

ساختار اسفنجی؛ معرفی و کاربردها

اکرم خالصی، مجید رحیمی و محمدعلی ارومیه‌چی‌ها*

پژوهش‌گر، پژوهشگاه توسعه فناوری‌های پیشرفته خواجه نصیرالدین طوسی، پژوهشکده افتاده، گروه رمز و امنیت اطلاعات

khalesikram@yahoo.com

rahimi@redat.ir

Orumiehchiha@redat.ir

چکیده

طراحی طیف وسیعی از الگوریتم‌های رمزگاری با استفاده از ساختار اسفنجی تنها با طراحی یک تبدیل یا جایگشت کاهش می‌یابد. در واقع با استفاده از ساختار اسفنجی می‌توان از یک تبدیل یا جایگشت با بیزگی‌های مشخص برای ساخت تابع چکیده‌ساز، الگوریتم رمز دنباله‌ای، الگوریتم رمزگاری احراز اصالت شده و مولد اعداد شبه‌تصادفی استفاده کرد. امکان استفاده از یک تبدیل یا جایگشت واحد در کاربردهای مختلف از یکسو موجب ساده‌سازی طراحی گستره‌ای از الگوریتم‌های رمزگاری می‌شود و از سوی دیگر پیاده‌سازی را نیز ساده‌تر می‌کند. ساختار اسفنجی، مشابه یک سبک عملکرد در رمزهای قالبی، مزایایی مانند امکان ارائه امنیت اثبات‌پذیر، سادگی در طراحی الگوریتم و پیاده‌سازی آن را به دست می‌دهد. در این نوشتار به معرفی این ساختار و انواع آن می‌پردازیم و کاربردها و نیازهای امنیتی آن را تشرییج می‌کنیم.

وازگان کلیدی: ساختار اسفنج، رمزهای دنباله‌ایی، توابع درهم‌ساز، الگوریتم رمزگاری احراز اصالت شده، مولد اعداد شبه‌تصادفی

برابر حملات اولیه روی توابع چکیده‌ساز با طول ثابت مشخص می‌شود، مانند $2^{n/2}$ برای برخورد و 2^n برای پیش تصویر(دوم).

برای اولیه‌های رمزگاری با طول خروجی متغیر، مانند RADIOGATUN¹. بیان امنیت بر اساس طول خروجی منطقی نیست؛ زیرا به طور ضمنی نتیجه می‌شود که با افزایش طول خروجی بدون محدودیت می‌توان امنیت را افزایش داد. علاوه‌بر بیان امنیت در برابر حملات اولیه روی توابع چکیده‌ساز، طراحان تصمیم گرفتند امنیت قابل حصول برای یک تابع ایده‌آل را بیان کنند. برای این منظور، در [۷] یک تابع ایده‌آل معرفی کردند؛ اما پس از انتشار مقاله، متوجه ایده‌آل‌بودن تابع شده و موضوع را عمیق‌تر مورد بررسی قرار دادند. هدف آنها تعریف تابعی با رفتاری مشابه یک پیش‌گوی تصادفی بود، تنها با این تفاوت که تابع مورد نظر شامل برخورد درونی^۲ خواهد بود. نتایج این مطالعات، بسط مفهوم توابع تصادفی اسفنجی است که مشابه پیش‌گوی تصادفی هستند. معرفی اسفنج: ساختار اسفنج یک ساختار تکرارشونده برای ساخت تابع F با طول ورودی متغیر و طول خروجی دلخواه مبتنی بر یک تبدیل یا جایگشت f با طول ثابت n بیت است (b) طول نامیده می‌شود.

۱- مقدمه

ساختار اسفنج^۱ و ساختار اسفنج دوتایی^۲ می‌توانند برای ساخت طیف گسترده‌ای از کاربردهای رمزگاری شامل رمزهای دنباله‌ای، توابع چکیده‌ساز^۳، مولد اعداد شبه‌تصادفی با امکان بارگذاری مجدد هسته^۴، تابع تولید کلید، کدهای احراز اصالت پیام و رمزگاری احراز اصالت شده^۵ مورد استفاده قرار گیرند. این امکان، استفاده از یک جایگشت واحد در کاربردهای مختلف را میسر می‌سازد؛ از این‌رو، حجم پیاده‌سازی الگوریتم‌های مورد نظر نیز کاهش می‌یابد. همچنین، طراحی و استفاده از یک جایگشت قوی بدون نگرانی در مورد اجزا (مانند طرح کلید در رمزهای قالبی) یک مزیت به‌شمار می‌رود.

ایده ساختار اسفنجی در زمان طراحی تابع چکیده‌ساز RADIOGATUN شکل گرفته است. طول ورودی و خروجی در این تابع چکیده‌ساز متغیر است. در زمان پیشنهاد این الگوریتم، طراحان می‌بایست امنیت طرح را مشخص می‌کردند. برای یک تابع چکیده‌ساز با طول چکیده ثابت n بیت، به‌طور عمومی (به‌طور ضمنی یا با صراحت) امنیت تابع به نسبت امنیت یک پیش‌گوی تصادفی که خروجی آن به n بیت بریده شده است، مقایسه می‌شود. بدین ترتیب، مقاومت در

¹ Sponge structure

² Duplex construction

³ Internal collision

حملات رایج برابر با امنیت یک پیشگوی تصادفی باشد، انتظاری منطقی بود؛ اما، این وضعیت بعد از انتشار حملات عمومی^۶ تغییر کرد. با انتشار این حملات، تأثیر حافظه محدود در امنیت تابع چکیده‌ساز معرفی شد که در پیش‌گوی تصادفی مورد استفاده، به عنوان یک مدل امنیتی مطرح نیست.

یک تابع تکرارشونده برای ذخیره حالت داخلی و پردازش قالب به قالب ورودی از یک حافظه محدود استفاده می‌کند. در هر مقطع زمانی، حالت داخلی تابع تکرارشونده قالب‌های ورودی را که تا کنون رسیده است خلاصه می‌کند. از آنجا که تعداد بیت‌های حالت داخلی محدود است، ممکن است در حالت داخلی برخورد رخ دهد. از سوی دیگر، پیش‌گوهای تصادفی فاقد برخورد در حالت داخلی بوده و چنین مفهومی در مورد آنها مطرح نمی‌شود. این مسئله دلیل اصلی عدم امکان استفاده مستقیم از پیش‌گوهای تصادفی برای بیان امنیت تابع با طول خروجی متغیر است: تأثیر حافظه محدود که در هر اولیه رمزنگاری وجود دارد در آنها نمایش داده نمی‌شود؛ از سوی دیگر، تابع اسفنجه تصادفی می‌توانند جایگزینی برای پیش‌گوهای تصادفی در بیان امنیت باشند. یک اسفنجه تصادفی نمونه‌ای از ساختار اسفنجه با b -انتخابی به صورت تصادفی از میان تبدیل‌ها (جایگشت‌ها) b -بیتی است. نشان داده شده است که امنیت یک تابع اسفنجه تصادفی، جز در مواردی که از حافظه محدود نتیجه می‌شود، برابر با یک پیش‌گوی تصادفی است. یک اسفنجه تصادفی می‌تواند به عنوان یک مدل مرجع برای بیان امنیت تابع چکیده‌ساز تکرارشونده و رمزهای دنباله‌ای مورد استفاده قرار گیرد.

۲- تعاریف

در بخش تعاریف، اصطلاحات و توابع کمکی در بحث ساختارهای اس芬جه را که در بخش‌های بعدی استفاده می‌شوند، معرفی می‌کنیم.

۲-۱- رشته‌بیت

طول رشته‌بیت M بر حسب بیت را با $|M|$ نمایش می‌دهیم. رشته‌بیت M می‌تواند به عنوان دنباله‌ای از قالب‌ها با طول ثابت x در نظر گرفته شود که در آن طول قالب آخر ممکن است کوتاه‌تر باشد. تعداد قالب‌های M با $|M|_x$ نمایش داده می‌شود. قالب‌های M با M_i نمایش داده شده و اندیس آن از صفر تا $|M|_x - 1$ تغییر می‌کند. طول یک رشته تهی برابر صفر بوده و بیتی ندارد. رشته تهی را می‌توان به صورت

⁶ Generic attacks

ساختار اسفنجه روی حالت داخلی b -بیتی کار می‌کند که در آن $c = r + b$. مقدار r نرخ بیتی^۱ و مقدار c ظرفیت^۲ اسفنجه نامیده می‌شود (شکل ۱).

ابتدا، تمامی بیت‌های حالت داخلی با مقدار صفر بارگذاری می‌شود. پیام ورودی، دنباله‌زنی شده و به قالب‌های b -بیتی بردیده می‌شود؛ سپس دو مرحله در ساختار اسفنجه اجرا می‌شود: مرحله جذب کردن^۳ و سپس مرحله فشرده شدن.^۴

- در مرحله جذب کردن، قالب‌های پیام b -بیتی با r بیت ابتدایی حالت داخلی (نرخ r) XOR شده و پس از XOR هر قالب پیام، یک مرتبه تابع f روی حالت داخلی $(b$ -بیتی) اسفنجه اجرا می‌شود. زمانی که تمامی قالب‌های پیام پردازش (جذب) شد، اسفنجه وارد مرحله فشرده شدن می‌شود.

- در مرحله فشرده شدن، z بیت ابتدایی حالت داخلی اسفنجه به عنوان خروجی برگردانده و پس از تولید هر z بیت خروجی یک مرتبه تابع f روی حالت داخلی اسفنجه اجرا و تعداد قالب‌های خروجی توسط کاربر مشخص می‌شود.

۵ بیت انتهایی حالت داخلی هرگز به صورت مستقیم از قالب‌های ورودی پیام تأثیر نگرفته و هرگز به صورت مستقیم به عنوان خروجی مورد استفاده قرار نمی‌گیرند. گفتنی است که در برخی از الگوریتم‌های طراحی شده با ساختار اس芬جه، ابتداء مقداری مخالف با مقدار تمام صفر در بیت‌های حالت داخلی اسفنجه بارگذاری می‌شود. این مسئله تأثیری در مطالعی که در ادامه می‌آید، نخواهد داشت.

اسفنجه به عنوان مرجع امنیتی: برای ارزیابی امنیت یک اولیه رمزنگاری، می‌توان خصوصیاتی را که بایستی در برابر آنها مقاوم باشد، به صورت جامع فهرست و سطح مقاومت مربوط به هر یک را مشخص کرد. روش دیگر، بررسی امنیت یک اولیه رمزنگاری^۵ با توجه به یک مدل امنیتی است. بدین معنا که احتمال موفقیت حمله روی آن اولیه رمزنگاری و مدل مورد نظر با یکدیگر مقایسه شود. این روش تمام خصوصیات امنیتی، حتی آنهایی را که در فهرست یادشده پیش‌بینی نشده‌اند، نیز پوشش می‌دهد.

