مروری بر حملات مکعبی، خطای تفاضلی و ساختارهای خطی بر روی تابع کچک(SHA-3)

محمدامین ارغوانی[۱] و معصومه صفخانی[۲]

[۱] دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه تربیت دبیر رجایی، تهران، ایران
maminarghavani@gmail.com

[۲] استادیار دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر رجایی، تهران، ایران
Safkhani@srttu.edu

چکیده

در سال‌های اخیر بررسی امنیت سومین تابع چکیده‌ساز امن[1] یکی از مهم‌ترین موضوعات مورد توجه پژوهش‌گران حوزه رمزنگاری بوده است. توابع چکیده‌ساز در رمزنگاری کاربردهای متعددی دارند. از این‌رو امنیت توابع چکیده‌ساز در مقابل حملات مختلف برای سامانه‌هایی که از آن استفاده می‌کنند، بسیار مهم است. حملات و تحلیل‌های متعددی تا امروز بر روی SHA-3 اعمال شده است؛ اما تاکنون هیچ‌کدام موفق به شکستن آن به‌صورت نظری و یا در عمل نشده‌اند. SHA-3 بر اساس تابع کچک[2] طراحی شده است. تابع کچک از خانواده توابع اسفنجی[3] است. در این مقاله، با تمرکز بر حمله تحلیل خطای تفاضلی[4] آخرین حملات و تحلیل‌های مطرح‌شده بر روی SHA-3 را به‌صورت خلاصه مرور خواهیم کرد. به‌صورت دقیق‌تر، در این مقاله، حملات مکعبی[5]، خطای تفاضلی و هم‌چنین حمله متمایزکننده‌های مجموع صفر[6] و پیش‌تصویر[7] با استفاده از ساختارهای خطی[8] شرح داده خواهند شد.

واژگان کلیدی: کچک، SHA-3، تحلیل خطای تفاضلی، حمله مکعبی، حمله متمایزکننده‌های مجموع صفر، حمله پیش‌تصویر، ساختارهای خطی.

۱- مقدمه

توابع چکیده‌ساز از جمله نیازمندی‌های زیربنایی برنامه‌های امنیتی است، که برای نمونه می‌توان از تولید و ارزیابی امضای دیجیتال، تولید کلید و تولید بیت تصادفی نام برد. تابع چکیده‌ساز تابعی است بر روی داده‌های دودویی (برای مثال رشته‌ای از بیت‌ها) که در آن طول خروجی برای داده‌های مختلف ثابت است. به ورودی تابع چکیده‌ساز، پیام[9] و به خروجی آن چکیده یا مقدار درهم‌سازی‌شده گفته می‌شود. چکیده به‌طور معمول به‌عنوان مقدار خلاصه‌شده‌ای از پیام مورد استفاده قرار می‌گیرد. تابع چکیده‌ساز باید به‌گونه‌ای طراحی شود تا در مقابل حملات برخورد[10]، حملات پیش‌تصویر نخست و دوم و هم‌چنین در برابر سایر حملات شناخته‌شده مقاوم باشد.

[1] Secure Hash Algorithm 3(SHA-3)
[2] KECCAK
[3] Sponge Functions
[4] Differential Fault Analysis(DFA)
[5] Cube Attack
[6] Zero-sum Distinguisher
[7] Pre-image
[8] Linear Structure
[9] Message
[10] Collision

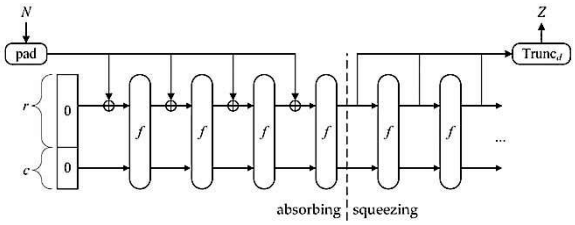
تابع چکیده‌ساز کچک [1] در سال ۲۰۱۲ به‌عنوان برنده مسابقه تعیین SAH-3 انتخاب شد. این تابع در استاندارد شماره ۲۰۲ انتشارات مرکزی استانداردها[1] چاپ شده است [2]. تابع چکیده‌ساز کچک در طول مسابقات و بعد از آن، از سوی جامعه رمزنگاری به‌شدت مورد توجه بوده است.

تحلیل‌های فراوانی بر روی تابع کچک انجام شده است از جمله [3]-[13]. نکته مهم آن که هیچ‌کدام از این تحلیل‌های متعدد نتوانسته به‌طور عملی و یا نظری امنیت تابع کچک را نقض کنند و هنوز حاشیه امنیت مناسبی بین تعداد دور شکسته‌شده و تعداد دورکامل (۲۴دور) الگوریتم وجود دارد.

در این مقاله با تمرکز بر حمله تحلیل خطای تفاضلی مروری بر آخرین حملات انجام‌شده بر روی تابع کچک خواهیم‌داشت. در ادامه و در بخش ۲ به تشریح تابع چکیده‌ساز کچک خواهیم پرداخت. در بخش ۳ حمله مکعبی و نتایج به دست آمده را از حمله مکعبی بر روی کچک بر اساس منبع [14] شرح می‌دهیم. حمله تحلیل خطای تفاضلی بر اساس منابع [15]،[16] در بخش ۴ آورده شده است. از آن‌جا که تمرکز کار ما بر روی حمله تحلیل خطای تفاضلی است، این حمله توسط ما دوباره بازسازی شده است. در بخش ۵، ساختار خطی و حملات پیش‌تصویر و تشخیص‌دهنده‌های مجموع صفر انجام شده بر روی تابع کچک، بر اساس تحلیل ساختارخطی و نتایج آن با توجه به منبع [17] آورده شده است و درنهایت مقاله در بخش ۶ با جمع‌بندی و نتیجه‌گیری به پایان می‌رسد.

۲- SAH-3

تابع کچک بر اساس ساختار اسفنجی[2] [29] ایجاد شده است. ساختار اسفنجی در واقع یک چهارچوب[3] برای تعیین توابعی است که بر روی داده‌های دودویی، خروجی دلخواه تولید کند [2]. در شکل (۱)، ساختار اسفنجی نشان داده شده است.



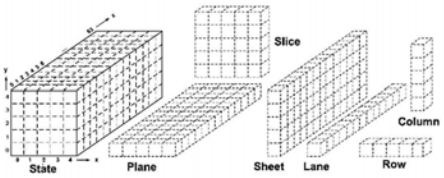
(شکل-۱): ساختار اسفنجی با استفاده از تابع f [2]

[1] Federal Information Processing Standards
[2] Sponge construction
[3] Framework

در ساختار اسفنجی، r نرخ[4] و c ظرفیت[5] است. مقدار ورودی و خروجی تابع f ثابت و برابر $b = r + c$ است. نکته این‌که مقدار r بیشتر، باعث افزایش سرعت و مقدار c بیشتر، باعث امنیت بیشتر می‌شود [15]. b می‌تواند مقادیر مختلفی داشته باشد که در حالت استاندارد مقدار آن $b = 1600$ است [2]. تابع f شامل ۲۴ دور است که هر دور شامل پنج نگاشت $\theta.\rho.\pi.\chi.\iota$ است. ترتیب اجرای این نگاشت‌ها در معادله (۱) مشخص است. در ادامه تعریف این نگاشت‌ها آورده شده است:

$$S_{i+1} = \iota \circ \chi \circ \pi \circ \rho \circ \theta(S_i).\, i \in \{0.1 \ldots .23\} \qquad (۱)$$

برای اجرای پنج نگاشت بالا، بیت‌ها به‌صورت یک ماتریس سه‌بعدی[x, y, z] ذخیره می‌شوند که در حالت استاندارد اندازه این ماتریس برابر [5,5,64] است. در شکل (۲) نام‌گذاری قسمت‌های مختلف ماتریس حالت تابع کچک آورده شده است. همان‌طور که در شکل مشخص است، اگر مقادیر x و y را ثابت و مقدار z را متغیر در نظر بگیریم(برای مثال [0, (0, z] یک سطر[6] ساخته‌ایم. با قراردادن پنج سطر با xهای متغیر(برای مثال [x, 0, z]) یک صفحه[7] ایجاد می‌شود.



