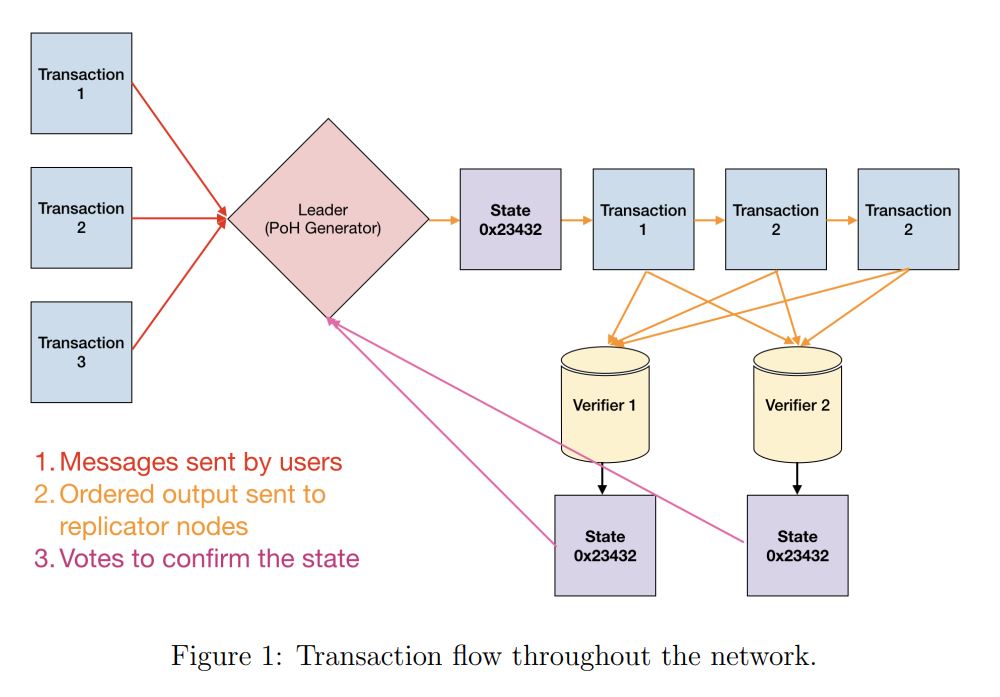
**Solana**

سولانا شبکه‌ای نسبتاً پیچیده است. استفاده از فناوری‌های نوین در آن باعث شده این بلاک‌چین یکی از بهینه‌ترین نمونه‌های موجود باشد. با این حال برای کارکردن با این شبکه و استفاده از توکن SOL نیازی به درک کامل نحوه عملکرد این شبکه و جزئیات فنی آن نخواهید داشت.

بلاک‌ها اسکلت هر بلاک‌چین هستند. با قرارگرفتن تراکنش‌ها و اطلاعات آنها در بلاک‌ها، زنجیره‌ای تشکیل می‌شود که بلاک‌چین نام دارد؛ در واقع بلاک‌ها مثل تکه‌های پازل هستند که در نهایت تصویری کلی به نام بلاک‌چین را می‌سازند.

در سولانا هر بلاک، اسلات (Slot) نام دارد و اعتبارسنجی خاص به نام لیدر (Leader) آنها را می‌سازد. هر اعتبارسنج در این شبکه عضوی از خوشه (Cluster) شبکه است. در این شبکه، هر خوشه تنها یک اعتبارسنج دارد و وظیفه آن اضافه‌کردن بلاک‌ها یا همان اسلات‌ها به دفتر کل بلاک‌چین است.

همانطور که در شکل 1 نشان داده شده است، در هر زمان یک نود سیستم به عنوان لیدر تعیین می شود تا یک اثبات تاریخچه (Proof of History) ایجاد کند، که ثبات خواندن سراسری شبکه و گذر زمان قابل تایید را فراهم می کند. Leader پیام های کاربر را ترتیب می دهد و آنها را به گونه ای مرتب می کند که بتوانند به طور موثر توسط نودهای دیگر در سیستم پردازش شوند و توان عملیاتی را به حداکثر برسانند. تراکنش ها را در حالت فعلی که در RAM ذخیره شده است اجرا می کند و تراکنش ها و امضای حالت نهایی را در نودهای تکراری (replications nodes) به نام Verifiers منتشر می‌کند. تأییدکنندگان همان تراکنش‌ها را بر روی نسخه‌های خود از state انجام می‌دهند و امضاهای محاسبه‌شده خود را از state به عنوان تأییدیه منتشر می‌کنند. تأییدیه های منتشر شده به عنوان رأی برای الگوریتم اجماع عمل می کنند.



