نویسنده

حمیدرضا عسگری

h.r.asgari52@gmail.com

منتشر شده توسط

کوین ایران

کوین ایران بزرگترین پایگاه خبری فارسی زبان در حوزه فناوری بلاکچین، ارزهای رمزنگاری شده و پلتفرم های مرتبط با بلاکچین است. این وب سایت در سال ۱۳۹۲ توسط بابک جلیلوند و آرش محبوب، با هدف اطلاع رسانی، آموزش و مشاوره به جامعه فارسی زبان علاقه مند به رمز ارزها راه اندازی شد و اولین مقاله آن در ۱۴ دی ماه همان سال، با عنوان «بیت کوین چیست؟» منتشر گردید . هدف کوین ایران از معرفی این فناوری ایجاد محیطی پژوهشی و آموزشی در راستای استفاده صحیح از این فناوری جهت ارائه تسهیلات و رفاه به جوامع فارسی زبان است.

www.coiniran.com

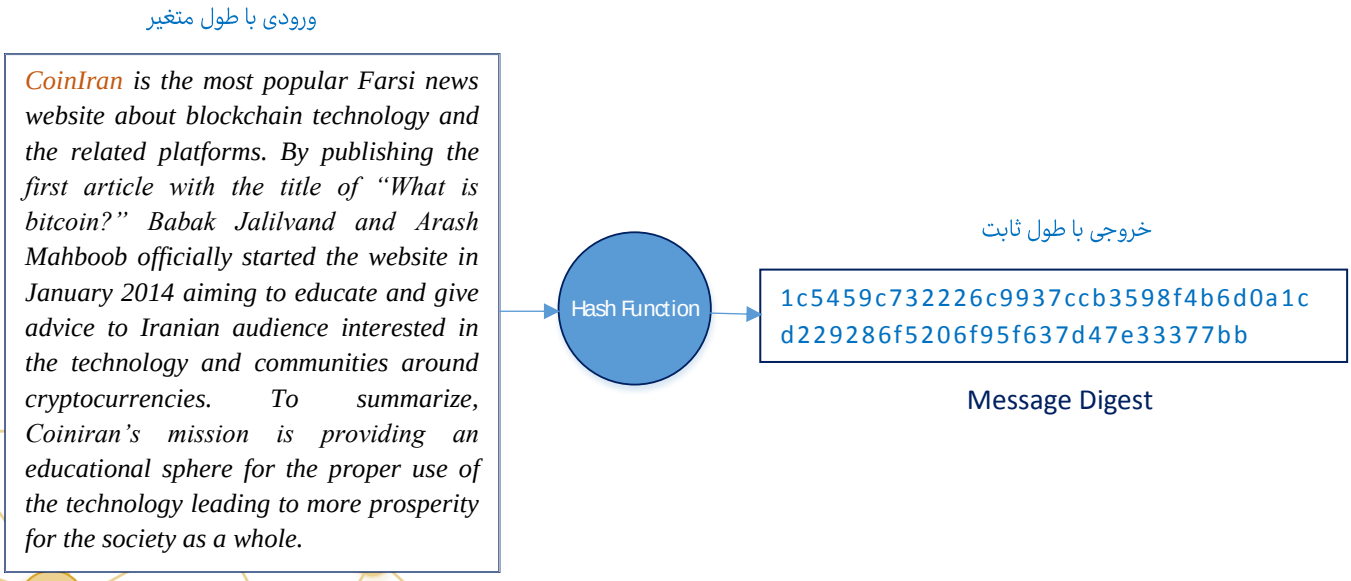
کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.

هَش چیست؟

یکی از توابع و مکانیزم هایی که در سیستم های رمزنگاری اطلاعات در کاربردهای مختلف بسیار با اهمیت است، توابع درهم سازی یا Hashing هستند. در واقع **هش** یکی از عناصر اصلی تشکیل دهنده سیستم ها و پروتکل های رمزنگاری است. از این رو لازم است تا قبل از ورود به مباحث رمزنگاری و رمز ارزها، آشنایی کافی با این مکانیزم حاصل شود.

