار درهم سازی، مانند حملهٔ ژو[۳].

۲. حملات معرفی شده برای تابع فشرده ساز به‌کاررفته در ساختار درهم سازی، مانند حملهٔ وانگ[۴].

[2] Singleton

[3] Joux

[4] Wang

روش پیشنهادی در این مقاله، موجب پایدار الگوریتمها در مقابل دستهٔ دوم از حملات می شود.

در ادامهٔ این بخش، تأثیر تغییر پیشنهادی در بهبود امنیت تعدادی از الگوریتمها گروه *MDx* در مقابل حملات ارائه شده برای تابع فشرده ساز این الگوریتمها مورد بررسی قرار می گیرند. لازم به ذکر است که در صورت بهکارگیری روش پیشنهادی بقیهٔ حملات ارائه شده برای این توابع فشرده ساز، که دراینجا ذکر نشدهاند، نیز کارگر نخواهند بود.

۴-۱ شرح الگوریتم *MD4*

الگوریتم *MD4* [۶] برای پیامهای با طول دلخواه، یک نتیجهٔ درهمسازی با طول ۱۲۸ بیت تولید می کند. از آنجایی که الگوریتم کلمات با طول ۳۲ بیت را مورد پردازش قرار می دهد، بنابراین مقدار زنجیره به چهار ثبات ۳۲ بیتی (A, B, C, D) تقسیم می شود. تابع فشردهساز قطعات پیام ۵۱۲ بیتی را مورد پردازش قرار می دهد. بنابراین یک قطعهٔ پیام به شانزده کلمهٔ ۳۲ بیتی تقسیم می شود که با W_j برای $j = 0,1,\ldots,15$ نمایش داده می شوند.

از نظر ساختار داخلی، تابع فشردهساز مشتمل بر ۴۸ مرحلهٔ متوالی است که در هر مرحله مقدار یک ثبات از چهار ثبات بهروز رسانی می شود.

۴-۲ تحلیل الگوریتم *MD4*

در منبع [۵] یک روش برای رسیدن به تلاقی در *MD4* ارئه شده است. در این روش برای دورهای اول و دوم الگوریتم*MD4*، تلاقی تقریبی داخلی پیدا می شود، به این معنی که بعد از دور دوم مقدار اختلاف اندک باشد. اختلاف بین دو پیام در کلمات W_{12} و W'_{12} در نظر گرفته شده است. در این حالت سعی می شود که اختلاف بین ثباتها بعد از دور دوم بسیار اندک باشد و این اختلاف اندک نیز با ترفند تحلیل تفاضلی در مرحلهٔ سوم با اعمال W_{12} و W'_{12} برای بار سوم حذف شود. این حمله یک حالت سادهسازی حملهٔ دوبرتین[۷] است. در این حمله روند حل معادلات تفاضلات تسهیل شده است.

می توان حملات مشابهی پیدا کرد که اختلاف پیامهای ورودی در W_{12} نباشد، بلکه در هر کدام از دیگر کلمات W_j باشد. دلیل اینکه در حملهٔ معرفی شده قبلی کلمهٔ W_{12} انتخاب شده است این است که پیچیدگی دو بخش مختلف حمله پایین می آید.

درصورتی که از روش بیان شده برای بسط پیام استفاده شود، حمله کننده قادر نخواهد بود که تفاضل مورد نظر را تنها محدود به تعدادی محدود کلمه از پیام بسط یافته نماید و در هر حالت حداقل ۲۴ کلمه از ۴۸ کلمهٔ پیام بسط یافته تغییر می کند. این حجم بالای اختلاف، امکان کنترل مشخصهٔ تفاضلی را بسیار ناچیز می کند و عملاً استراتژی بیان شده برای حمله، کارگر نخواهد بود.

