مبانی رایانش امن
الگوریتم های یکپارچگی
تابع های چکیده ساز- اصالت سنجی پیام

محسن هوشمند
دانشکده تکنولوژی اطلاعات و علم رایانه
دانشگاه تحصیلات تکمیلی علوم پایه زنجان

تصدیق پیام

رمزگذاری
- محافظت در مقابل حملهٔ منفعل
  - شنود

نیاز به محافظت در مقابل حملهٔ فعال
- تحریف داده و ارسال

تصدیق پیام
- Message Authentication
- اصالت‌سنجی پیام؟

جنبه‌ها
- بررسی تغییر نیافتن محتوای پیام
- اعتبار فرستندهٔ پیام
- همچنین
  - به موقع بودن زمان پیام
  - درستی ترتیب پیام

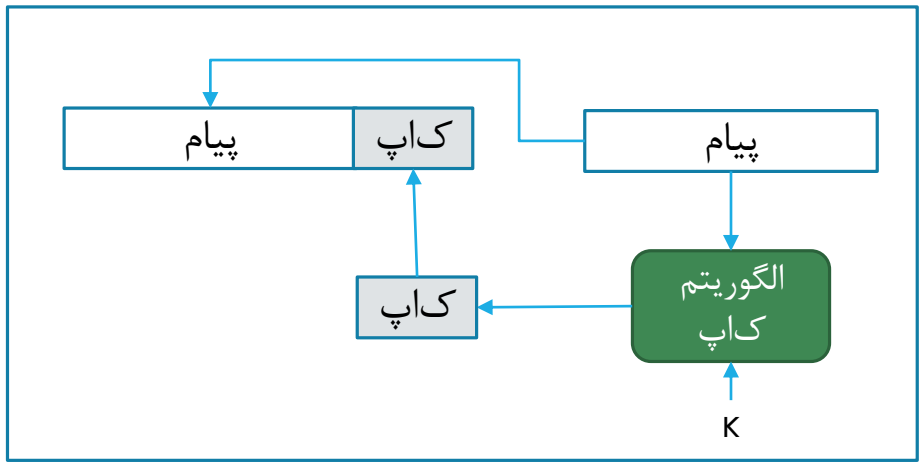
Message Authentication

کد اصالت‌سنجی پیام

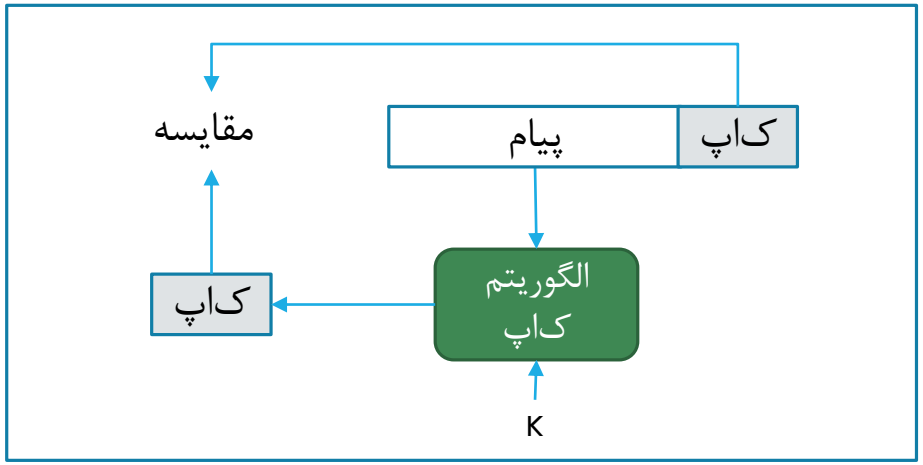
Message Authentication Code

تصدیق پیام

- غالبا با استفاده از کد اصالت‌سنجی پیام (کاپ)MAC
- همچنین مشهور به تابع درهم‌ساز با کلید
- استفاده از اصالت‌سنج بین دو طرف دارای کلید مشترک