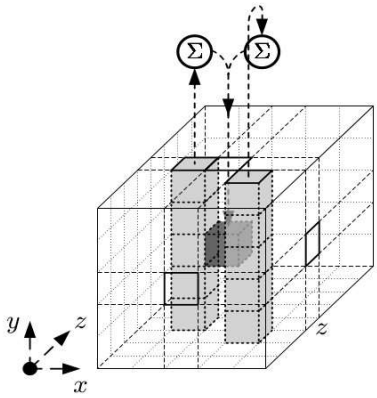
(شکل-۲): ساختار و نام‌گذاری ماتریس حالت در کچک [16]

۲-۱- نگاشت θ

تأثیر نگاشت θ به این‌صورت است که هر بیت در ماتریس حالت با دو ستون دیگر از ماتریس حالت XOR می‌شود (شکل۳)[2]. مراحل این جایگشت در الگوریتم زیر آمده است:

1. For all pairs (x, z) such that $0 \le x < 5$ and $0 \le z < w$, let
$C[x.z] = A[x.0.z] \oplus A[x.1.z] \oplus A[x.2.z] \oplus A[x.3.z] \oplus A[x.4.z]$
2. For all pairs (x, z) such that $0 \le x < 5$ and $0 \le z < w$, let
$D[x.z] = C[(x-1)\ mod\ 5.\ z] \oplus C[(x+1) mod\ 5.\ (z-1)\ mod\ w]$.
3. For all triples (x, y, z) such that $0 \le x < 5$, and $0 \le y < 5$, and $0 \le z < w$, let
$A'[x.y.z] = A[x.y.z] \oplus D[x.z]$.
Return A'

[4] Rate
[5] Capacity
[6] Lane
[7] Plane



(شکل-۳): ساختار نگاشت θ [2]

۲-۲- نگاشت ρ

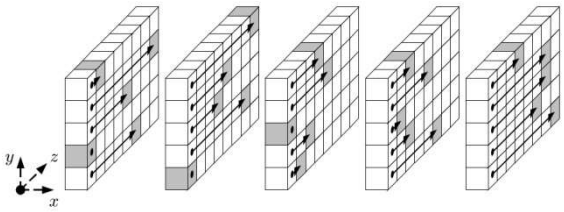
تأثیر این نگاشت به این‌صورت است که بیت‌های هر سطر (شکل۲) را با یک طول ثابت که آن را آستانه[1]می‌نامند و بستگی به اندیس‌های x وy دارد، چرخش می‌دهد. به‌عبارت-دیگر برای هر بیت در یک سطر ، اندیس z به اندازه باقی‌مانده اندازه سطر به آستانه تغییر می‌کند. مراحل این الگوریتم به‌صورت زیر است [2].

1. For all z such that 0 ≤ z < w, let A′ [0, 0, z] = A [0, 0, z].
2. Let (x, y) = (1, 0).
3. For t from 0 to 23:
4. for all z such that 0 ≤ z <w, let A′ [x, y, z] = A [x, y, (z – (t + 1) (t + 2)/2) mod w];
let (x, y) = (y, (2x + 3y) mod 5).
Return A′.

مقادیر آستانه‌ها که برای هر سطر محاسبه می‌شود و در قدم ۴ از الگوریتم نشان داده شده است، در جدول (۱) قرار دارد.

(جدول-۱): مقادیر آستانه‌ها [2]

	x = 3	x = 4	x= 0	x = 1	x = 2
y = 2	153	231	3	10	171
y = 1	55	276	36	300	6
y = 0	28	91	0	1	190
y = 4	120	78	210	66	253
y = 3	21	136	105	45	15



(شکل-۴): ساختار نگاشت ρ [۲]

[1] Offset

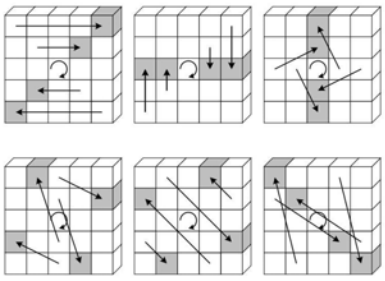
۳-۲- نگاشت π

تأثیر نگاشتπ به این‌صورت است که مکان سطرهای موجود در هر قطعه[2] (شکل۲) را تغییر می‌دهد. الگوریتم این نگاشت به‌صورت زیر است [2].

1. For all triples (x, y, z) such that 0 ≤ x < 5, 0 ≤ y <5, and 0 ≤ z < w, let

$$A'[x.y.z] = A[(x + 3y) mod\ 5.\ x.\ z]$$

Return A′.



(شکل-۵): ساختار نگاشت π [2]

۴-۲- نگاشت χ

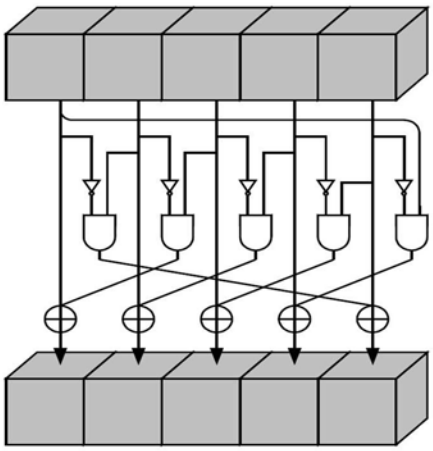
تأثیر نگاشت χ به این‌صورت است که هر بیت به‌وسیلهٔ یک معادله غیرخطی با دو بیت دیگر در سطر[3] (شکل۲) یای انحصاری (XOR) می‌شود [2].

1. For all triples (x, y, z) such that 0 ≤ x <5, 0 ≤ y < 5, and 0 ≤ z < w, let

$$A'\ [x.y.z] = A[x.y.z] \oplus \big((A[(x+1) mod\ 5.y.z] \oplus 1).A[(x+2) mod\ 5.y.z]\big).$$

Return A′.

عملکرد این نگاشت در شکل (۶) آورده شده است.



(شکل-۶): عملکرد نگاشت χ بر روی یک سطر[2]

[2] Slice
[3] Row

۵-۲- نگاشت$_\iota$

تأثیر نگاشت$_\iota$ به این‌صورت است که تعدادی از بیت‌های Lane(0, 0) را که البته به اندیس دور i_r بستگی دارد، تغییر می‌دهد. بیست‌وچهار سطر دیگر توسط این‌نگاشت تغییر نمی‌کند [2].

تابع کچک دارای چهار مد با خروجی ثابت با نام‌های SHA3-224, SHA3-256, SHA3-384 و SHA3-512 است [2]. هم‌چنین دارای دو مد با خروجی متغیر[1] با نام‌های SHAKE128 و SHAKE256 است که مقدار عددی انتهای نام آن‌ها نشان‌دهنده قدرت امنیتی آن‌ها است [2].

۳- حملهٔ مکعبی[2]

حملهٔ مکعبی درواقع یک حمله متن‌انتخابی[3]است که هدف آن الگوریتم‌های کلیددار است. این حمله نخستین‌بار توسط دینور[4] و شامیر[5] در ۲۰۰۸ ارائه شد [18]. به‌طورکلی روش حمله به این‌صورت است که جمع مجموعه‌هایی از بیت‌های خروجی را می‌یابیم که برابر با جمع زیرمجموعه‌هایی از بیت‌های ورودی باشند، به‌طوری که نتایج جمع‌ها، رابطه‌های خطی از بیت‌های خروجی باشند. با تکرار این روش، به مجموعه‌هایی از روابط خطی بین بیت‌های خروجی می‌رسیم و در ادامه با حل روابط خطی به‌دست‌آمده به مقادیر اصلی بیت‌های رمز شده خواهیم رسید.

از زمان معرفی حمله مکعبی تاکنون، بر روی تعدادی از الگوریتم‌های رمزنگاری اعمال شده است [19]-[21] در ادامه روش اعمال حمله مکعبی به‌صورت خلاصه آورده شده است. برای اطلاعات بیشتر در این خصوص به منبع [18] مراجعه نمایید.

در حملهٔ مکعبی فرض می‌شود که بیت‌های خروجی الگوریتم رمز، یک چندجمله‌ای از بیت‌های ورودی(متغیرها) است (معادله ۲) [14].

$$f : X^n \to \{0.1\} \qquad (۲)$$

مشاهدهٔ اصلی در حملهٔ مکعبی زمانی است که چندجمله‌ای یادشده (۲) دارای درجه جبری پایین(d) باشد؛ سپس با جمع خروجی‌های حاصله از 2^{d-1} ورودی، یک تابع خطی حاصل شود؛ به‌طوری‌که زیرمجموعـــه‌ای از ورودی‌ها (برای مثال یک مکعب) با اندازه $d-1$ برای تمام حـالات ممکن برقرار باشد و سایر ورودی‌ها ثابت باشند (نظریه۱).