۴-۳ شرح الگوریتم *MD5*

الگوریتم *MD5* [۸] بعنوان یک نسخهٔ تقویت شده از الگوریتم *MD4* ارائه شد. از آنجایی که نتیجهٔ درهمسازی الگوریتم ۱۲۸ بیتی است، بنابراین پارامتر زنجیره به چهار ثبات ۳۲ بیتی (A, B, C, D) تقسیم می شود. طراحی *MD5* بسیار شبیه *MD4* است و تنها دارای اختلافات زیر است:

- تابع فشردهساز در این الگوریتم متشکل از ٦٤ مرحلهٔ متوالی است در قالب٤ دور در حالیکه *MD4* دارای ٤٨ مرحله در قالب ۳ دور بود.
- عمل انجام شده در هر قدم تا حدودی متفاوت است.
- در دورهای دوم و سوم، ترتیب بهکارگیری کلمات W_j متفاوت از ترتیب استفاده شده در *MD4* است.
- تابع بولی مورد استفاده در دور های متفاوت تا حدودی تغییر کرده است.
- در هر مرحله از یک مقدار جمعشوندهٔ منحصر به فرد برای آن مرحله استفاده می شود.

- مقدار ثابتهای چرخش تغییر پیدا کرده است و در دورهای متفاوت از مقادیر یکسان برای ثبت چرخش استفاده نمی کنند.

۴-۴ تحلیل الگوریتم *MD5*

دوبرتین در منبع [۹] حمله‌ای را معرفی کرده است که تلاقی را در تابع فشرده‌ساز *MD5* پیدا میکند. این به معنی پیدا شدن دو قطعه پیام $\{W_j\}$ و $\{W'_j\}$ و $(0 \le j \le 15)$ که به ازای یک مقدار اولیه زنجیره مشابه، منجر به خروجی یکسان می شوند، است. ترفند استفاده شده در این حمله مشابه با آنچه در *MD4* بکار رفت می باشد. تلاقی برای دو پیام پیدا می شود که دارای اختلاف اندکی و تنها در یک کلمه هستند:

$$W'_{14} = W_{14} + 1^{<<9}, \qquad (۳)$$
$$W'_j = W_j \quad (j \neq 14).$$

شمای عمومی حمله به این صورت است:

- پیدا کردن تلاقی داخلی در دورهای ۱ و ۲ (مراحل ۱۵ تا ۲۶)؛
- پیدا کردن تلاقی داخلی در دورهای ۲ و ۳ (مراحل ۳۶ تا ۵۱)؛
- اتصال دو تلاقی داخلی.

لازمهٔ پیدا کردن تلاقی داخلی مورد نیاز دو بخش اول حمله، ساختن و حل کردن یک دستگاه معادلات تفاضلات، مشابه با حمله انجام شده بر روی *MD4* است.

در ایجا نیز مشاهده میشود که شرط لازم برای قابل اجرا بودن حمله، کنترل دقیق تفاضل ورودی پیام بسط یافته به دورهای مختلف است. این تفاضل باید به گونه‌ای فراهم شود که تنها ۳ کلمه از پیام و آن هم با شرایط مشخص اختلاف داشته باشند. این در حالی است که درصورت استفاده از روش پیشنهاد شده برای بسط پیام ، حمله کننده قادر نخواهد بود که تفاضل مورد نظر را تنها محدود به سه کلمه از پیام بسط یافته نماید و در هر حالت حداقل ۳۲ کلمه از ۶۴ کلمهٔ پیام بسط یافته تغییر می کنند. بنابراین، با توجه به روند پیشنهادی برای بسط پیام، تفاضل ورودی متناظر با دو پیام متفاوت حداقل در ۳۲ کلمه متفاوت خواهند بود. این حجم بالای اختلاف، امکان کنترل مشخصهٔ تفاضلی را بسیار ناچیز می کند و عملاً استراتژی بیان شده برای حمله، کارگر نخواهد بود.

۴-۵ شرح الگوریتم *SHA*

الگوریتم *SHA-1* برای هر پیام با طول دلخواه که طول آن کمتر از 2^{64} بیت باشد، یک نتیجهٔ درهم‌سازی ۱۶۰ بیتی تولید می کند[۱۰و۱۱]. مشابه با بقیهٔ الگوریتمهای خانوادهٔ *MDx* در اینجا نیز طول کلمات مورد پردازش ۳۲ بیت می باشد. طول قطعه پیامهایی که تابع فشرده‌ساز آن را مورد پردازش قرار می دهد، ۵۱۲ بیت است که در قالب شانزده کلمهٔ ۳۲ بیتی W_j برای $J = 0,1,...,15$ قرار می گیرد.