انتقال



کد اصالت‌سنجی پیام

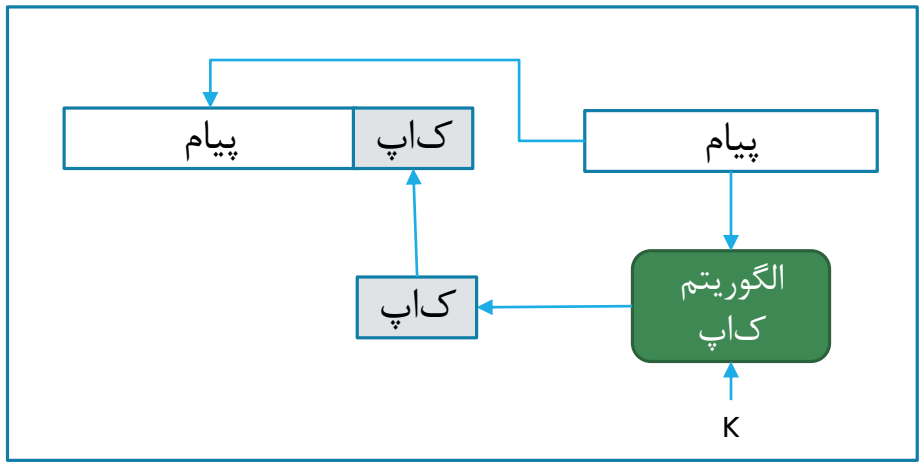
اطمینان گیرنده از عدم‌تغییر پیام

- با فرض محرمانه بودن کلید مشترک

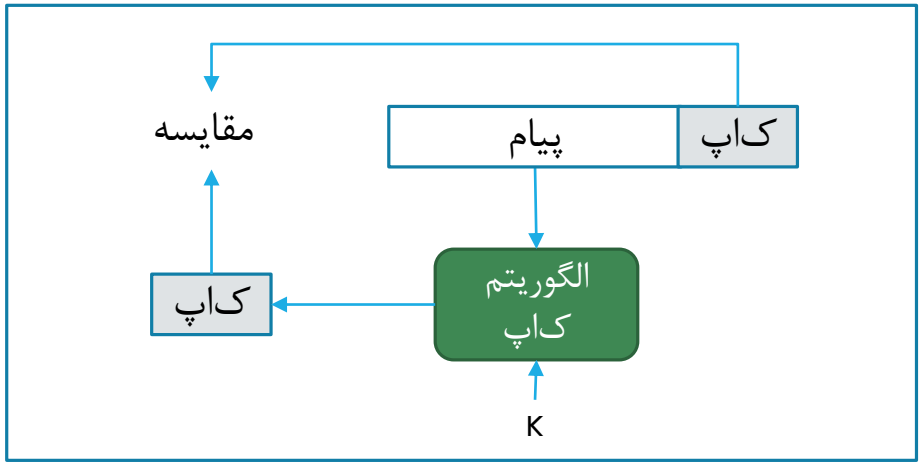
اطمینان دریافت از فرستندهٔ اصلی

در صورت شماره‌داری

- اطمینان از ترتیب درست



انتقال

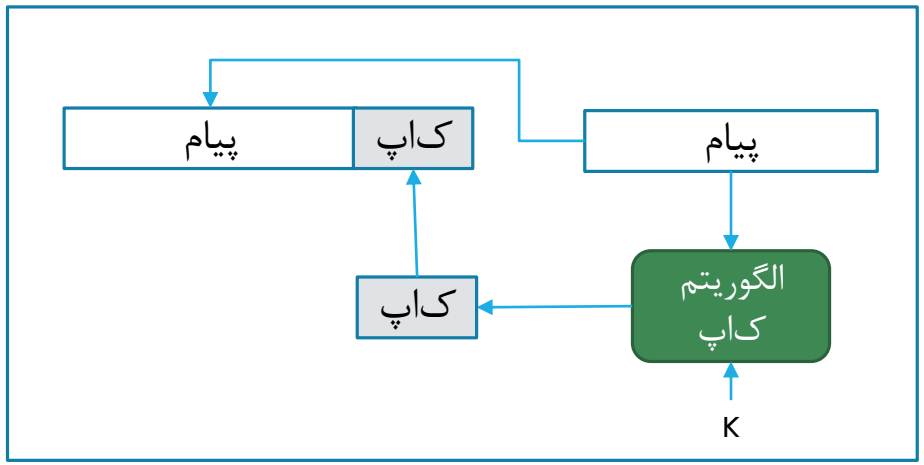


کد اصالت‌سنجی پیام

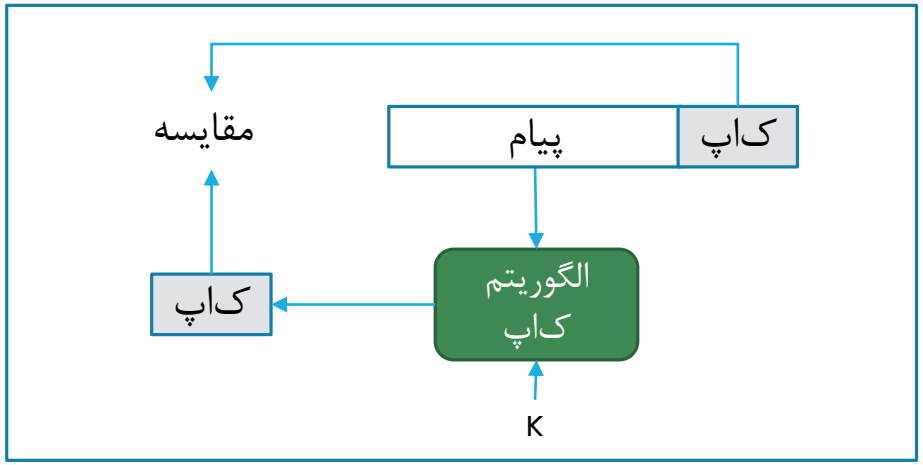
پس امکان استفاده از رمزنگاری برای یکپارچگی

عدم لزوم برگشت‌پذیری تابع اصالت‌سنجی

- آسیب کمتر



انتقال



توابع درهم‌ساز یک‌طرفه

تابع درهم‌ساز
- روشی دیگر برای اصالت‌سنجی پیام
- پذیرش بلوک داده‌ای با اندازه متغیر
- عدم لزوم وجود کلید رمز
- برگشت مقدار درهم با طول ثابت

ویژگی تابع درهم خوب
- توزیعی بودن خروجی‌های حاصل از اعمال بر مجموعه داده بزرگ
- همچنین خروجی دارای نمودی تصادفی

هدف اصلی تابع درهم
- یکپارچگی داده
- تغییری در بیتی از پیام ورودی با احتمال بالا منجر به تغییر در مقدار درهم

تابع درهم دارای انواعی
- موردی که در امنیت و رمزنگاری کاربرد دارد
  - نوع تابع درهم یک‌طرفه (یا تابع درهم رمزنگاری)

تابع درهم‌ساز یک‌طرفه
- خاصیت یک‌طرفگی- عدم امکان یافتن ورودی با داشتن تابع درهم
- خاصیت بدون تصادمی- عدم امکان یا غیرمحتمل بودن با درجهٔ بالا یافتن دو مقدار با مقدار یکسان درهم

کاربردها

تابع درهم‌ساز رمزنگاری محتملا همه‌فن‌حریف‌ترین الگوریتم سپهر رمزنگاری

کاربرد در گسترهٔ سخت وسیعی از کاربردهای امنیتی و پروتکل‌های اینترنت

- اصالت‌سنجی یا تصدیق پیام
  - سازوکار یا خدمتی جهت تایید اعتبار یکپارچگی پیام
  - به دنبال اطمینان بخشی به یکسانی دادهٔ دریافتی با دادهٔ ارسالی
  - به دیگر سخن، عدم وجود هیچ تغییری در مقدار در اثر افزودن، کاهیدن، یا تغییر چینش
- سازوکار و احراز تائیدگر درستی هویت فرستنده

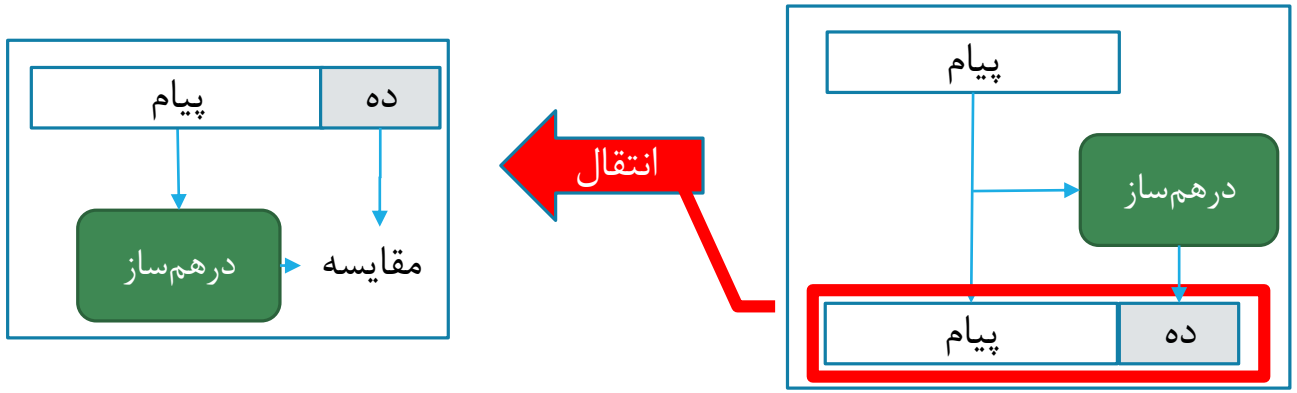
مواقعی که تابع درهم به عنوان احراز صحت پیام استفاده می‌شود،

- خوانده شدن مقدار خروجی تابع با فشردهٔ (افشره- مخلص- خلاصه)

دلیل ضرورت استفاده از تابع درهم در یکپارچگی

- فرستنده مقدار درهم را حساب می‌کند
- ارسال هر دوی پیام و مقدار درهم
- اعمال محاسبات یکسانی از تابع درهم بر پیام در گیرنده
- مقایسه با مقدار دریافتی درهم
- اطلاع از تغییر در پیام (یا مقدار درهم) در صورت وجود عدم‌تطابق