در توابع چکیده‌ساز با طول چکیده ثابت، امنیت مورد نیاز در برابر حملات بر اساس طول چکیده بیان می‌شود. تا چندی پیش انتظار اینکه امنیت یک تابع چکیده‌ساز در برابر

¹ Bitrate

² Capacity

³ Absorbing

⁴ Squeezing

⁵ concrete

یک به همراه کمترین تعداد بیت با مقدار صفر به همراه یک بیت با مقدار یک را برای آنکه طول بیتی رشته حاصل ضریبی از طول قالب باشد با $pad101^*$ نمایش می‌دهیم.

واضح است که این دنباله‌زنی منطبق بر اسنج و یک‌به‌یک بوده و یک رشته تهی و یا رشته‌ای با آخرین قالب تمام صفر نتیجه نمی‌دهد. دنباله‌زنی چندترخی حداقل دو بیت و حداقل یک واحد بیشتر از تعداد بیت‌های یک قالب به یک رشته اضافه می‌کند.

۲-۳- پیش‌گوی تصادفی، تبدیل و جایگشت

یک پیش‌گوی تصادفی را با RO نمایش داده و آن را بصورت زیر تعریف می‌کنیم:

تعریف ۴: یک پیش‌گوی تصادفی RO رشته دودویی را با طول دلخواه به عنوان ورودی گرفته و برای هر ورودی یک رشته تصادفی نامحدود تولید می‌کند؛ بدان معنی که پیش‌گوی تصادفی نگاشتی از Z_2^* به Z_2^∞ است.

فراخوانی RO با ورودی M را که در آن خروجی l بیت ابتدایی بریده شده است، به صورت $Z = RO(M, l)$ نمایش می‌دهیم.

تعریف ۵: یک تبدیل تصادفی با طول داده شده b ، تبدیلی است که به صورت تصادفی و یکنواخت از میان 2^{b^2} تبدیل b -بیتی انتخاب شده است.

تعریف ۶: یک جایگشت تصادفی با طول داده شده b ، جایگشتی است که به صورت تصادفی و یکنواخت از میان 2^b جایگشت b -بیتی انتخاب شده است.

۳- ساختار اسفنجی و ساختار دوتایی

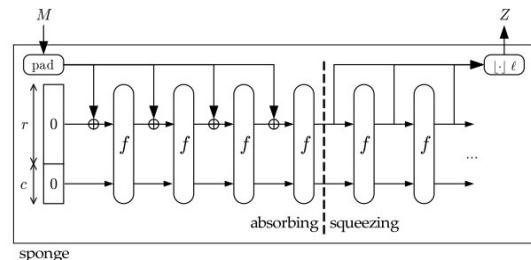
در این بخش به معرفی ساختار اسفنجی و انواع آن می‌پردازیم.

۳-۱- ساختار اسفنجی

ساختار اسفنجی،تابع $[f, pad, r]$ $SPONGE[f, pad, r]$ را با دامنه Z_2^* و Z_2^∞ می‌سازد که در آن f تبدیل یا جایگشت b -بیتی است. pad دنباله‌زنی منطبق بر اسنج و r پارامتر نرخ بیتی است. ساختار اسفنجی یک حالت داخلی b -بیتی به نام s دارد. ابتدا تمام بیت‌های حالت داخلی با صفر مقداردهی می‌شوند. پیام ورودی دنباله‌زنی شده و به قالب‌های b -بیتی بریده می‌شود؛ سپس دو گام اجرا می‌شود: گام جذب کدن و گام فشرده شدن. در این گام‌ها، با r بیت ابتدایی و باقی $b-r$ بیت از حالت داخلی s متفاوت رفتار می‌شود. r بیت ابتدایی حالت داخلی را بخش بیرونی^۵ نماید و آن را با ℓ نمایش می‌دهیم. همچنین، $b-r$

دنباله‌ای بدون قالب و یا یک قالب با طول صفر در نظر گرفت. در این مستند، در صورتی که اشاره صریح نشده باشد، رشته تهی را بدون قالب در نظر می‌گیریم.

تفکیک رشته‌بیت M به I بیت ابتدایی آن را با $M|I$ نمایش می‌دهیم. یک رشته‌بیت متشکل از n صفر را با 0^n و الحاق^۱ دو رشته‌بیت M و N را با $M||N$ نمایش می‌دهیم.



(شکل-۱): ساختار اسفنجی $SPONGE[f, pad, r](M, I)$

مجموعه رشته‌بیت‌ها شامل رشته تهی و بدون رشته تهی را به ترتیب با Z_2^* و Z_2^+ نمایش می‌دهیم. مجموعه رشته‌بیت‌ها با طول نامحدود را با Z_2^∞ نمایش می‌دهیم.

۲-۲- قوانین دنباله‌زنی

برای قوانین دنباله‌زنی^۲ از نمادگذاری زیر استفاده می‌کنیم: دنباله‌زنی پیام M به یک دنباله از قالب‌های x -بیتی را با $M||pad[x](M)$ نمایش می‌دهیم. برای دنباله‌زنی‌های یک‌به‌یک، عنوان دنباله‌زنایی^۳ را برای بازیابی از $M = M||pad[x](M)$ استفاده می‌کنیم.

تعریف ۱: یک دنباله‌زنی منطبق بر اسنج است، اگر هرگز رشته تهی نتیجه ندهد و در شرط زیر صدق کند:

$$\forall n \geq 0, \forall M, M' \in Z_2^*: M \neq M' \Rightarrow M||pad[r](M) \neq M' \\ || pad[r](M') || 0^{nr}$$

حال دنباله‌زنی‌های ساده‌ای را که منطبق بر اسنج هستند، معرفی می‌کنیم.

تعریف ۲: دنباله‌زنی ساده که در آن یک بیت با مقدار یک به همراه کمترین تعداد بیت با مقدار صفر برای آن که طول $pad101^*$ رشته حاصل ضریبی از طول قالب باشد، با $SPONGE[f, pad, r](M)$ نمایش می‌دهیم.

دنباله‌زنی ساده حداقل یک بیت و حداقل به اندازه

تعداد بیت‌های یک قالب به انتهای رشته اضافه می‌کند.

تعریف ۳: دنباله‌زنی چندترخی^۴ که در آن یک بیت با مقدار

¹ Concatenation

² Padding

³ Unpadding

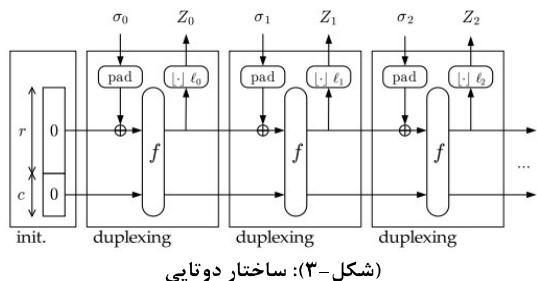
⁴ Multi-rate Padding

گریک قاعده دنباله‌زنی و پارامتر نرخ بیتی^۱ برای ساخت یک طرح رمزگاری استفاده می‌کند. ساختار دوتایی یک شیء^۲ را نتیجه می‌دهد که یک رشته ورودی را گرفته و خروجی را برمی‌گرداند؛ با این ویژگی که خروجی یادشده به تمام ورودی‌هایی که تا کنون اعمال شده وابسته است. نمونه‌ای برای یک ساختار دوتایی را یک شیء دوتایی^۳ نامیده و آن را با D نمایش می‌دهیم. فراخوانی یک شیء دوتایی مشخص D را با پیشوند D و یک نقطه نمایش می‌دهیم.

یک شیء دوتایی D یک حالت داخلی σ -بیتی دارد. در ابتدای مرحله مقداردهی اولیه حالت داخلی با صفر مقداردهی شده و پس از اتمام مقداردهی اولیه فراخوانی به صورت $D.duplexing(\sigma, l)$ که σ رشته ورودی و l تعداد بیت‌های درخواستی خروجی است، انجام می‌شود. بیشترین تعداد بیت قابل درخواست l برابر با r بوده و طول رشته ورودی σ بایستی حداقل به اندازه‌ای باشد که بعد از دنباله‌زنی تنها یک قالب σ -بیتی نتیجه دهد. بزرگ‌ترین طول r نرخ دوتایی بیشینه^۴ نامیده و آن را با (pad, r) نمایش می‌دهیم.

$\rho_{max}(pad, r) = \min\{x: x + |pad[r](x)| > r\} - 1$

واضح است که نرخ دوتایی بیشینه از نرخ بیتی کوچک‌تر بوده و با انتخاب دنباله‌زنی‌ای که کمترین تعداد بیت ممکن را به رشته ورودی اضافه می‌کند، بیشینه می‌شود. با دریافت یک فراخوانی $D.duplexing(\sigma, l)$ شیء دوتایی رشته ورودی σ را دنباله‌زنی کرده و آن را با بخش بیرونی حالت داخلی XOR می‌کند؛ سپس f روی حالت داخلی اسفنج اعمال شده و σ بیت ابتدایی از بخش بیرونی حالت داخلی به عنوان خروجی معرفی می‌شود. فراخوانی با ورودی رشته تهی σ را فراخوانی خالی^۵ و فراخوانی با $\sigma = 0$ را فراخوانی صامت^۶ نامیم. ساختار دوتایی در شکل (۳) نشان داده شده و الگوریتم (۲) (شکل ۴) تعریفی برای آن ارائه می‌کند.



(شکل-۳): ساختار دوتایی

⁴ Object

⁵ Duplex object

⁶ Maximum duplex rate

⁷ Blank call

⁸ Mute call

بیت باقی از حالت داخلی σ را بخش درونی^۱ نامیده و آن را با $\hat{\sigma}$ نمایش می‌دهیم. طول بخش درونی^۲ $b-r$ بیت بوده و ظرفیت c نامیده می‌شود. دو گام یادشده به شرح زیر است:

گام جذب کردن: قالب‌های σ -بیتی ورودی با بخش بیرونی حالت داخلی XOR شده و پس از جذب هر قالب σ -بیتی تابع f اجرا می‌شود. زمانی که تمام قالب‌های پیام پردازش (جذب) شد، ساختار اسفنجی وارد گام فشرده‌شدن می‌شود.

گام فشرده‌شدن: بخش بیرونی حالت داخلی به عنوان خروجی برگردانده شده و پس از تولید هر σ -بیت خروجی، تابع f روی حالت داخلی اسفنج اجرا می‌شود. تعداد دفعات تکرار با توجه به طول خروجی درخواستی l تعیین و درنهایت خروجی اسفنج به عنوان خروجی معرفی می‌شود. c بیت حالت داخلی هرگز به طور مستقیم از قالب‌های ورودی تأثیر نگرفته و هرگز در گام فشرده‌شدن به طور مستقیم به عنوان خروجی معرفی نمی‌شود. اثبات می‌شود که ظرفیت c تعیین‌کننده میزان امنیت قالب حصول ساختار است.

