به نام خدا

دانشکده‌ی علوم ریاضی

مقدمه‌ای بر رمزنگاری ۱۹ تیر ۱۳۹۳

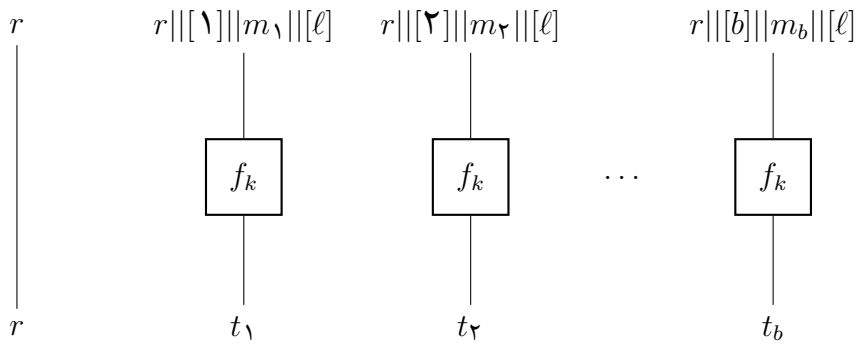
جلسه‌ی ۱۷: توابع چکیده‌ساز

مدرس: دکتر شهرام خزائی *نگارنده: سعید محلوجی فر، محمد آقامیر*

هدف ما در این جلسه ارائه ادامه مبحث کد اصالت‌سنجی پیام [۱] است و همچنین معرفی توابع چکیده‌ساز [۲] و استفاده از آن برای بهبود کدهای اصالت‌سنجی پیام، مباحث آخر این جلسه خواهندبود.

کد اصالت‌سنج برای پیام با طول غیرثابت

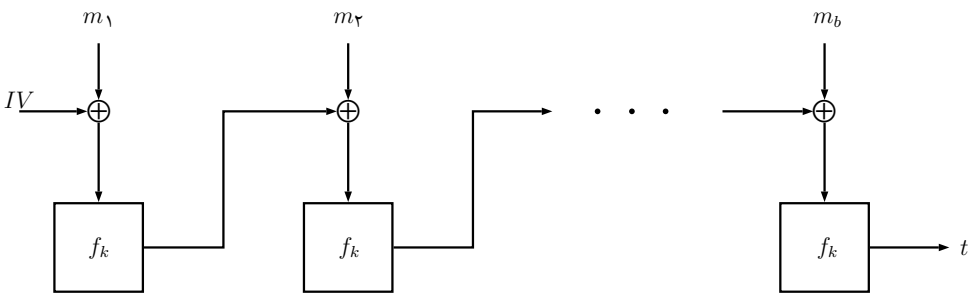
در جلسه قبل دیدیم که برای MAC کردن پیام با طول ثابت n ، از خانواده توابع شبه تصادفی f_k می‌توانیم به شکل زیر استفاده کنیم:



$$\langle r, t_1, \ldots, t_b \rangle \leftarrow \mathsf{Mac}_k(m)$$

مشکل این کد اصالت‌سنج این است که طول MAC و همچنین کارائی تابع شبه تصادفی $\frac{4\ell}{n}$ می‌باشد. حال ما می‌خواهیم که این مقدار را از $\frac{4\ell}{n}$ به $\frac{\ell}{n}$ برسانیم.
یک راه‌کار برای حل این مشکل این است که به جای استفاده از مد کاری ECB از مد کاری CBC استفاده کنیم.

[۱] Message Authentication Code
[۲] Hash Functions

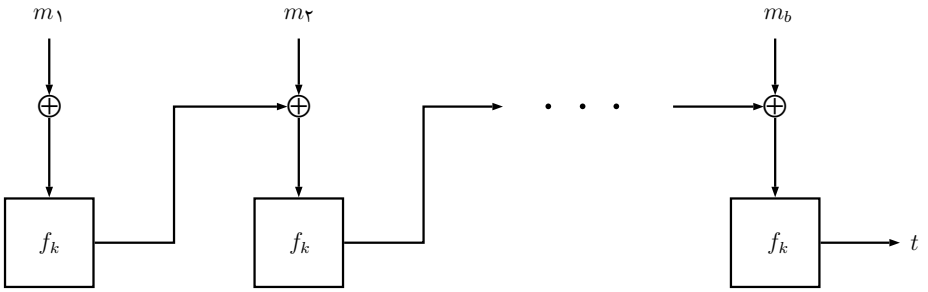


این راه‌کار امن نیست:

۱. حمله کننده پیام m را ارسال کرده و برچسب $\langle IV, t\rangle$ را دریافت می‌کند.

