مجموعه مقالات پانزدهمین کنفرانس بین‌المللی انجمن رمز ایران
دانشکده مهندسی برق، دانشگاه تربیت دبیر شهید رجایی، تهران، ۶-۷ شهریور ۱۳۹۷
**(مقاله مدعو)**

# رمزنگاری مقاوم در برابر دستکاری

سید امیر مرتضوی

sa.mortezavi @gmail.com

## چکیده

بررسی امنیت سامانه‌های رمزنگاری در برابر انواع حملات پیاده سازی، شامل حملات فعال و غیرفعال، با توجه به کثرت این نوع از حملات در دنیای واقعی بسیار حائز اهمیت است. از سال‌های 2000 میلادی به بعد رمزنگاران تلاش برای مدل سازی حملات فعال و غیرفعال به پیاده سازی الگوریتم‌های رمزنگاری را آغاز کردند. ابتدا مهاجمینی با توانایی دسترسی به اطلاعات نشتی از مشخصه‌های مخفی سامانه‌های رمزنگاری مورد بررسی قرار گرفتند که تلاشی برای مدل‌سازی حملات کانال جانبی نظیر حملات تحلیل توانی بودند که منجر به *رمزنگاری مقاوم در برابر نشت* شد. سپس حملات دستکاری در مشخصه‌های سامانه‌های رمزنگاری مورد توجه قرار گرفت که منجر به *رمزنگاری مقاوم در برابر دستکاری* شد. یکی از ابزارهای رمزنگاری پیشنهادشده برای مقابله با انواع دستکاری استفاده از کدهای *چکش‌ناپذیر* است که در برابر تغییرات دستکاری مقاوم هستند.

در این سخنرانی کدهای چکش‌ناپذیر و امنیت چکش‌ناپذیری تعریف و بررسی خواهند شد. به عنوان نمونه سامانه کدگذاری با امنیت چکش‌ناپذیری پیوسته FMNV معرفی خواهد شد. سامانه کدگذاری FMNV اولین سامانه کدگذاری با امنیت چکش‌ناپذیری پیوسته است که در آن مهاجم توانایی اجرای حملات دستکاری متعددی را دارد. سامانه‌های کدگذاری با امنیت چکش‌ناپذیری پیوسته می‌توانند با سایر سامانه‌های رمزنگاری ترکیب شده و این سامانه‌ها را در برابر حملات دستکاری به سامانه محافظت کنند.

## کلمات کلیدی

امنیت اثبات‌پذیر، دستکاری، چکش ناپذیری، رمزنگاری مقاوم در برابر دستکاری.

## ۱- مقدمه

ایده سامانه‌های رمزنگاری اثبات‌پذیر بعد از مقاله دیفی هلمن [۲] مطرح شد و انواع مختلف سامانه‌های رمزنگاری با امنیت قابل‌اثبات مطرح شدند. این سامانه‌ها با قبول بعضی فرضیات امنیت سامانه را اثبات می‌کردند ولی هیچ گونه ضمانتی برای امنیت در برابر حملات پیاده‌سازی نداشتند. حملات پیاده‌سازی در دو شکل فعال و غیرفعال بر پیاده‌سازی سامانه‌های رمزنگاری اعمال می‌شود .

حملات غیرفعال نظیر حملات کانال جانبی از ضعف های موجود در پیاده‌سازی سامانه‌ها استفاده می‌کردند. حملات کانال جانبی از اطلاعات کمکی حاصل از اندازه‌گیری و پردازش مشخصه‌هایی از سامانه استفاده می‌کنند. به این نوع اطلاعات، *اطلاعات نشتی* می‌گویند. مشخصه‌های مختلف سامانه از قبیل زمان پردازش، توان مصرفی، دما، تشعشعات مغناطیسی، صدا و ... می‌توانند اطلاعات نشتی را فراهم کنند. برای اولین بار در دهه ۹۰ میلادی کوچر در [۳,۴] حملات کانال جانبی را بیان کرد که در مقاله اول حملات زمانی مطرح و در مقاله دومی حملات توانی ساده معرفی شدند.

همزمان با وقوع حملات کانال جانبی بحث مقابله با حملات کانال جانبی نیز مطرح شد. هدف مقابله، ایجاد تغییراتی در پیاده‌سازی بود تا پیاده‌سازی‌ها را در برابر حملات کانال جانبی امن کنند و انواع روش‌های مقابله با حملات کانال جانبی در مقالات ابداع شدند. هدف روش‌های مقابله با حملات کانال جانبی، جلوگیری و یا بی‌ثمر کردن اطلاعات نشتی است.

استفاده از روش‌های اثبات‌پذیر برای مقابله با حملات کانال جانبی ایده‌ای بود که مورد توجه محققان قرار گرفت. مرجع [۵] نمونه‌هایی از کارهای اولیه در این ارتباط است. اولین کار جدی در این راستا در سال ۲۰۰۳ در [۶] معرفی شد که در آن میکالی و همکارانش اولین مدل نشت اطلاعات را بیان

* برخی از مطالب این گزارش از مرجع [۱] ذکر می‌شود.