از اصطلاح اسفنج تصادفی برای اسفنجی با یک تبدیل یا جایگشت تصادفی f استفاده می‌کنیم؛ همچنین منظور از حمله عمومی روی ساختار اسفنجی در این مستند به شرح زیر است:

تعريف ۷: اگر یک حمله روی یک تابع اسفنجی از خصوصیات f استفاده نکند حمله‌ای عمومی خواهد بود.

ساختار اسفنجی در شکل (۱) نشان داده شده و الگوریتم (۱) در شکل (۲) تعریفی برای آن ارائه می‌کند.

Algorithm 1 The sponge construction SPONGE[f, pad, r]

Require: $r < b$

```

Interface:  $Z = \text{sponge}(M, \ell)$  with  $M \in \mathbb{Z}_2^*$ , integer  $\ell > 0$  and  $Z \in \mathbb{Z}_2^\ell$ 
 $P = M || \text{pad}[r](|M|)$ 
 $s = 0^b$ 
for  $i = 0$  to  $|P|_r - 1$  do
     $s = s \oplus (P_i || 0^{b-r})$ 
     $s = f(s)$ 
end for
 $Z = [s]_r$ 
while  $|Z|_r < \ell$  do
     $s = f(s)$ 
     $Z = Z || [s]_r$ 
end while
return  $[Z]_\ell$ 

```

(شکل-۲): الگوریتم ساختار اسفنجی [۱]

۲-۳- ساختار دوتایی

مشابه ساختار اسفنجی، ساختار دوتایی σ از یک تبدیل یا جایگشت با طول ثابت $DUPLEX[f, pad, r]$

¹ Inner part

² Inner state

³ Duplex construction

$Z_j = \overline{absorb}(P||0^{r_j}), j \geq 0$
 که در آن $|P| = M||pad[r]|M|$ است.
 تابع جذب برای بیان تغییر حالت داخلی در هر دو صورت جذب ورودی M و فشرده شدن قابل استفاده است.

Algorithm 3 The absorbing function $ABSORB[f, r]$
Require: $r < b$

```
Interface:  $s = absorb(P)$  with  $P \in \mathbb{Z}_{2^r}^*$  and  $s \in \mathbb{Z}_2^b$ 
 $s = 0^b$ 
for  $i = 0$  to  $|P|_r - 1$  do
     $s = s \oplus (P_i || 0^{b-r})$ 
     $s = f(s)$ 
end for
return  $s$ 
```

(شکل-۵): الگوریتم تابع جذب [۱]

خروجی فراخوانی دوتایی خروجی تابع اسفنجی بوده و ثابت می شود که امنیت ساختار دوتایی برابر با امنیت ساختار اسفنجی است.

Algorithm 2 The duplex construction $DUPLEX[f, pad, r]$

Require: $r < b$
Require: $\rho_{\max}(pad, r) > 0$

```
Interface:  $D.\text{initialize}()$ 
 $s = 0^b$ 
```

```
Interface:  $Z = D.\text{duplexing}(\sigma, \ell)$  with  $\ell \leq r$ ,  $\sigma \in \bigcup_{n=0}^{\rho_{\max}(pad, r)} \mathbb{Z}_{2^n}^*$ , and  $Z \in \mathbb{Z}_2^\ell$ 
 $P = \sigma || pad[r] || \sigma'$ 
 $s = s \oplus (P || 0^{b-r})$ 
 $s = f(s)$ 
return  $[s]_\ell$ 
```

(شکل-۴): الگوریتم ساختار دوتایی [۱]

۲-۱-۴- تابع فشردن

تابع کمکی دوم که به جهت ساختاری مشابه تابع جذب کردن است، تابع فشردن $Z = SQUEEZE[f, r](s, l)$ است. از $Z = squeeze(s, l)$ برای نمایش تابع فشردن با جایگشت با تبدیل f مشخص و پارامتر نزدیکی r مشخص استفاده می کنیم.

برای یک حالت داخلی داده شده s $squeeze(s, l)$ معرف خروجی بریده به طول l بیت از تابع اسفنجی، با حالت داخلی s در ابتدای گام فشردن است. تعریف تابع فشردن در الگوریتم (۴) (شکل ۶) آورده شده است.

Algorithm 4 The squeezing function $SQUEEZE[f, r]$

Require: $r < b$

```
Interface:  $Z = squeeze(s, \ell)$  with  $s \in \mathbb{Z}_2^b$ , integer  $\ell > 0$  and  $Z \in \mathbb{Z}_2^\ell$ 
 $Z = [s]_r$ 
while  $|Z|_r < \ell$  do
     $s = f(s)$ 
     $Z = Z || [s]_r$ 
end while
return  $[Z]_\ell$ 
```

(شکل-۶): الگوریتم تابع فشردن [۱]

۲-۲-۴- حملات اولیه روی تابع اسفنجی

در این بخش تعدادی از حملات روی توابع اسفنجی را بررسی می کنیم. گفتگویی است که این حملات ناشی از حالت داخلی محدود توابع اسفنجی بوده، لذا در مورد پیشگویی تصادفی مصدق ندارد. حملات مورد نظر باند بالایی برای امنیت قابل ارائه توسط یک تابع اسفنجی قرار می دهند و حملات پایه ای هستند. به این دلیل این حملات را حملات اولیه آمیزمیم.

³ Primary attacks

۴- امنیت

تا کنون مقالات زیادی در زمینه ارزیابی امنیت اسفنج به چاپ رسیده است. برخی از این مقالات به ارائه امنیت اثبات پذیر پرداخته اند [۸، ۹ و ۱۰] و برخی دیگر نیز به ارزیابی امنیت انواع الگوریتم های مبتنی بر این ساختار پرداخته اند [۱۱، ۱۲]. در این بخش ابتدا تابع کمکی مورد استفاده در ارزیابی امنیت تابع اسفنجی معرفی و سپس حملات اولیه شامل برخورده بازیابی حالت داخلی و انقیاد خروجی^۱ معرفی می شود.

۴-۱- توابع کمکی

در اینجا دو تابع کمکی^۲ تعریف می کنیم که تعاریف و استدلالها را در بررسی حملات روی ساختار اسفنجی ساده می کنند.

۴-۱-۱- تابع جذب و مسیر

تابع کمکی نخست تابع جذب $s = ABSORB[f, r](P)$ است. این تابع رشته P که $|P| = P$ ضربی از r است را به عنوان P گرفته و مقدار حالت داخلی را بعد از جذب $s = absorb(P)$ بر می گرداند. تابع جذب در الگوریتم (۳) (شکل ۵) تعریف شده است. از $s = absorb(P)$ برای نمایش تابع جذب با جایگشت با تبدیل f مشخص و پارامتر نزدیکی r مشخص استفاده می کنیم.

تعریف $s = P$ را مسیری به وضعیت s می کوییم اگر $s = absorb(P)$ باشد.

قالب زیر خروجی از یک تابع اسفنجی متناظر با ورودی M برابر است با:

¹ Output binding

² Auxiliary function

۴-۲-۲- انقیاد خروجی و بازیابی حالت داخلی
در حالت کلی، یافتن حالت داخلی s به‌گونه‌ای که $squeeze(s, |Z|) = Z$ ، برای رشته‌های طولانی Z دشوار است. بسته به دنباله مورد بررسی Z و هدف مهاجم، دو حالت داخلی را در نظر می‌گیریم: انقیاد خروجی و بازیابی حالت داخلی.

در انقیاد خروجی رشته مورد بررسی Z به‌الازام نتیجه فشرده شدن یک حالت داخلی نبوده و بنابراین ممکن است، در عمل پاسخی وجود نداشته باشد.

تعریف ۱۱: برای یک رشته داده شده Z ، انقیاد خروجی یافتن حالت داخلی s است به‌گونه‌ای که $squeeze(s, |Z|) = Z$. تعداد حالت‌های داخلی که با فشرده شدن، رشته داده شده Z را نتیجه می‌دهند، برابر است با $2^{|Z|}$. اگر $b > 2^{|Z|}$ باشد، احتمال وجود چنین وضعیتی به‌طور تقریبی برابر است با $2^{|Z|}$.

در بازیابی حالت داخلی برخلاف انقیاد خروجی، رشته Z با فشرده شدن یک حالت داخلی به‌دست آمده است. ممکن است، حالت‌های داخلی دیگری $s' \neq s$ باشد که با فشرده شدن، رشته Z را نتیجه دهد، اما حالت داخلی s به‌عنوان تنها پاسخ در نظر گرفته می‌شود.

تعریف ۱۲: بازیابی حالت داخلی شامل یافتن حالت داخلی s برای رشته $Z = squeeze(s, |Z|)$ است. اگر $|Z| > b$ باشد، با احتمال بالای تنها یک پاسخ وجود داشته و انقیاد خروجی بازیابی حالت داخلی s که رشته Z را نتیجه دهد، اگر $b < |Z|$ باشد، به‌طور عمومی چند حالت داخلی وجود دارد که فشرده شدن آمها رشته Z را نتیجه می‌دهد و انقیاد خروجی به‌الازام بازیابی حالت داخلی را نتیجه نمی‌دهد.

۵- سبک عملکرد در اس芬ج

تابع اس芬جی تنها یک ورودی M که رشته‌ای با طول دلخواه است را دریافت می‌کند. برخلاف برخی ساختارها، ساختار اس芬جی ورودی دیگری به‌عنوان بردار اولیه ندارد. سبک‌های عملکرد مختلف تابع اس芬جی شامل روش‌های نگاشت انواع ورودی‌ها همچون کلید، بردار اولیه، پیام و ... به ورودی اس芬ج (M) و برش خروجی به طول دلخواه است. در این بخش هستند، معرفی می‌کنیم.

یک ساختار اس芬جی به صورت زنجیره‌ای از توابع دنباله‌زنی، جذب و فشردن است. برای $Z = SPONGE[f, pad, r](M, l)$

$$\begin{aligned} P &= M \parallel [pad[r](|M|)] \\ s &= ABSORB[f, r](P) \\ Z &= SQUEEZE[f, r](s, l) \end{aligned}$$

وارون‌سازی توابع کمکی ساختار اس芬جی پیچیده بوده و در حالت کلی، یافتن یک مسیر P به یک حالت داخلی داده شده s و نیز یافتن حالت داخلی s برای یک خروجی داده شده Z دشوار است. علاوه‌براین، یافتن دو مسیر متفاوت به یک حالت داخلی (برخورد حالت داخلی) نیز پیچیده است.