اگر بخواهیم از نظر ساختار داخلی بررسی کنیم، تابع فشرده‌ساز آن به ۸۰ مرحلهٔ متوالی تقسیم می شود. از نگاهی دیگر این تابع فشرده‌ساز متشکل از چهار دور که هر دور متشکل از ۲۰ مرحله است، می باشد. در هر مرحله مقدار دو ثبات از پنج ثبات عوض می شود. تنها تفاوتی که بین دو نسخهٔ *SHA-0* و *SHA-1* وجود دارد مربوط به همین نحوهٔ بروز رسانی است. مقدار چرخش یک بیت به سمت چپ، در *SHA-0* لحاظ نشده بود[۱۱].

۴-۶ تحلیل الگوریتم *SHA*

۴-۶-۱ حملهٔ چابود و ژوکس به الگوریتم *SHA-0*

در تحلیلی که به وسیلهٔ چابود و ژوکس در منبع [۱۲] انجام شده است، یک حملهٔ تلاقی بصورت تئوری برای الگوریتم *SHA-0* پیدا شده است که پیچیدگی آن برابر با 2^{61} عملیات تابع فشرده‌ساز آن است.

رویهٔ عمومی بکار رفته در این حمله به این صورت است که گسترش یک آشفتگی محلی را پیگیری می کند و دنبال پوششهای تفاضلی می گردد که بتواند به قطعه ۸۰ کلمه‌ای

اضافه شود، در حالیکه خروجی با احتمال بالایی بدون تغییر بماند. در مرجع [۱۲] تعدادی از حالتهای ساده شده *SHA-0* نیز مورد تجزیه و تحلیل قرار گرفته اند.

لازم به ذکر است که علی رقم تشابه بسیار زیاد الگوریتم‌های *SHA-1* و *SHA-0* حملات آشوب بیان شده قابل اعمال به الگوریتم *SHA-1* نیستند. دلیل آن هم این است که یک بیت چرخشی که در ساختار بسط پیام استفاده شده برای الگوریتم *SHA-1* به کار رفته است باعث می شود که تابع خطی گسترش آن دیگر در سطح بیت عمل نکند. در واقع این چرخش یک بیتی باعث می شود که تغییر یک بیت به مکانهای دیگر کلمات نیز نفوذ کند. همین نکته باعث می شود که رویهٔ حملهٔ بکار رفته در منبع [۱۲] دیگر کارگر نباشد و یک دلیل قوی برای این نکته است که گذر از *SHA-0* به *SHA-1* موجب افزایش سطح امنیتی شده است.

دلیل اصلی عدم کارگرشدن حمله برای *SHA-1*، بالا رفتن میزان نفوذ در اثر استفاده از یک بیت چرخش و درنتیجه غیر قابل کنترل شدن تفاضلات مختلف بوسیله حمله کننده است. این در حالی است که درصورت استفاده از روش پیشنهاد شده برای بسط پیام ، حمله کننده قادر نخواهد بود که تفاضل مورد نظر را تنها محدود به تعدادی محدود کلمه از پیام بسط یافته نماید و در هر حالت حداقل ۴۰ کلمه از ۸۰ کلمهٔ پیام بسط یافته تغییر می کنند. بنابراین، با توجه به روند پیشنهادی برای بسط پیام، تفاضل ورودی متناظر با دو پیام متفاوت حداقل در ۴۰ کلمه متفاوت خواهند بود. این حجم بالای اختلاف، امکان کنترل مشخصهٔ تفاضلی را بسیار ناچیز می کند و عملاً استراتژی بیان شده برای حمله، کارگر نخواهد بود.

۴-۶-۲ حملهٔ بیهام و چن

در مرجع [۱۳] حملهٔ بیان شده در قسمت قبل را با استفاده از ترفندی موسوم به ترفند بیتهای بی اثر بهبود داده‌اند. برای آگاهی از این تکنیک می توانید به مرجع [۱۳] مراجعه کنید.