کردند و سعی کردند تعاریف مفاهیم رمزنگاری اثبات‌پذیر نظیر توابع یک‌طرفه و مولد کلید و ... را به مدل مقاوم در برابر نشت بسط دهند. در ادامه مدل‌های مختلفی برای نشت اطلاعات در مقالات بیان شدند که این مدل‌ها در نوع ویژگی‌های مورد نظر برای تابع نشت با یکدیگر تفاوت دارند. در مقالات [۷,۸] انواع مختلفی از مدل‌های نشت بحث شده است.

در حین تلاش برای مدل‌سازی حملات نشت تلاش هایی برای مدل سازی حملات دستکاری توسط مهاجمین فعال نیز انجام یافته است. در این مدل سازی سعی بر این است که مهاجمی با توانایی دستکاری و تغییر در سامانه فرض شود و سامانه رمزنگاری را در برابر این نوع مهاجم امن بکنیم. سامانه رمزنگاری مقاوم در برابر دستکاری[۱] سامانه‌ای است که در آن مهاجم فعالی در نظر گرفته می‌شود که توانایی دستکاری و یا تغییر بعضی از مشخصه‌های سامانه رمزنگاری را دارد. دستکاری در هر یک از مشخصه‌های سامانه از قبیل کلید مخفی و متغیرهای تصادفی می‌تواند رخ دهد. در [۹] اثر تغییر دستکاری در منبع تصادفی سامانه بررسی شده است و در [۱۰] اثر تغییر کلید مخفی سامانه مورد بررسی قرار گرفته است.

در [۹] فرض شده است که مهاجم توانایی تغییر توزیع منبع تصادفی سامانه به منبع تصادفی دیگری را دارد. البته مهاجم توانایی تغییر دلخواه منبع تصادفی را ندارد، زیرا در این صورت مهاجم می‌تواند با انتخاب منبع ثابت، سامانه رمزنگاری احتمالاتی را به سامانه رمزنگاری تعینی تبدیل کند و امنیت تمایزناپذیری در این حالت دیگر معنی ندارد. در این مقاله نشان داده شده است که با تغییر توزیع منبع تصادفی سامانه، امنیت برای رمزگذاری متقارن، رمزگذاری نامتقارن و پروتکل های هیچ‌آگاهی و تعهد وجود نداشته ولی می‌توان امضای دیجیتال ایمنی را داشت. در این مقاله فرض می‌شود که ویروسی در سامانه موجود است که می‌تواند رشته‌بیت تصادفی را به رشته‌بیت دستکاری شده تغییر بدهد. این ویروس با مهاجم در ارتباط نیست و به صورت برخط رشته بیت را تغییر می‌دهد و در نتیجه مهاجم می‌تواند امنیت تمایزناپذیری را نقض کند.

یک روش مقابله با حملات فعال دستکاری کلید سعی در شناسایی عملیات دستکاری در سامانه است. به‌طوری که اگر کلید مخفی $sk$ سامانه به $T(sk)$ کلید تبدیل شود، سامانه متوجه این دستکاری شده و متوقف می‌شود و به اصطلاح وارد سبک انتحاری[۲] می‌شود و دیگر به سایر پرسمان‌های دستکاری جوابی نخواهد داد.

در [۱۱] کامپایلری معرفی شده است که هر مدار رمزنگاری را به مدار دیگری تبدیل می‌کند که قابلیت شناسایی دستکاری در کلید و سایر مشخصه‌های سامانه رمزنگاری را دارد. اگر مدار سامانه رمزنگاری را با $CS(st)$ نشان دهیم که در آن $st$ حالت اولیه سامانه است. این کامپایلر مدار $CS$ را به مدار $CS'$ با حالت اولیه جدید $st'$ تبدیل می‌کند. مهاجم به سامانه جدید $CS'$ دسترسی دارد و می‌تواند تابع دستکاری خود را انتخاب و سامانه جدید را راه‌اندازی کند. در این کامپایلر امضای دیجیتال حالت اولیه $st$ برابر $st'$ است و به این ترتیب هر تغییری در حالت اولیه جدید شناسایی می‌شود ولی باید فرض کنیم که مهاجم توانایی دستکاری کلید مخفی امضا گذار را ندارد.

در [۱۰] راه‌حل بدون نیاز به کلید و مبتنی بر کدهای *چکش‌ناپذیر* ارائه شده است که در این روش نیازی به داشتن کلید مخفی مقاوم (لازم برای تأیید امضا و شناسایی دستکاری) در برابر دستکاری نیست. کدهای چکش‌ناپذیر از لحاظ ساختاری شبیه به کدهای تصحیح خطا هستند. کدهای چکش‌ناپذیر، کدهایی هستند که با اعمال دستکاری توسط مهاجم و تغییر کلمه‌کد معتبر به کلمه‌کد دستکاری شده، پیام حاصله همان پیام اولیه و یا پیامی مستقل از پیام اولیه باشد.

اولین کد چکش‌ناپذیر در [۱۰] معرفی شده است که در آن مهاجم توانایی دستکاری بیت به بیت کلید مخفی را به شکل مستقل از هم دارد. یعنی مهاجم برای هر بیت کلید می‌تواند تصمیم بگیرد که آن بیت صفر یا یک باشد و یا ثابت و یا مکمل بشود و برای بیت دوم به صورت مستقل از بیت قبلی تصمیم‌گیری می شود. مهاجم باید بین توابع دستکاری مجاز توابعی را انتخاب و دستکاری را انجام دهد.