نظریه ۱:

با داشتن چندجمله‌ای $f : X^n \to \{0.1\}$ از درجه d، فرض کنید که $0 < k < d$ و t یک تک‌جمله‌ای[6] به‌صورت $x_0 \dots x_{k-1}$ باشد، آن‌گاه تابع معادله (۳) برقرار است:

$$f(x) = t \cdot P_t(x) + Q_t(x) \qquad (۳)$$

که در آن هیچ‌کدام از جملات $Q_t(x)$ قابل تقسیم به t نیستند. توجه داشته باشید که درجه P_t در رابطه ۴ صدق می‌کند.

$$P_t \le d - k \qquad (۴)$$

آن‌گاه مجموع f برای تمام مقادیر مکعبی که با t تعریف شده است برابر است با:

$$\sum_{x'=(x_0,\dots,x_{k-1})\in C_t} f(x'.x) = P_t(1.\dots.1(k).x_k.\dots.x_{n-1}) \qquad (۵)$$

که درجه در بیش‌ترین حالت برابر $d-k$ و یا برابر یک اگر $k = d-1$ باشد. هم‌چنین مکعب C_t شامل تمام بردارهای دودویی[7] با طول k است. اثبات این نظریه در منبع [18] آورده شده است.

۳-۱- پیش‌پردازش(مرحله برون‌خط[8])

این‌مرحله برای هر الگوریتم رمز فقط یک‌بار انجام می‌شود و ارتباطی با کلید رمزنگاری ندارد. ابتدا متغیرهای عمومی(متغیرهایی که توسط حمله‌کننده کنترل می‌شوند) برای مثال پیام را با $\upsilon = (v_1.\dots.v_{d-1})$ و متغیرهای کلید رمز با $x = (x_1.\dots.x_n)$ نشان داده می‌شوند. خروجی (بیت‌های رمز، بیت‌های جریان کلید یا بیت‌های چکیده) توسط یک چندجمله‌ای $f(v.x)$ نشان‌داده می‌شود. برای مکعب‌های C_t از معادله ۶ استفاده می‌شود به طوری‌که $L(x)$ یک فوق چندجمله‌ای[9] از C_t باشد [14].

---

[1] Extendable-output function(XOF)
[2] Cube attack
[3] Chosen plaintext
[4] Dinur
[5] Shamir
[6] Monomial
[7] Binary Vectors
[8] Offline Phase
[9] Super polynomial

$$\sum_{v\in C_t} f(v.x) = L(x) \quad (۶)$$

با درنظرگرفتن درجهٔ d برای $f(v.x)$ با توجه به مشاهدات می‌توان نوشت:

$$L(x) = a_1x_1 + \cdots + a_nx_n + c \quad (۷)$$

در حمله مکعبی در مرحله پیش‌پردازش، فوق چندجمله‌ای‌های خطی $L(x)$ یافت می‌شود، که درنهایت به حمله‌کنندگان کمک می‌کند تا معادله‌های خطی را بر اساس متغیرهای رمز بسازند. درنهایت ضرایب خطی $L(x)$ را به‌صورت زیر جای‌گذاری می‌کنند [14].

ثابت c که با استفاده از رابطهٔ ۸ محاسبه می‌شود:

$$c = \sum_{v\in C_t} f(v.0) \quad (۸)$$

- a_i که با استفاده از رابطهٔ ۹ محاسبه می‌شود:

$$a_i = \sum_{v\in C_t} f(v.0.\ldots.1(x_i).0.\ldots.0) = a_i \quad (۹)$$

توجه داشته باشید که در بیشتر موارد، جزئیات $f(v.x)$ نامشخص است و باید درجه آن(d) را در یک مرحله جداگانه و پیچیده به‌دست آورد. این مرحله با بررسی مکعب‌هایی با ابعاد مختلف انجام می‌شود [14].

۳-۲- مرحله برخط[1]

مرحله برخط بعد از مشخص‌شدن کلید رمز یا کلید تابع چکیده‌ساز در حالت MAC انجام می‌شود. در این‌مرحله، حمله‌کننده می‌تواند مقادیر مختلفی را برای متغیر عمومی v انتخاب‌کند. برای هر مکعب C_t ، حمله‌کننده مقدار دودویی b_t را با عملیات جمع بر روی مکعب C_t به‌دست می‌آورد به‌عبارت دیگر:

$$\sum_{v\in C_t} f(v.x) = b_t \cdot \quad (۱۰)$$

برای مکعب C_t داده شده، b_t برابر عبارت خطی $L(x)$ است که در قسمت ۳-۱ به‌دست آمده بود؛ بنابراین یک معادله خطی به‌صورت زیر حاصل خواهد شد:

$$a_1x_1 + \cdots + a_nx_n + c = b_t \cdot \quad (۱۱)$$

با درنظرگرفتن تعدادی مکعب C_t، حمله‌کننده می‌تواند تعداد کافی معادله‌خطی ایجاد کند. اگر تعداد معادلات‌خطی غیروابسته برابر متغیرهای مجهول(رمز) باشد، دستگاه با حذف گوسی[2] قابل حل است [14]. در قسمت بعد یک نمونه از حمله مکعبی اعمال‌شده بر روی کچک در حالت MAC برای بازیابی‌کلید شرح داده خواهد شد.

۳-۳- حمله مکعبی بر روی حالت‌کلیددار SHA-3 (MAC) تا پنج دور

مرحلهٔ پیش‌پردازش:

درجه جبری هر دور از تابع کچک دو است. بنابراین بعد از پنج دور درجه جبری $2^5 = 32$ خواهد بود و برای هر مکعب با ۳۱ متغیر، فوق چندجمله‌ای فقط شامل عبارات خطی خواهد بود. بنابراین می‌توان مرحله آزمون فوق چندجمله‌ای برای وجود عبارات خطی را انجام نداد. باید دقت شود که فوق چندجمله‌ای می‌تواند مقادیر ثابت باشد که برای بازیابی کلید مناسب نیستند؛ زیرا ثابت‌ها شامل اطلاعاتی از کلید نیستند [14].

برای بهتر پیداکردن مکعب‌هایی‌که به بازیابی‌کلید کمک می‌کنند از روش زیر استفاده شده‌است. ابتدا به صورت تصادفی ۳۱ متغیر از ۱۲۸ متغیر عمومی انتخاب و سپس بررسی می‌شود که مکعب شامل متغیرهای رمز است و یا مقدار ثابت دارند. با چند روز محاسبه بر روی یک سامانه ساده می‌توان ۱۱۷ فوق چندجمله‌ای مناسب یافت. البته این زمان قابل بهبود است [14].

مرحلهٔ برخط:

در مرحله برخط، حمله‌کننده مقدار دودویی واقعی فوق چندجمله ای را با جمع مکعب‌های به‌دست‌آمده در خروجی محاسبه می‌کند. در این حمله از نوزده مکعب که هر مکعب دارای ۳۱ متغیر است، استفاده شده است. بنابراین حمله‌کننده $19 \cdot 2^{31} \cong 2^{35}$ خروجی برای پنج دور از تابع کچک را به‌دست خواهد آورد [14]. با محاسبه مقادیر فوق چندجمله‌ای، حمله‌کننده ۱۱۷ رابطه خطی غیروابسته را ساخته و ۱۲۸ بیت کلید رمز را بازیابی می‌کند (با حدس‌زدن مقادیر ۱۱ رابطه خطی غیروابسته). در کل پیچیدگی مرحله برخط 2^{35} خواهد بود [14].

۳-۴- نتایج

در منبع [۱۴]، حمله مکعبی با روش‌های جبری‌ترکیب و بر

[1] Online Phase

[2] Gaussian Elimination

روی کچک در حالت کلیددار (MAC) اعمال شده است. هم‌چنین حملات انجام‌شده بر روی حالت پایه که توسط طراح پیشنهاد شده انجام شده است که از نوع حمله متن انتخابی است. درواقع حمله‌کننده می‌تواند به تعداد دلخواه پیام و خروجی معادل آن را در اختیار داشته باشد. تحلیل موجود در منبع [۱۴] به سه قسمت تقسیم می‌شود. در قسمت نخست بررسی امنیتی حالت کلیددار (MAC) تابع کچک در برابر حمله مکعبی و موفقیت تا شش دور انجام شده و با رایانه‌های شخصی پیاده‌سازی شده است. در قسمت دوم به بررسی امنیتی حالات کلیددار (MAC) تابع کچک در مقابل آزمون-کننده‌های مکعبی پرداخته شده است. برخلاف حمله مکعبی آزمون‌کننده‌های مکعبی کلید را بازیابی نمی‌کنند و در عوض به‌دنبال یافتن رفتارهای غیرتصادفی هستند [۱۴]. جزئیات بیشتر در منبع [۱۹] آورده شده است. در قسمت آخر حملات بازبینی شده و نشان داده شده است که چطور کلید تابع کچک را تا هفت دور با پیچیدگی زمانی کمتر از جستجوی کامل بازیابی شود. نتایج حملات مکعبی در جدول (۲) آورده شده است.