تعریف تابع هش

تابع هش از یک ورودی با طول متغیر (و بدون محدودیت) از اطلاعات، یک خروجی درهم سازی شده و با طول ثابت تولید می‌کند.



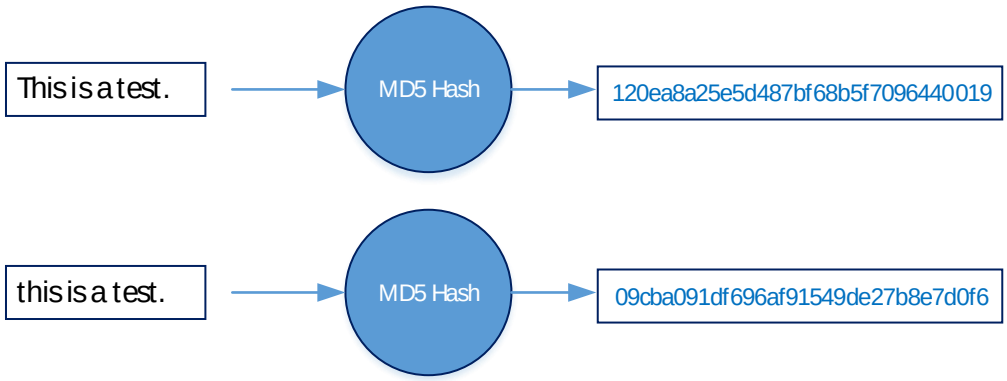
خروجی بصورت یک رشته درهم سازی شده و ناخوانا است که به آن Message Digest گفته می‌شود. پس همانگونه که در شکل فوق می‌بینید مهم نیست طول رشته ورودی چقدر باشد، در نهایت یک خروجی با طول ثابت حاصل خواهد شد. پس خروجی یک رشته از حروف و اعداد ناخوانا است. آیا می‌توانید حدس بزنید خروجی هش زیر از چه ورودی ایجاد شده است؟!

ec85e70b019d8220756f40e4a8893f429153f3b7b5bbf5d5b3cb0e9ebc50a6ac

کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.

نکته:

به ازای کوچترین تغییری در ورودی، خروجی تابع هش به کلی متفاوت خواهد شد. به مثال زیر توجه نمایید:



همانطور که در شکل مشاهده نمودید تنها با تغییر حرف T به t از کلمه اول، یک خروجی هش کاملا متفاوت ایجاد شده است.

مثال کاربردی: امضای دیجیتال Digital Signature

یک کاربرد جالب از تابع هش، در امضای دیجیتال است. مثلا فرض کنید که شما یک سند مهم دارید و می‌خواهید آن را برای کسی بفرستید و در عین حال می‌خواهید مطمئن شوید که این سند دست نخواهد خورد و تغییری در آن صورت نخواهد گرفت. کافیست که شما سند خود را با یکی از مکانیزم های رایج هش کرده و هش خروجی را به همراه سند برای طرف مقابل ارسال نمایید. طرف مقابل نیز پس از دریافت سند می‌تواند با اطمینان یک بار دیگر هش آن را محاسبه کرده و با هش ارسالی از شما مقایسه کند و چنانچه هر دو هش برابر بودند اطمینان می‌یابد که این سند همانی است که شما برای او فرستاده اید و کوچکترین تغییری در آن حاصل نشده است.

به این مکانیزم عدم تغییر در محتویات، Tamper resistance mechanism گفته می‌شود.

کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.

گستره وسیعی از تنوع توابع هش

امروزه توابع هش متنوعی برای کاربردهای مختلف وجود دارد و تعدادی از آنها در تکنولوژی های رمزنگاری مورد استفاده واقع شده است. از این تعداد نیز نمونه هایی به علت پیشرفت تکنولوژی در شکستن آنها و همچنین ضعف های امنیتی کنار گذاشته شده اند. به عنوان مثال MD5 یکی از این توابع است که امروزه در کاربردهای حساس دیگر استفاده نمی‌شود. در جدول زیر چند نمونه از این توابع و طول هرکدام ارائه شده است. هریک از این موارد مکانیزم متفاوتی برای محاسبه هش خروجی دارند.