۴-۱-۲- برحورد

تعریف ۹: برای زوجی از مسیرهای متفاوت ($P \neq Q$) برحورد حالت داخلی عبارت است از:

$$\begin{aligned} absorb(P) &= absorb(Q) \\ \text{برخورد حالت داخلی را از نظر گام اجرایی تابع اس芬جی} \\ \text{در زمان وقوع برحورد (جذب کردن و یا فشرده شدن) و نیز} \\ \text{بیت‌هایی که برحورد در آنها رخ می‌دهد (بیت‌های بخش} \\ \text{درونی و یا بخش بیرونی) می‌توان بررسی کرد.} \end{aligned}$$

بسته به اینکه برحورد حالت داخلی در چه گامی رخ دهد، نتایج مختلف محدودبودن حالت داخلی را نشان می‌دهد. برحورد در گام جذب کردن می‌تواند خروجی‌های یکسان نتیجه دهد؛ به‌عنوان مثال اگر $absorb(Q) = absorb(P)$ با فشرده شدن خروجی یکسان $\overline{Q} = absorb(Q \parallel 0^{rj})$ برای هر مقدار r نتیجه می‌شود. برحورد در گام فشرده شدن می‌تواند بیان‌گر دوره در دنباله خروجی باشد؛ به‌عنوان نمونه، اگر به‌ازای مقداری برای P و d داشته باشیم $absorb(P \parallel 0^{dr}) = absorb(P \parallel 0^{dr})$ ، دنباله خروجی متناوب خواهد بود.

در بررسی موقعیت بیت‌هایی که برحورد در آنها اتفاق می‌افتد داریم:

تعریف ۱۰: یک برحورد داخلی زوجی از مسیرهای متفاوت $absorb(P) \neq Q$ به حالت داخلی یکسان است: $\overline{absorb(Q)}$

واضح است که برحورد حالت داخلی بیان‌گر برحورد داخلی نیز هست؛ اما وارون این مسئله برقرار نیست. گفتی است، ایجاد برحورد حالت داخلی از یک برحورد داخلی ساده است. برای $P \neq Q \parallel B$ داده شده به‌گونه‌ای که $absorb(P) = absorb(Q)$ ، می‌توان برحورد حالت داخلی با $\overline{absorb(P) \oplus A} = A, B \in Z_2^r$ با شرط $absorb(Q) \oplus B$ برای هر را یافت.

¹ State collision

² Inner collision

قرار دهیم. این روش برای سبک‌های عملکردی است که در آن برخی از ورودی‌ها در بسیاری از اجراهای تغییر نمی‌کنند، مانند کلید یا اسم یک فضای اسم.

۶- کاربردهای اسفنج

در این بخش به معرفی نمونه‌هایی برای سبک‌های عملکرد ساختار اسفنج شامل تابع چکیده‌ساز^۱،^۲ رمزگاری احراز اصالت شده^۳ و مولد اعداد شبه‌تصادفی^۴ [۵] که طیف وسیعی از توابع رمزگاری را نتیجه می‌دهند، می‌پردازیم.

۶-۱- تابع چکیده‌ساز

چکیده‌ساز Keccak [۲]، یا همان الگوریتم استاندارد SHA3 از نمونه‌های شناخته شده برای استفاده از ساختار اسفنجی در ساخت الگوریتم‌های رمزگاری است. با برش خروجی یک تابع اسفنجی می‌توان از آن بهصورت یک تابع چکیده‌سازⁿ-بیتی استفاده کرد. اگر بخواهیم از تابع چکیده‌ساز بهصورت چکیده‌ساز تصادفی استفاده کنیم، یک مقام تصادفی (برای مثال salt) به پیام افزوده می‌شود. در صورت نیاز به چند تابع چکیده‌ساز می‌توان از ایده تفکیک دامنه استفاده کرد.

یک تابع چکیده‌ساز را می‌توان بهصورت فرآیند محاسبه چکیده پیام‌ها با فرآخوانی یک تابع تعريف کرد. تابع یادشده می‌تواند یک تابع فشرده‌ساز با طول ورودی ثابت، یک جایگشت و یا یک تابع چکیده‌ساز باشد. دو هدف برای یک سبک عملکرد چکیده‌ساز وجود دارد. هدف نخست ساخت یک تابع چکیده‌ساز با طول ورودی متغیر با استفاده از یک تابع با طول ورودی ثابت و هدف دوم ساخت یک تابع چکیده‌ساز درختی^۳ است.

در چکیده‌ساز درختی امکان پردازش همزمان بخش‌های مختلف پیام وجود دارد. ازین‌رو، چکیده‌ساز درختی می‌تواند برای سرعت‌بخشیدن به محاسبات یک تابع چکیده‌ساز با استفاده از موازی‌سازی در معماری‌های مدرن استفاده شود. در ادامه به معرفی این ساختار و اصطلاحات مورد نیاز آن می‌پردازیم.

یک درخت را در نظر بگیرید که گره‌های آن را با Z_α نمایش داده و α اندیس گره نامیده می‌شود. اصطلاحات مورد استفاده در بحث چکیده‌ساز درختی بهصورت زیر است:

- یک گره می‌تواند یک گره والد^۴ واحد داشته باشد. اندیس گره والد گره‌ای با اندیس α بهصورت parent(α) است.

۱-۵- تفکیک دامنه

به دلیل ویژگی‌های تصادفی بودن و ورودی‌های با طول دلخواه، یک پیش‌گویی تصادفی می‌تواند با استفاده از سازوکار تفکیک دامنه^۱ برای پیاده‌سازی چند پیش‌گویی تصادفی مورد استفاده قرار گیرد. برای این منظور کافی است دامنه را به دسته‌هایی تفکیک کنیم. تفکیک رشته‌ها به دو دسته شامل رشته‌هایی که با صفر شروع می‌شوند و رشته‌هایی که با یک شروع می‌شوند، نمونه‌ای از تفکیک دامنه است. با یک پیش‌گویی تصادفی مشخص RO می‌توان دو پیش‌گویی تصادفی مستقل RO₀(M) \triangleq RO(0||M) و RO₁(M) \triangleq RO(1||M) تعریف کرد. این یک ابزار قدرتمند برای ساخت توابع متنوع با استفاده از یک پیش‌گویی تصادفی است. سازوکار تفکیک دامنه روی توابع اسفنجی نیز قابل اجرا است.

۱-۶- استفاده از کلید

با گنجاندن کلید در ورودی یک تابع اسفنجی می‌توان آن را به یک تابع کلیددار تبدیل کرد. در ساده‌ترین شکل، ورودی M شامل پیوست کلید K و ورودی M' بهصورت $M = M' \parallel K$ یا $M = M' \parallel K \parallel M'$ است. چنین تابعی شبه‌تصادفی بوده و آن را با $(PRF)_K(M')$ نمایش می‌دهیم. اگر تابع اسفنجی مشابه یک پیش‌گویی تصادفی رفتار کند، PRF برای کسی که به کلید دسترسی نداشته و تنها به تابع اسفنج دسترسی دارد مشابه یک تابع تصادفی عمل می‌کند. کلید را می‌توان قبل یا بعد از پیام قرار داد. قراردادن کلید قبل از پیام امکان پیش‌محاسبات حالت داخلی^۲ را فراهم کرده و مقاومت بیشتری در برابر حملات عمومی خواهد داشت.

۱-۷- پیش‌محاسبات حالت داخلی

یک تابع اسفنجی ورودی M را بهصورت قالب‌های α -بیتی پردازش می‌کند. در مرحله پردازش و جذب ورودی می‌توان از نوعی دنباله‌زنی استفاده کرد. برای مثال، اگر در یک اسفنج کلیددار، کلید K بهصورت یک قالب ورودی کامل دنباله‌زنی شود، می‌توان حالت داخلی حاصل پس از جذب کلید را محاسبه و ذخیره کرد. بدین ترتیب، در زمان اجرای اسفنج با این کلید بهخصوص، می‌توان به طور مستقیم محاسبات را از مقدار ذخیره شده برای حالت داخلی اسفنج شروع کرده و از فرآخوانی یک تابع β اضافی جلوگیری کرد. از این نظر، بهتر است پارامترهای ورودی که کمترین میزان تغییرات را دارند، در ابتدا

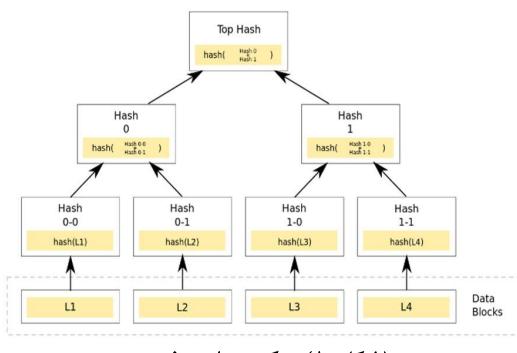
^۱ Tree hash function

^۲ Parent node

^۱ Domain separation

^۲ State precomputation

چکیده‌ساز درختی با اعمال F روی گره نهایی که به یک خروجی با طول نامعلوم منتج می‌شود، به دست می‌آید (شکل ۸). فراخوانی F توسط گره‌های مختلف، داده‌های مختلف را پردازش کرده و قابلیت موازی‌سازی دارد. از آنجا که طول پیام ورودی، دلخواه و از پیش نامعلوم است، لازم است تا نحوه رشد درخت و یا چگونگی پذیرش قالب‌های ورودی (که تعداد آنها افزایش می‌یابد) توسط درختی با تعداد گره ثابت تعیین شود.



شکل ۸: چکیده‌ساز درختی

۶-۱- خصوصیات

سبک عملکرد چکیده‌ساز درختی از دو حالت داخلی پشتیبانی می‌کند:

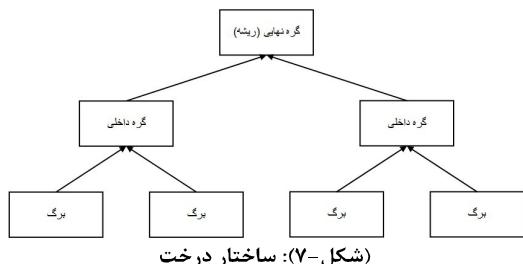
رشد گره نهایی: درجه گره نهایی به صورت تابعی از طول پیام ورودی رشد کرده و تعداد برگ‌ها به نسبت زیاد می‌شود. **جانمایی برگ‌ها:** اندازه درخت و تعداد برگ‌ها با پارامترهای سبک عملکرد درخت و مستقل از طول پیام تعیین می‌شود؛ اما قالب‌های پیام میان برگ‌ها جانمایی می‌شوند.

سبک عملکرد دودویی پیام M و مجموعه‌ای از پارامترها که با A نمایش داده می‌شود:

- سبک عملکرد رشد درخت شامل رشد گره نهایی و یا جانمایی برگ‌ها
- ارتفاع درخت H, $H > 0$
- درجه D گره‌ها
- اندازه قالب برگ‌ها B
- اندازه مقدار زنجیره‌ای C.

زمانی که رشد درخت از نوع جانمایی برگ‌هاست، درخت یک درخت ریشه‌دار با ارتفاع H است: درجه تمام گره‌های داخلی برابر D است. زمانی که رشد درخت از نوع رشد گره نهایی است، درجه گره نهایی به طول پیام بستگی داشته و درجه سایر گره‌ها برابر D است.

در درخت، تمام گره‌ها به جز یک گره، یک والد دارند. گره بدون والد را گره نهایی نامیده و آن را با * نمایش می‌دهیم.
- سایر گره‌ها گره‌های داخلی^۱ نامیده می‌شود (شکل ۷).