۲. سپس حمله کننده برچسب قابل قبول $\langle m \oplus \Delta, \langle IV \oplus \Delta, t\rangle\rangle$ را تولید می‌کند.

همان‌گونه که مشاهده می‌کنید مشکل اصلی IV است. دو راه‌کار برای حل این مشکل وجود دارد یا این که خود IV را نیز به تابع شبه تصادفی بدهیم و خروجی آن را قرار دهیم و راه کار دوم که ما آن را ترجیح می‌دهیم این است که اصلا IV نداشته باشیم. حال سوال این است که آیا ساختار بالا بدون IV امن است یا خیر؟



جواب خیر است و این ساختار نیز دارای امنیت غیرقابل جعل نیست:

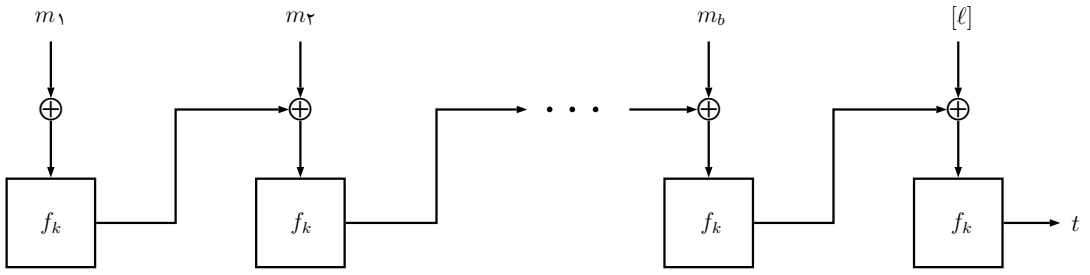
۱. حمله کننده پیام‌های m_1، $m_1 m_2$ و m'_1 دلخواه را ارسال کرده و برچسب‌های

$m_1 \longrightarrow t_1$
$m_1 m_2 \longrightarrow t_2$
$m'_1 \longrightarrow t_3$

را دریافت می‌کند.

۲. سپس حمله کننده برچسب قابل قبول $\langle m'_1 \parallel (m_2 \oplus t_3 \oplus t_1), t_2\rangle$ را تولید می‌کند.

به عنوان راه حل، می‌توان طول پیام (ℓ) را به عنوان آخرین تکه پیام اضافه کرد و MAC را روی آن هم اعمال کرد.



اما این ساختار نیز قابل جعل است:

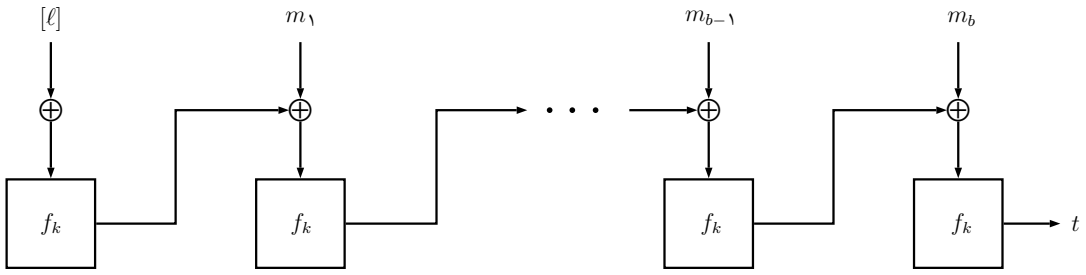
۱. حمله کننده پیام‌های $m_۱ \parallel m_۱$، $m_۲ \parallel m_۲ \parallel [۲n] \parallel m_۱ \parallel m_۱$ و $m_۳ \parallel m_۳$ را ارسال کرده و برچسب‌های

$$m_۱ \parallel m_۱ \longrightarrow MAC(m_۱ \parallel m_۱) = t_۱$$
$$m_۱ \parallel m_۱ \parallel [۲n] \parallel m_۲ \parallel m_۲ \longrightarrow MAC(m_۱ \parallel m_۱ \parallel [۲n] \parallel m_۲ \parallel m_۲) = t_۲$$
$$m_۳ \parallel m_۳ \longrightarrow MAC(m_۳ \parallel m_۳) = t_۳$$

را دریافت می‌کند.