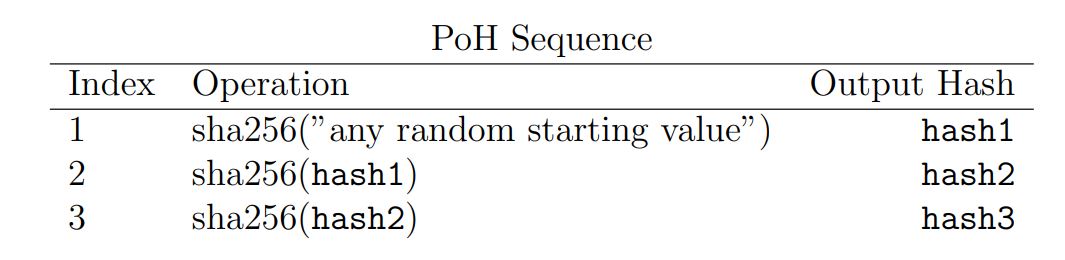
در حالت غیرپارتیشن بندی شده، در هر زمان معین، یک Leader در شبکه وجود دارد. هر نود Verifier دارای قابلیت‌های سخت‌افزاری یکسانی با یک Leader است و می‌تواند به عنوان یک Leader انتخاب شود، این کار از طریق انتخابات مبتنی بر PoS انجام می‌شود. از نظر قضیه CAP، سازگاری تقریباً همیشه در یک رویداد پارتیشن به جای Availability انتخاب می شود. در مورد یک پارتیشن بزرگ، این مقاله مکانیزمی را برای بازیابی کنترل شبکه از یک پارتیشن با هر اندازه پیشنهاد می‌کند.

**اثبات تاریخچه**

اثبات تاریخ، دنباله ای از محاسبات است که می تواند راهی برای تأیید رمزنگاری گذر زمان بین دو رویداد ارائه دهد. از یک تابع امن رمزنگاری استفاده می کند که خروجی را نمی توان از ورودی پیش بینی کرد و برای تولید خروجی باید به طور کامل اجرا شود. این تابع به صورت متوالی بر روی یک هسته اجرا می شود، خروجی قبلی آن به عنوان ورودی جریان است، به صورت دوره ای خروجی جریان را ضبط می کند و چند بار فراخوانی شده است. سپس خروجی را می توان مجدداً محاسبه و توسط رایانه های خارجی به طور موازی با بررسی هر بخش دنباله روی یک هسته جداگانه تأیید کرد. داده‌ها را می‌توان با اضافه کردن داده‌ها (یا هش برخی از داده‌ها) به وضعیت تابع، در این دنباله مهر زمانی گذاشت. ثبت وضعیت، فهرست و داده‌ها همانطور که به دنباله‌ها اضافه شده است، یک مهر زمانی ارائه می‌دهد که می‌تواند تضمین کند که داده‌ها قبل از ایجاد هش بعدی در دنباله ایجاد شده‌اند. این طرح همچنین از مقیاس بندی افقی پشتیبانی می کند زیرا چندین ژنراتور می توانند با ترکیب کردن حالت خود در توالی های یکدیگر در بین یکدیگر همگام شوند.

سیستم برای کار به صورت زیر طراحی شده است. با یک تابع هش رمزنگاری، که خروجی آن بدون اجرای تابع قابل پیش بینی نیست (به عنوان مثال sha256، ripemd، و غیره)، تابع را از مقداری شروع تصادفی اجرا کنید و خروجی آن را بگیرید و دوباره به عنوان ورودی به همان تابع ارسال کنید. تعداد دفعات فراخوانی تابع و خروجی را در هر تماس ضبط کنید. مقدار تصادفی انتخاب شده شروع می‌تواند هر رشته‌ای باشد، مانند سرفصل نیویورک تایمز برای آن روز.