- گره با اندیس α فرزند گره با اندیس $\text{parent}(\alpha)$ نامیده می‌شود. هر گره می‌تواند صفر، یک و یا بیشتر فرزند داشته باشد. تعداد فرزندان یک گره را درجه^۲ آن گره نامیده و گره بدون فرزند را برگ^۳ می‌نامیم.

- گره Z_α را نیای^۴ گره Z_β می‌نامیم اگر $\alpha = \text{parent}(\beta)$ و Z_α نیای والد گره Z_β باشد. به عبارت دیگر، Z_α را نیای گره Z_β می‌گوییم اگر دنباله‌ای از اندیس‌ها $\alpha_0, \alpha_1, \dots, \alpha_{d-1}$ وجود داشته باشد به گونه‌ای که $\alpha = \text{parent}(\alpha_i)$ و $\alpha_{i-1} = \text{parent}(\alpha_i)$, α_0 را نوهد^۵ گره Z_α نامیده و d فاصله بین Z_α و Z_β است. واضح است که گره نهایی بدون نیای بوده و یک گره برگ فاقد نوه است.

- هر گره Z_α از نوادگان گره نهایی بوده و فاصله بین هر گره از گره نهایی ارتفاع^۶ آن نامیده می‌شود. بنابراین، ارتفاع گره نهایی صفر بوده و ارتفاع درخت برابر با بیشترین ارتفاع گره‌های آن است.

در یک نگاه، این ساختار به صورت زیر کار می‌کند. یک درخت ریشه‌دار با گره‌های داخلی و برگ‌ها را در نظر بگیرید. گره نهایی ریشه درخت نامیده می‌شود. پیام ورودی به قالب‌هایی بریده شده و در میان برگ‌ها پخش می‌شود. روی هر برگ F اعمال شده و C بیت از خروجی برای تولید زنجیره بریده می‌شود. یک گره داخلی مقادیر زنجیره را از فرزندانش جمع‌آوری کرده، آنها را به هم پیوست کرده و با اعمال مجدد F مقدار زنجیره را محاسبه می‌کند. این عملیات به صورت بازگشته تا رسیدن به گره نهایی تکرار می‌شود. خروجی تابع

¹ Inner nodes

² Degree

³ Leaf

⁴ Ancestor

⁵ Descendant

⁶ Height

چکیده‌ساز درختی به یک چکیده‌ساز موازی ساده کاهش می‌باید که در آن قالب‌های B-بیتی از پیام ورودی به D تابع اسفنجی مختلف ارسال می‌شوند. D نتیجه حاصل نیز در گرهنهایی با یکدیگر ترکیب شده و خروجی نهایی تولید می‌شود. علاوه بر سبک‌های عملکرد رشد گره نهایی و جانمایی برگ‌ها، می‌توان درخت را با افزایش ارتفاع (H)، تا جایی که تعداد برگ‌ها (L) به نسبت طول پیام باشد، بزرگ کرد. تنظیم سبک عملکرد جانمایی برگ‌ها در این شرایط قالب‌های پیام را جانمایی نمی‌کند، اما درخت را ثابت می‌کند.

سبک عملکرد چکیده‌ساز درختی دارای ویژگی درستی^۱ است. ویژگی درستی برای یک تابع چکیده‌ساز بیان گر آن است که برتری تمایز یادشده از یک تابع چکیده‌ساز ایده‌آل دارای باند بالایی برابر با $\frac{q^2}{2^{n+1}}$ است، که در آن q تعداد پرسمان‌ها از تابع چکیده‌ساز و n طول مقادیر زنجیره‌ای است. در [۱] برای درست‌بودن یک تابع چکیده‌ساز چهار شرط بیان شده که سبک عملکرد مورد نظر شرایط یادشده را برآورده کرده و لذا دارای ویژگی درستی است.

۶-۲- رمزنگاری احراز اصالت شده

رمزنگاری احراز اصالت شده (AE) در دهه گذشته مورد توجه جامعه رمزنگاری بوده است. برگزاری مسابقه CAESAR در این زمینه گواهی بر این ادعای است [۱۴]. استفاده از رمزهای قالبی در سبک‌های عملکرد زیادی برای این منظور ارائه شده‌اند که بسیاری از آنها از امنیت اثبات‌پذیر در برابر حملات عمومی نیز برخوردار هستند [۱۵، ۱۶، ۱۷، ۱۸، ۱۹]. استفاده از ساختار اسفنجی در طرح یک الگوریتم رمزنگاری احراز اصالت شده مشابه استفاده از رمزهای قالبی در سبک عملکرد است، با این تفاوت که بجای طراحی یک رمز قالبی، تنها به طراحی یک جایگشت نیاز است.

رمزنگاری احراز اصالت شده برای انتقال کلید محramانه با حفظ تمامیت آن قابل استفاده است. این عمل که پوشش کلید^۲ نامیده می‌شود، در بحث مدیریت کلید از اهمیت بالایی برخوردار است. پوشش کلید، در صورتی که یک شناسه واحد به هر کلید اختصاص داده شود، با استفاده از ساختار اسفنجی نیز قابل انجام است.

¹ Soundness

² Key wrapping

درخت H+1 سطح از گره‌ها دارد که با k اندیس‌گذاری می‌شود. در سطح صفر، گره‌های برگ را داریم که حاوی بیت‌های پیام هستند. گره‌ها در سطوح‌های دیگر $k > 1$ بیت از اینها نتیجه پیوست مقادیر زنجیره‌ای هستند که هر یک از آنها درخت اعمال F روی گره‌ها در سطح $k-1$ هستند که C بیت از خروجی آنها بریده شده است. در سطح H تنها یک گره که همان گره نهایی است وجود دارد. خروجی نهایی حاصل اعمال F روی این گره است؛ حال می‌توان با تشریح نحوه شکل‌گیری گره‌ها به معرفی این سبک عملکرد پرداخت. تمامی گره‌ها به دو نوع فریم بیت ختم می‌شوند. بیت نخست تعیین می‌کند که گره یک برگ است (۱) یا خیر (۰) و بیت دوم تعیین می‌کند که گره یادشده گره نهایی است (۱) یا خیر (۰). در ادامه نحوه شکل‌گیری باقی بیت‌های هر گره را تشریح می‌کنیم.

تعداد قالب‌های B-بیتی در پیام را با $|M|_B$ نمایش داده و قالب‌های پیام را از صفر تا $|M|_B - 1$ با $|M|_B$ اندیس‌گذاری می‌کنیم. توجه کنید که ممکن است، قالب آخر کمتر از B بیت داشته باشد. تعداد گره‌های برگ را با L نمایش می‌دهیم. اندیس‌گذاری گره‌ها در هر سطح را از صفر شروع می‌کنیم. گره با اندیس i در سطح k با $H < k < i$ شامل دنباله‌ای از D مقدار زنجیره‌ای متناظر با گره‌ها در سطح $k-1$ با اندیس‌های iD تا $(i+1)D - 1$ است.

اگر سبک عملکرد جانمایی برگ‌ها باشد، داریم $L = D^H$ و در سطح k تعداد گره‌ها برابر با D^{H-k} است. برگ با اندیس i حاوی دنباله قالب‌های پیام $M_i, M_{i+L}, M_{i+2L}, \dots$ را دارد. گره نهایی مقادیر دنباله‌ای D گره در سطح $H-1$ را دارد و به دنبال آن دو بایت حاوی پارامترهای درخت H، C و B و D کدگذاری شده و یک بایت حاوی سبک عملکرد جانمایی برگ‌ها قرار داده می‌شود.

اگر سبک عملکرد رشد گره نهایی باشد، داریم $L = R D^{H-1}$ با $R = \left\lceil \frac{|M|_B}{D^{H-1}} \right\rceil$ گره در سطح k وجود دارد. اگر $|M|_B < i$ ، برگ با اندیس i حاوی قالب پیام i است و در غیر این صورت بدون قالب پیام آن دو مقادیر دنباله‌ای R گره در سطح $H-1$ را دارد و بدنبال آن دو بایت حاوی پارامترهای درخت H، C و B و D کدگذاری شده و یک بایت حاوی سبک عملکرد رشد گره نهایی قرار داده می‌شود.

اگر تعداد بهینه پردازش‌ها معلوم باشد، به سادگی می‌توان از سبک عملکرد جانمایی برگ‌ها با $H=1$ و D بزرگ‌تر یا مساوی با تعداد پردازش‌ها استفاده کرد؛ در این صورت،

رمزگذاری و محاسبه برچسب را به صورت زیر انجام می‌دهند، امکان‌پذیر است:

رمزگذاری: رمزگذاری با جمع بیتی B با دنباله کلید انجام می‌شود. دنباله کلید یادشده خروجی پیش‌گوی تصادفی RO_1 به رشتہ s_k است که از (K, A) با یک تابع کدگذاری یک به یک محاسبه شده است. اگر (K, A) در نقش تکشمار باشد، دنباله کلید برای داده‌های ورودی مختلف نتیجه فراخوانی RO_1 با ورودی‌های متفاوت بوده و بدین ترتیب، یک دنباله کلید اطلاعاتی در مورد سایر دنباله کلیدها افشا نخواهد کرد.

محاسبه برچسب: برچسب، خروجی پیش‌گوی تصادفی RO_2 به رشتہ h_t است که از (K, A, B) با یک تابع کدگذاری یک به یک محاسبه شده است. برچسب محاسبه شده برای پیام‌های مختلف نتیجه فراخوانی RO_2 با رشتہ ورودی متفاوت خواهد بود.

از آنجا که دنباله کلید و برچسب با فراخوانی پیش‌گوی‌های تصادفی متفاوتی تولید شده‌اند، اطلاعاتی از دیگری افشا نمی‌کنند؛ علاوه بر این، از آنجا که کلید تنها به عنوان ورودی به پیش‌گوی تصادفی اعمال شده است، نیاز به عملی نبودن بازیابی کلید نیز برآورده می‌شود. دو پیش‌گوی تصادفی RO_1 و RO_2 با استفاده از روش تقسیک دامنه روی یک پیش‌گوی تصادفی RO قابل دستیابی است.

ساده‌ترین راه برای ساخت یک سامانه واقعی که به عنوان یک مدل مرجع عمل می‌کند، جایگزین پیش‌گوی تصادفی با تابع اسفنجی است؛ اما چنین راه کاری به اجرای دو تابع اسفنجی نیاز دارد، یکی برای تولید دنباله کلید و دیگری برای تولید برچسب. سبک عملکرد SPONGEWRAP راه کاری است که تنها به یک مرتبه فراخوانی بهازی هر قالب ورودی نیاز دارد.

۶-۴-۲-۶- سبک عملکرد رمزگاری احراز اصالت شده SPONGEWRAP

در این بخش به معروفی سبک عملکرد رمزگاری احراز اصالت شده SPONGEWRAP می‌پردازیم. در اینجا نیز مشابه ساختار دوتایی، نمونه‌ای برای یک ساختار SPONGEWRAP را یک شیء SPONGEWRAP^۵ نامیده و آن را با W نمایش می‌دهیم.