استفاده از تابع درهم‌ساز برای وارسی یکپارچگی

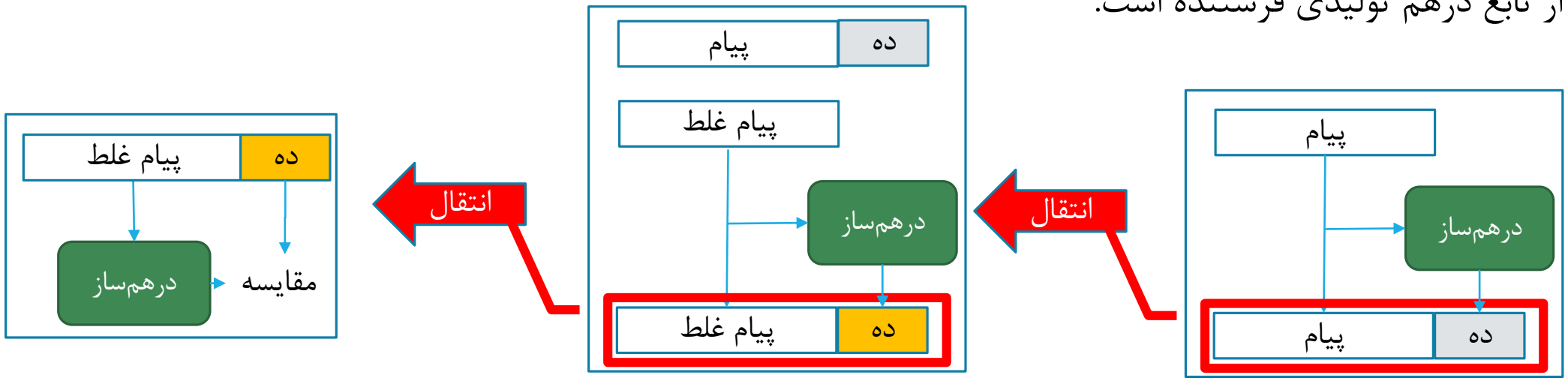


ارسال مقدار درهم

امکان تغییر مقدار درهم در میان راه جهت گمراه‌سازی گیرنده

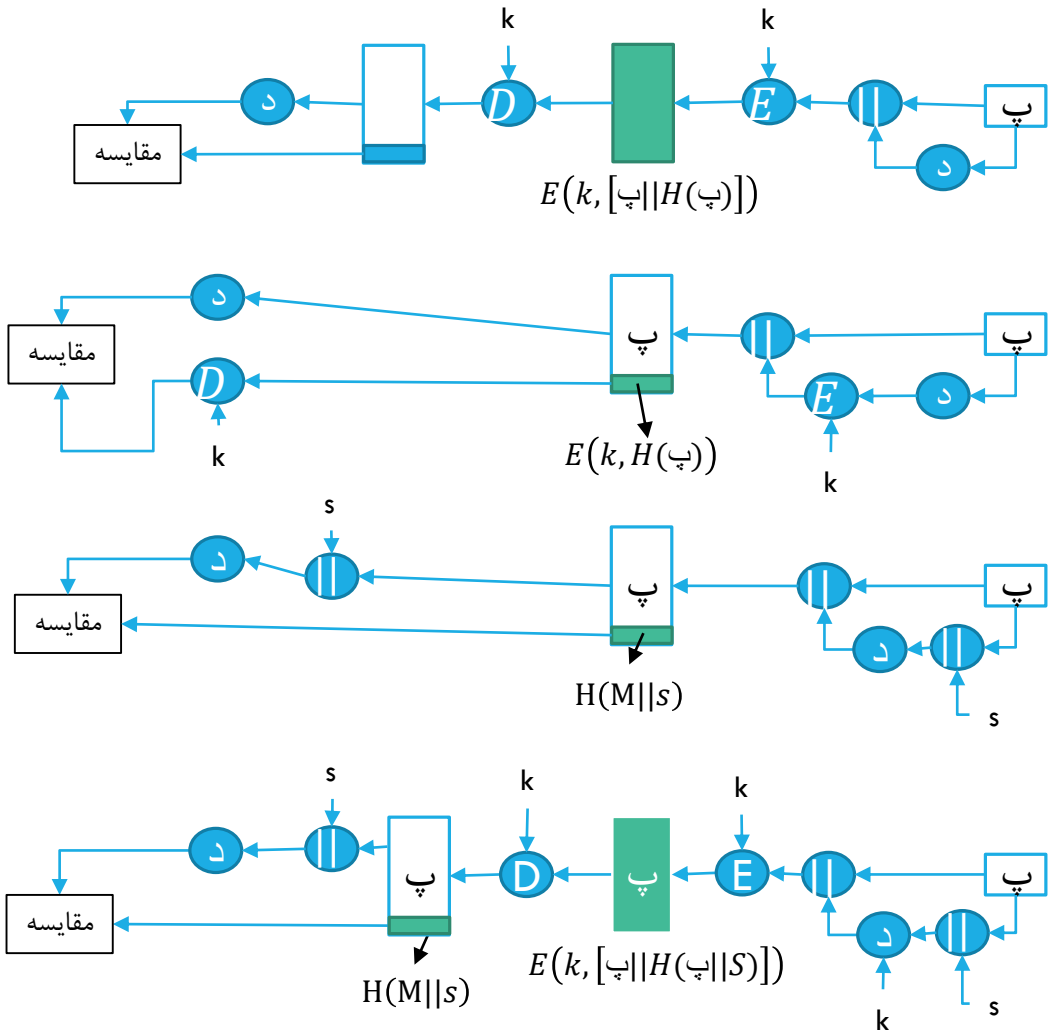
لزوم ارسال امن مقدار درهم

تصویر زیر مورد اخیر را نشان میدهد. شنونده میان راه پیام را تغییر و متناسب با آن تابع درهم را نیز تغییر می‌دهد. بنابرانی گیرنده دستکاری داده را تشخیص نمی‌دهد. جهت جلوگیری از چنین حمله‌ای، نیاز به محافظت از تابع درهم تولیدی فرستنده است.



حملهٔ نشسته در میان

انواع راه‌های کاربرد درهم به مثابهٔ اصالت‌سنج پیام



۱- رمز پیام و مقدار درهم الحاقی به آن با رمزنگاری متقارن

- به دلیل اشتراک کلید مخفی بین الف و ب
- پیام باید از الف آمده باشد و در میانه تغییری نکرده باشد.
- مقدار درهم نیز ساختار یا افزونگی لازم جهت احراز پیام را داراست
- وجود محرمانگی به دلیل اعمال رمزنگاری به تمامی پیام و درهم

۲- رمز مقدار درهم رمز با رمزنگاری متقارن

- موجب کاهش سربار محاسباتی در موقعیت‌هایی بدون‌نیاز به محرمانگی

۳- فرض بر اشتراک مقدار مخفی S بین فرستنده وگیرنده

- الحاق مقدار مخفی S به پیام اصلی و محاسبهٔ ردهم آنها
- سپس پیام و مقدار درهم را الحاق میکند و ارسال بدون رمز
- بدلیل عدم ارسال مقدار مخفی S رقیب میان راه نمی‌تواند تغییری ایجاد کند.