Name	Length	Sample input	Output hash
MD2	128 bits	This is a test.	f6e409c1db988993c4d482250c2e0d4a
MD5	128 bits		120ea8a25e5d487bf68b5f7096440019
GOST	256 bits		e3cfc9d806e29f30e3887844dea0211b73d0eb57e174eb23b8fa02d54132062b
RIPEMD	128 bits		e42592307e02098ab66beee19a2a24a8
SHA-1	160 bits		afa6c8b3a2fae95785dc7d9685a57835d703ac88
SHA-256	256 bits		a8a2f6ebe286697c527eb35a58b5539532e9b3ae3b64d4eb0a46fb657b41562c
SHA-512	512 bits		f3bf9aa70169e4ab5339f20758986538fe6c96d7be3d184a036cde8161105fcf53516428fa096ac56247bb88085b0587d5ec8e56a6807b1af351305b2103d74b

کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.

سه مشخصه ویژه تابع هش

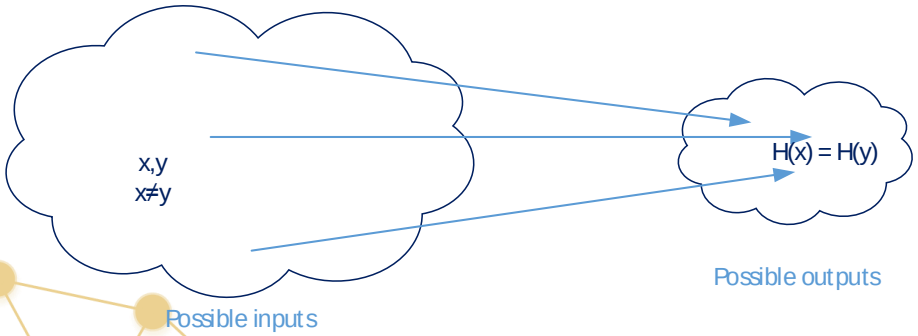
حال که با تعریف تابع هش آشنا شدیم، به تشریح سه خاصیت مهم تابع هش می‌پردازیم. این مشخصات عبارتند از:

۱. تناظر بین ورودی و خروجی – Collision Resistance
۲. مخفی سازی – Hiding
۳. مقاوم در برار حدس زدن مقدار ورودی – Puzzle Friendliness

تناظر بین ورودی و خروجی – Collision Resistance

این خاصیت بیان می‌کند که **یافتن** ورودی متفاوت دیگری که همان مقدار هش را تولید کند غیر ممکن است. به زبان ریاضی، نمی‌توان دو عدد x و y را پیدا کرد به طوری که $x \neq y$ باشد اما هش آنها یکسان شوند یعنی $H(x) = H(y)$

توجه به این نکته حائز اهمیت است که این خاصیت نمی‌گوید چنین x و y وجود ندارد، بلکه می‌گوید پیدا کردن این دو مقدار غیر ممکن است. از نظر ریاضی به علت اینکه x و y از یک مجموعه بزرگتر انتخاب می‌شوند و خروجی هش از مجموعه کوچکتری است، چنین اعدادی وجود دارد، اما پیدا کردن آن نیاز به صرف زمان نجومی دارد که عملا غیر ممکن است.



متدها و الگوریتم های ریاضی هش به گونه ای است که پیدا کردن x و y با خاصیت فوق بسیار بسیار سخت و مستلزم صرف زمان بسیار طولانی است. لذا می‌توان عملا آن را غیر ممکن نامید.

چقدر طول می‌کشد که بتوان خروجی یک هش را حدس زد؟!