در مرحله مقداردهی اولیه به یک شیء SPONGEWRAP، کلید K در آن بارگذاری می‌شود؛ پس از آن می‌توان درخواست‌هایی برای پوشش یا واپیچی داده ارسال

⁵ SPONGEWRAP object

۶-۲-۱- مدل کردن رمزگاری احراز اصالت شده

رمزگاری احراز اصالت شده به صورت فرآیندی که کلید K ، سرآیند داده A و بدن داده B را دریافت کرده و متن رمزشده C و برچسب T را بر می‌گرداند، در نظر گرفته می‌شود. این عملیات را پوشش^۱ نامیده و عمل دریافت سرآیند داده A ، متن رمزشده C و برچسب T و بازگرداندن بدن داده B را در صورتی که برچسب T صحیح باشد، واپیچیدن^۲ می‌نامیم. متن رمزشده، حاصل رمزگذاری بدن داده B با کلید K بوده و برچسب یادشده یک MAC محاسبه شده از سرآیند و بدن داده با کلید K است.

در اینجا فرض بر آن است که عملیات پوشش و واپیچی قطعی‌آشده؛ بنابراین، دو ورودی یکسان (A, B) و (A', B') ، با کلید یکسان K ، همواره خروجی یکسان (C, T) را نتیجه می‌دهند. در صورتی که این ویژگی مشکل‌ساز باشد، می‌توان با استفاده از تکشمار^۳ بر آن چیره شد.

۶-۲-۲- نیازهای امنیتی

طرح رمزگاری احراز اصالت شده که بهازی کلید انتخاب شده به صورت تصادفی K ، نیازهای امنیتی زیر را برآورده کند، بسیار مفید خواهد بود:

عملی نبودن بازیابی کلید: احتمال موفقیت یافتن کلید در حمله‌ای شامل بررسی N مقدار برای کلید بیشتر از $N^{2^{-|K|}}$ نباشد.

عملی نبودن بازیابی جعل برچسب: بدون بازیابی کلید، احتمال موفقیت جعل برچسب برای هر (A, B) منتخب برابر با 2^{-T} است. این احتمال برای مهاجمی که به متن رمزشده C و خروجی (C_i, T_i) متناظر با متن انتخابی (A_i, B_i) تنها با شرط $(A_i, B_i) \neq (A, B)$ دسترسی دارد نیز یکسان است.

عملی نبودن بازیابی متن اصلی: برای خروجی (C, T) داده شده متناظر با (A, B) ، بهازی A منتخب و B نامعلوم، حتی برای مهاجمی که به خروجی (C_i, T_i) متناظر با متن انتخابی (A_i, B_i) تنها با شرط $(A_i, B_i) \neq (A, B)$ دسترسی دارد، مؤثرترین راه برای به دست آوردن اطلاعات در مورد B (به جز طول آن) بازیابی کلید است.

۶-۲-۳- یک سامانه ایده‌آل

تعريف یک سامانه مرجع که این ویژگی‌ها را داشته باشد، با استفاده از یک زوج پیش‌گوی تصادفی (RO_1, RO_2) که

¹ Wrapping

² Unwrapping

³ Deterministic

⁴ Noncc

باند بالای نرخ برابر با $1 - \rho_{max}(pad, r)$ است. تعریف SPONGEWRAP در الگوریتم (۵) (شکل ۹) آورده شده است.

```

Algorithm 5 The authenticated encryption mode SPONGEWRAP[f, pad, r, ρ].
Require:  $\rho \leq \rho_{max}(pad, r) - 1$ 
Require:  $D = \text{DUPLEX}[f, pad, r]$ 
This algorithm treats  $A, B$  or  $C$  instances equal to the empty string as a single (empty) block

Interface:  $W.\text{initialize}(K)$  with  $K \in \mathbb{Z}_2^+$ 
D.initialize()
for  $i = 0$  to  $|K|_\rho - 2$  do
     $D.\text{duplexing}(K_i||1, 0)$ 
end for
 $D.\text{duplexing}(K_{|K|_\rho - 1}||0, 0)$ 

Interface:  $(C, T) = W.\text{wrap}(A, B, \ell)$  with  $A, B \in \mathbb{Z}_2^*$ , integer  $\ell > 0$ ,  $C \in \mathbb{Z}_2^{|\ell|}$  and  $T \in \mathbb{Z}_2^\ell$ 
for  $i = 0$  to  $|A|_\rho - 2$  do
     $D.\text{duplexing}(A_i||0, 0)$ 
end for
 $Z = D.\text{duplexing}(A_{|A|_\rho - 1}||1, |B_0|)$ 
 $C = B_0 \oplus Z$ 
for  $i = 0$  to  $|B|_\rho - 2$  do
     $Z = D.\text{duplexing}(B_i||1, |B_{i+1}|)$ 
     $C = C|| (B_{i+1} \oplus Z)$ 
end for
 $Z = D.\text{duplexing}(B_{|B|_\rho - 1}||0, \rho)$ 
while  $|Z| < \ell$  do
     $Z = Z||D.\text{duplexing}(0, \rho)$ 
end while
 $T = [Z]_\ell$ 
return  $(C, T)$ 

Interface:  $B = W.\text{unwrap}(A, C, T)$  with  $A, C \in \mathbb{Z}_2^*$ ,  $T \in \mathbb{Z}_2^+, B \in \mathbb{Z}_2^{|C|} \cup \{\text{error}\}$ 
for  $i = 0$  to  $|A|_\rho - 2$  do
     $D.\text{duplexing}(A_i||0, 0)$ 
end for
 $Z = D.\text{duplexing}(A_{|A|_\rho - 1}||1, |C_0|)$ 
 $B_0 = C_0 \oplus Z$ 
for  $i = 0$  to  $|C|_\rho - 2$  do
     $Z = D.\text{duplexing}(B_i||1, |C_{i+1}|)$ 
     $B_{i+1} = C_{i+1} \oplus Z$ 
end for
 $Z = D.\text{duplexing}(B_{|C|_\rho - 1}||0, \rho)$ 
while  $|Z| < \ell$  do
     $Z = Z||D.\text{duplexing}(0, \rho)$ 
end while
if  $T = [Z]_\ell$  then
    return  $B_0||B_1||\dots||B_{|B|}$ 
else
    return Error
end if
```

(شکل ۹): الگوریتم SPONGEWRAP

قابل بیتی در الگوریتم (۵) برای دو هدف استفاده شده است:

تفکیک دامنه: ورودی‌های ساختار دوتایی برای تولید قالب‌های دنباله کلید و قالب‌های برچسب از دامنه‌های متفاوت هستند. هر پاسخ ساختار دوتایی که برای رمزگذاری قالب‌های پیام استفاده می‌شود، پاسخ به رشته ورودی است که به یک ختم می‌شود؛ درحالی که هر پاسخ ساختار دوتایی که برای تولید برچسب استفاده می‌شود، پاسخ به رشته ورودی است که به صفر ختم می‌شود.

کدگشایی: کلید، سرآیند و قالب‌های بدن از دنباله ورودی به ساختار دوتایی قابل بازیابی است. این بدان معنی است که دو رشته متفاوت $K, A^{(0)}, B^{(0)}, A^{(1)}, B^{(1)}, \dots$ و $K', A'^{(0)}, B'^{(0)}, A'^{(1)}, B'^{(1)}, \dots$ دو رشته ورودی یکسان به ساختار دوتایی را نتیجه نمی‌دهند.

کرد. قالب‌های دنباله کلید و برچسب به کلید K و داده‌های ارسال شده در تمامی درخواست‌های قبلی وابسته هستند. مراحل عنوان شده برای پوشش یا واپیچی داده تنها با یک درخواست در سبک عملکرد SPONGEWRAP قابل اجراست.

شیء W SPONGEWRAP، از یک شیء دوتایی D استفاده می‌کند. در مقداردهی اولیه به یک شیء SPONGEWRAP، شیء یادشده D را مقداردهی اولیه کرده و قالب‌های کلید (دنباله‌زنی شده) را با فراخوانی صامت به ارسال می‌کند.

درخواست پوشش با ورودی‌های سرآیند A ، بدن B و طول برچسب خروجی ℓ با فراخوانی شیء $W.\text{wrap}(A, B, \ell)$ بهصورت (C, T) ارسال می‌شود. متن رمزشده C قالب به قالب بهصورت $C_i = B_i \oplus Z_i$ که در آن Z_i پاسخ D به فراخوانی قبلی $D.\text{duplexing}()$ است، تولید می‌شود. بیت برچسب T ، پاسخ D به آخرین قالب داده است (درصورتی که 1 بزرگ باشد، فراخوانی صامت نیز خواهیم داشت). در انتها متن رمزشده C و برچسب T بهعنوان خروجی برگردانده می‌شود.

درخواست واپیچی با ورودی‌های سرآیند A ، متن رمزشده C و برچسب T با فراخوانی شیء $W.\text{unwrap}(A, C, T)$ بهصورت (B, ℓ) دریافت می‌شود، سرآیند B به D ارسال می‌شود. بدن B قالب به قالب بهصورت $B_i = C_i \oplus Z_i$ رمزگشایی می‌شود که در آن Z_i پاسخ D به فراخوانی قبلی $D.\text{duplexing}()$ است. پاسخ D به آخرین قالب بدنه داده با برچسب T که بهعنوان ورودی دریافت شده است مقایسه می‌شود. درصورتی که برچسب معتبر باشد، بدنه داده B برگردانده و در غیر این صورت خطای برگردانده می‌شود. توجه کنید که در زمان پیاده‌سازی می‌توان محدودیت‌هایی همچون فقط پیاده‌سازی پوشش یا واپیچی داده را نیز اعمال کرد؛ همچنانی، شیء SPONGEWRAP می‌تواند محدودیتی را برای کمینه طول برچسب دریافت شده قبل از واپیچی در نظر بگیرد.

هر کلید، سرآیند، داده و یا متن رمزشده قبل از ارجاع به D با قابل بیتی^۱ توسعی داده می‌شوند. اگر A, B و یا C رشته‌های تهی باشند، بهصورت یک قالب شامل رشته تهی در نظر گرفته می‌شوند. نرخ ρ برای سبک عملکرد SPONGEWRAP تعیین کننده اندازه قالب‌ها و در نتیجه بیشترین تعداد بیت پردازش شده در هر بار فراخوانی f است.

¹ Farme bit

بدنه داده استفاده کرد. در سامانه‌های مدیریت کلید، هر کلید یک شناسه منحصر به‌فرد دارد. بدین ترتیب، کافیست شناسه هر کلید در سرآیند A قرار داده شود. در این صورت، دو کلید اصلی هیچگاه با یک دنباله کلید رمز نمی‌شود. در پوشش کلید، می‌توان از کلید عمومی متناظر و یا چکیده محاسبه شده از کلید مورد نظر به‌عنوان شناسه استفاده کرد.