۴- افزودن محرمانگی به روش سوم

انواع راه‌های کاربرد درهم به مثابهٔ اصالت‌سنج پیام

در صورت عدم لزوم محرمانگی
- ارجحیت روش ۲ بر روش ۱ و ۴
- به دلیل نیاز به محاسبات کمتر

اقبال عمومی به روش‌های فاقد رمزنگاری
- چند دلیل بر اساس تسود۹۲ (مشخص با سرخ)
  - کندی نسبی نرم‌افزار رمزنگاری
  - هر چقدر هم که مقدار داده جهت رمزنگاری در هر پیام کوچک باشد، باز ممکن است که دنباله مداومی از پیام‌ها به و از سیستم وجود داشته باشد.
- کم بودن هزینه‌های سخت‌افزار رمزنگاری
  - پیاده‌سازی تراشه‌ای ارد موجود است
  - اما شیب بالای هزینه با افزودن گره‌های جدید به شبکه
- بهینه بودن سخت‌افزار رمزنگاری برای داده‌های حجم بالا
  - جهت بلوک‌های کوچک داده، بیشتر زمان صرف افزونگی‌های مقداردهی اولیه و جز این
- ثبت تجاری رمزنگاری‌ها و نیاز به پرداخت هزینه برای کسب پروانه

امضای دیجیتال

کاربرد مهم دیگر تابع درهم

- مشابه اصالت‌سنجی پیام
- فرایند امضای دیجیتال شبیه کاپ
  - اما دارای تفاوت‌هائی

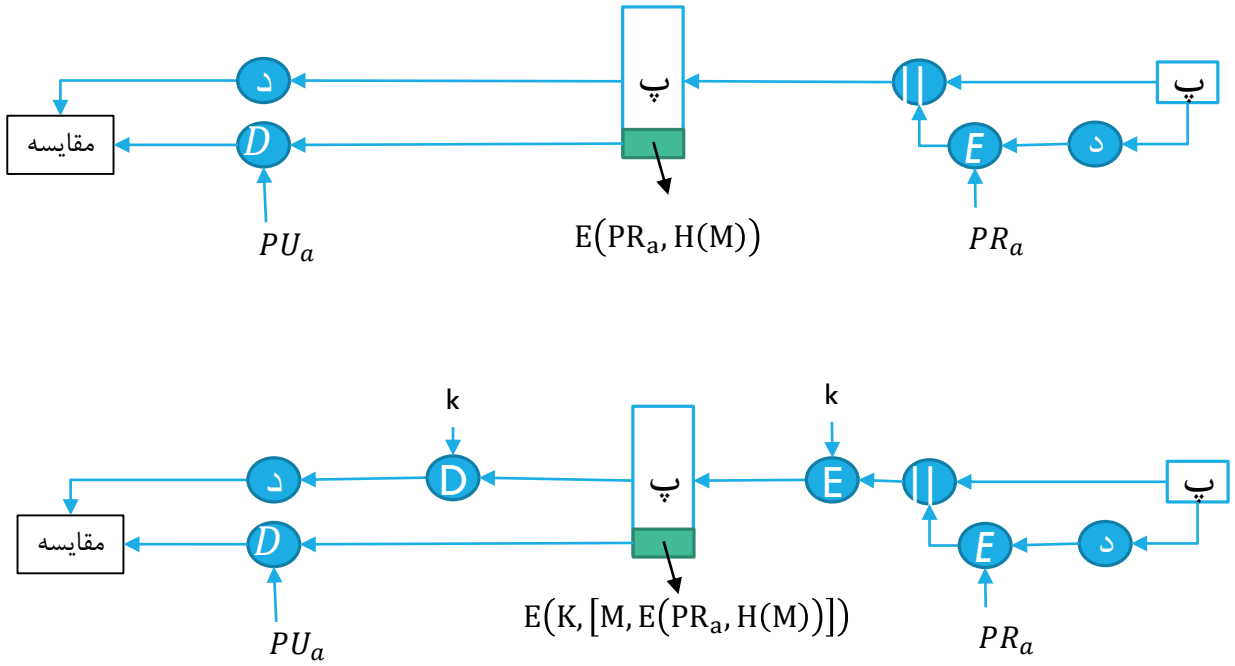
امضای دیجیتال

- رمز مقدار درهم پیام با کلید خصوصی فرستنده
- طرف دیگر کلید عمومی فرستنده را می‌داند
- حفظ یکپارچگی پیام با امضای دیجیتال
- در این موقعیت، مهاجمی که قصد تغییر پیام را دارد نیاز است که کلید خصوصی فرستنده را بداند.

۱- رمز مقدار درهم با رمزنگاری کلید-عمومی و کلید خصوصی فرستنده

۲- در صورت نیاز به محرمانگی، امکان وجود رمز متن و رمز مقدار دهم با رمزنگاری متقارن یا غیرمتقارن

امضای دیجیتال



پاکت دیجیتال

ضعف‌های نشانیِ

- رمزگذاری کلید عمومی
  - محاسبات کند
  - کاهش سرعت انتقال
  - افزایش زمان پردازش
- رمزگذاری متقارن
  - خطوط انتقال ناامن
- استفاده رمزگذاری کلید متقارن جهت رمزکردن داده
- استفاده از رمزگذاری کلید عمومی جهت رمزگذاری و ارسال کلید متقارن

کاربردهای دیگر

استفاده معمول از تابع درهم در ایجاد فایل رمز یک‌طرفه

- one-way password file
- استفاده در بیشتر سیستم عامل‌ها
  - جهت جلوگیری از دستیابی مهاجمان به رمزها
  - بررسی کنید

تشخیص نفوذ intrusion detection

تشخیص ویروس Virus detection

- H(F) هر فایل را روی سیستم ذخیره و مقدار درهم را امن کنید.
- بعدا می‌توان بررسی کرد که آیا فایل تغییر کرده است یا نه
- انجام با بازمحاسبهٔ H(F)

ایجاد تابع شبه‌تصادفی یا مولد عدد شبه‌تصادفی

تابع‌های درهم‌ساز

تابع‌های درهم‌ساز

ابتدا معرفی دو روش غیرامن تابع درهم ساز

عمل تمامی تابع‌های درهم‌ساز مبنی بر اصل‌های معمولی

- تقسیم ورودی به بلوک‌های n بیتی
- پردازش کل هر بلوک در یک زمان
- تولید مقدار درهم n بیتی

تابع‌های درهم‌ساز

یاانحصاری بیت به بیت هر بلوک

- از ساده‌ترین تابع‌های درهم‌ساز

$$C_i = b_{i1} \oplus b_{i2} \oplus \cdots \oplus b_{im}$$

به طوری که C_i برابر با i-امین بیت از کد درهم‌ساز

m تعداد بلوک‌های n بیتی

b_{ij} بیت i-ا،م از بلوک j-اُم

$\oplus$ عملگر یاانحصاری

مشابه بیت توازن برای موقعیت هر بیت

- شهره به وارسی افزونگی طولی

تابع‌های درهم‌ساز

شیفت چرخشی تک بیتی یا چرخش

- راه دیگر برای بهبود

الف- مقداردهی اولیه مقدار درهم n-بیتی به صفر

ب- پردازش هر بلوک n بیتی داده به شرح زیر

۱- چرخش مقدار فعلی درهم‌ساز به چپ به اندازه یک بیت

۲- یای انحصاری بلوک با مقدار درهم

کاملتر بودن اثر تصادفی‌سازی ورودی

غلبه بر وجود هر نظم و الگوئی در ورودی

- اما این روش نیز در امنیت داده محل اعراب ندارد. چرا؟

نیازمندی‌ها و امنیت

در هر تابع درهم h=H(x) داریم:

x را پیش‌تصویر preimage مقدار h می‌خوانیم. در واقع x بلوک داده‌ای است که مقدار درهم آن با تابع H برابر با h است. چون H نگاشتی چند به یک است، به ازای هر h چندین مقدار پیش‌تصویر وجود دارد.