می‌توان نشان داد که در این کد چکش‌ناپذیر محدودیت‌های زیادی برای انتخاب الگوریتم دستکاری وجود دارد و برای مثال با انتخاب مجموعه تمام الگوریتم‌های PPT که مجموعه مورد علاقه ما است هیچ کد چکش‌ناپذیری در این مدل نمی‌توان داشت. برای مقابله با این محدودیت‌ها در [۱۲] کد چکش‌ناپذیر جدیدی ارائه شده است که در آن کلمه‌کد دارای دو بخش $A, B$ است و مهاجم می‌تواند به صورت *مستقل* از هم الگوریتم دستکاری را به دو بخش کلمه‌کد اعمال کند. در این طرح از اثبات‌های هیچ‌آگاهی غیرتعاملی[۳] استفاده شده است. این طرح به کد تفکیک‌شده دو حالته[۴] مشهور است و در مقالات کاربرد بسیار زیادی دارد.

کدهای چکش‌ناپذیر در مدل بیت‌مبنا[۵] در [۱۰] معرفی شده‌اند و کدهای چکش‌ناپذیر برای دستکاری قالب‌مبنا[۶] در [۱۳] بررسی شده‌اند. در [۱۴] با فرض وجود تابع یک‌طرفه[۷] کامپایلری معرفی شده است که هر کد چکش‌ناپذیر تفکیک شده دو حالته را به کد چکش‌ناپذیر با نرخ 1 در مدل استاندارد محاسباتی تبدیل می‌کند. در [۱۵] نشان داده شده است که برای هر خانواده از توابع دستکاری $F$ با اندازه $|F| \leq 2^{poly(n)}$ یک کد چکش‌ناپذیر کارایی وجود دارد که در برابر تمام توابع دستکاری $f \in F$ مقاوم است.

کدهای چکش ناپذیر از دید مهاجم نظریه اطلاعاتی[۸] نیز در مراجع مختلفی از قبیل [۱۶] بررسی شده‌اند. در [۱۷] کد چکش‌ناپذیر کارایی با نرخ 1 در مدل تئوری اطلاعاتی برای یک خانواده از توابع جایگشتی معرفی شده است و در [۱۸] کد چکش‌ناپذیر تفکیک شده در مدل تئوری اطلاعاتی با نرخ کدینگ ثابتی معرفی شده است.

## ۲- حملات دستکاری

حملات دستکاری[۹] جزو حملات فعال پیاده‌سازی هستند. در این حملات مهاجم توانایی دسترسی به بعضی از مشخصه‌های داخلی سامانه رمزنگاری را دارد و می‌تواند برخی از این مشخصه‌ها را تغییر و یا دستکاری کند و با مشاهده رفتار سامانه بعد از اعمال این تغییرات، عملیات تحلیل رمزنگاری را انجام دهد. این دسته از حملات فعال با روش‌های نرم‌افزاری و سخت‌افزاری قابل اعمال هستند. در حوزه نرم‌افزار مهاجم می‌تواند با نفوذ در داخل رایانه، برخی از خانه‌های حافظه رایانه را تغییر دهد و به این ترتیب حمله دستکاری را انجام می‌دهد. در حوزه سخت‌افزار نیز می‌توان با اعمال تغییراتی در توان مصرفی سامانه، تداخلات الکترومغناطیسی، دسترسی مستقیم به داخل سامانه و تغییر در بعضی از مشخصه ها و ... دستکاری را انجام داد. حملاتی نظیر حملات کلید مرتبط؛[۱۰] و حملات تزریق خطا؛[۱۱] می‌توانند جزو حملات دستکاری طبقه‌بندی شوند.

به شاخه‌ای از رمزنگاری که در آن مهاجم تعریف شده, توانایی *دستکاری* م شخ صه‌های سامانه رمزنگاری را دارد، *رمزنگاری مقاوم در برابر د ستکاری*[۱۲] اطلاق می‌شود.

برای آشنایی بیشتر با حملات دستکاری، ابتدا حمله دستکاری از نوع تزریق خطا به سامانه امضای دیجیتال RSA را در ادامه بررسی خواهیم کرد. در این حمله نشان داده شده است که چگونه با اعمال خطا، حتی یک بیت خطا در حین عملیات وارسی امضای دیجیتال، می‌توان به سامانه امضای دیجیتال RSA حمله کرد و یا به عبارت دقیقتر نمای توان‌رسانی N در امضای دیجیتال RSA را تجزیه کرد.

## ۲-۱- حمله تزریق خطا به سامانه RSA

سامانه امضای دیجیتال RSA را با $N = pq$ و کلید عمومی $e, N$ و کلید خصوصی $d$ را در نظر می‌گیریم. امضای دیجیتال برای پیام $m \in \mathbb{Z}_N^*$ (عدد صحیحی در بازه 1 تا $N$) برابر است با $S = \mathrm{m}^{\mathrm{d}}\ mod\ \mathrm{N}$.

امنیت این امضای دیجیتال بر پایه سختی تجزیه عدد $N$ است. البته امنیت امضای دیجیتال RSA در مدل پیشگوی تصادفی و بر اساس سختی مسئله RSA در [۱۹] اثبات شده است که این فرض از سختی مسئله تجزیه عدد $N$ فرض قویتری است.