اگر یک تابع هش دارای خروجی ۲۵۶ بیت باشد، در بدترین حالت می‌بایست $2^{256} + 1$ و به طور متوسط 2^{128} حالت محاسبه شود. با فرض داشتن سخت افزاری برای محاسبه ۱۰،۰۰۰ هش در ثانیه (h/s)، محاسبه این تعداد در حدود 10^{27} **سال** طول خواهد کشید. با تکنولوژی امروز با توانایی محاسبه **1 Peta hash/s** یعنی 10^{15} **هش در ثانیه**، زمان فوق در حدود **10,000,000,000,000,000 سال**!!! خواهد بود که یک عدد نجومی است و عملا غیر ممکن است.

کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.

مخفی سازی – Hiding

توضیح ساده خاصیت مخفی سازی در مکانیزم هش بیان می‌کند که از روی هش خروجی نمی‌توان دیتای ورودی را محاسبه کرد یا حدس زد، زیرا تابع هش یک تابع یک طرفه است. این مکانیزم در کاربردهای رمزنگاری و امنیتی بسیار مهم است. توضیح آن به زبان ریاضی دارای پیچیدگی هایی است که از حوصله این بحث خارج است اما به طور ساده بیان این مطلب به زبان ریاضی چنین است:

اگر بتوان یک مقدار r از یک توزیع بسیار یکنواخت (High-min entropy) به طور تصادفی انتخاب کرد، در آنصورت از خروجی هش تابع H(r|x)، حدس زدن مقدار x (یعنی اطلاعات ورودی) غیر ممکن است. پایه گذار این نظریه، Claude Elwood Shannon است که در سال ۱۹۴۸ برای اولین بار در مقاله ای با عنوان A Mathematical Theory of Communication این نظریه را مطرح نمود.

برای روشن تر شدن مطلب فوق، به این مثال توجه نمایید:

مثال کاربردی

در دنیای واقعی فرض کنید شما اطلاعاتی مثلا یک عدد یا هرچیز دیگری را در یک پاکت قرار می‌دهید تا بعدا از آن استفاده کنید. برای اینکه کسی نتواند آن اطلاعات را ببیند شما پاکت را لاک و مهر کرده و آن را روی میزتان می‌گذارید تا بعدا به آن مراجعه کنید. در طول این مدت همه افراد می‌توانند فقط روی پاکت را ببینند اما از درون آن اطلاعی ندارند. به این ترتیب با این کار شما نوعی تعهد داده اید (commitment) که اطلاعات داخل پاکت محرمانه و بدون تغییر خواهد ماند تا موقعی که بخواهید از آن استفاده کنید.

حال ببینیم در فضای دیجیتال پیاده سازی این مثال چگونه است. در اینجا فرض کنید که یک API داریم که کار های زیر را با آن انجام می‌دهیم:

Commitment API

```
(com, key) := commit(msg)
Match := verify(com, key, msg)
```

پس این API دو تابع دارد، تابع commit(*msg*) که رشته ورودی یعنی *msg* را گرفته و دو مقدار key و *com* را تولید می‌کند. از این تابع برای مخفی سازی *msg* (یعنی همان عمل لاک و مهر در مثال فوق) استفاده می‌کنیم و سپس مقدار *key* و *com* را منتشر می‌کنیم.

کسی که می‌خواهد نامه را باز کند یا به عبارتی محتوای *msg* را ببیند، با استفاده از مقدار *com* و *key* که در اختیار اوست صحت اطلاعات را توسط تابع verify چک می‌کند. چنانچه خروجی این تابع درست باشد (TRUE) در آنصورت مطمئن است که *msg* معتبر است و چنانچه خروجی تابع verify نادرست (FALSE) باشد، *msg* نامعتبر خواهد بود.

دو نکته امنیتی از اقدام فوق حاصل می‌شود:

- ✓ **با داشتن *com* حدس زدن یا پیدا کردن *msg* غیر ممکن است.**
- ✓ **نمی‌توان *msg'* دیگری پیدا کرد به طوری که msg ≠ msg' اما verify(commit(*msg*), *msg'*) == TRUE**

کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.