(جدول-۲): نتایج حملات مکعبی [14]

مد[1]	تعداد دور	نوع حمله	پیچیدگی عمومی	پیچیدگی حمله
MAC	۵	بازیابی کلید[2]	$۲^{۱۲۸}$	$۲^{۳۶}$
MAC	۶	بازیابی کلید	$۲^{۱۲۸}$	$۲^{۳۶}$
MAC	۷	بازیابی کلید	$۲^{۱۲۸}$	$۲^{۹۷}$
MAC	۷	جعل[3]	$۲^{۱۲۸}$	$۲^{۶۵}$
MAC	۸	جعل	$۲^{۲۵۶}$	$۲^{۱۲۹}$
AE (Keyak[22])	۶	بازیابی کلید	$۲^{۱۲۸}$	$۲^{۳۶}$
AE (Keyak[22])	۷	بازیابی کلید	$۲^{۱۲۸}$	$۲^{۷۶}$
AE (Keyak[22])	۷	جعل	$۲^{۱۲۸}$	$۲^{۶۵}$
Stream Cipher	۶	بازیابی کلید	$۲^{۱۲۸}$	$۲^{۳۵}$
Stream Cipher	۸	پیش‌بینی جریان کلید[4]	$۲^{۲۵۶}$	$۲^{۱۲۸}$
Stream Cipher	۹	پیش‌بینی جریان کلید	$۲^{۵۱۲}$	$۲^{۲۵۶}$

با اجرای حملات مکعبی اخیر به‌نظر می‌رسد از حاشیه امنیتی تابع کچک در حالت کلیددار (MAC) کاسته شده است. البته نسبت به ۲۴ دور حالت استاندارد تابع کچک هنوز حاشیه امنیتی زیادی وجود دارد [14].

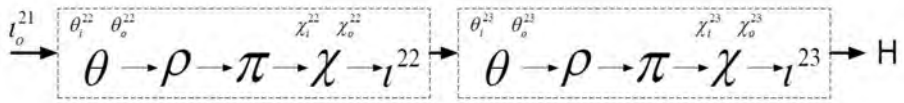
۴- حمله تحلیل خطای تفاضلی[5]

در این قسمت به توضیح حمله تحلیل خطای تفاضلی برعلیه تابع چکیده‌ساز کچک در حالت کلیددار (MAC) بر مبنای منابع [15] و [16] پرداخته می‌شود. در ادامه برای راحتی آن‌ها را حملهٔ نخست [15] و حمله دوم [16] می‌نامیم. حمله خطای تفاضلی نخستین‌بار توسط بیهام[6] و شامیر معرفی شد [23]. آن‌ها در سال ۱۹۹۰این حمله را بر روی DES[7] پیاده کردند و توانستند با استفاده از ۵۰ تا ۲۰۰ متن ساده و رمزشده، کلیدکامل DES را استخراج کنند. هم‌چنین بیان کردند که اگر این حمله بر روی ۳DES نیز انجام گیرد، پیچیدگی آن ثابت خواهد بود. حمله تفاضلی توانایی شکستن بسیاری از الگوریتم‌ها را دارد. نکته قابل‌توجه درباره حمله تحلیل خطای تفاضلی این است که درصورتی‌که تفاوت‌های کمی در مدل‌های القای خطا وجود داشته باشد t پیچیدگی حمله به‌صورت قابل‌توجهی افزایش می‌یابد [23]. این حمله از نوع حملات متن رمزشده انتخابی[8] است و در آن جفت‌متن‌های رمزشده به‌نحوی انتخاب می‌شوند که یای انحصاری (XOR) متن‌های متناظر اصلی‌شان دارای مقدارخاصی باشد [24].

در متداول‌ترین حملات القای خطای تفاضلی در حالت پیش‌فرض مکان و مقدار خطا مشخص نیست. در توابع چکیده‌ساز، حمله تفاضلی زمانی معنا پیدا می‌کند که تابع چکیده‌ساز از نوع IV-MAC[26],HMAC[25], NMAC [25] باشد. در این حالت حمله تفاضلی می‌تواند برای بازسازی کلید رمزنگاری و یا جعل‌کردن مقدار کد احراز اصالت پیام[9] استفاده شود [15]. از آن‌جا که مدل حمله تفاضلی در هر دو مقاله بر روی دو دور آخر است؛ لذا در ادامه از نام‌گذاری‌های شکل (۷) استفاده خواهیم کرد.

[1] Mode
[2] Key Recovery
[3] Forgery
[4] Keystream Prediction

[5] Differential Fault Analysis (DFA)
[6] Biham
[7] Data Encryption Standard
[8] Chosen Cipher Text Attack
[9] Message Authentication Code(MAC)



(شکل-۷): نامگذاری دو دور آخر [۱۶]

۴-۱- مقایسهٔ روش‌های حمله تفاضلی

تفاوت اصلی بین دو مقاله [15]،[16] در این است که در حمله نخست یک بیت خطا و در حمله دوم یک بایت خطا اعمال می‌شود. به‌نظر می‌رسد با توجه به روش‌های سخت‌افزاری اعمال خطا از قبیل خطای پالس[1] و تغییرات ولتاژ که بر روی گروهی از بیت‌ها تأثیرمی‌گذارند، اعمال یک بیت خطا کمی غیرواقعی به‌نظر برسد و حالت یک بایت خطا واقعی‌تر است [16]. هم‌چنین حمله نخست بر روی حالت‌های SHA3-384 و SHA3-512 اعمال شده است، درصورتی‌که تمرکز حمله دوم بر روی حالت‌های SHA3-224 و SHA3-256 است. در حالت‌های SHA3-224 و SHA3-256 از آن‌جا که حمله‌کننده یک صفحه‌کامل (شکل ۲) را در خروجی ندارد، اعمال حمله دشوارتر خواهد بود.

۴-۲- مقدمات حمله

اگر نگاشت‌های تابع کچک را بر اساس خطی[2]بودن و تغییر بیت‌ها دسته‌بندی‌کنیم به جدول (۳) خواهیم رسید.

(جدول-۳): مقایسه جایگشت‌ها بر اساس ویژگی‌ها

نگاشت	θ	ρ	π	χ	ι
خطی	✓	✓	✓		✓
تغییر بیت	✓			✓	

در حمله نخست با بررسی جایگشت χ مشاهدات زیر به‌دست آمده است که اساس حمله در هر دو مقاله است.

مشاهده ۱:

فرض کنیم یک بیت خطا در ورودی نگاشت χ ایجاد شود. با محاسبه تفاضل ورودی و خروجی به‌راحتی می‌توان به دو بیت از ورودی رسید. این نتیجه به‌علت عملیات بیتی"و "(AND) است که با ایجاد خطا باعث بروز اطلاعات از ورودی خود می‌شود [15].

مشاهده ۲:

با داشتن هر مقدار ورودی از نگاشتχ به یک مجموعه روابط خطی از ورودی در خروجی نگاشت خواهیم رسید. اگر x_i را ورودی نگاشتχ و y_i را خروجی نگاشتχ در نظر بگیریم با داشتن یک بیت از ورودی به جدول (۴) خواهیم‌رسید [15].

(جدول -۴): روابط بین ورودی و خروجی‌های نگاشتχ با دانستن یک بیت از دو ورودی [15]

x_0	x_1	x_2	x_3	x_4	y_0	y_1	y_2	y_3	y_4
x_0	x_1	x_2	x_3	0	NL	NL	x_2	$x_3 \oplus x_0$	NL
x_0	x_1	x_2	x_3	1	NL	NL	$x_2 \oplus \bar{x}_3$	x_3	NL

مشاهده ۳

با فرض این‌که تعدادی از بیت‌های ورودی χ را می‌دانیم، خروجی نگاشتχ غیرخطی است اگر و فقط اگر دو بیت پشت سرهم از ورودی نامعلوم باشد. این قانون به‌طور مستقیم از رابطه محاسبه نگاشت χ یعنی:

$$y_i = x_i \oplus (\overline{x_{i+1}} \& x_{i+2}) \qquad (۱۲)$$

به‌دست آمده است. در این رابطه تنها قسمت غیرخطی نگاشتAND است و چون از دو بیت پی‌درپی در محاسبه آن استفاده شده است، قانون بالا اثبات خواهد شد. برای نمونه یک مثال در جدول (۵) آورده شده است [15].