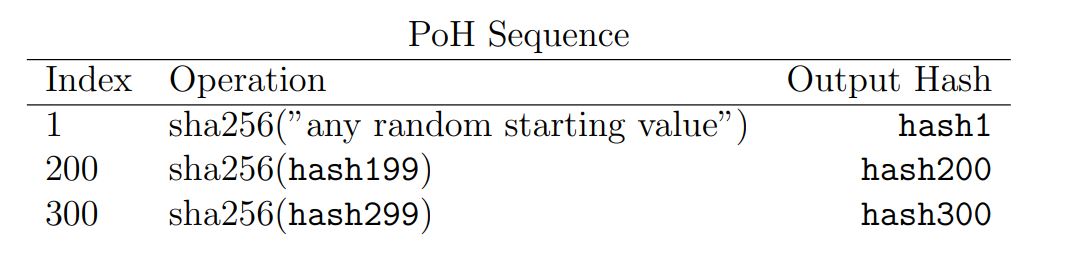
به عنوان مثال:



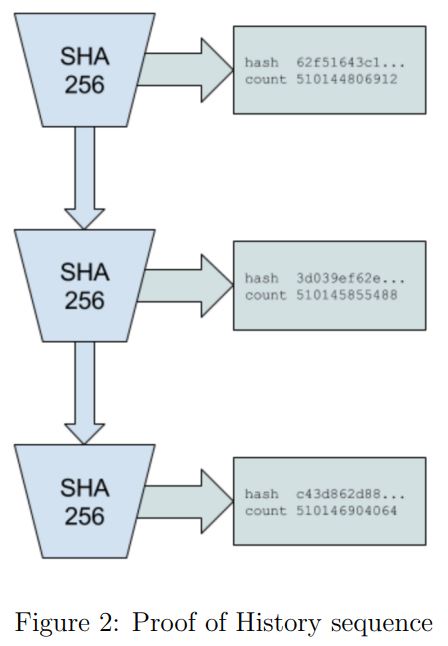
جایی که hashN خروجی هش واقعی را نشان می دهد.

فقط لازم است زیر مجموعه ای از هش ها و شاخص ها در یک بازه زمانی منتشر شود.

به عنوان مثال:



تا زمانی که تابع هش انتخاب شده در برابر حمله مقاوم باشد، این مجموعه از هش ها را فقط می توان به ترتیب توسط یک رشته کامپیوتری محاسبه کرد. این از این واقعیت ناشی می شود که هیچ راهی برای پیش بینی مقدار هش در شاخص 300 وجود ندارد بدون اینکه الگوریتم را از مقدار شروع 300 بار اجرا کنید. بنابراین ما می توانیم از ساختار داده استنباط کنیم که زمان واقعی بین شاخص 0 و شاخص 300 سپری شده است. در مثال در شکل 2، هش 62f51643c1 در تعداد 510144806912 تولید شد و هش c43d862d88 بر روی تعداد 510146904064 مورد بحث قبلی تولید شد. با الگوریتم PoH، می توانیم اعتماد کنیم که زمان واقعی بین تعداد 510144806912 و تعداد 510146904064 گذشته است.



تقریباً در تمام شبکه‌های توزیع‌شده، بزرگترین چالش پیش رو آن است که نودها نمی‌توانند به برچسب‌های زمانی که از نودهای دیگر دریافت می‌کنند، اعتماد کنند؛ چراکه این برچسب‌ها را گاهی خودِ نودها به بلاک‌ها اضافه می‌کنند یا از روی ساعت سخت‌افزار آنها به بلاک‌ها افزوده می‌شود.