پیاده سازی واقعی این مثال توسط تابع hash انجام می‌شود.

```
commit(msg) := ( H(key | msg), key )
verify(com, key, msg) := ( H(key | msg) == com )
```

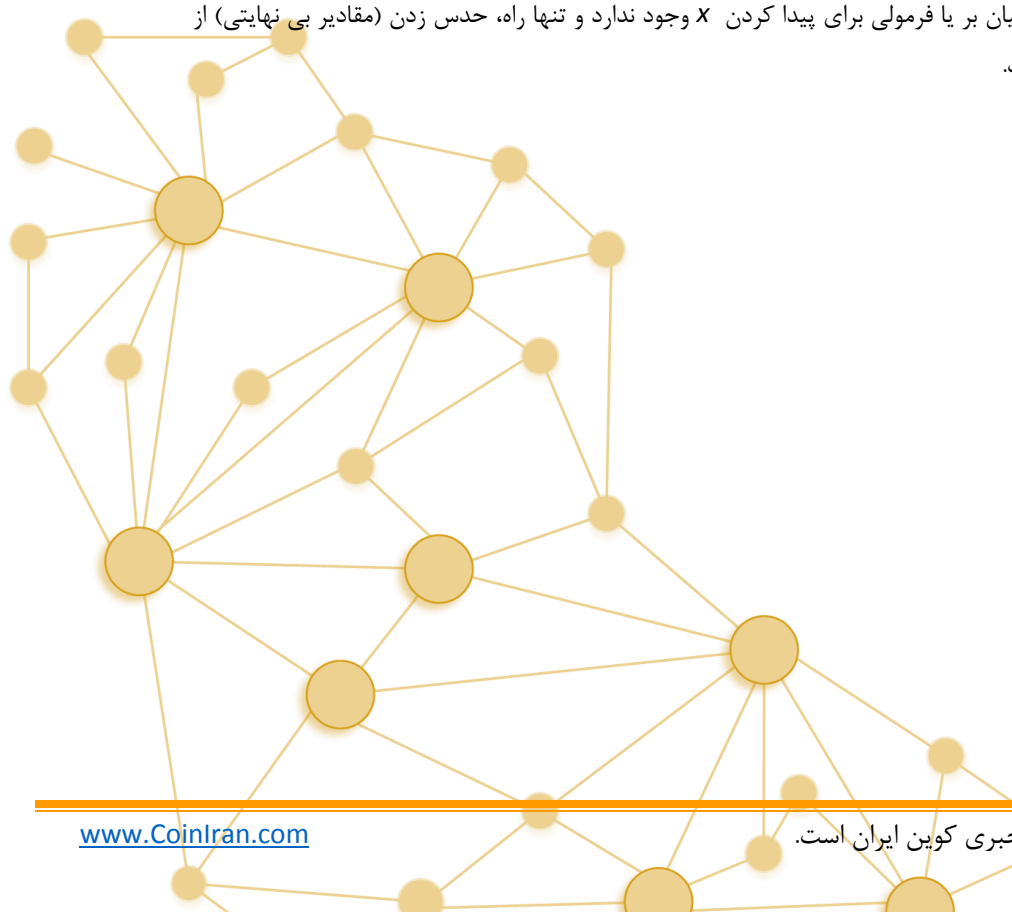
علامت | به معنی ادغام (concatenate) دو عبارت است. *key* یک مقدار تصادفی ۲۵۶ بیتی است و *com* نیز همان خروجی تابع hash است. پس بر اساس خاصیت دوم تابع هش، با داشتن خروجی H(*key* | *msg*) پیدا کردن مقدار *msg* غیر ممکن است.

همچنین غیر ممکن است بتوان مقدار دیگری مثل *msg'* یافت به طوری که *msg* ≠ *msg'* اما خروجی هش آن دو یکی باشد یعنی

H(*key* | *msg*) == H(*key* | *msg'*).

مقاوم در برار حدس زدن مقدار ورودی – Puzzle Friendliness

این خاصیت نیز بر اساس تئوری های ریاضی است و نسبتا پیچیده است. بر اساس این ویژگی، به ازای هر خروجی محتمل *y* اگر مقدار *k* از یک توزیع بسیار یکنواخت و تصادفی (high min-entropy) انتخاب شود، در آنصورت غیر ممکن است بتوان مقدار *x* را پیدا نمود به طوری که H(*k* | *x*) = *y* باشد. به عبارت دیگر هیچ راه میان بر یا فرمولی برای پیدا کردن *x* وجود ندارد و تنها راه، حدس زدن (مقادیر بی نهایتی) از حالت های مختلف است که عملا غیر ممکن است.