تصادم برابر است با $x \neq y, H(x) = H(y)$

به دلیل استفاده از تابع‌های درهم در یکپارچگی داده، تصادم امری نامطلوب است.

فرض کنید طول مقدار درهم n بیت است. و تابع H مقادیری b بیتی به عنوان ورودی دارد و b>n. بنابراین، تعداد کل پیام‌ها برابر ۲^bو تعداد کل درهم‌ها ۲^n است.

به طور میانگین، هر مقدار درهم متناظر با ۲^{b-n} پیش‌تصویر است.

اما بهتر است بیشتر تحلیل کنیم

نیازمندی‌های امنیت برای تابع‌های درهم رمزنگاری

نیازمندی	توضیح	
اندازه متغیر ورودی	اعمال H به هر اندازه از بلوک داده ممکن باشد.	
اندازه ثابت خروجی	H خروجی با طول ثابت تولید کند	
کارائی	محاسبهٔ H(x) برای هر ورودی x نسبتا ساده باشد تا هر دو پیاده‌سازی نرم‌افزاری و سخت‌افزاری را عملی کند.	
مقاومت پیش‌تصویر (خاصیت یکطرفگی)	برای مقدار داده‌شدهٔ h، یافتن y که H(y)=h باشد رایانشن ناممکن باشد.	[MENE97]
مقاومت پیش‌تصویر دوم (مقاومت تصادم ضعیف)	برای هر مقدار بلوک داده شدهٔ x یافتن $y \neq x, H(x) = H(y)$ رایانشن ناممکن باشد.	[MENE97]
مقاومت تصادم(مقاومت تصادم قوی)	یافتن هر جفتی (x,y) که $y \neq x, H(x) = H(y)$ رایانشن ناممکن باشد.	[MENE97]
شبه‌تصادف	خروجی H آزمون استانده شبه‌تصادف را بگذراند.	[JOHN05]

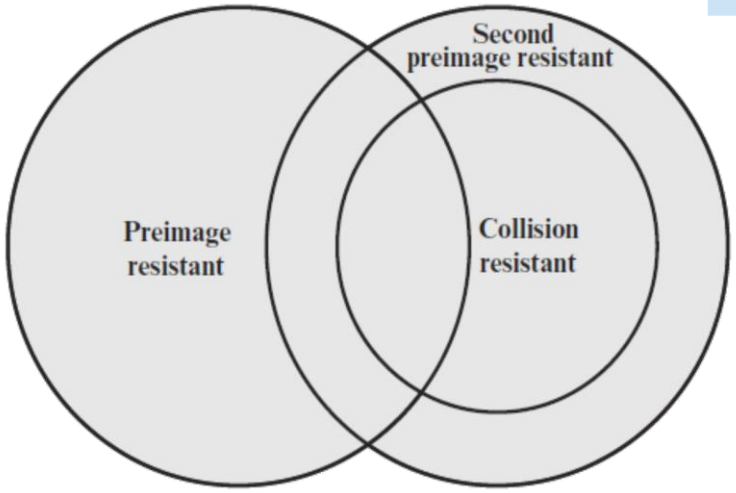


Figure 11.6 Relationship Among Hash Function Properties

نیازمندی‌های امنیت برای تابع‌های درهم رمزنگاری

نیازمندی‌های مقاومتی کاربردهای تابع درهم

	مقاومت پیش‌تصویر	مقاومت دوم پیش‌تصویر	مقاومت تصادم
درهم+ امضای دیجیتال	بله	بله	بله
تشخیص نفود و تشخیص ویروس		بله	
درهم+رمزنگاری متقارن			
فایل گذرواژهٔ یک‌طرفه	بله		
ک‌اپ	بله	بله	بله

حملات جستجو کامل

صرفا وابسته به طول بیت مقدار درهم

حملات پیش‌تصویر و پیش‌تصویر دوم-

- دو نوع حملات مذکور یافتن y به طوری‌که H(y) برابر با مقدار درهم داده‌شده h
- روش جستجوی کامل مقداری تصادفی از y را انتخاب می‌کند
- ادامه تا یافتن مقدار تصادم
- تعداد تلاش‌ها برابر با ۲^m است. پیوست U

حملات مقاومت تصادم-

- به دنبال یافتن دو بلوک داده که منجر به تابع درهم یکسان شود.

الگوریتم درهم‌ساز امن (شا)

پراستفاده‌ترین تابع درهم در سال‌های اخیر

توسعه در سازمان ملی استاندارد و فناوری

با یافتن ضعف‌هائی در نسخه شا-0
- معرفی شا-۱ در سال‌های بعدی
  - شا-۱ مقدار درهم ۱۶۰ بیتی

در سال ۲۰۰۲ استانداردهٔ دیگری
- دارای سه نسخهٔ ۲۵۶، ۳۸۴، ۵۱۲ بیتی بود
- به ترتیب به شا-۲۵۶، شا-۳۸۴، و شا-۵۱۲
- تمامی نسخه‌های مذکور مشهور به شا-۲ هستند.
- شا-۲ ساختاری شبیه به شا-۱

نسخه بعدی در سال ۲۰۰۸
- ۲۲۴ بیتی

معرفی نسخه‌های دیگر در سال ۲۰۱۵
- کچاک
- الگوریتم اسفنجی
- شا-۳

Secure Hash Algorithm (SHA)