بیهام و چن، در منبع[۱۳]، با استفاده از شیوهٔ بیان شده توانسته‌اند برای *SHA-0* گسترش داده شده به ۸۲ مرحله تلاقی به‌دست آورند. علاوه بر این در منبع [۱۴] کاربرد این شیوه برای نسخهٔ کاهش یافتهٔ *SHA-1* ارائه شده است که منجر به تلاقی برای ۴۳ مرحله شده است و در نتیجه گیری آمده است که باید برای ۵۳ مرحلهٔ انتهایی نیز امکان پذیر باشد. در منبع [۱۵] ژوکس و دیگران برای پیدا کردن تلاقی واقعی در *SHA-0* با ۸۰ مرحله، این ترفند را با ترکیب ٤ مسیر تفاضلی اینچنینی، با استفاده از دو پیام که هرکدام از ٤ قطعه پیام تشکیل شده‌اند، بکار گرفته‌اند.

مشابه با حملهٔ چابود و ژوکس، در اینجا نیز دلیل اصلی عدم کارگرشدن حمله برای *SHA-1*، بالا رفتن میزان نفوذ در اثر استفاده از یک بیت چرخش و درنتیجه غیر قابل کنترل شدن تفاضلات مختلف بوسیله حمله کننده است. بنابراین حمله کننده قادر به فراهم کردن شرایط لازم برای حمله نمی شود. این در حالی است که در اینجا نیز درصورت استفاده از روش پیشنهاد شده برای بسط پیام ، حمله کننده قادر نخواهد بود که تفاضل مورد نظر را تنها محدود به تعدادی محدود کلمه از پیام بسط یافته نماید و در هر حالت حداقل ۴۰ کلمه از ۸۰ کلمهٔ پیام بسط یافته تغییر می کنند. بنابراین، با توجه به روند پیشنهادی برای بسط پیام، تفاضل ورودی متناظر با دو پیام متفاوت حداقل در ۴۰ کلمه متفاوت خواهند بود. این حجم بالای اختلاف، امکان کنترل مشخصهٔ تفاضلی را بسیار ناچیز می کند و عملاً استراتژی بیان شده برای حمله، کارگر نخواهد بود.

۴-۶-۳ حمله وانگ و دیگران

مهمترین اتفاقی که اخیراً توابع درهم ساز را با چالش روبرو کرد رشته تحلیلهایی بودند که در قالب منابع [۱۶-۱۹] توسط وانگ و دیگران ارائه شدند. این رشته از حملات که امنیت توابع درهم ساز استاندارد مبتنی بر گروه *MDx* را هدف قرار داده بود موجب جلب توجه جامعه جهانی به این بخش شد. در این راستا *NIST* اقدام به برگزاری یک فراخوان برای انتخاب استاندارد جدید درهم سازی کرده است[۲۰].

یک تفاوت عمده‌ای که بین حملهٔ بکار رفته توسط وانگ ودیگران با حمله‌ای که در بخش قبل بیان شد وجود دارد، این

است که در این حمله، به جای استفاده از تفاضل انحصاری، از تفاضل پیمانه‌ای استفاده شده است.

همهٔ تلاقی‌هایی که اخیراً بر مبنی این حمله در منابع [۱۶-۱۹] منتشر شده‌اند، تلاقی در توابع درهم‌سازی هستند که از جایگشت مکان کلمات پیام بعنوان تابع گسترش دهندهٔ پیام استفاده کرده‌اند. این به این معنی است در هر دور، هر کدام از کلمات پیام، یک‌بار بعنوان یکی از Wj ها اعمال می شود[۲۱].

همانگونه که برای حملهٔ چابود و ژوکس بیان شد، وانگ و دیگران نیز حملهٔ خود را با گشتن دنبال یک مسیر تفاضلی شروع می کنند. اما در این حمله پیدا کردن این مسیر تفاضلی مورد نظر در دو بخش انجام می شود. ابتدا سعی می شود که یک مسیر تفاضلی ورودی مناسب، که در یک بخش (مثلاً در آخرین دور) دارای رفتار تفاضلی مناسب باشد، را پیدا کنند، آنگاه برای مراحل باقی مانده مسیر تفاضلی متناظر را پیدا کنند.