۶-۳- تولید دنباله اعداد شبه‌تصادفی با امکان بارگذاری مجدد بذر

اعداد شبه‌تصادفی کاربردهای فراوانی در رمزگاری دارند. تولید کلید یکی از کاربردهای متداول این‌گونه دنباله از اعداد است. اگرچه خاصیت تصادفی بودن از یک منبع فیزیکی قابل استخراج است، اما اغلب به تعداد بیت‌های بیشتر از آنتروپی یک منبع فیزیکی نیاز است. یک مولد دنباله اعداد شبه‌تصادفی راه حلی برای این نیاز است. چنین مولدی با یک بذر^۲ که به صورت مخفی و یا به‌طور کامل تصادفی تولید شده بارگذاری شده و آن را به دنباله‌ای از بیت‌ها توسعی می‌دهد.

یکی از ویژگی‌های مهم برای دنباله اعداد شبه‌تصادفی در کاربردهای رمزگاری آن است که حتی در صورت افسای بخشی از دنباله، پیش‌گویی سایر قسمت‌های آن ممکن نیاشد. از این جنبه، مولد اعداد شبه‌تصادفی مشابه یک رمز دنباله‌ای است؛ علاوه‌بر این، در برخی کاربردها ویژگی امنیت پیشرو^۳ باشیست فراهم شود. این ویژگی بدان معناست که آشکارشدن حالت داخلی کنونی امکان آشکارکردن بیت‌های قبلی تولیدشده را فراهم نکند.

حالت داخلی مولد اعداد شبه‌تصادفی باشیست آنتروپی لازم را داشته باشد. این بدان معناست که پیش‌بینی بیت‌های خروجی به‌سادگی با حدس حالت داخلی ممکن نیاشد؛ بنابراین، دنباله‌ای که به‌عنوان بذر مورد استفاده قرار می‌گیرد، باشیست آنتروپی کافی را فراهم کند. دنباله تولیدشده به‌عنوان بذر با استفاده از منابع فیزیکی به‌عنوان مولد خاصیت تصادفی، بدليل نامتنازن بودن و یا همیستگی بیت‌ها، عموماً آنتروپی بالای ندارند. برای افزایش آنتروپی، می‌توان از بذرهای مختلف که از منابع تصادفی مختلف به‌دست آمده استفاده کرد؛ اما این آنتروپی باشیست به حالت داخلی محدود مولد اعداد شبه‌تصادفی منتقل شود؛ بنابراین به روشهایی برای ترکیب آنتروپی تولید شده از منابع مختلف در حالت داخلی مولد نیاز داریم. بارگذاری بذرهای مختلف در مولد اعداد شبه‌تصادفی دنباله‌های متفاوتی را نتیجه می‌دهد. این بدان معناست که

² Seed

³ Forward secrecy

مزایا و محدودیت‌ها:

مزایای سبک عملکرد رمزگاری احرار اصالتشده عبارتند از:

- این سبک عملکرد تنها به طراحی یک جایگشت با طول ثابت نیاز دارد، برخلاف برخی از سبک‌های عملکرد رمزگاری احرار اصالتشده که به طراحی یک رمز قالبی نیاز دارند.

- این سبک عملکرد علاوه بر رمزگاری احرار اصالتشده، برای کاربردهایی که در آنها فقط به احرار اصالش نیاز است، نیز قابل استفاده است.

- قابلیت تولید برچسب‌های میانی وجود دارد.

- یک طرفه است.

- برای هر قالب ورودی تنها یکبار فراخوانیتابع f مورد نیاز است.

- انتخاب نرخ بیتی متنوع، مادامکه ظرفیت از مقدار معینی کمتر نباشد، از ویژگی‌های این سبک عملکرد است.

در مقایسه با سبک‌های عملکرد رمزگاری احرار اصالتشده مبتنی بر رمزهای قالبی، SPONGEWRAP دارای محدودیت‌هایی نیز هست.

محدودیت‌های سبک عملکرد رمزگاری احرار اصالتشده عبارتند از:

- این سبک عملکرد به صورت سریال بوده و قابلیت پیاده‌سازی موازی در سطح الگوریتم را ندارد.

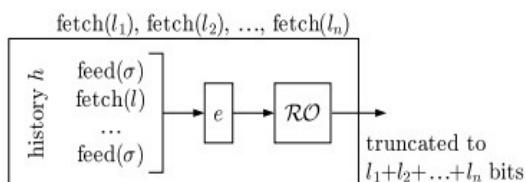
- اگر سامانه محدودیتی به‌عنوان الزام استفاده از تکشمار در A را نداشته باشد، مهاجم می‌تواند دو پرسمان (A, B) و (B') را انجام دهد. در این صورت، نخستین قالب متفاوت در B، به‌عنوان مثال B_i و B'_i با کلید یکسانی رمز شده و اطلاعاتی در مورد جمع بیتی آنها را برای مهاجم افشا می‌کند. این در حالی است که برخی از سبک‌های عملکرد رمزگاری احرار اصالتشده برای رمزهای قالبی به‌گونه‌ای طراحی شده‌اند که حتی در صورتی که سامانه شرط استفاده از تکشمار را منظور نکند، تها اطلاعاتی که برای مهاجم افشا می‌شود، تساوی یا عدم تساوی دو ورودی به الگوریتم است [۲۰].

کاربرد:

پوشش کلید عبارتست از حفظ محرمانگی و تمامیت کلید در انتقال یا ذخیره آن. کلید اصلی با کلید رمزگذاری کلید KEK^۴ پوشش داده می‌شود. می‌توان از سبک عملکرد SPONGEWRAP با KEK به‌عنوان K و کلید اصلی به‌عنوان

⁴ Key-encrypting key

سبک عملکرد یادشده تمامی درخواست‌های تغذیه و واکشی را در حافظه h نگه می‌دارد. با دریافت درخواست $\text{feed}(\sigma)$ ، حافظه h به روزرسانی می‌شود. با دریافت درخواست (l) ، یک پرسمان از پیش‌گوی تصادفی با رشتۀای که حافظه h را کدگذاری کرده صورت می‌گیرد. پاسخ پیش‌گوی تصادفی از بیت x تا بیت $x+1-1$ به عنوان خروجی به پرسش‌گر برگردانده می‌شود، که در آن x تعداد بیت‌های درخواستی در درخواست‌های واکشی قبلی از زمان آخرین درخواست تغذیه است. بنابراین، پیوست پاسخ‌ها به دنباله‌ای از درخواست‌ها معادل با پاسخ پیش‌گوی تصادفی تنها به یک پرسمان است. این وضعیت در شکل (۱۰) نشان داده شده است. سبک عملکرد یادشده، سبک عملکرد حافظه‌دار با تابع کدگذاری $e(h)$ ^۳ نامیده می‌شود؛ بنابراین، تعریف سبک عملکرد مورد نظر به تعریف این تابع کدگذاری کاهش می‌یابد.



(شکل ۱۰): پاسخ یک مولد شبۀ تصادفی به درخواست‌های واکشی

از آنجا که خروجی مولد شبۀ تصادفی بایستی به تمامی داده بذر وابسته باشد، لازم است تا تابع کدگذاری یک به یک باشد. به بیان دیگر، برای دو دنباله از درخواست‌ها با بذرها مختلف، دو خروجی متفاوت از تابع کدگذاری بایستی داشته باشیم. این ویژگی تمامیت بذر^۳ نامیده می‌شود. تعریف ۱۳: یک مولد شبۀ تصادفی ایده‌آل یک پیشگوی تصادفی در سبک عملکرد حافظه‌دار با تابع کدگذاری $e(h)$ با ویژگی تمامیت بذر است.

گفتنی است که مولد شبۀ تصادفی مورد نظر فاقد ویژگی امنیت پیشرو است (افشای حالت داخلی کنونی مهاجم را قادر به بازیابی بیت‌های خروجی قبلی نمی‌کند).

۶-۲-۳- SpongePRG

یک روش ساده برای ساخت یک سامانه واقعی که مانند سامانه مرجع فوق عمل می‌کند، جایگزین کردن پیش‌گوی تصادفی با تابع اسفنجی است. در نگاه نخست، نیاز به ذخیره تمامی پرسمان‌های قبلی مستلزم حافظه زیاد بوده و این راه کار عملی به نظر نمی‌رسد؛ اما با استفاده از ساختار دوتایی این مانع رفع می‌شود.

بذرها مختلف حالت‌های داخلی مختلف را نتیجه می‌دهند؛ از این جبهه، یک مولد اعداد شبۀ تصادفی مانند یک تابع چکیده‌ساز مقاوم در برابر برخورد است.

تحقیق امکان بارگذاری مجدد بذر برای یک مولد اعداد شبۀ تصادفی ساده است. برای این منظور، کافی است امکان ورود بذرها اضافه بعد از تولید بیت‌های شبۀ تصادفی فراهم شود. بارگذاری مجدد بذر، به جای حذف کردن حالت داخلی کنونی، آن را با بذر جدید ترکیب می‌کند. از دیدگاه کاربر، یک مولد اعداد شبۀ تصادفی با امکان بارگذاری مجدد بذر به صورت یک جعبه سیاه با یک واسطه برای درخواست دنباله بیت‌های شبۀ تصادفی و یک واسطه برای ورودی بذرها جدید است. در ادامه، منظور از مولد اعداد شبۀ تصادفی مولد اعداد شبۀ تصادفی با امکان بارگذاری مجدد بذر است.

۶-۱-۳- مدل کردن یک مولد شبۀ تصادفی ایده‌آل

مولد شبۀ تصادفی را به صورت یک واحد حافظه‌دار با پشتیبانی از دو نوع درخواست تعریف می‌کنیم.

- درخواست تغذیه^۱، $\text{feed}(\sigma)$ ، که رشتۀ ناتهی $\sigma \in Z_2^+$ حاوی بذر را به حالت داخلی مولد شبۀ تصادفی تزریق می‌کند.

- درخواست واکشی^۲، $\text{fetch}(l)$ ، که برگرداندن ۱ بیت را از مولد شبۀ تصادفی درخواست می‌کند.

داده تشکیل‌دهنده بذر حاصل الحقق σ ه در تمامی درخواست‌های تغذیه است.

نیازمندی‌های یک مولد شبۀ تصادفی بدین صورت قابل بیان است: نخست، خروجی آن بایستی به تمامی داده تشکیل‌دهنده بذر وابسته باشد. دوم، برای مهاجمی که از بذر آگاهی نداشته و بخشی از خروجی را مشاهده کرده است، یافتن اطلاعاتی در مورد سایر بخش‌های خروجی غیرعملی باشد.