منطق شا-۵۱۲

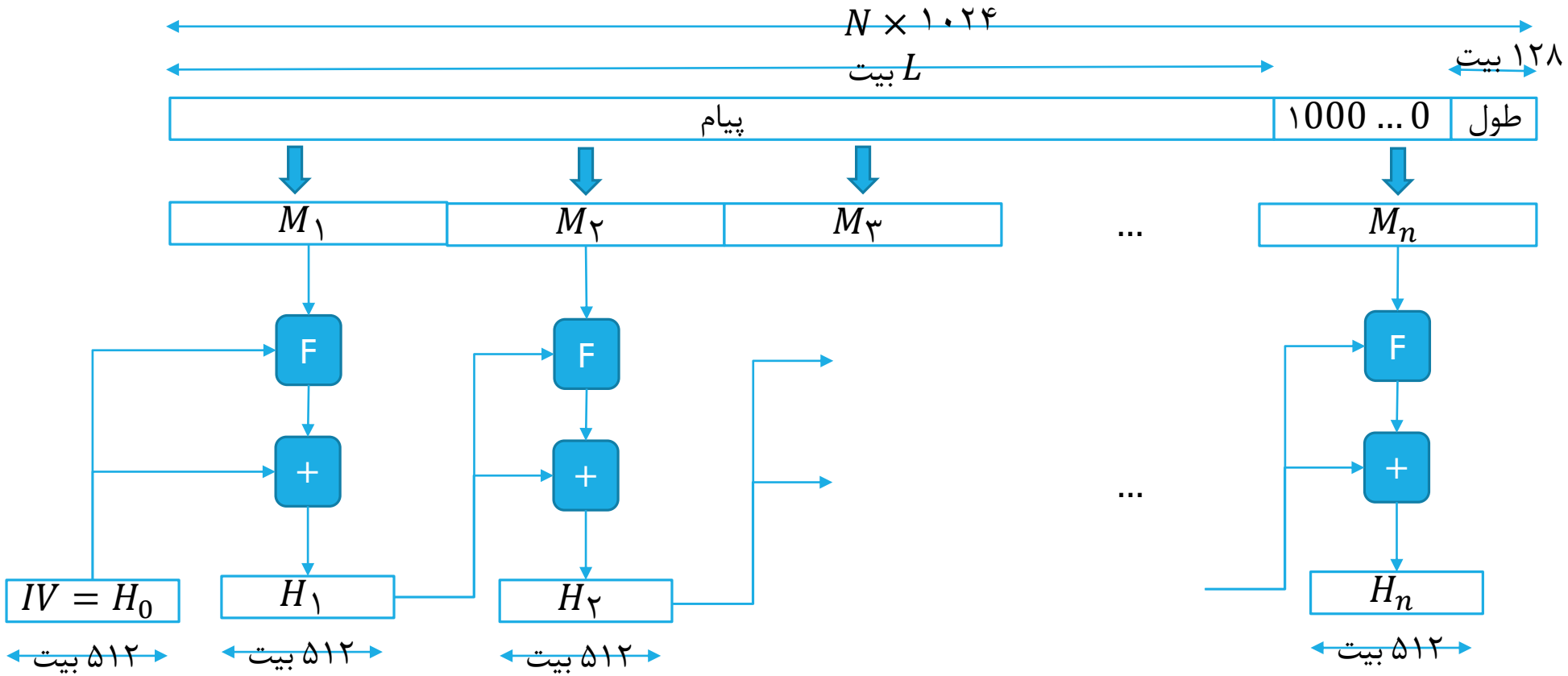
ورودی الگوریتم پیامی با طول کمتر از $۲^{۱۲۸}$ بیت

تولید خروجی چکیده پیام ۵۱۲ بیتی

تنظیم ورودی در قابل بلوک‌های ۱۰۲۴ بیتی

فرایند کلی تولید چکیده از پیام

منطق شا-۵۱۲


$N \times ۱۰۲۴$
L بیت
۱۲۸ بیت
پیام
۱000 ... 0
طول
$M_۱$
$M_۲$
$M_۳$
...
M_n
F
+
$IV = H_0$
$H_۱$
$H_۲$
H_n
۵۱۲ بیت

منطق شا-۵۱۲

۱- افزودن بیت‌های لاگذاری- پیام به اندازه‌ای لاگذاری می‌شود که هم‌ارز ۸۹۶ بیت

۲- افزودن طول- ۱۲۸ بیت در انتهای پیام

- مقدار مذکور عددصحیح بدون علامت UNSIGNED INTEGER در نظر میآیند و طول پیام اصلی را نشانی می‌دهد.

نتیجه لاگذاری و مقدار طول منجر به پیامی با طول مضربی از ۱۰۲۴ بیت خواهد شد.

۳- مقداردهی اولیهٔ بافر درهم- بافری ۵۱۲ بیتی جهت نگهداری مقداری میانی و نهایی تابع درهم استفاده می‌شود. بافر را می‌توان با هشت ثبات ۶۴ بیتی a و b و c و d و e و f و g و h نمایش داد. مقداردهی اولیهٔ ثبات‌ها با رقم شانزده‌شانزدهی به شرح زیر است

a = 6A09E667F3BCC908

b = BB67AE8584CAA73B

c = 3C6EF372FE94F82B

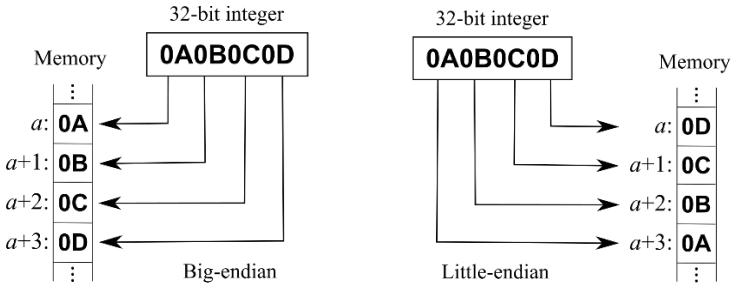
d = A54FF53A5F1D36F1

e = 510E527FADE682D1

f = 9B05688C2B3E6C1F

g = 1F83D9ABFB41BD6B

h = 5BE0CD19137E2179



مقادیر به صورت آخر-بزرگ در حافظه ذخیره می‌شوند. کلمات از ۶۴ بیت آعازین بخش صحیح ریشهٔ دوم هشت عدد اول گرفته شده‌اند.

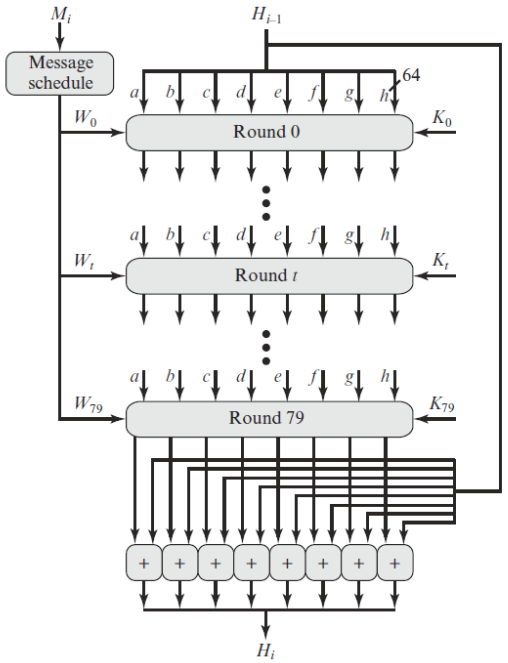


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

۴- پردازش پیام در بلوک‌های ۱۰۲۴ بیتی (۱۲۸ بایتی)-

- قلب مدل شامل پیمانه‌ای با هشتاد دور
- نمایش پیمانهٔ مذکور با F
- چگونگی انجام کار در شکل
- در هر دور ورودی
  - گرفتن ۵۱۲ بیتی از مقدار بافر abcdefgh
  - سپس بروز کردن محتوای بافر