سولانا با ارائه مدل اثبات تاریخچه در صدد حل این مشکل برآمده است. اثبات تاریخچه در واقع یک منبع امن و رمزنگاری‌شده است که یک مرجع سراسری در شبکه برای زمان ایجاد می‌کند. اثبات تاریخچه در واقع یک تابع تأخیر قابل‌تأیید (Verifiable Delay Function) با بسامد بالاست. در این نوع از توابع، مراحل مشخصی نیاز است تا اعتبار خروجی تأیید شود؛ اما هر کسی می‌تواند اعتبار خروجی را به‌سادگی تأیید کند.

اثبات تاریخچه به نودها این امکان را می‌دهد که بدون هماهنگی با دیگر اعضای شبکه، بلاک بعدی را بسازند. در این روند دیگر خبری از اعتماد به برچسب‌های زمانی است. در نتیجه استفاده از اثبات تاریخچه، سرعت تأیید تراکنش‌ها تا حد چشمگیری افزایش پیدا می‌کند.

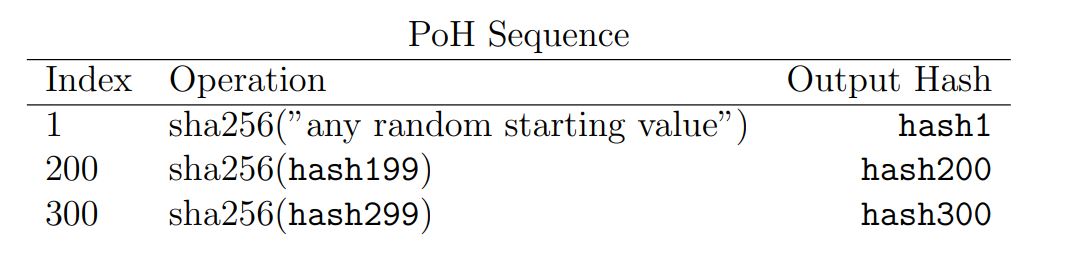
اثبات تاریخچه در واقع یک مرجع سراسری در شبکه سولانا برای ثبت زمان است. هدف اصلی این فناوری برقراری امنیت شبکه بدون فداکردن سرعت است.

تمام تراکنش‌ها و رخدادهای شبکه با تابع هش SHA256 هش می‌شوند. با استفاده از این تابع، خروجی منحصربه‌فرد بوده و حدس‌زدن آن تقریباً غیرممکن است. پس از هش‌شدن، هر خروجی به‌عنوان ورودی هش بعدی مورد استفاده قرار می‌گیرد.

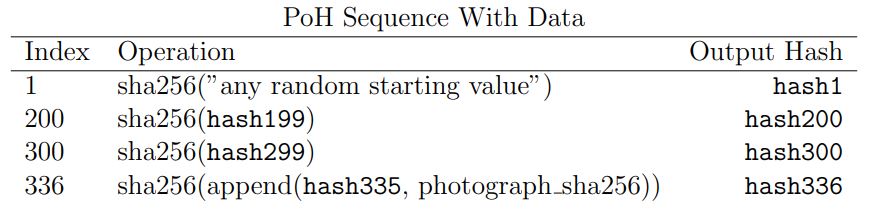
**برچسب‌های زمانی (Timestamp)برای Event ها**

این توالی هش همچنین می تواند برای ثبت اینکه برخی از داده ها قبل از ایجاد یک شاخص هش خاص ایجاد شده است استفاده شود. استفاده از تابع "combine" برای ترکیب قطعه داده با هش فعلی در شاخص فعلی. داده ها می توانند به سادگی یک هش رمزنگاری منحصر به فرد از داده های رویداد دلخواه باشند. تابع combine می تواند یک ضمیمه ساده از داده ها یا هر عملیاتی باشد که در برابر حمله مقاوم است. هش تولید شده بعدی نشان‌دهنده یک برچسب‌ زمانی (Timestamp) از داده‌ها است، زیرا تنها پس از درج آن قطعه خاص از داده می‌توانست تولید شود.