در پیاده‌سازی‌های عملی سامانه RSA, برای افزایش کارایی محاسبات عملیات نمایی $S = m^d\ mod\ N$ به طور معمول از قضیه باقیمانده چینی (CRT)[۱۳] استفاده می‌شود. با صرف‌نظر از جزئیات داریم:

$$\mathrm{S_p} = \mathrm{m^d}\ \mathrm{mod\ p} = \mathrm{m}^{d\ mod\ (p-1)}\ \mathrm{mod\ p}$$
$$\mathrm{S_q} = \mathrm{m^d}\ \mathrm{mod\ q} = \mathrm{m}^{d\ mod\ (q-1)}\ \mathrm{mod\ q},$$

مقادیر $a, b$ به نحوی انتخاب می‌شوند که:

$$a = 1\ mod\ p\ and\ a = 0\ mod\ q$$
$$b = 0\ mod\ p\ and\ b = 1\ mod\ q,$$

در نتیجه خواهیم داشت:

$$S = m^d\ mod\ N\ = [(aS_p\ +\ bS_q) mod\ N.$$

می‌توان نشان داد که با این روش سرعت عملیات تولید امضا در مقایسه با روش تولید امضا بدون استفاده از CRT در حدود چهار برابر سریعتر می‌شود [۲۰].

فرض کنیم دو امضای دیجیتال بر روی پیام دلخواه $m \in \mathbb{Z}_N^*$ داشته باشیم:

$$S = aS_p + bS_q\ \ mod\ N$$
$$\hat{S} = a\hat{S}_p + b\hat{S}_q\ \ mod\ N.$$

فرض کنید که S یک امضای دیجیتال بدون خطا ولی $\hat{S}$ امضای خطاداری باشد که در آن تنها در طی محاسبه مؤلفه $\hat{S}_p$ خطایی رخ داده باشد و مؤلفه دوم بدون خطا محاسبه شده باشد; یعنی $S_q = \hat{S}_q$ و $S_p \neq \hat{S}_p$ باشد. خطا می‌تواند به صورت تصادفی و یا عامدانه توسط مهاجمی اعمال شده باشد.

در این حالت خواهیم داشت:

$$S - \hat{S}\ = \left(aS_p + bS_q\right) - \left(a\hat{S}_p + b\hat{S}_q\right)$$
$$= a\left(S_p - \hat{S}_p\right).$$

با توجه به روابط بالا داریم:

$$\gcd(\mathrm{S} - \hat{\mathrm{S}}, \mathrm{N}) = \gcd\left(\mathrm{a}\left(\mathrm{S_p} - \hat{\mathrm{S}}_\mathrm{p}\right), \mathrm{N}\right) = \mathrm{q},$$

به این ترتیب عدد N می‌تواند تجزیه شود (در آخرین رابطه از عبارت $\mathrm{a} = 0\ \mathrm{mod\ q}$ استفاده شده است) [۲۱].

## ۲-۲- دستکاری رشته‌بیت تصادفی سامانه رمزنگاری

در [۹] امنیت سامانه‌های رمزنگاری در حضور مهاجمی فعالی با توانایی د ستکاری ر شته بیت ت صادفی سامانه برر سی شده ا ست. در این مقاله حمله دستکاری با احتمال $p$[۱۴] معرفی شده است که در ادامه به اختصار بیان می‌شود. برای بررسی جزئیات بیشتر به [۹] مراجعه کنید.

ابتدا منبع تصادفی سانتا-وزیرانی[۱۵] را تعریف می‌کنیم [۲۲].

متغیر تصادفی $X = (X_1; \dots ; X_n)$ با $X_i \in \{0,1\}$ یک منبع تصادفی سانتا-وزیرانی SV از مرتبه $\delta$ است، اگر برای هر $i \in [n]$ و $(x_1, \dots, x_i) \in \{0,1\}^i$ رابطه زیر برقرار باشد:

$$\delta \leq\ Pr[X_i = x_i |\ X_1 = x_1, \dots, X_{i-1} = x_{i-1}] \leq\ \ 1 - \delta\ .$$

در حمله دستکاری با احتمال $p$، مهاجم هر بیت $x_i$ را با احتمال $p$ می‌تواند دستکاری کند و در غیر این صورت هیچ نوع دستکاری انجام نمی‌دهد. برای انجام دستکاری، مهاجم با علم بر مقادیر بیت‌های قبلی $\{x_1, x_2, \dots, x_{i-1}\}$ مقدار $x_i$ را تغییر می‌دهد.

**تو ضیح:** متغیر ت صادفی حا صل از اعمال حمله د ستکاری با احتمال p روی یک منبع تصادفی یکنواخت یک منبع تصادفی سانتا-وزیرانی با مشخصه $\frac{1-\mathrm{p}}{2}$ است.