پیام i: M_i

- *با استفاده از زمان‌بندی تقسیم پیام به هشتاد* W_t

استفاده از مقدارهائی ثبات در هر دور K_t و $0 \leq t \leq ۷۹$

- بدست آمده از ۶۴ بیت نخست ریشهٔ سوم هشتاد عدد اول نخست
- عرضهٔ الگوئی تصادفی

K_t

استفاده از مقدارهائی ثبات در هر دور K_t و $0 \leq t \leq ۷۹$

- بدست آمده از ۶۴ بیت نخست ریشهٔ سوم هشتاد عدد اول نخست
- عرضهٔ الگوئی تصادفی

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

۵- خروجی

$$H_0 = IV$$
$$H_i = (H_{i-۱} + abcdefgh_i)\%۲^{۶۴}$$
$$MD = H_N$$

IV مقدار اولیهٔ بافر

$abcdefgh_i$ خروجی آخرین دور پردازش بلوک i-ام

N تعداد بلوک‌های پیام

جمع جداگانهٔ هر کلمه از حفت ورودی‌ها به پیمانهٔ $۲^{۶۴}$

MD چکیدهٔ نهائی

تابع دور شا-۵۱۲

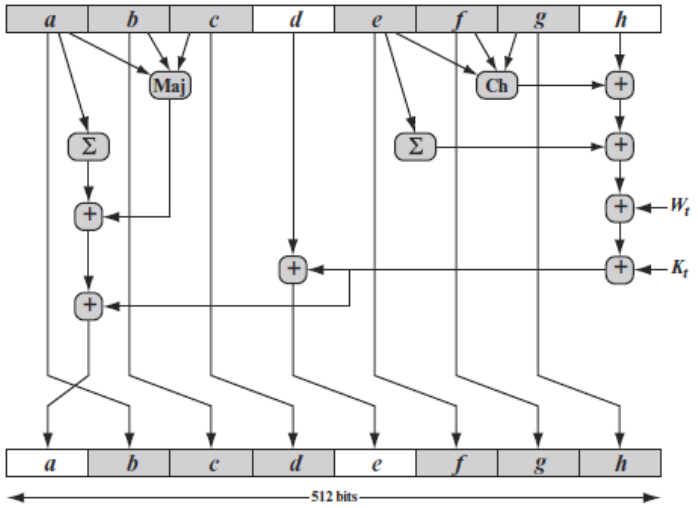


Figure 11.11 Elementary SHA-512 Operation (single round)

هر دور با مجموعه شامل توابع زیر

- $0 \leq t \leq ۷۹$
- $Ch(e,f,g)$ تابع شرط اگر e آن‌گاه f وگرنه g
- $Maj(a,b,c)$ درست در صورتی که اکثریت ورودی‌ها درست باشند (دو یا سه)
- $\sum_{۱}^{۵۱۲} a = RoTr^{۲۸}(a) \oplus RoTr^{۳۴}(a) \oplus RoTr^{۳۹}(a)$
- $\sum_{۱}^{۵۱۲} e = RoTr^{۱۴}(e) \oplus RoTr^{۱۸}(e) \oplus RoTr^{۴۱}(e)$
- $RoTr^{n}(x)$ شیفت چرخشی راست ورودی ۶۴ بیتی به اندازهٔ nبیت

- $T_۱ = h + Ch(e,f,g) + \left(\sum_{۱}^{۵۱۲} e\right) + W_t + K_t$
- $T_۲ = \left(\sum_{۱}^{۵۱۲} a\right) + Maj(a,b,c)$
- $h = g$
- $g = f$
- $f = e$
- $e = d + T_۱$
- $d = c$
- $c = b$
- $b = a$
- $a = T_۱ + T_۲$

مشاهدات

الف- جایگشت شش کلمه از هشت کلمه

ب- صرفا تغییر دو مقدار a و e با جانشینی

- کلمهٔ e از متغیرهای ورودی d,e,f,g,h و کلمه دور W_t و ثابت K_t
- کلمهٔ a تابعی از تمامی متغیرهای ورودی به جز d و کلمه دور W_t و ثابت K_t

حال نیاز به دانستن نحوهٔ استخراج کلمهٔ ۶۴ بیتی W_t از پیام ۶۴ بیتی

شکل زیر نگاشت را نشان می‌دهد.

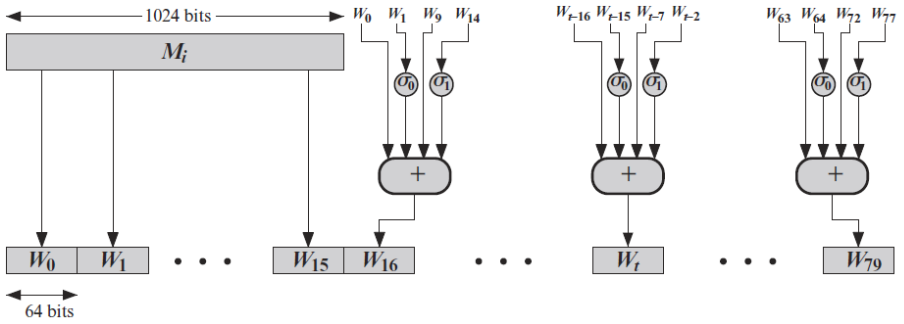


Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

شانزده مقدار اولیهٔ کلمات متناظر با بلوک ۲۰۱۴ بیتی است. برای ۶۴ قدم بعدی، مقدار W_t شامل شیفت چرخشی چپ

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\begin{aligned}
\sigma_0^{512}(x) &= \mathrm{ROTR}^1(x) \oplus \mathrm{ROTR}^8(x) \oplus \mathrm{SHR}^7(x) \\
\sigma_1^{512}(x) &= \mathrm{ROTR}^{19}(x) \oplus \mathrm{ROTR}^{61}(x) \oplus \mathrm{SHR}^6(x) \\
\mathrm{ROTR}^n(x) &= \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits} \\
\mathrm{SHR}^n(x) &= \text{right shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros on the left} \\
+ &= \text{addition modulo } 2^{64}
\end{aligned}$$

اقدامات مذکور میزان زیادی افزانگی و وابتسی به یکدیگر ایجاد می‌کند. و فرایند یافتن پیام اصلی را پیچیده می‌کند.

شا-۵۱۲ دارای این خاصیت است که هر بیت کد درهم تابعی از هر بیت ورودی است. تکار تابع نیز موجب ترکیب کامل بیت‌ها می‌شود و بنابراین حتی دو پیامی که درای شباهت‌هایی باشند غیرمحتمل است که نمیاش درهم یکسانی داشته باشند.

مثال مقدار اسکی abc

01100001 01100010 01100011

با لاگذاری خواهیم داشت:

0000000000000000 0000000000000000 0000000000000000 6162638000000000

0000000000000000 0000000000000000 0000000000000000 0000000000000000

0000000000000000 0000000000000000 0000000000000000 0000000000000000

0000000000000018 0000000000000000 0000000000000000 0000000000000000

تخصیص بلوک به کلمات W0 تا W۱۵ به تریتب زیر

W0 = 6162638000000000 W8 = 0000000000000000

W1 = 0000000000000000 W9 = 0000000000000000

W2 = 0000000000000000 W10 = 0000000000000000

W3 = 0000000000000000 W11 = 0000000000000000

W4 = 0000000000000000 W12 = 0000000000000000

W5 = 0000000000000000 W13 = 0000000000000000

W6 = 0000000000000000 W14 = 0000000000000000

W7 = 0000000000000000 W15 = 0000000000000018

مقداردهی اولیه a تا h به H00 تا H07

a	6a09e667f3bcc908	f6afceb8bcfcddf5	1320f8c9fb872cc0
b	bb67ae8584caa73b	6a09e667f3bcc908	f6afceb8bcfcddf5
c	3c6ef372fe94f82b	bb67ae8584caa73b	6a09e667f3bcc908
d	a54ff53a5f1d36f1	3c6ef372fe94f82b	bb67ae8584caa73b
e	510e527fade682d1	58cb02347ab51f91	c3d4ebfd48650ffa
f	9b05688c2b3e6c1f	510e527fade682d1	58cb02347ab51f91
g	1f83d9abfb41bd6b	9b05688c2b3e6c1f	510e527fade682d1
h	5be0cd19137e2179	1f83d9abfb41bd6b	9b05688c2b3e6c1f