(جدول-۵): روابط بین ورودی و خروجی‌های نگاشت χ با فرض نامشخص بودن x_0 و x_2 [15]

x_0	x_1	x_2	x_3	x_4	y_0	y_1	y_2	y_3	y_4
x_0	0	x_2	0	0	$x_0 \oplus x_2$	0	x_2	x_0	0
x_0	0	x_2	0	1	$x_0 \oplus x_2$	0	$\bar{x}_2$	0	1
x_0	0	x_2	1	0	$x_0 \oplus x_2$	$\bar{x}_2$	x_2	$\bar{x}_0$	0
x_0	0	x_2	1	1	$x_0 \oplus x_2$	$\bar{x}_2$	x_2	1	1
x_0	1	x_2	0	0	x_0	1	x_2	x_0	$\bar{x}_0$
x_0	1	x_2	0	1	x_0	1	$\bar{x}_2$	0	$\bar{x}_0$
x_0	1	x_2	1	0	x_0	$\bar{x}_2$	x_2	$\bar{x}_0$	$\bar{x}_0$
x_0	1	x_2	1	1	x_0	$\bar{x}_2$	x_2	1	$\bar{x}_0$

۴-۳- مدل حمله

مدل حمله تفاضلی در هر دو مقاله به‌طور کلی شامل موارد زیر است:

- اعمال خطا در ابتدای دور ۲۲(θ_i^{22})
- محاسبه مسیر تفاضلی(عبور بیت‌های ماتریس تـفاضل از

[1] Clock glitch
[2] Linear

نگاشت‌های کچک) ماتریس حالت تا نقطه مقایسه در θ_o^{23} در منبع [15] و χ_i^{23} در منبع [16]

- بازگرداندن مقادیر خروجی تا نقطه مقایسه
- یافتن محل اعمال خطا
- بازیابی بیت‌های ماتریس حالت در ابتدای دور ۲۲
- اعمال خطا در دور آخر برای بهبود نتایج (فقط در حمله دوم)

۴-۴- توضیح حمله

با توجه به مدل حمله ابتدا خطا در ابتدای دور ۲۲ تزریق می‌شود. در حمله نخست چون مقدار خطا یک بیت است، در کل ۱۶۰۰ مکان یا حالت برای خطا وجود دارد؛ درصورتی‌که در حمله دوم چون مقدار خطا یک بایت است شامل دویست مکان و هر مکان ۲۵۶ حالت و در کل ۵۱۲۰۰ حالت خطا وجود خواهد داشت.

با استفاده از مشاهده یک و با توجه به جدول (۳) با عبور خطا از جایگشت‌ها و با بازنویسی جایگشت‌های χ و θ الگوهای خطا در نقاط مقایسه به‌دست خواهند آمد. با درنظرگرفتن نقطه مقایسه در θ_o^{23} ما الگوهای یک بیت و یک بایت خطا را دوباره پیاده‌سازی‌کرده و به خروجی‌های شکل (۸) و شکل (۹) رسیده‌ایم. در شکل‌های (۸) و (۹)، ۱۶۰۰ بیت ماتریس حالت که هر سطر نمایان‌گر یک سطر است، آورده شده است. نام هر سطر بر اساس $[x.y]$ نشان داده شده است.

```
theta_f_23
[0,0]---> xx100000 00xx0001 00000x10 0000x000 00000000 0x00x1x0 00000000 00000000
[0,1]---> xx100000 00xx0001 00000x10 0000x000 00000000 0x00x1x0 00000000 00000000
[0,2]---> xx100000 00xx0001 00000x10 0000x000 00000000 0x00x1x0 00000000 00000000
[0,3]---> xx100000 00xx0001 00000x10 0000x000 00000000 0x00x1x0 00000000 00000000
[0,4]---> xx100000 00xx0001 00000x10 0000x000 00000000 xx00x1x0 00000000 00000000
[1,0]---> 110x0000 0xx10000 x0000x00 00001000 00000000 x100x0x0 00000000 00000000
[1,1]---> 110x0000 00x10000 x0000x00 00001000 00000000 x100xxx0 00000000 00000000
[1,2]---> 110x0000 0xx10000 x0000x00 00001000 00000000 x100x0x0 00000000 00000000
[1,3]---> 110x0000 00x10000 x0000x00 00001000 00000000 x100x0x0 00000000 00000000
[1,4]---> 110x0000 00x10000 x0000x00 00001000 00000000 x100x0x0 00000000 00000000
[2,0]---> 0xxx0000 0xx0000x x0000100 00000x00 00000000 x0001x10 00000000 00000000
[2,1]---> 0xxx0000 0xx00000 x0000100 00000x00 00000000 x0001x10 00000000 00000000
[2,2]---> 0xxx0000 0xx00000 x0000100 00000x00 00000000 x0001x10 00000000 00000000
[2,3]---> 0xxx0000 0xx00000 x0000100 00000x00 00000000 x0001x10 00000000 00000000
[2,4]---> 0xxx0000 0xx00000 x0000100 00000x00 00000000 x0001x10 00000000 00000000
[3,0]---> xxx10000 0x10000x 100000x0 00000x00 00000000 10000x00 00000000 00000000
[3,1]---> 0xx10000 0x10000x 100000x0 00000x00 00000000 10000x00 00000000 00000000
[3,2]---> 0xx10000 0x10000x 100000x0 00000x00 00000000 10000x00 00000000 00000000
[3,3]---> 0xx10000 0x10000x 100000x0 0000xx00 00000000 10000x00 00000000 00000000
[3,4]---> 0xx10000 0x10000x 100000x0 00000x00 00000000 10000x00 00000000 00000000
[4,0]---> xxx00000 010x000x 000000x0 0000x100 00000000 0x00xx00 00000000 00000000
[4,1]---> xxx00000 010x000x 00000xx0 0000x100 00000000 0x000x00 00000000 00000000
[4,2]---> xxx00000 010x000x 000000x0 0000x100 00000000 0x000x00 00000000 00000000
[4,3]---> xxx00000 010x000x 000000x0 0000x100 00000000 0x000x00 00000000 00000000
[4,4]---> xxx00000 010x000x 000000x0 0000x100 00000000 0x000x00 00000000 00000000
```

(شکل-۸): الگوی خطای یک بیت(خطا در بیت A[0,0,0])

در این الگوها سه نوع بیت خواهیم داشت، بیت‌هایی با مقدار 1 که همان بیت‌های فعال[1] می‌باشند، بیت‌هایی با مقدار0 و بیت‌هایی با مقدار X که بیت‌هایی‌می‌باشند که با توجه به فرمول جایگشتχ به مقدار ورودی جایگشت وابسته هستند.

```
theta_f_23
[0,0]---> xxxxx100 00xxxxx1 11100xxx x100xxxx 00000000 0xxxxxxx xx000000 00000000
[0,1]---> xxxxx100 00xxxxx1 11100xxx x100xxxx 00000000 0xxxxxxx xx000000 00000000
[0,2]---> xxxxx100 00xxxxx1 11100xxx x100xxxx 00000000 0xxxxxxx xx000000 00000000
[0,3]---> xxxxx100 00xxxxx1 11100xxx x100xxxx 00000000 0xxxxxxx xx000000 00000000
[0,4]---> xxxxx100 00xxxxx1 11100xxx x100xxxx 00000000 xxxxxxxx xx000000 00000000
[1,0]---> 100xxxx0 00xxxx10 xxxx0xxx x0001111 00000000 xxxxxxxx xx000000 00000000
[1,1]---> 100xxxx0 00xxxx10 xxxx0xxx x0001111 00000000 xxxxxxxx xx000000 00000000
[1,2]---> 100xxxx0 0xxxxx10 xxxx0xxx x0001111 00000000 xxxxxxxx xx000000 00000000
[1,3]---> 100xxxx0 00xxxx10 xxxx0xxx x0001111 00000000 xxxxxxxx xx000000 00000000
[1,4]---> 100xxxx0 00xxxx10 xxxx0xxx x0001111 00000000 xxxxxxxx xx000000 00000000
[2,0]---> 0xxxxxx0 0xxxxx0x xxxx0111 10000xxx x0000000 xxxx1xxx x1000000 00000000
[2,1]---> 0xxxxxx0 0xxxxx00 xxxx0111 10000xxx x0000000 xxxx1xxx x1000000 00000000
[2,2]---> 0xxxxxx0 0xxxxx00 xxxx0111 10000xxx x0000000 xxxx1xxx x1000000 00000000
[2,3]---> 0xxxxxx0 0xxxxx00 xxxx0111 10000xxx x0000000 xxxx1xxx x1000000 00000000
[2,4]---> 0xxxxxx0 0xxxxx00 xxxx0111 10000xxx x0000000 xxxx1xxx x1000000 00000000
[3,0]---> xxxxxx10 0xxxx10x xxx100xx xx000xxx x0000000 11110xxx x0000000 00000000
[3,1]---> 0xxxxx10 0xxxx10x xxx100xx xx000xxx x0000000 11110xxx x0000000 00000000
[3,2]---> 0xxxxx10 0xxxx10x xxx100xx xx000xxx x0000000 11110xxx x0000000 00000000
[3,3]---> 0xxxxx10 0xxxx10x xxx100xx xx00xxxx x0000000 11110xxx x0000000 00000000
[3,4]---> 0xxxxx10 0xxxx10x xxx100xx xx000xxx x0000000 11110xxx x0000000 00000000
[4,0]---> xxxxxx00 011xxxxx xxx000xx xx00xxxx 10000000 0xxxxxxx x0000000 00000000
[4,1]---> xxxxxx00 011xxxxx xxx00xxx xx00xxxx 10000000 0xxxxxxx x0000000 00000000
[4,2]---> xxxxxx00 011xxxxx xxx000xx xx00xxxx 10000000 0xxxxxxx x0000000 00000000
[4,3]---> xxxxxx00 011xxxxx xxx000xx xx00xxxx 10000000 0xxxxxxx x0000000 00000000
[4,4]---> xxxxxx00 011xxxxx xxx000xx xx00xxxx 10000000 0xxxxxxx x0000000 00000000
```