وانگ و دیگران، با استفاده از این روش توانسته‌اند که مسیرهای تفاضلی و یک مجموعه شرایط برای مقادیر ثابتها به‌دست آورند که برای پیدا کردن تلاقی واقعی به‌کار ببرند. آنها توانسته‌اند این روش را با موفقیت به توابع درهم‌سازی که دارای سه دور هستند، نظیر *MD4* و *HAVAL-128* اعمال کنند و آنها را بشکنند. با توجه به روش استفاده‌شده، به‌نظر می رسد با استفاده از این روش هر تابع فشرده‌ساز سه‌دوریی با مشخصات مورد نظر را بتوان شکست. این در حالی است که می توان گفت توابع دارای بیشتر از سه‌دور را تنها درصورتی می توان با استفاده از این شیوه شکست که دارای یک ضعف خاص در ساختار خود باشند که بتوان از آن برای شکستن الگوریتم بهره گرفت. بعنوان نمونه آنها توانسته‌اند برای الگوریتم *RIPEMD-0* که متشکل از دو شاخهٔ موازی است، که هر شاخه از سه دور تشکیل شده است و بنابراین کل الگوریتم از شش دور تشکیل شده است، تلاقی پیدا کنند. ضعفی که در اینجا وجود دارد این است که دو شاخهٔ موازی، تقریباً یکسان هستند و این امکان که یک مسیر تفاضلی پیدا کرده و به‌صورت همزمان به‌هر دوشاخه اعمال کرد، وجود دارد.

نقطه ضعف خاصی که در حملهٔ وانگ به *MD5* به‌کار گرفته شده است، امکان تولید تفاضل خروجی برابر با 2^{31} ، با اعمال تفاضلی خاص در ورودی است. این تفاضل در دور سوم از مرحلهٔ بعد و در هر مرحله با احتمال ۱ و در بخش زیادی از دور چهارم آن با احتمال $\frac{1}{2}$ برای هر مرحله، گسترش پیدا می کند. در نتیجه، این امکان فراهم می شود که یک مسیر تفاضلی ورودی به وجود می آید، که با احتمال بالایی منجر به یک تفاضل خروجی در دورهای سوم و چهارم شود. بنابراین، با استفاده از شیوه‌ای که در قسمت قبل توضیح داده شد، این امکان که به این تابع درهم ساز چهار دوری حمله کرد، وجود دارد.

شرایط لازم برای کارگر شدن حمله در اینجا بسیار دقیق تر از حملات قبلی است. در صورتی که تابع پیشنهاد شده برای بسط پیام به کار گرفته شود، حمله کننده قادر نخواهد بود که تفاضل مورد نظر را تنها محدود به تعدادی محدود کلمه از پیام بسط یافته نماید و در هر حالت حداقل ۴۰ کلمه از ۸۰ کلمهٔ پیام بسط یافته تغییر می کنند. بنابراین، با توجه به روند پیشنهادی برای بسط پیام، تفاضل ورودی متناظر با دو پیام متفاوت حداقل در ۴۰ کلمه متفاوت خواهند بود. این حجم بالای اختلاف، امکان کنترل مشخصهٔ تفاضلی را بسیار ناچیز می کند و عملاً استراتژی بیان شده برای حمله، کارگر نخواهد بود. دلیل این امر را باید در این نکته دانست که حمله کننده در این شرایط قادر نخواهد بود تفاضل بیان شده در شرایط حمله را فراهم کند.

۵. نتیجه گیری

در این مقاله روشی برای بهبود پایداری توابع درهم‌ساز در مقابل حملات تلاقی ارائه گردید. در این روش رویهٔ بسط پیام استفاده شده در الگوریتم‌های گروه *MDx* با استفاده از کدهای اصلاح خطای خطی با مشخصهٔ $[em,m,d]$ انجام شد. مزیت استفاده از این روش فعال سازی تعداد زیادی از کلمات پیام درصورت تغییر قطعه پیام ورودی و در نتیجه مشکل شدن امکان تدارک حمله برای الگوریتم‌های بهبود یافته است. اضافه شدن این بخش به الگوریتمهای موجود تا حدودی بار

محاسباتی را افزایش می دهد و درنتیجه الگوریتم بهبود یافته از بازدهی پایینتری برخوردار خواهد بود. با این وجود، و با توجه ضعفهایی که در نسخهٔ اصلی الگوریتمها پیدا شده است و با توجه به اینکه بسیاری از الگوریتمهای این خانواده در گسترهٔ وسیعی استفاده شده‌اند، استفاده از الگوریتم بهبود یافته، در کنار تأمین امنیت لازم کمترین تغییرات را به سیستمهای موجود اعمال می کند.