حمله دستکاری با احتمال $p$ می‌تواند با مزیت از مرتبه $\mathrm{O(p)}$ برای سامانه‌های زیر تمایزپذیری را انجام بدهد:

- سامانه‌های رمزگذاری متقارن و نامتقارن با امنیت تمایزناپذیری CPA
- ویژگی هیچ‌آگاهی هر الگوریتم هیچ‌آگاهی
- ویژگی نهان‌سازی سامانه‌های تعهد[۱۶]

در بالا برخی از نتایج منفی حمله دستکاری با احتمال $p$ را بیان کردیم و در ادامه نیز چند نتیجه مثبت را بیان خواهیم کرد:

با فرض وجود مهاجمی با توانایی حمله دستکاری با احتمال p و با فرض وجود تابع یک‌طرفه، امکان مقاوم کردن هر نوع سامانه امضای دیجیتال و سامانه‌های شناسایی[۱۷] در برابر این نوع از حمله وجود دارد. به منظور مقاوم‌سازی سامانه امضای دیجیتال در برابر حمله فوق‌الذکر از یک تابع شبه‌تصادفی با هسته[۱۸] اولیه کوچک ا ستفاده می شود که این تابع شبه‌تصادفی جایگزین رشته‌بیت تصادفی مورد استفاده در سامانه امضای دیجیتال می‌شود. در این مقاله از نوعی روش تحلیل فوریه[۱۹] برای محاسبه توزیع خروجی توابع محدود با ورودی دستکاری شده با احتمال p را استفاده می‌شود.

## ۳- روش‌های مقابله با دستکاری

برای مقابله با مهاجم با توانایی دستکاری سامانه رمزنگاری در مقالات دو راه‌کار عمده پیشنهاد شده است:

- استفاده از روش‌های کلیددار
- استفاده از روش‌های بدون کلید

در روش‌های نیازمند کلید، فرض می‌شود که یک مشخصه یا کلید مخفی و غیرقابل‌د ستکاری در سامانه وجود دارد و با ا ستفاده از این کلید سایر بخش‌های سامانه رمزنگاری را در برابر حمله دستکاری مقاوم می‌کنند. در [۱۱] با فرض وجود چنین کلیدی، کامپایلری معرفی شده است که می‌تواند سامانه $S$ را به سامانه معادل $S'$ تبدیل کند که این دو سامانه از لحاظ کارکرد معادل یکدیگر و از لحاظ امنیت، سامانه دومی در برابر حملات دستکاری نیز

مقاوم است. این کامپایلر با استفاده از کلید مخفی خود، از حالت داخلی سامانه، یک امضای دیجیتال تولید می‌کند و اگر مهاجمی بتواند حالت داخلی سامانه را تغییر دهد، در این صورت با توجه به امنیت امضای دیجیتال این دستکاری قابل شناسایی خواهد بود. البته باید دقت کرد که پیش‌فرض وجود مشخصه مخفی و غیرقابل‌دستکاری، فرض سنگینی است و به این دلیل این روش خیلی مورد استقبال قرار نگرفته است.

استفاده از روش‌های بدون کلید، یعنی بدون پیش فرض وجود مشخصه مخفی و غیرقابل‌دستکاری، از دید جامعه تحقیقاتی بسیار جذابتر است. در این دسته از روش‌های مقابله با دستکاری، استفاده از کدهای چکش‌ناپذیر پیشنهاد شده است که اکثریت پژوهش‌های در این حوزه را شامل می‌شود. در بخش بعدی این نوع از کدها بررسی خواهند شد.

## ۳-۱- کدهای چکش‌ناپذیر

یک سامانه کدگذاری در حالت کلی با نماد $\Pi = (\text{Init}, \text{Enc}, \text{Dec}, \kappa, \ell)$ نشان داده می‌شود و به شرح زیر بیان می‌شود:

Init : ماشین تورینگ غیریکنواخت احتمالاتی و زمان چندجمله‌ای است که مشخصه‌ها و اطلاعات عمومی لازم را تولید و مقادیر اولیه را مقداردهی می‌کند.

Enc : کدگذار $Enc$ ماشین تورینگ غیریکنواخت احتمالاتی و زمان چندجمله‌ای است که با گرفتن $s \in \{0,1\}^{\kappa}$ مقدار $x \in \{0,1\}^{\ell}$ را تولید می‌کند. مقدار $x$ را کلمه‌کد[۱۰] می نامند.

$Dec$: کدگشای $Dec$ ماشین تورینگ غیریکنواخت احتمالاتی و زمان چندجمله‌ای است که با گرفتن $x \in \{0,1\}^{\ell}$ مقدار $s \in \{0,1\}^{\kappa}$ یا $\perp$ را منتشر می‌کند به طوری که,

$$\forall s \in \{0,1\}^{\kappa}, \Pr[\text{Dec}(\text{Enc}(s)) = s] = 1.$$

احتمال مذکور بر روی رشته‌بیت تصادفی مورد استفاده در الگوریتم $Enc$ انجام می‌شود.

در تعریف بالا سمبل ویژه $\perp$ نشان‌دهنده‌ی خطایی در عملیات کدگشایی است.

به صورت شهودی یک طرح کدگذاری چکش‌ناپذیر نامیده می شود; اگر کدگشایی هر کلمه‌کد دستکاری شده (تغییر یافته توسط مهاجم) به کلمه‌کد اصلی یا به کلمه‌کد کاملاً نامرتبط و متفاوتی تبدیل شود. یعنی اگر مهاجمی کلمه‌کدی را دستکاری کند، پیام حاصل بعد از کدگشایی همان متن اولیه و یا پیامی کاملاً مستقل و نامرتبط از متن اولیه باشد [۱۰].