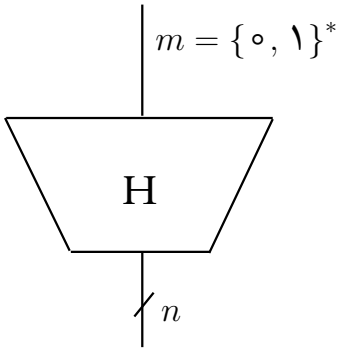
۲. سپس حمله کننده برچسب قابل قبول $\langle m_۳ \parallel m_۳ \parallel [۲n] \parallel m_۴ \parallel m_۲, t_۲ \rangle$ را تولید می‌کند که $m_۴ = t_۱ \oplus t_۲ \oplus m_۲$ است.

اما اگر همین ℓ را اول پیام بگذاریم؛ امن خواهدشد.



توابع چکیده‌ساز

تابع (الگوریتم) چکیده‌ساز هر تابعی است که ورودی با طول دلخواه را به خروجی با طول ثابت می‌نگارد، طوری که با تغییرات در ورودی نتوان خروجی را کنترل کرد و تغییرات کوچک در ورودی باعث تغییرات بزرگ در خروجی شود. این توابع در رمزنگاری و همچنین طراحی الگورتم‌ها کاربرد گسترده دارند. به طور دقیق‌تر یک تابع چکیده‌ساز تابعی است مانند $H : \{۰, ۱\}^* \to \{۰, ۱\}^n$ که n طول چکیده پیام نامیده می‌شود.



به صورت ایده‌آل یک تابع چکیده‌ساز باید مانند یک اوراکل تصادفی[۳] عمل کند که به هر پیام دلخواه یک چکیده کاملاً تصادفی n بیتی اختصاص می‌دهد. اما پیاده‌سازی چنین تابعی غیر عملی است. در ادامه مهم‌ترین ویژگی‌های توابع چکیده‌ساز که به دنبال آن هستیم را معرفی می‌کنیم.

برخوردتابی: این ویژگی که مقاومت در برابر برخورد[۴] نیز نامیده می‌شود به این مفهوم است که یافتن زوج m, m' طوری که $m \neq m'$ و $H(m) = H(m')$ از نظر محاسباتی غیر عملی باشد. به چنین زوجی یک برخورد برای H می‌گوییم. لازم به ذکر است با توجه به این که فضای ورودی بزرگتر از فضای خروجی است، برخورد وجود دارد و هر مهاجمی با محاسبه حدود $\mathbf{2}^{\frac{n}{2}}$ پیام متفاوت می‌تواند با احتمال بالایی یک برخورد برای H بیابد. بنابراین عدم توانایی محاسباتی مهاجم برای یافتن برخورد به معنای عدم وجود برخورد نیست.

پیش‌تصویرتابی: این ویژگی که مقاومت در برابر پیش‌تصویر[۵] نیز نامیده می‌شود به این مفهوم است که به ازای یک رشته‌ی h از فضای خروجی تابع H، نتوان ورودی m را یافت طوری که $H(m) = h$. در این حالت به m یک پیش‌تصویر برای h نسبت به تابع H می‌گوییم.

پیش‌تصویر دوم‌تابی: این ویژگی که مقاومت در برابر پیش‌تصویر دوم[۶] نیز نامیده می‌شود به این مفهوم است که به ازای یک رشته‌ی h از فضای خروجی و یک رشته‌ی m از فضای ورودی تابع H، طوری که $H(m) = h$، نتوان یک رشته m' از فضای ورودی پیدا کرد طوری که $m \neq m'$ و $H(m') = H(m) = h$. توجه شود که مقاومت در برابر پیش‌تصویر دوم قابل کاهش به مقاومت در برابر پیش‌تصویر است.