بیان دقیق‌تر نیازمندی‌های امنیتی یک مولد شبۀ تصادفی اغلب با تعریف یک سامانه مرجع ایده‌آل صورت می‌گیرد. سامانه مرجع مناسب برای توابع اسفنجی، توابع چکیده‌ساز و رمزهای دنباله‌ای پیش‌گوی تصادفی است؛ اما از آنجا که پیشگوی تصادفی واسطه متفاوتی با مولد شبۀ تصادفی با امکان بارگذاری مجدد بذر دارد، نمی‌توان از آن به عنوان سامانه مرجع مناسبی برای مولد شبۀ تصادفی استفاده کرد. با این حال، یک مولد شبۀ تصادفی ایده‌آل را به صورت سبک عملکرد خاصی از پیش‌گوی تصادفی تعریف می‌کنیم.

¹ Feed

² Fetch

³ Seed-completeness

اگر f یک جایگشت باشد، SpongePRG برگشتپذیر خواهد بود و افشاری حالت داخلی، امکان برگشتن به حالت داخلی در آخرین اعمال بذر را فراهم می‌کند. البته راه کارهایی برای این مشکل وجود دارد که نمونهای از آن در [۱] معرفی شده است. سبک عملکرد SpongePRG در الگوریتم (۶) (شکل ۱۱) تعریف شده است.

مزایا و محدودیت‌ها:

ایده اصلی مبنی بر ترکیب منابع مختلف بذر و تولید بیت‌های شبه‌تصادفی است. تنها نیازمندی برای داده بذر آن است که به صورت دنباله‌ای از بیت‌ها باشد، که بدون هیچ‌گونه پیش‌پردازش اضافی در دسترس است. بنابراین، اعمال بذر و تولید بیت‌های شبه‌تصادفی در یک سبک عملکرد پیوسته قابل پیاده‌سازی بوده و برای اعمال داده بذر اضافی نیازی به تکرار اضافه‌تر تابع به روزرسانی نیست.

اگر f یک جایگشت باشد، طول حالت داخلی را می‌توان کوتاه نگه داشت. این امکان از دو ویژگی زیر فراهم می‌شود. ویژگی نخست، حفظ آنتروپی حالت داخلی با استفاده از جایگشت و ویژگی دوم، وجود باند بالای قوی برای احتمال موقوفیت حملات عمومی روی اسفنج کلیددار است.

عدم تأمین ویژگی امنیت پیشرو در ساختار معرفی شده ضعف آن به شمار می‌رود. این ضعف به دلیل در بسیاری از کاربردها مشکل‌ساز نیست؛ نخست آن که اعمال بذر به صورت منظم و با آنتروپی کافی مانع پیشروی مهاجم در جهت عقب می‌شود. دوم آن که تجهیزی که مولد شبه‌تصادفی مورد نظر روى آن پیاده می‌شود باقیتی محروم‌گی داده را حفظ کرده و از این رو بازخوانی حالت داخلی مشکل خواهد بود. نمونه‌ای از استفاده از ساختار اسفنج برای تولید اعداد شبه‌تصادفی در [۵] معرفی شده است.

۷- نتیجه‌گیری

در این نوشتار به معرفی ساختار اسفنجی که در سال‌های اخیر به یکی از ساختارهای پرکاربرد در طراحی اولیه‌های رمزنگاری تبدیل شده پرداختیم. برای این منظور، ساختار اسفنجی پایه و ساختار دوتایی را تعریف و حملات اولیه روى آنها را بررسی کردیم. در ادامه به تشریح نحوه استفاده از اسفنج در طراحی توسعه چکیده‌ساز، الگوریتم‌های رمزنگاری احراز اصالت شده و

یک شیء دوتایی به عنوان یک مولد شبه‌تصادفی با امکان بارگذاری مجدد بذر قابل استفاده است. در خواست تغذیه با ورودی σ با فراخوانی $D.duplexing()$ صورت گرفته و خروجی به عنوان بیت‌های شبه‌تصادفی قابل استفاده است. بیت‌های شبه‌تصادفی بدون داشتن بذر با فراخوانی $D.duplexing()$ قابل تولید است. تنها محدودیت این ساختار آن است که رشته حلوی بذر بایستی به طول ρ_{max} بریده شده و نیز در هر بار فراخوانی تنها r بیت خروجی قابل تولید است.

سبک عملکرد SpongePRG به صورت زیر کار می‌کند. این سبک عملکرد از یک شیء D استفاده کرده و دارای دو بافر است: بافر ورودی B_{in} و بافر خروجی B_{out} . بذر در در خواست‌های تغذیه به B_{in} فرستاده شده و با فراخوانی $D.duplexing()$ به شیء D ارسال می‌شود. در صورتی که تعداد بیت‌های بذر بیشتر از ρ_{max} باشد، بیت‌های اضافی در B_{in} می‌ماند. با دریافت درخواست واکشی، در صورت خالی‌بودن بافر ورودی، بیت‌های باقی مانده از بذر به D ارسال شده و بیت‌های درخواست‌شده خروجی داده می‌شود (در صورت نیاز تعداد دفعات فراخوانی $D.duplexing()$ قابل افزایش است). بیت‌های اضافی خروجی در بافر خروجی نگهداری شده و در درخواست واکشی بعدی ابتدا این بیت‌ها برگردانده می‌شود. توجه کنید که در هر زمان یکی از بافرهای ورودی و یا خروجی خالی است.

Algorithm 6 Pseudo random bit sequence generator mode SPONGEPRG[f, pad, r, ρ]

Require: $\rho \leq \rho_{max}(pad, r)$
Require: $D = \text{DUPLEX}[f, pad, r]$

Interface: $P.initialize()$

$D.initialize()$

$B_{in} = \text{empty string}$

$B_{out} = \text{empty string}$

Interface: $P.feed(\sigma)$ with $\sigma \in \mathbb{Z}_2^+$

$M = B_{in} || \sigma$

for $i = 0$ to $|M|_\rho - 2$ do

$D.duplexing(M_i, 0)$

end for

$B_{in} = M_{|M|_\rho - 1}$

$B_{out} = \text{empty string}$

Interface: $Z = P.fetch(\ell)$ with integer $\ell \geq 0$ and $Z \in \mathbb{Z}_2^\ell$

while $|B_{out}| < \ell$ do

$B_{out} = B_{out} || D.duplexing(B_{in}, \rho)$

$B_{in} = \text{empty string}$

end while

$Z = [B_{out}]_\ell$

$B_{out} = \text{last } (|B_{out}| - \ell) \text{ bits of } B_{out}$

return Z

Interface: $P.forget()$

$Z = D.duplexing(B_{in}, \rho)$

$B_{in} = \text{empty string}$

for $i = 1$ to $\lfloor c/\rho \rfloor$ do

$Z = D.duplexing(Z, \rho)$

end for

$B_{out} = \text{empty string}$

(شکل-۱۱): الگوریتم سبک عملکرد SpongePRG

[Bookmark not defined.]

مولد اعداد شبه تصادفی پرداخته و مزایا و محدودیت های هر کدام را مورد ارزیابی قرار دادیم.

-۸- مراجع

- [13] G.Bertoni, J. Daemen, M. Peeters, and G. V Assche, "Sufficient Conditions for Sound Tree and Sequential Hashing Modes", *International Journal of Information Security*, pp.335-353, 2014.
- [14] CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. Retrieved from <http://competitions.cr.y-p.to/caesar.html/>.
- [15] M.Bellare and C.Namprempre, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm", *ASIACRYPT 2000, LNCS 1976*, pp. 531-545, Berlin,2000.
- [16] V. D. Gligor and P. Donescu, "Fast Encryption and Authentication: XCBC Encryption and XECB Authentication Modes", *FSE 2001, LNCS 2355*, pp. 92-108. Berlin, 2001.
- [17] C. S. Jutla, "Encryption Modes with Almost Free Message Integrity", *EUROCRYPT 2001, LNCS 2045*, pp. 529-544. Berlin,2001.
- [18] P.Rogaway, M. Bellare, J.Black, and T.Krovetz, "OCB: A Block Cipher Mode of Operation for Efficient Authenticated Encryption", *ACM Conference on Computer and Communication Security*, pp. 196-205, 2001.
- [19] NIST, Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality. NIST special publication 800-38C. 2007
- [20] P.Rogaway and T.Shrimpton, "A Provable-Security Treatment of the Key-Wrap Problem", *EUROCRYPT 2006. LNCS 4004*, pp. 373-390. Berlin: Springer, 2006.
- [1] G.Bertoni, J.Daemen, M.Peeters, and G.V.Assche, (2011), "Cryptographic Sponge Functions", Retrieved from <http://sponge.noekeon.org/>.
- [2] G.Bertoni, J. Daemen, M. Peeters, and G. V.Assche, (2013). Keccak. *EUROCRYPT 2013, LNCS 7881*, pp. 313-314.
- [3] W.Li, G. Liao, Y.Wen, and Z.Gong, "Spong MPH: A New Multivariate Polynomial Hash Function bascd on the Sponge Construction", *Second International Conference on Data Science in Cyberspace (DSC).Shenzhen: IEEE*, (2017).
- [4] N.Abdoun, S. El Assad, K. Hammoud, R.Assaf, M.Khalil, and O. Disforges, "New Keyed Chaotic Neural Network Hash Function Based on Sponge Construction", *International Conference for Internet Technology and Secured Transactions (ICITST). Cambridge: IEEE*, (2018).
- [5] P.Kumar Singh, A. V. Monsy, R. Garg, S.Dey, and S.Nandi, "JSpongeGen: A Pseudo Random Generator for Low Resource Devices", *International Conference on Distributed Computing and Internet Technology, ICDCIT 2019*, pp. 410-421, 2019.
- [6] R.AlTawy, R.Rohit, M. He, K.Mandal, G.Yang, and G.Guang, "sLiSCP: Simeck-Based Per-mutations for Lightweight Sponge Cryptographic Primitives", *International Conference on Selected Areas in Cryptography. LNCS 10719*, pp. 129-150 , 2018.
- [7] G.Bertoni, J.Daemen, G. V.Assche, and M.Peeters, Radiogatun, a Belt-and-Mill Hash Function, Second Cryptographic Hash Workshop, Available: <http://radiogatun.noekeon.org/>. [Accessed:2006].
- [8]P.Jovanovic, A. Luykx, and B.Mennink, "Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes", *ASIACRYPT 2014, LNCS 8873*, pp. 85-104. 2014.
- [9] B.Mennink, R.Reyhanitabar, and D.Vizar," Security of Full-State Keyed Sponge and Duplex: Applications to Authenticated Encryption", *ASIACRYPT 2015, LNCS 9453*, pp. 465-489. 2015.
- [10] Y.Naito and K.Yasuda, "New Bounds for Keyed Sponges with Extendable Output: Independence Between Capacity and Message Length", *FSE 2016, LNCS 9783*, pp. 3-22, Berlin, 2016.
- [11] L.Song and J.Guo, "Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP", *IACR Transactions on Symmetric Cryptology*, pp. 182-214, 2018.
- [12] R.Kumar, M. S. Rajasree, and H. Alkhzaimi, "Cryptanalysis of 1-Round KECCAK. AFRI-