اگر مهاجم مجاز به انتخاب تمامی الگوریتم‌های $PPT$ برای دستکاری باشد به سادگی می‌توان نشان داد که هیچ ساختار کدگذاری امنی وجود نخواهد داشت، زیرا مهاجم می‌تواند با انتخاب الگوریتم دستکاری معادل با الگوریتم کدگشایی را روی کلمه‌کد اعمال و پیام اولیه را به دست آورد و سپس دوباره این پیام را کدگذاری کند و امنیت سامانه را نقض کند. به این ترتیب می‌توان دید که در حضور مهاجمی با دسترسی به تمامی الگوریتم‌های $PPT$ با تعریف بالا امکان ایجاد هیچ ساختار چکش‌ناپذیر امنی نیست و باید مجموعه الگوریتم‌های مجاز دستکاری را محدود کرد. روش مورد پسند در مقالات استفاده از کدهای تفکیک‌شده[۲] است. در این حالت فرض می‌شود کلمه‌کد خروجی الگوریتم کدگذاری از $t$ قسمت تشکیل شده است و مهاجم هر قسمت را مستقل از سایر قسمت‌ها و بدون دانستن مقدار قسمت‌های دیگر دستکاری می‌کند. به این نوع سامانه کدگذاری، کدهای چکش‌ناپذیر تفکیک‌شده از مرتبه $t$ می‌گویند. پرکاربردترین کدگذاری استفاده شده در مقالات این حوزه، استفاده از روش تفکیک‌شده از مرتبه 2 است.

## ۴- تعریف چکش‌ناپذیری

تعاریف مختلفی برای چکش‌ناپذیری در مقالات موجود است که در این بخش چکش‌ناپذیری قوی[۳] و پیوسته توضیح داده خواهند شد.

چکش‌ناپذیری قوی با آزمایش $\text{SNM}_{(A,\mathcal{T})}(n)$ نشان داده می‌شود که در این نماد $A$ مهاجم و $\mathcal{T}$ خانواده ماشین‌های تورینگی است که مهاجم مجاز به دستکاری با استفاده از یکی از این ماشین‌های تورینگ است. آزمایش $\text{SNM}_{(A,\mathcal{T})}(n)$ به‌شرح زیر تعریف می‌شود [۱۲]:

**تعریف:** آزمایش تمایزناپذیری $\text{SNM}_{(A,\mathcal{T})}(n)$

۱. الگوریتم $Init(1^n)$ اجرا می‌شود تا مشخصه‌های عمومی $\Omega$ تولید شود.

۲. مهاجم $A$ با گرفتن $\Omega$ زوج پیام $m_0, m_1$ را انتخاب و منتشر می‌کند.

۳. بیت تصادفی $b \in \{0,1\}$ انتخاب می‌شود و مقدار $(x_0, x_1) \leftarrow Enc(m_b)$ محاسبه می‌شود.

۴. مهاجم ماشین تورینگ‌های $T_0 \in \mathcal{T}, T_1 \in \mathcal{T}$ را انتخاب و به عنوان پرسمان دستکاری سؤال می‌کند. (دقت کنید که مهاجم حق تنها یک پرسمان دستکاری را دارد.)

a. مقادیر $x_0' := T_0(x_0), x_1' := T_1(x_1)$ و $x' := Dec(x_0', x_1')$ محاسبه می‌شوند.

b. اگر $(x_0, x_1) = (x_0', x_1')$ باشد، مهاجم مقدار $same^*$ را دریافت می‌کند و در غیر این صورت مقدار $x'$ به مهاجم داده می‌شود.

۵. مهاجم A بیت $b' \in \{0,1\}$ را به عنوان خروجی منتشر می‌کند. خروجی آزمایش 1 است اگر $b' = b$ و در غیر این صورت خروجی آزمایش 0 است.

سامانه کدگذاری $\Pi = (\text{Init}, \text{Enc}, \text{Dec}, \kappa, \ell)$ دارای امنیت چکش‌ناپذیری قوی است اگر و فقط اگر برای تمام مهاجمین مانند A تابع ناچیزی مانند negl موجود باشد به طوری که،

$$\Pr[\text{SNM}_{(A,\mathcal{T})}(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

اگر مهاجم این آزمایش مهاجم احتمالاتی و چندجمله‌ای باشد در این صورت تعریف بالا یک مهاجم محاسباتی را در نظر می گیرد و اگر مهاجم مورد نظر مهاجم نامحدود باشد تعریف بالا مهاجم تئوری اطلاعاتی را دربر خواهد گرفت.

در چکش ناپذیری *پیوسته*[۴] مهاجم توانایی اعمال $q$ پرسمان دستکاری را دارد و تعریف امنیتی چکش‌ناپذیری پیوسته شبیه تعریف بالا است و تنها به مهاجم اجازه q دستکاری داده می‌شود. این تعریف با نماد $\text{CNM}_{A,\mathcal{T},q}(n)$ نشان داده می شود که در آن A مهاجم و $\mathcal{T}$ خانواده مجاز دستکاری و q حد بالای پرسمان دستکاری و تابعی چند جمله‌ای از مشخصه امنیتی n است [۲۳].