۶. مراجع

[1] D. Hong, S. Jaechul, S. Hong, S. Lee, and D. Moon. "A new dedicated 256-bit hashfunction: FORK-256." First NIST Workshop on Hash Functions, 2005.

[2] F.J. Mac Williams and N.J.A. Sloane, ″The theory of error correcting codes″ ,North-Holland Publishing Company, 1977.

[3] V.Rijmen, J. Daemen, B. Preneel, A. Bosselaers, and E. De Win. ″The cipher SHARK. Fast Software Encryption,″ LNCS 1039, D. Gollmann, Ed., Springer-Verlag, pp. 99-112, 1996.

[4] J. Daemen, V. Rijmen, " aes Proposal: Rijndael, " http://csrc.nist.gov/encryption /aes/rijndael/Rijndael.pdf

[5] B. Van Rompay, " Analysis and Design of Cryptographic Hash Functions, MAC Algorithms and Block Ciphers." Doctoral dissertation, K. U. Leuven, Juni. 2004.

[6] R. L. Rivest, "The MD4 message digest algorithm." in Advances in Cryptology – Crypto'90 (A. Menezes and S. A. Vanstone, eds.), no. 537 in Lecture Notes in Computer Science, pp. 303–311, Springer-Verlag, 1991.

[7] H. Dobbertin, "Cryptanalysis of MD4." Journal of Cryptology, vol. 11, no. 4, pp. 253–271, 1998.

[8] Ronald L. Rivest, "The MD5 message-digest algorithm," IETF RFC 1321, 1992.Available from www.ietf.org/rfc/rfc1321.txt

[9] H. Dobbertin, "The status of MD5 after a recent attack." CryptoBytes,vol. 2, no. 2, pp. 1,3–6, 1996.

[10] FIPS 180, "Secure Hash Standard (SHS)." National Institute of Standards and Technology, May 1993.

[11] FIPS 180-1, "Secure Hash Standard (SHS)." National Institute of Standards and Technology, Apr. 1995.

[12] F. Chabaud and A. Joux, "Differential collisions in SHA-0." in Advances in Cryptology - Crypto'98 (H. Krawczyk, ed.), no. 1462 in Lecture Notes in Computer Science, pp. 56-71, Springer-Verlag, 1998.

[13] E. Biham and R. Chen. "Near-Collisions of SHA-0. " In Advances in Cryptology - CRYPTO 2004, volume 3152 of LNCS, pages 290-305. Springer-Verlag, 2004.

[14] E. Biham, R. Chen, A. Joux, P. Carribault, W. Jalby, and C. Lemuet. " Collisions of SHA-0 and Reduced SHA-1. " In Advances in Cryptology - EUROCRYPT 2005, LNCS. Springer–Verlag, 2005.

[15] A. Joux, P. Carribault, W. Jalby, and C. Lemuet. "Collisions in SHA-0. " Presented at the rump session of CRYPTO 2004, August 2004.

[16] X. Wang, X. Lai, D. Feng, and H. Yu. "Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD. " Presented at the rump session of CRYPTO 2004, August 2004. http://eprint.iacr.org/2004/199

[17] X.Wang, X. Lai, D. Feng, H. Chen, and X. Yu. "Cryptanalysis for Hash Functions MD4 and RIPEMD. " In Advances in Cryptology - EUROCRYPT 2005, LNCS. Springer-Verlag, 2005.

[18] X. Wang and H. Yu. "How to Break MD5 and Other Hash Functions. " In Advances in Cryptology - EUROCRYPT 2005, LNCS. Springer-Verlag, 2005.

[19] X. Wang, Y. L. Yin, and H. Yu. "Collision Search Attacks on SHA1." preprint, February 2005. http://www.infosec.sdu.edu.cn/paper/sha-attack-note.pdf

[20] National Institute of Standards and Technology, Tentative Timeline of the Development of New Hash Functions, 2007,Available at: http://www.csrc.nist.gov/pki/HashWorkshop/timeline.htm

[21] Magnus Daum, "Cryptanalysis of Hash Functions of the MD4-Family" PhD Thesis, Bochum university, 2005.