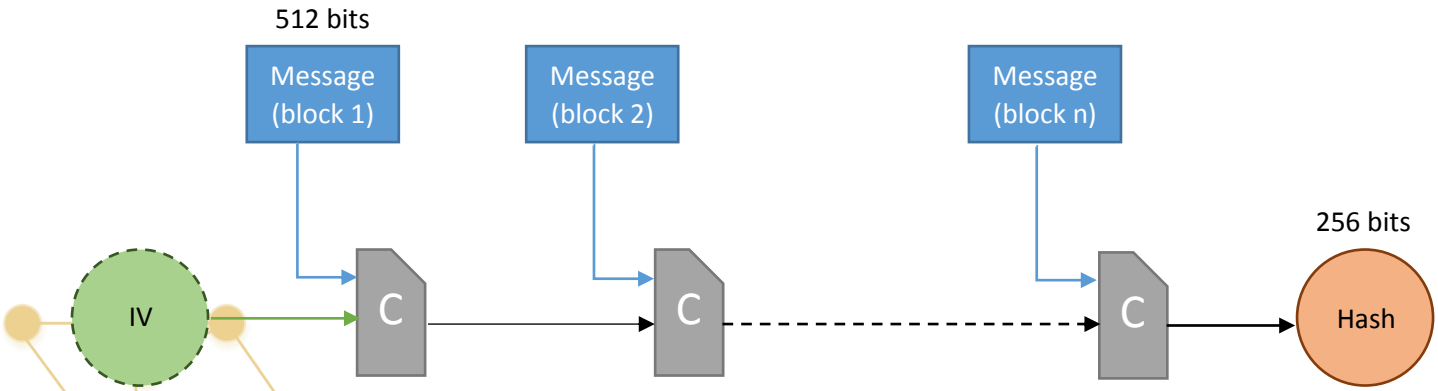
کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.

تابع هَش SHA-256 مورد استفاده در پروتکل بیتکوین

همانگونه که قبلا نیز اشاره شد خروجی این تابع هش ۲۵۶ بیتی است اما با هر مقدار طول اطلاعات ورودی کار می‌کند. برای این کار از یک تابع فشرده سازی استفاده می‌شود که دیتای ورودی با طول m را به بلاک هایی با طول m-n تبدیل می‌کنند (m > n) به این مکانیزم Merkle-Damgard transform گفته می‌شود. لذا برای هر یک از بلاک ها، هش جداگانه محاسبه و در نهایت یک هش کلی از همه آنها تولید می‌شود. در تابع SHA-256 ورودی به بلاک های ۵۱۲ بیت تقسیم می‌شود. این مکانیزم طوری عمل می‌کند که همواره ورودی را تبدیل به مضرب صحیحی از ۵۱۲ نموده و سپس آن را به بلاک های کوچکتر تقسیم می‌کند.

از نظر ریاضی اثبات شده هرگاه هریک از تکه های هش مربوط به هر بلاک کوچک دارای خاصیت collision resistance باشد، در آنصورت کل مکانیزم هش نیز دارای این خاصیت خواهد بود.

در شکل زیر نحوه کار این سیستم نشان داده شده است.



مقدار IV برای مکانیزم تابع SHA-256 بر اساس استاندارد FIPS 180-3 به اینصورت محاسبه می‌شود:

ریشه دوم هریک از اعداد اول ۲ تا ۱۹ محاسبه و اولین ۳۲ بیت بخش اعشار هرکدام از این ۸ عبارت را برداشته و در کنار هم قرار داده و یک مقدار ۲۵۶ بیتی به عنوان IV حاصل می‌شود.

کلیه حقوق این مستند متعلق به پایگاه خبری کوین ایران است.