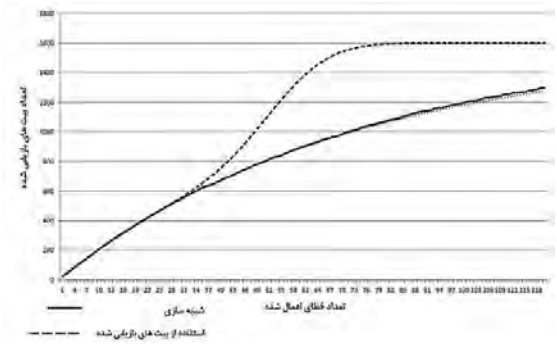
(شکل-۹): الگوی خطای یک بایت (خطا در بایت اول با مقدار[1,1,1,1,0,0,0,0])

برای بازگرداندن بیت‌های خروجی تا نقطه مقایسه در حمله نخست با توجه به معکوس‌پذیربودن جایگشت‌ها تا نقطه مقایسه و داشتن یک صفحه کامل (شکل۲) به‌راحتی قابل انجام است؛ اما در حمله دوم، کار کمی متفاوت‌تر است. در SHA3-256 بیت‌های سطر آخر و در SHA3-224 بیت‌های یک‌ونیم سطر آخر نامشخص است. در کل برای بازگرداندن سطرهای صفحه آخر، حمله‌کننده می‌تواند بیت‌های سطرهای نخست و دوم را با احتمال 75% و بیت‌های سطرهای سوم و چهارم را با احتمال 50% بازیابی‌کند [16] و بیت‌های سطرآخر نامعلوم خواهندماند. در نهایت به‌صورت نظری در حالت SHA3-256 حمله‌کننده قادر به بازیابی ۱۶۰ بیت و در حالت SHA3-224 ۱۲۰ بیت را بازیابی خواهد کرد [16]. با مقایسه الگوهای خطا برای تمام حالات و هم‌چنین خروجی‌های بازگردانده‌شده در نقطه مقایسه می‌توان محل و مقدار خطا را مشخص کرد. بنابر ادعای طراحان، در حمله نخست احتمال یافتن محل خطا، ۱۰۰ درصد ولی در حمله دوم در بدترین حالت ۶۶ درصد است که دلیل آن هم تعداد کمتر بیت‌های بازگردانده‌شده و هم‌چنین افزایش تعداد حالات ممکن است، خطاهایی را که در محدوده این ۶۶ درصد قرار می‌گیرند را خطاهای مؤثر می‌نامیم [16].

بعد از مشخص‌شدن مکان و مقدار خطا با استفاده از بیت‌های فعال (1) قادر به بازیابی بیت‌های ماتریس حالت در ابتدای دور ۲۲ خواهیم بود. در حالت القاء یک بیت خطا چون در نقطه مقایسه به‌طور دقیق ۲۲ بیت وجود دارد، قادر به بازیابی ۲۲ بیت از حالت درونی برای هر خطا خواهیم بود. از

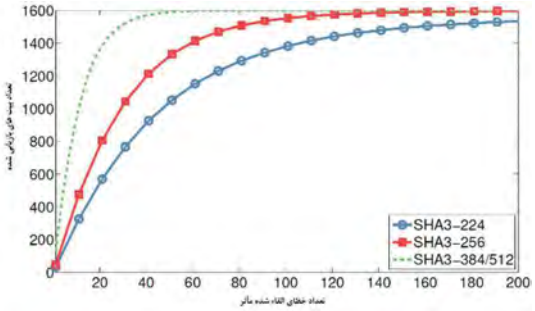
[1] Active Bits

آن‌جا که بعد از اعمال خطاهای متعدد، احتمال تکراری بودن بیت‌های بازیابی‌شده بیشتر است، بعد از مدتی با استفاده از بیت‌های بازیابی‌شده و با توجه به مشاهده ۲ و ۳ سرعت بازیابی افزایش خواهد یافت. این مهم در شکل (۱۰) آورده شده است. در کل در حمله نخست با اعمال ۸۰ خطا قادر به بازیابی ۱۵۹۲ بیت از ماتریس حالت خواهیم بود.



(شکل-۱۰): تعداد بیت‌های بازیابی شده بر اساس تعداد خطا [15]

در حالت القای یک بایت خطا از آن‌جا که به‌طور میانگین بیت‌های فعال بیشتری تولید می‌شود، سرعت بازیابی بیت‌ها به‌ازای خطاهای اعمال‌شده بیشتر است. در حمله دوم برای بهبود نتیجه، حمله‌کننده در دور آخر نیز خطا القا کرده است. نتایج به‌دست‌آمده در مقاله دوم بعد از بهبود نتیجه، در شکل (۱۱) آمده است [16].



(شکل-۱۱): تعداد بیت‌های بازیابی‌شده بر اساس تعداد خطا[16]

۵- اعمال ساختارهای خطی[1] روی تابع کچک

در منبع [17] روش جدیدی با عنوان ساختار خطی مطرح شده است. این روش باعث خطی‌شدن جایگشت χ کچک تا سه دور خواهد شد. در این منبع در ادامه با استفاده از روش ساختار خطی دو حمله پیش‌تصویر[2] و تشخیص‌دهنده‌های مجموع صفر[3](به‌اختصار تشخیص‌دهنده) بر روی کچک اعمال شده است و به‌عنوان بهترین نتیجه موفق به یافتن پیش‌تصویر بر روی سه دور از SHAKE128 با پیچیدگی زمانی ۱ شده است.

۵-۱- تحلیل ویژگی‌های نگاشت χ

روابط موجود بین پنج بیت ورودی و خروجی جایگشت χ در زیر آورده شده است:

$$b_0 = a_0 \oplus (a_1 \oplus 1) \cdot a_2. \quad (۱۳)$$

$$b_1 = a_1 \oplus (a_2 \oplus 1) \cdot a_3. \quad (۱۴)$$

$$b_2 = a_2 \oplus (a_3 \oplus 1) \cdot a_4. \quad (۱۵)$$

$$b_3 = a_3 \oplus (a_4 \oplus 1) \cdot a_0. \quad (۱۶)$$

$$b_4 = a_4 \oplus (a_0 \oplus 1) \cdot a_1 \cdot \quad (۱۷)$$

حال اگر دو بیت متوالی از خروجی در دسترس باشد، می‌توان یک رابطه خطی بر اساس بیت‌های ورودی به‌دست آورد. برای مثال فرض کنید که b_0 و b_1 در دست است، آن‌گاه بر اساس رابطه (۱۴) خواهیم داشت:

$$b_1 \cdot a_2 = (a_1 \oplus (a_2 \oplus 1) \cdot a_3) \cdot a_2 \quad (۱۸)$$

و بر اساس رابطه (۱۳) خواهیم داشت:

$$b_0 = a_0 \oplus (b_1 \oplus 1) \cdot a_2 \cdot \quad (۱۹)$$

هم‌چنین اگر سه بیت متوالی b_0، b_1 و b_2 را داشته باشیم، رابطه خطی بیست به روابط قبلی اضافه خواهد شد.

$$b_1 = a_1 \oplus (b_2 \oplus 1) \cdot a_3 \cdot \quad (۲۰)$$

رابطه تعداد معادلات خطی و بیت‌های متوالی قابل مشاهده از خروجی در جدول (۶) آورده شده است.