در تعاریف بالا اگر مهاجم دسترسی به اطلاعات نشتی از سهم های کلمه کد را نیز داشته باشد با تعریف چکش‌ناپذیری نشت تاب قوی و پیوسته مواجه خواهیم بود که به ترتیب با نمادهای $\mathrm{SNMLR}_{(\mathrm{A},\ell,\mathcal{T})}(\mathrm{n})$ و $\mathrm{CNMLR}_{\mathrm{A},\ell,\mathcal{T},\mathrm{q}}(\mathrm{n})$ نشان داده خواهند شد. در این نمادها منظور از $\ell$ حد بالای اطلاعات نشتی است. در حالت وجود نشت فرض می شود که پیشگوهای نشتی مانند $\mathrm{O}^{\ell}(\mathrm{x}_0)$ و $\mathrm{O}^{\ell}(\mathrm{x}_1)$ وجود دارد که مهاجم با پرسمان الگوریتم‌های $PPT$ می‌تواند تا $\ell$ بیت اطلاعات از سهم‌های $x_0, x_1$ به دست آورد.

## ۵- سامانه کدگذاری FMNV

اولین ساختار کدگذاری چکش‌ناپذیر پیوسته نشت‌تاب در [۲۳] معرفی شده است. این ساختار در این گزارش با اختصار اختار FMNV نامیده می‌شود که مخفف حروف اول نام نویسندگان هست.

**تعریف:** ساختار FMNV

ساختار کدگذاری $\Pi = (Init, Enc, Dec)$ بر اساس طرح انباره نشت‌تاب قوی SLRS[۵] و تابع چکیده‌ساز برخوردتاب و یک اثبات هیچ‌آگاهی غیرتعاملی مقاوم برچسبدار[۶] در مدل CRS بیان می‌شود. خانواده توابع چکیده‌ساز به شکل $\mathcal{H} = \{H: \{0,1\}^n \to \{0,1\}^k\}$ و اثبات هیچ‌آگاهی مقاوم برای زبان $L_{t,\mathcal{H}} = \{h: \exists s \ s.t. \ h = H_t(s)\}$ به شکل $\Pi' = (\mathrm{Init}_{NIZK}, \mathrm{P}, \mathrm{V}, \mathrm{Sim}_1, \mathrm{Sim}_2, \mathrm{Extract})$ تعریف می‌شود [۲۴]. اگر $\Pi'' = (\mathrm{LRS}, \mathrm{LRS}^{-1})$ طرح انباره نشت‌تاب قوی از مرتبه $\ell'$ و $q$ حد بالای تعداد سؤال و جواب مهاجم از پیشگوی دستکاری باشد، در این صورت سامانه کدگذاری FMNV به شکل چندتایی $\Pi = (Init, Enc, Dec)$ نمایش داده می‌شود و در ادامه نیز شرح داده می‌شود.

**توضیح:** سامانه کدگذاری انباره نشت‌تاب, LRS, یک سامانه کدگذاری تفکیک‌شده از مرتبه 2 با امنیت *نشت‌تابی* است. مهاجم این سامانه, مهاجم غیرفعالی است که تنها به اطلاعات نشتی از سهم‌های کلمه‌کد دسترسی دارد [۲۵].

- $: Init(1^n)$

این الگوریتم به صورت یکنواخت $t \leftarrow \{0,1\}^k$ را انتخاب و $\Omega \leftarrow Init_{NIZK}$ را اجرا می‌کند.

- $: Enc(\Omega, x)$

۱- مهاجم ابتدا مقدار $(s_0, s_1) \leftarrow LRS(x)$ را تشکیل می‌دهد و سپس عبارت‌های $h_0 = H_t(s_0), h_1 = H_t(s_1)$ و $\lambda_0 = h_0, \lambda_1 = h_1$ را محاسبه و در انتها با استفاده از الگوریتم اثبات هیچ‌آگاهی دو مقدار $\pi_0 = P^{\lambda_0}(\Omega, h_1, s_1)$ و $\pi_1 = P^{\lambda_1}(\Omega, h_0, s_0)$ را محاسبه می‌کند.

۲- دو سهم کلمه‌کد به شکل زیر محاسبه می‌شوند.

$$\mathrm{X}_0 = (\mathrm{s}_0, \mathrm{h}_1, \pi_0, \pi_1)$$
$$\mathrm{X}_1 = (\mathrm{s}_1, \mathrm{h}_0, \pi_0, \pi_1)$$

- $: \mathrm{Dec}(\mathrm{X}_0, \mathrm{X}_1)$

۱- سهم $\mathrm{X}_b$ برای $b \in \{0,1\}$ به شکل $(\mathrm{s}_b, h_{1-b}, \pi_0, \pi_1)$ تجزیه می‌شود.

۲- وارسی‌های محلی

صحت اثبات‌های هیچ آگاهی به شکل $V^{\lambda_1}(\Omega, h_0, \pi_0)$ و $V^{\lambda_0}(\Omega, h_1, \pi_1)$ بررسی می‌شود.

۳- وارسی‌های تقاطعی

صحت عبارت‌های $h_0 \stackrel{?}{=} H_t(s_0), h_1 \stackrel{?}{=} H_t(s_1)$ و برابری $\pi_0, \pi_1$ در دو سهم دریافتی انجام می‌شود.

۴- اگر هر کدام از وارسی‌ها بالا صادق نباشد، مقدار $\perp$ برگردانده می‌شود و در غیر این صورت اگر تمام وارسی‌ها صادق باشند مقدار $\mathrm{LRS}^{-1}(\mathrm{s}_0, s_1)$ به عنوان خروجی برگردانده می‌شود.