هنگامی که بحث‌های مربوط به امنیت مطرح می‌شود، به تعریف خانواده‌ای از توابع چکیده‌ساز علاقه‌مند می‌شویم. برای این منظور خانواده‌ی $\{H_s\}_{s\in\{0,1\}^*}$ از توابع چکیده‌ساز تعریف می‌شود:

$$H : \{0,1\}^{\kappa(n)} \times \{0,1\}^* \to \{0,1\}^{\lambda(n)}$$

در این تعریف n پارامتر امنیتی است. می‌توان تصور کرد که s یک کلید است اما قرار نیست مخفی باشد و تنها برای مشخص کردن یک عضو $H_s(\cdot) = H(s, \cdot)$ از خانواده بکار می‌رود. همچنین $\kappa(n)$ و $\lambda(n)$ توابعی بر حسب پارامتر امنیتی می‌باشند که به ترتیب طول کلید و طول چکیده را مشخص می‌کنند. با استفاده از این مفاهیم می‌توان با آزمایش‌هایی مناسب، ویژگی‌های تابع چکیده‌ساز را به طور دقیق تعریف کرد.

کاربرد توابع چکیده‌ساز

توابع چکیده‌ساز کاربرد بسیاری در رمزنگاری دارد طوری که در اکثر سیستم‌های رمزنگاری شناخته شده به نحوی از این توابع استفاده شده است. خواص سه‌گانه گفته شده باعث می‌شود با احتمال بسیار زیادی دو ورودی متمایز به خروجی های متمایزی گماشته شوند و به این ترتیب می‌توان ورودی های مختلف را با استفاده از خروجی آنها تمایز داد.
نقش سه شرط گفته شده را با دو مثال بررسی می‌کنیم. سیستم های امضای دیجیتال را درنظر بگیریم. در این سیستم‌ها ورودی با انجام اعمال سنگین ریاضی مانند همنهشتی و توان به یک امضا با طول ثابت نگاشته می‌شوند.

[۳] Random Oracle
[۴] Collision Resistance
[۵] Pre-Image Resistance
[۶] Second Pre-Image Resistance

در مواردی که اندازه ورودی بزرگ باشد امکان انجام این اعمال به طور مستقیم بر روی ورودی غیر عملی است. بنابراین باید طول ورودی را به نحوی کاهش داد. برای این کار به جای امضا کردن ورودی اصلی، ابتدا آن را از یک تابع چکیده‌ساز عبور می‌دهند و سپس آن را امضا می‌کنند. واضح است که اگر تابع چکیده‌ساز مورد استفاده هر یک از سه شرط بالا را نداشته باشد سیستم دیگر دارای امنیت دلخواه شده نخواهد بود.

مورد دیگر ذخیره‌سازی رمز عبور است. در سرورهای اینترنتی به جای ذخیره‌سازی رمز عبور کاربران آن را از یک تابع چکیده‌ساز عبور می‌دهند و سپس ذخیره می‌کنند. دلیل این موضوع آن است که ممکن است افراد زیادی به سرور دسترسی داشته باشند و بتوانند اطلاعات را بدست آورند. به همین دلیل ذخیره‌سازی رمزهای عبور کار عاقلانه‌ای نخواهد بود. به این ترتیب هنگامی که کاربر رمزعبور خود را وارد می‌کند این رمز از تابع چکیده‌ساز عبور داده می‌شود و با مقدار ذخیره شده مطابقت داده می‌شود. در این روش مقاومت در برابر برخورد و پیش‌تصویر شرط لازم برای امنیت است.

به عنوان آخرین مثال، پروتکل تعهد را بیان می‌کنیم. یک پروتکل تعهد شامل دو مرحله تعهد و آشکارسازی است. در مرحله تعهد، فرستنده خود را به پیامی متعهد می‌کند. بعداً، در مرحله آشکارسازی فرستنده، پیام خود را برای گیرنده آشکار می‌کند. به عنوان کاربرد، تصور کنید شخصی ادعا می‌کند که پیروز جام جهانی بعدی را می‌داند. یک روش فیزیکی برای انجام این کار بدین صورت است که آن شخص نام تیم مورد نظر را در درون صندوقی که امکان باز کردن آن صرفاً برای خودش وجود دارد بگذارد و صندوق را در اختیار شما قرار دهد. این همان مرحله تعهد است. بدین ترتیب، شخص متعهدشونده دیگر نمی‌تواند نام تیم مورد نظر خود را تغییر دهد. همچنین شما نمی‌توانید اطلاعاتی در مورد نام تیم به‌دست آورید زیرا لازمه این کار باز کردن صندوق است. حال پس از مشخص‌شدن نتیجه، شخص متعهدشونده با باز کردن صندوق برای شما، ثابت می‌کند نام تیم پیروز را می‌دانسته‌است (یا مشخص می‌شود که اشتباه کرده‌است). این همان مرحله آشکارسازی است. در رمزنگاری، پروتکل تعهد به صورت دیجیتال انجام می‌شود. برای این منظور از یک تابع تعهد استفاده می‌شود که n پارامتر امنیتی است:

$$\text{commit} : \{۰,۱\}^n \times \{۰,۱\}^* \to \{۰,۱\}^*$$

این تابع در زمان چندجمله‌ای و به صورت قطعی عمل می‌کند. در واقع پروتکل تعهد از دو مرحله زیر بین فرستنده و گیرنده تشکیل شده‌است:

مرحله تعهد: در این پروتکل، فرستنده خود را به مقدار $m \in \{۰,۱\}^*$ متعهد می‌کند؛ برای این کار با انتخاب مقدار تصادفی $r \leftarrow \{۰,۱\}^n$ مقدار $c = \text{commit}(r,m)$ را محاسبه و برای گیرنده ارسال می‌کند. حال گیرنده c را ذخیره و برای استفاده در آینده نگه‌می‌دارد.

مرحله آشکارسازی: در این پروتکل، فرستنده با ارسال پیام m و مقدار تصادفی r به گیرنده، عملاً c را باز می‌کند. گیرنده با محاسبه‌ی $\text{commit}(r,m)$ و مقایسه‌ی آن با c از صحت m اطمینان می‌یابد.

یک روش پیاده‌سازی تابع تعهد استفاده از یک تابع چکیده‌ساز H به صورت $\text{commit}(r,m) = H(r||m)$ است.

سؤال ۱ *آیا به نظر شما برای امن بودن پروتکل تعهد، ویژگی‌های سه‌گانه تابع چکیده‌ساز کافی است؟*

توابع چکیده‌ساز معروف

- MD5 که طول خروجی آن ۱۲۸ بیت است و در سال ۲۰۰۸ برای آن برخورد پیدا شد.

- SHA-1 توسط NIST طراحی شده که طول خروجی اش ۱۶۰ بیت است و تا به حال برخوری برای آن پیدا نشده است. اما با توجه به پیشرفت توان محاسباتی به نظر می‌رسد به زودی برای این تابع یک برخورد پیدا شود.
- SHA-2 توسط NIST طراحی شده با طول متغیر از ۲۵۶ بیت تا ۵۱۲ بیت ولی خیلی مورد استفاده قرار نمی‌گیرد.
- SHA-3 با همان طول ۲۵۶ تا ۵۱۲ بیت که در مسابقه‌ای به نام SHA-۳ که توسط NIST برگزار شد برنده شد و آن را با نام Keccak هم می شناسند.

ساخت توابع چکیده‌ساز

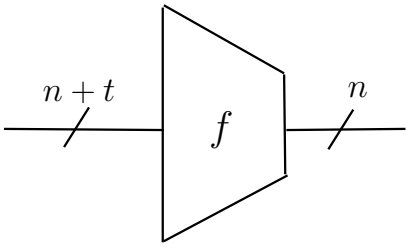
ساختن توابع چکیده‌ساز با ویژگی‌های گفته شده کار مشکلی است و به همین دلیل توابع چکیده‌ساز مورد استفاده‌ی زیاد به چند مورد بیان شده خلاصه می‌شود. برای ساخت توابع چکیده‌ساز دو روش عمده وجود دارد. مد مرکل-دمگارد[۷] و مد اسفنجی[۸] دو روشی هستند که برای ساخت توابع چکیده‌ساز از آنها استفاده می‌شود.

مد مرکل-دمگارد

برای معرفی این مد ابتدا باید توابع فشرده‌ساز را معرفی کنیم.