(جدول-۶): تعداد معادلات خطی به‌دست‌آمده از خروجی جایگشت χ [17]

تعداد بیت‌های متوالی قابل مشاهده از خروجی	۲	۳	۴	۵
تعداد رابطه‌های خطی	۱	۲	۴	۵

هم‌چنین برای افزایش روابط خطی می‌توان از دو روش دیگر برای زمانی که کمتر از چهار بیت از خروجی در دسترس است، نیز استفاده کرد. در روش نخست می‌توان یک بیت از ورودی را حدس زد. در این حالات به دو رابطه خطی خواهیم رسید. از طرفی باید توجه داشت که هزینه محاسباتی دو برابر خواهد شد. در روش دوم از احتمالات استفاده شده است. برای مثال رابطه $b_i = a_i$ با احتمال ۷۵ درصد برقرار است [17].

[1] Linear Structures
[2] Pre-image
[3] Zero-sum Distinguisher

خطی‌سازی برای حالت معکوس نگاشت χ نیز قابل انجام است. درجه جبری حالات معکوس نگاشت χ برابر سه است. در زیر روابط حالت معکوس نگاشت χ آورده شده است. برای جزئیات بیشتر به منبع [17] رجوع شود.

$$a_0 = b_0 \oplus (b_1 \oplus 1) \cdot (b_2 \oplus (b_3 \oplus 1) \cdot b_4). \quad (۲۱)$$
$$a_1 = b_1 \oplus (b_2 \oplus 1) \cdot (b_3 \oplus (b_4 \oplus 1) \cdot b_0). \quad (۲۲)$$
$$a_2 = b_2 \oplus (b_3 \oplus 1) \cdot (b_4 \oplus (b_0 \oplus 1) \cdot b_1). \quad (۲۳)$$
$$a_3 = b_3 \oplus (b_4 \oplus 1) \cdot (b_0 \oplus (b_1 \oplus 1) \cdot b_2). \quad (۲۴)$$
$$a_4 = b_4 \oplus (b_0 \oplus 1) \cdot (b_1 \oplus (b_2 \oplus 1) \cdot b_3). \quad (۲۵)$$

۵-۲- اعمال ساختار خطی روی کچک

با استفاده از نتایج به‌دست‌آمده در تحلیل‌های قسمت ۵-۱ و با افزودن یک مقدار ثابت به یک ستون[1] از ماتریس حالت می‌توان دو دور از تابع کچک را خطی نگاه داشت. هم‌چنین با روشی مشابه می‌توان سه دور از تابع کچک را هم خطی نگاه داشت. جزئیات بیشتر در منبع [17] آورده شده است. توجه داشته باشید که تنها نگاشت غیرخطی در هر دور از تابع کچک نگاشت χ است (جدول۳) که درجه جبری آن از ورودی به سمت خروجی برابر دو و برای حالت معکوس برابر سه است [17].

۵-۳- استفاده از ساختار خطی در حمله تشخیص‌دهنده

تشخیص‌دهنده‌ها بر روی یک تابع درواقع روشی است که برای مجموعه‌ای از بیت‌ها با جمع صفر در جای دیگر نیز دارای جمعی برابر صفر باشند. به‌عبارت دیگر برای مجموعه S و تابع f رابطه ۲۶ برقرار باشد [17].

$$\sum_{x \in S} x = 0 \rightarrow \sum_{x \in S} f(x) = 0 \quad (۲۶)$$

بر اساس روش شناخته‌شده در حمله تشخیص‌دهنده با افزودن یک دور به تابع کچک درجه جبری برای به‌دست‌آوردن خروجی‌ها از ورودی دو برابر و در حالت معکوس یعنی به‌دست‌آوردن ورودی از خروجی‌ها درجه جبری سه برابر خواهد شد [17]. حمله تشخیص‌دهنده از دورهای میانی تابع کچک شروع شده و در جهت جلو n دور و در جهت برعکس m دور گسترش می‌یابد. درواقع حمله در دو جهت گسترش می‌یابد تا بتواند کمترین محدوده عملیاتی را پیدا کند [17].

[1] Column

در کارهای منتشرشده قبل از منبع [17] ثابت شده بود که می‌توان تا یک دور را خطی نگاه داشت. با توجه به قسمت ۵-۲ می‌توان به خطی‌سازی تا سه دور هم دست یافت [17]. درواقع با ثابت‌نگاه‌داشتن پیچیدگی، ساختار خطی برای $(m+3)$ دور قابل انجام است. به‌طور دقیق‌تر $(n+2)$ دور به جلو و $(m+1)$ دور به عقب. درنهایت نتایج به‌دست‌آمده در جدول (۷) نشان داده شده است.

(جدول-۷): نتایج حمله تشخیص‌دهنده بر روی تابع کچک [17]

	بهبود		بهبود بیشتر	
تعداد دور	عقب+جلو	پیچیدگی	عقب+جلو	پیچیدگی
۷	۴+۳	$۲^{۱۰}$	۵+۲	$۲^{۹}$
۸	۵+۳	$۲^{۱۷}$	۵+۳	$۲^{۱۰}$
۹	۵+۴	$۲^{۲۸}$	۶+۳	$۲^{۱۷}$
۱۰	۶+۴	$۲^{۳۳}$	۶+۴	$۲^{۲۸}$
۱۱	۷+۴	$۲^{۶۵}$	۷+۴	$۲^{۳۳}$
۱۲	۷+۵	$۲^{۸۲}$	۸+۴	$۲^{۶۵}$
۱۳	۸+۵	$۲^{۱۲۹}$	۸+۵	$۲^{۸۲}$
۱۴	۸+۶	$۲^{۲۴۴}$	۹+۵	$۲^{۱۲۹}$
۱۵	۹+۶	$۲^{۲۵۷}$		

۵-۴- استفاده از ساختار خطی در حمله پیش‌تصویر

با استفاده از تحلیل بخش ۵-۲ و هم‌چنین خطی‌سازی نگاشت χ حمله پیش‌تصویر بر روی مدهای مختلف تابع کچک اعمال شده است. به‌طور کلی برای یافتن پیش‌تصویر بر روی متن پیام اصلی با طول کمتر از $(r-2)$ بیت، می‌بایست بیت $(r-1)$ را یک قرار داد تا بیت‌های افزوده شده[2] به‌صورت یک قطعه در نظر گرفته شوند [17]. در منبع [17] برای به‌دست‌آوردن پیچیدگی زمانی کمتر، از روش‌های مختلف ساختارهای خطی و هم‌چنین روش‌های مختلف معکوس‌کردن نگاشت χ برای مدهای مختلف تابع کچک استفاده شده است. پایه ایده اصلی در حمله پیش‌تصویر ساخت معادلات خطی و سپس حل آن‌ها است. پیچیدگی در حمله پیش‌تصویر بر اساس تعداد دفعات لازم برای حل معادلات خطی در نظر گرفته شده است. در جدول (۸)، نتایج به‌دست‌آمده از حمله پیش‌تصویر با استفاده از ساختار خطی و در مقایسه با کارهای پیشین آورده شده است [17].

[2] Padding Bit

(جدول-۸): نتایج حمله پیش‌تصویر تا چهار دور در تابع کچک [17]

تعداد دور	مد تابع کچک	پیچیدگی زمانی	منبع
۲	۲۵۶/۲۲۴/۱۲۸	$۲^{۳۳}$	[۲۷]
۲	۲۵۶/۲۲۴/۱۲۸	۱	[۱۷]
۲	۳۸۴	$۲^{۱۲۹}$	[۱۷]
۲	۵۱۲	$۲^{۳۸۴}$	[۱۷]
۳	۱۲۸	$۲^{۲۶.۶}$	[۱۷]
۳	۱۲۸	۱	[۱۷]
۳	۲۲۴	$۲^{۹۷}$	[۱۷]
۳	۲۵۶	$۲^{۱۹۲}$	[۱۷]
۳	۳۸۴	$۲^{۳۲۲}$	[۱۷]
۳	۵۱۲	$۲^{۴۸۲}$	[۱۷]
۳	۵۱۲	$۲^{۵۰۶}$	[۲۸]
۴	۱۲۸	$۲^{۱۰۶}$	[۱۷]
۴	۲۲۴	$۲^{۲۱۳}$	[۱۷]
۴	۲۵۶	$۲^{۲۵۱}$	[۱۷]
۴	۲۵۶/۲۲۴	$۲^{۲۵۲}/۲^{۲۲۱}$	[۲۸]
۴	۵۱۲/۳۸۴	$۲^{۵۰۶}/۲^{۳۷۸}$	[۲۸]

۶- نتیجه‌گیری

در این مقاله ما به بررسی تعدادی از حملات انجام‌شده بر روی تابع چکیده‌ساز کچک پرداختیم. این حملات شامل حمله مکعبی، حمله تفاضلی، حمله پیش‌تصویر و حمله تشخیص‌دهنده‌های مجموع صفر بودند. در حملات پیش‌تصویر و تشخیص‌دهنده‌های مجموع صفر از روش ساختار خطی استفاده شد. در کل، وجه مشترک کلیه حملات بررسی‌شده تمرکز بر نوعی خطی‌سازی تنها جایگشت غیرخطی تابع چکیده‌ساز کچک یعنی χ است.