سامانه کدگذاری پیشنهادی FMNV به دلیل برخوردتابی تابع چکیده‌ساز و امنیت سامانه اثبات هیچ‌آگاهی مقاوم, در برابر حملات دستکاری مقاوم است. به طور مثال اگر مهاجمی بخواهد اثبات $\pi_0$ را به اثبات $\pi_0'$ تغییر دهد باید در سهم دوم کلمه‌کد بدون دانستن شاهد یعنی $s_0$ اثباتی معتبر را تولید کند که امری دشوار است. اثبات کامل امنیت این سامانه در [۲۳] موجود است.

## ۶- نتیجه

سامانه‌های کدگذاری دارای کاربردهایی در ابزارهای نیازمند به ویژگی ضد دستکاری برای محافظت از حافظه و سخت‌افزار سامانه در برابر حملات دستکاری هستند. در این‌گونه از ابزارها، استفاده از کدهای چکش‌ناپذیر یک نوع محافظت الگوریتمی و نرم‌افزاری هست. در حال حاضر اکثر روش‌های محافظت در برابر دستکاری مبتنی بر محافظت سخت‌افزاری هست ولی با استفاده از کدهای چکش‌ناپذیر امکان محافظت نرم‌افزاری نیز فراهم می شود. با توجه به وجود پیش‌فرض استقلال دستکاری در سهم‌های مختلف کلمه‌کد تولید شده در سامانه‌های کدگذاری تفکیک شده، بهتر است استفاده از این نوع سامانه‌های کدگذاری به‌عنوان یک لایه امنیتی اضافی به ساختار دفاع در عمق[۷] افزوده می‌شود.

## مراجع


[1] مرتضوی، سید امیر، طراحی سامانه‌های کدگذاری نشت‌تاب و مقاوم در برابر دستکاری با امنیت اثبات‌پذیر، رساله دکتری، دانشگاه صنعتی شریف، تهران, ایران، ۱۳۹۷.

[2] Whitfield Diffie and Martin Hellman. New directions in cryptography. IEEE transactions on Information Theory, 22(6):644–654, 1976.

[3] Paul C Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Annual International Cryptology Conference, pages 104–113. Springer, 1996.

[4] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Advances in cryptology—CRYPTO'99, pages 789–789. Springer, 1999.

[5] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Annual International Cryptology Conference, pages 463–481. Springer, 2003.

[6] Silvio Micali and Leonid Reyzin. Physically observable cryptography. In Theory of Cryptography Conference, pages 278–296. Springer, 2004.

[7] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In Foundations of Computer

[8] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. SIAM Journal on Computing, 41(4):772–814, 2012.

[9] Per Austrin, Kai-Min Chung, Mohammad Mahmoody, Rafael Pass, and Karn Seth. On the impossibility of cryptography with tamperable randomness. In International Cryptology Conference, pages 462–479. Springer, 2014.

[10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In ICS, pages 434–452, 2010.

[11] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In TCC, volume 2951, pages 258–277. Springer, 2004.

[12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. Advances in Cryptology–CRYPTO 2012, pages 517–532, 2012.

[13] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: Built-in tamper resilience. In International Conference on the Theory and Application of Cryptology and Information Security, pages 740–758. Springer, 2011.

[14] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Theory of Cryptography Conference, pages 393–417. Springer, 2016.

[15] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key derivation for poly-size tampering circuits. IEEE Transactions on Information Theory, 62(12):7179–7194, 2016.

[16] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Advances in Cryptology–CRYPTO 2013, pages 239– 257. Springer, 2013.

[17] Shashank Agrawal, Divya Gupta, Hemanta K Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In Annual Cryptology Conference, pages 538–557. Springer, 2015.

[18] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Nonmalleable reductions and applications. In Proceedings of the forty-seventh annual ACM symposium on Theory of computing, pages 459–468. ACM, 2015.

[19] Mihir Bellare and Phillip Rogaway. The exact security of digital signatures-how to sign with rsa and rabin. In International Conference on the Theory and Applications of Cryptographic Techniques, pages 399–416. Springer, 1996.

[20] Jonathan Katz and Yehuda Lindell. Introduction to modern cryptography. CRC press, 2014.

[21] Dan Boneh, Richard DeMillo, and Richard Lipton. On the importance of checking cryptographic protocols for faults. In Advances in Cryptology—EUROCRYPT'97, pages 37–51. Springer, 1997.

[22] MS Satha and UV Vazirani. Generating quasi-random sequences from semi-random sounces. Journal of Computer and System Science, 33:75–87, 1986.

[23] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In TCC, volume 8349, pages 465–488, 2014.

[24] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Crypto, volume 1, pages 566–598. Springer, 2001.

[25] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In SCN, volume 6280, pages 121–137. Springer, 2010.

## زیرنویس‌ها

[1] tamper-resilient cryptography
[2] Self-destruction
[3] non interactive zero knowledge
[4] two split-state
[5] bit-wise
[6] block-wise
[7] one-way
[8] information-theoretic
[9] tampering
[10] related-key attack
[11] fault injection
[12] tamper-resilient cryptography
[13] Chinese Remainder Theorem
[14] p-tampering

[15] Santha-Vazirani
[16] commitment scheme
[17] identification system
[18] seed
[19] Fourier analytic
[20] non-malleable
[21] codeword
[22] split-state
[23] strong non-malleability
[24] continuous non-malleability
[25] strong leakage resilient storage
[26] robust Non-interactive zero knowledge with label
[27] defense in depth