توابع فشرده ساز

توابع فشرده ساز توابعی هستند که ورودی های با طول ثابت را به خروجی با طول ثابت کوچکتر می‌نگارد. ویژگی این توابع این است که یک‌طرفه هستند. به این مفهوم که محاسبه‌ی ورودی آنها از روی خروجی به راحتی امکان پذیر نیست. توابع فشرده ساز عمدتا به وسیله رمزهای بلوکی ساخته می‌شوند و روش های گوناگونی برای ساخت این توابع از روی رمز های بلوکی وجود دارد که در انتها به این روش ها می‌پردازیم.

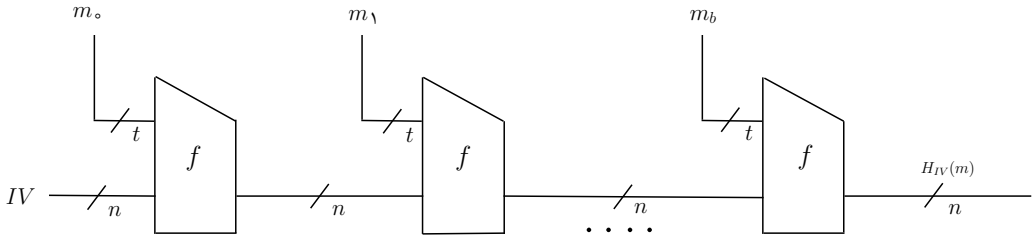


مرکل-دمگارد

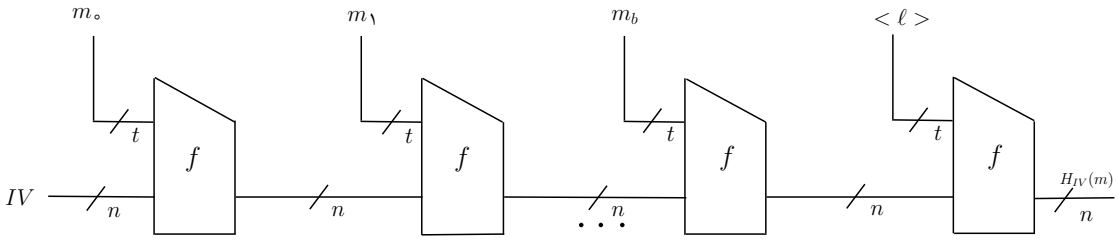
در مد مرکل-دمگارد با استفاده از یک تابع فشرده ساز یک تابع چکیده‌ساز ساخته می‌شود. فرض کنید f یک تابع فشرده‌ساز مقاوم دربرابر برخورد باشد که ورودی با طول $n+t$ را به خروجی با طول n می‌نگارد. و فرض کنیم

[۷] Merkle–Damgård Mode
[۸] Sponge Mode

$m10^* = m_1 m_2 ... m_b$ گسترش‌یافته‌ی ورودی m باشد طوری که طول هر m_i برابر t باشد. اولین ساختاری که به ذهن می‌رسد این است به شکل زیر عمل کنیم.



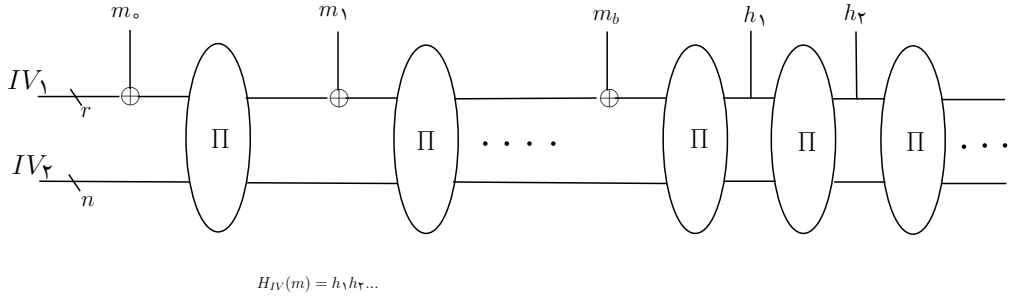
که IV یک مقدار ثابت است. سوالی که مطرح می‌شود این است که اگر f مقاوم دربرابر برخورد باشد آیا H_{IV} نیز مقاوم در برابر برخورد است؟ جواب این سوال خیر است. برای اینکه به یک تابع چکیده‌ساز مقاوم دربرابر برخورد برسیم کافی است یک تغییر کوچک در شکل بالا ایجاد کنیم. فرض کنیم ℓ طول پیام m باشد. مد مرکل-دمگارد به شکل زیر عمل می‌کند.