درنهایت با توجه به ساختار پنج جایگشت درونی تابع کچک (جدول ۴) به‌نظر می‌رسد با تمرکز بر جایگشت غیر خطی χ و تحلیل آن بتوان برای بهبود نتایج تمامی انواع حملات بهره جست. با توجه به بررسی‌های ما تا زمان این مقاله حملات انجام‌شده تا امروز نتوانسته‌اند امنیت تابع کچک را نقض کنند. هم‌چنین فاصله بین تعداد دور شکسته‌شده و تعداد دور کامل از تابع در حالت استاندارد قابل‌توجه است. با انجام تحلیل و تغییر پارامترهای تمامی حملات انجام‌شده فضا برای بهبود نتایج حملات بر روی تابع کچک وجود دارد.

۷-مراجع

[1] http://keccak.noekeon.org, (2016), Last access at 1/2/2017.

[2] The U.S. National Institute of Standards and Technology: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standard, FIPS 202 (5th August 2015), Last access at 1/2/2017.

[3] Aumasson, Jean-Philippe, and Meier, Willi. "Zero-sum distinguishers for reduced Keccak-f and for the core functions of Luffa and Hamsi." *rump session of Cryptographic Hardware and Embedded Systems-CHES* 2009: 67.

[4] Benoît, Olivier, and Peyrin, Thomas. "Side-channel analysis of six SHA-3 candidates." *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer Berlin Heidelberg, 2010.

[5] Boura, Christina, and Canteaut, Anne. "A zero-sum property for the Keccak-f permutation with 18 rounds." *2010 IEEE International Symposium on Information Theory*, (ISIT), 2010, pp. 2488-2492.

[6] Boura, Christina, and Canteaut, Anne. "Zero-sum distinguishers for iterated permutations and application to Keccak-f and Hamsi-256." *International Workshop on Selected Areas in Cryptography*. Springer Berlin Heidelberg, 2010, pp. 1-17.

[7] Boura, Christina, Canteaut, Anne, and De Canniere, Christophe. "Higher-order differential properties of Keccak and Luffa." *International Workshop on Fast Software Encryption*. Springer Berlin Heidelberg, 2011, pp. 252-269.

[8] Das, Sourav, and Meier, Willi. "Differential biases in reduced-round Keccak." *International Conference on Cryptology in Africa*, vol. 8469 of the series Lecture Notes in Computer Science, pp. 69-87.

[9] Dinur, Itai, Dunkelman, Orr, and Shamir, Adi. "New attacks on Keccak-224 and Keccak-256." *Fast Software Encryption*. Springer Berlin Heidelberg, 2012, pp. 442-461.

[10] Duc, Alexandre and et al. "Unaligned rebound attack: application to Keccak." *Fast Software Encryption*. Springer Berlin Heidelberg, 2012, pp. 402-421.

[11] Jean, Jérémy, and Nikolić, Ivica. "Internal differential boomerangs: practical analysis of the round-reduced Keccak-f permutation." *International Workshop on Fast Software Encryption*. Springer Berlin Heidelberg, 2015.

[12] Kölbl, Stefan and et al. "Differential cryptanalysis of Keccak variants." *IMA International Conference on Cryptography and Coding*. Springer Berlin Heidelberg, 2013, pp. 141-157.

[13] Naya-Plasencia, María, Röck, Andrea and Meier, Willi. "Practical analysis of reduced-round keccak." *International Conference on Cryptology in India ,INDOCRYPT 2011,* vol. 7107 of the series Lecture Notes in Computer Science, pp 236-254.

[14] Dinur, Itai and et al. "Cube attacks and cube-attack-like cryptanalysis on the round-reduced Keccak sponge function." *Annual International Conference on the Theory and Applications of Cryptographic Techniques,* EUROCRYPT 2015, vol. 9056 of the series Lecture Notes in Computer Science, pp. 733-761.

[15] Bagheri, Nasour, Ghaedi, Navid and Kumar Sanadhya, Somitra. "Differential fault analysis of SHA-3." *International Conference in Cryptology in India*, INDOCRYPT 2015, vol. 9462 of Lecture Notes in Computer Science, pp. 253-269.

[16] Luo, Pei and et al. "Differential Fault Analysis of SHA3-224 and SHA3-256." *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2016 Workshop on*. IEEE, 2016.

[17] Guo, Jian, Liu, Meicheng and Song, Ling. "Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak." *Advances in Cryptology–ASIACRYPT 2016,* vol. 10031 of Lecture Notes in Computer Science, pp. 249-274.

[18] Dinur, Itai, and Shamir, Adi. "Cube attacks on tweak able black box polynomials." *Annual International Conference on the Theory and Applications of Cryptographic Techniques. EUROCRYPT 2009*, pp. 278-299.

[19] Aumasson, Jean-Philippe and et al. "Cube testers and key recovery attacks on reduced-round MD6 and Trivium." *Fast Software Encryption*, 2009, pp. 1-22.

[20] Bard, Gregory V. and et al. "Algebraic, AIDA/cube and side channel analysis of KATAN family of block ciphers." *International Conference on Cryptology in India, INDOCRYPT 2010,* vol. 6498 of Lecture Notes in Computer Science, pp. 176-196.

[21] Lathrop, Joel. "Cube attacks on cryptographic hash functions." Master's Thesis in Computer Science, (2009).

[22] http://keyak.noekeon.org, (2016), Last access 1/2/2017.

[23] Biham, Eli, and Shamir, Adi. "Differential fault analysis of secret key cryptosystems." *Annual International Cryptology Conference,* CRYPTO '97, 1997, pp. 513-525.

[24] Biham, Eli, and Shamir, Adi. "Differential cryptanalysis of DES-like cryptosystems." *Journal of CRYPTOLOGY*, vol. 4, no. 1, 1991, pp. 3-72.

[25] Bellare, Mihir, Canetti, Ran and Krawczyk, Hugo. "Keying hash functions for message authentication." *Crypto 96 Proceedings*, vol. 1109 of Lecture Notes in Computer Science, 1996.

[26] Preneel, Bart, and van Oorschot, Paul C.. "On the security of iterated message authentication codes." *IEEE Transactions on Information theory*, vol. 45, no.1, pp. 188-199, 1999.

[27] Naya-Plasencia, María, Röck, Andrea, and Meier Willi. "Practical analysis of reduced-round Keccak." *International Conference on Cryptology in India INDOCRYPT 2011*, vol. 7107 of Lecture Notes in Computer Science, pp. 236-254.

[28] Morawiecki, Paweł, Pieprzyk, Josef, and Srebrny, Marian. "Rotational cryptanalysis of round-reduced Keccak." *International Workshop on Fast Software Encryption*. Springer Berlin Heidelberg, 2013.

[29] Bertoni, Guido and et al. "Cryptographic sponges." Submission to NIST (Round 3), *available at: http://sponge. noekeon. org* (2011).

محمدامین ارغوانی تحصیلات خود را در مقاطع کارشناسی و کارشناسی ارشد مهندسی کامپیوتر (نرم‌افزار) به‌ترتیب در سال ۱۳۸۸ در دانشگاه شهید ستاری و در سال ۱۳۹۵ در دانشگاه تربیت دبیر شهید رجایی به پایان رساند. زمینهٔ پژوهشی مورد علاقه وی تحلیل امنیتی الگوریتم‌های رمزنگاری است.

معصومه صفخانی در حال حاضر استادیار دانشکده مهندسی کامپیوتر در دانشگاه تربیت دبیر شهید رجایی است. ایشان نویسنده بیشتر از سی مقاله در زمینه تحلیل و طراحی پروتکل‌های امنیتی و طرح‌های رمزنگاری هستند. در حال حاضر تمرکز ایشان روی امنیت RFID، امنیت اینترنت اشیا و امنیت طرح‌های رمزگذاری قابل جستجو است. اطلاعات بیشتر در نشانی زیر قرار دارد:

http://www.srttu.edu/english-cv-dr-safkhani/