خروجی دور آخر برابر است با

73a54f399fa4b1b2 10d9c4c4295599f6 d67806db8b148677 654ef9abec389ca9

d08446aa79693ed7 9bb4d39778c07f9e 25c96a7768fb2aa3 ceb9fc3691ce8326

مقدار درهم از طریق زیر بدست می‌اید

$H1,0 = 6a09e667f3bcc908 + 73a54f399fa4b1b2 = ddaf35a193617aba$

$H1,1 = bb67ae8584caa73b + 10d9c4c4295599f6 = cc417349ae204131$

$H1,2 = 3c6ef372fe94f82b + d67806db8b148677 = 12e6fa4e89a97ea2$

$H1,3 = a54ff53a5f1d36f1 + 654ef9abec389ca9 = 0a9eeee64b55d39a$

$H1,4 = 510e527fade682d1 + d08446aa79693ed7 = 2192992a274fc1a8$

$H1,5 = 9b05688c2b3e6c1f + 9bb4d39778c07f9e = 36ba3c23a3feebbd$

$H1,6 = 1f83d9abfb41bd6b + 25c96a7768fb2aa3 = 454d4423643ce80e$

$H1,7 = 5be0cd19137e2179 + ceb9fc3691ce8326 = 2a9ac94fa54ca49f$

چکیده پیام ۵۱۲ بیتی بربر است با

ddaf35a193617aba cc417349ae204131 12e6fa4e89a97ea2 0a9eeee64b55d39a

2192992a274fc1a8 36ba3c23a3feebbd 454d4423643ce80e 2a9ac94fa54ca49f

فرض کنید ورودی را به مقدار زیر تغییر دهیم

6362638000000000 0000000000000000 0000000000000000 0000000000000000

0000000000000000 0000000000000000 0000000000000000 0000000000000000

0000000000000000 0000000000000000 0000000000000000 0000000000000000

0000000000000000 0000000000000000 0000000000000000 0000000000000018

آنگاه چکیده برابر است با

531668966ee79b70 0b8e593261101354 4273f7ef7b31f279 2a7ef68d53f93264

319c165ad96d9187 55e6a204c2607e27 6e05cdf993a64c85 ef9e1e125c0f925f

خلاصه

متن لاگذاری شده شامل بلوک‌های M_1، M_2، ...، M_N است. هر پیام M_i دارای شانزده کلمهٔ ۶۴ بیتی M_{i0}، M_{i1}، ...، M_{i15} است. تمامی جمع‌ها به پیمانه 2^{64} انجام می‌پذیرد

$H_{00} = 6A09E667F3BCC908$	$H_{04} = 510E527FADE682D1$
$H_{01} = BB67AE858CAA73B$	$H_{05} = 9B05688C2B3E6C1F$
$H_{02} = 3C6EF372FE94F82B$	$H_{06} = 1F83D9ABFB41BD6B$
$H_{03} = A54FF53A5F1D36F1$	$H_{07} = 5BE0CD19137E2179$

for $i = 1$ **to** N

1. Prepare the message schedule W

for $t = 0$ **to** 15

$W_t = M_{i,t}$

for $t = 16$ **to** 79

$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$

2. Initialize the working variables

$a = H_{i-1,0}$ $\quad e = H_{i-1,4}$

$b = H_{i-1,1}$ $\quad f = H_{i-1,5}$

$c = H_{i-1,2}$ $\quad g = H_{i-1,6}$

$d = H_{i-1,3}$ $\quad h = H_{i-1,7}$

3. Perform the main hash computation

for $t = 0$ **to** 79

$$T_1 = h + \text{Ch}(e, f, g) + \left(\Sigma_1^{512} e\right) + W_t + K_t$$

$$T_2 = \left(\Sigma_0^{512} a\right) + \text{Maj}(a, b, c)$$

$h = g$

$g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

4. Compute the intermediate hash value

$H_{i,0} = a + H_{i-1,0}$ $\quad H_{i,4} = e + H_{i-1,4}$

$H_{i,1} = b + H_{i-1,1}$ $\quad H_{i,5} = f + H_{i-1,5}$

$H_{i,2} = c + H_{i-1,2}$ $\quad H_{i,6} = g + H_{i-1,6}$

$H_{i,3} = d + H_{i-1,3}$ $\quad H_{i,7} = h + H_{i-1,7}$

return $\{H_{N,0} \| H_{N,1} \| H_{N,2} \| H_{N,3} \| H_{N,4} \| H_{N,5} \| H_{N,6} \| H_{N,7}\}$

شا-۳

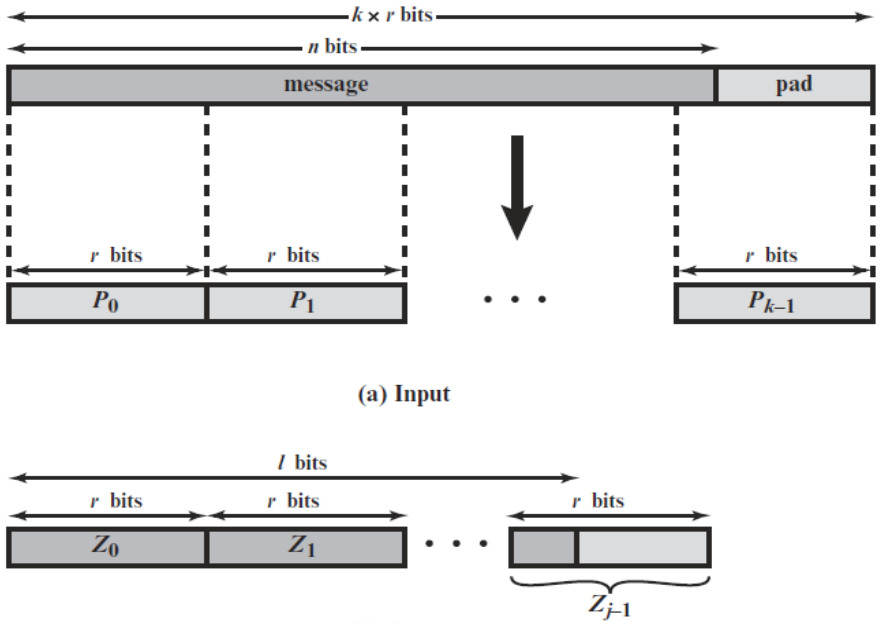


Figure 11.14 Sponge Function Input and Output

کچّاک به عنوان شا-۳ (شا-۳) انتخاب شد.

انعطاف بالای الگوریتم مذکور

- اجازهٔ انجام تمامی اعمال ابتدائی رمزنگاری متقارن
- اعم از رمز دنباله، درهم، درهم‌سازی درخت، تولید عدد شبه‌تصادفی، رمزنگاری اصالت‌سنجی، و جز اینها.

ایجاد اسفنج- دارای ساختاری شبیه تابع درهم دوری

تابع اسفنج ورودی را دریافت و تقسیم آن به بلوک‌هائی با اندازه ثابت

منابع

[استالینگز۱۶]

[استالینگز۱۷]