در این حالت می‌توان بررسی کرد که اگر f مقاوم در برابر برخورد باشد آنگاه H_{IV} نیز مقاوم دربرابر برخورد است. اثبات این موضوع به خواننده واگذار می‌شود(راهنمایی: کافی است بر اساس طول دو پیامی که برخورد دارند دو حالت در نظر بگیرید و سعی کنید در یکی از بلوک ها یک برخورد برای f پیدا کنید.).

مد اسفنجی

در مد اسفنجی با استفاده از یک جایگشت یک تابع چکیده‌ساز تولید می‌کنیم. فرض کنید Π یک جایگشت باشد که طول ورودی و خروجی آن $n + r$ است. و فرض کنیم $m10^* = m_1 m_2 ... m_b$ گسترش‌یافته‌ی ورودی m باشد طوری که طول هر m_i برابر r باشد. مد اسفنجی به این شکل عمل می‌کند.



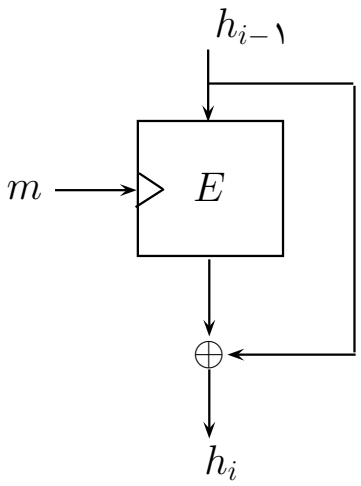
می‌توان این عملیات را تا حد لازم ادامه داد و به میزان لازم خروجی گرفت. می‌توان ثابت کرد اگر Π شبه تصادفی باشد آنگاه H_{IV} مقاوم در برابر برخورد است و علاوه بر آن شبه تصادفی نیز هست و به شکل یک اوراکل تصادفی عمل می‌کند. در مورد ساخت جایگشت‌های شبه‌تصادفی نیز قبلا در مبحث رمز های بلوکی بحث شده است و به عنوان مثال می‌توان از AES با یک کلید ثابت استفاده کرد.

ساخت توابع فشرده‌ساز

برای ساخت توابع فشرده‌ساز از رمزهای بلوکی استفاده می‌کنیم. دو روش دیویس-مایر[۹] و اُسیس-مایر[۱۰] را برای ساخت توابع فشرده‌ساز از روی رمزهای بلوکی معرفی می‌کنیم.

دیویس-مایر

در این روش یک تابع فشرده ساز می‌سازیم که ورودی m, h_{i-1} را به h_i می‌نگارد. در این روش به شکل زیر عمل می‌کنیم.

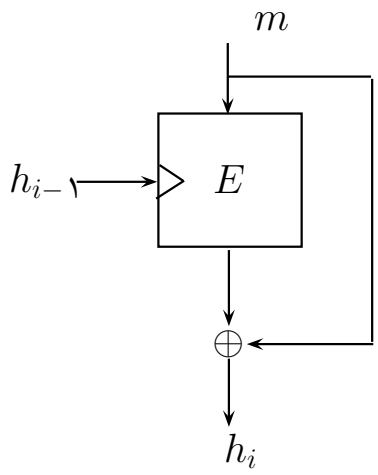


می‌توان ثابت کرد اگر E شبه تصادفی باشد تابع تولید شده در این روش مقاوم در برابر برخورد است.

اُسیس-مایر

در این روش یک تابع فشرده ساز می‌سازیم که ورودی m, h_{i-1} را به h_i می‌نگارد. در این روش به شکل زیر عمل می‌کنیم.

[۹] Davies-Meyer Scheme
[۱۰] Oasis-Meyer Scheme



می‌توان ثابت کرد اگر E شبه تصادفی باشد تابع تولید شده در این روش مقاوم در برابر برخورد است. در این روش نسبت به روش دیویس-مایر محدودیتی وجود دارد زیرا باید طول کلید و طول ورودی در E برابر باشد. بنابراین تابع تولید شده طول ورودی را نصف می‌کند.