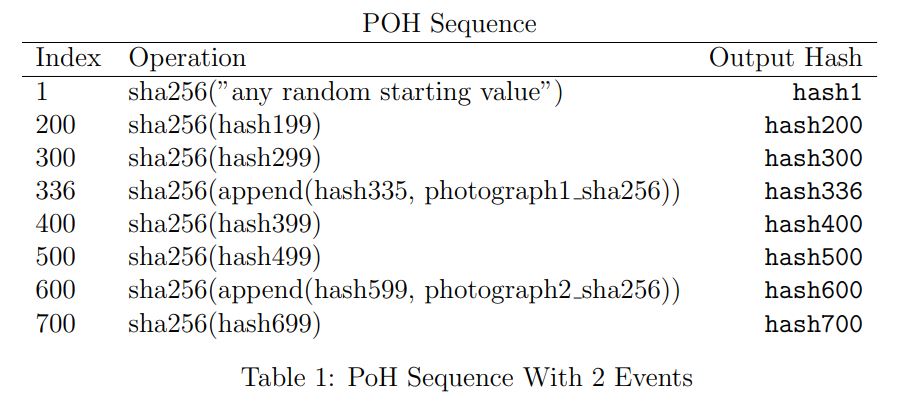
برای مثال:



برخی از رویدادهای خارجی رخ می دهد، مانند یک عکس گرفته شده، یا هر گونه داده دیجیتال دلخواه ایجاد شده است:



Hash336 از داده های باینری پیوست شده hash335 و sha256 عکس، محاسبه می شود. ایندکس و sha256 عکس به عنوان بخشی از خروجی توالی ثبت می شود. بنابراین هر کسی که این دنباله را تأیید می کند می تواند این تغییر را در دنباله دوباره ایجاد کند. تأیید همچنان می تواند به صورت موازی انجام شود، زیرا فرآیند اولیه هنوز متوالی است، سپس می توانیم بگوییم که موارد وارد شده به دنباله باید زمانی قبل از محاسبه مقدار هش شده آینده رخ داده باشند.

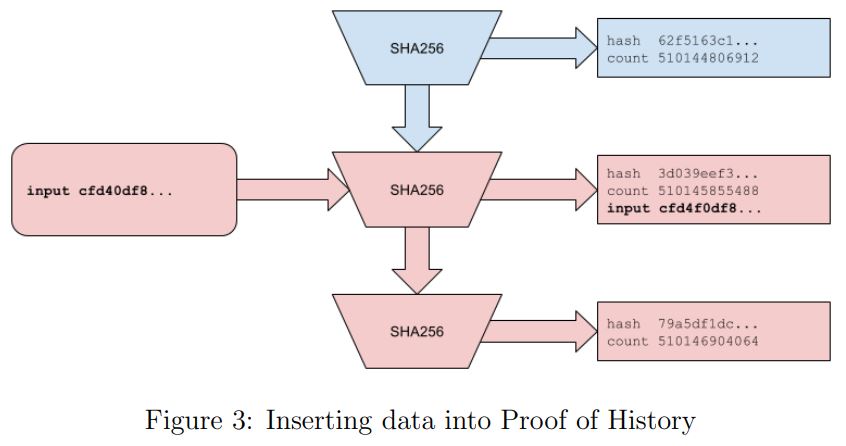


در دنباله ای که در جدول 1 نشان داده شده است، Photo2 قبل از hash600 و Photo1 قبل از hash336 ایجاد شده است. درج داده ها در دنباله هش ها منجر به تغییر در تمام مقادیر بعدی در دنباله می شود. تا زمانی که تابع هش مورد استفاده در برابر حمله مقاوم باشد و داده ها اضافه شده باشند، از نظر محاسباتی نمی توان از قبل هر توالی آینده را بر اساس دانش قبلی از اینکه چه داده هایی در دنباله ادغام می شود، محاسبه کرد.

داده‌هایی که در توالی ترکیب می‌شوند می‌توانند خود داده‌های خام یا فقط هش داده‌ها با ابرداده‌های همراه باشند.

در مثال شکل 3، ورودی cfd40df8... در دنباله اثبات تاریخ درج شد. تعداد درج شده 510145855488 و حالتی که در آن درج شده است 3d039eef3 است. تمام هش های تولید شده در آینده با این تغییر به دنباله اصلاح می شوند، این تغییر با تغییر رنگ در شکل نشان داده می شود.

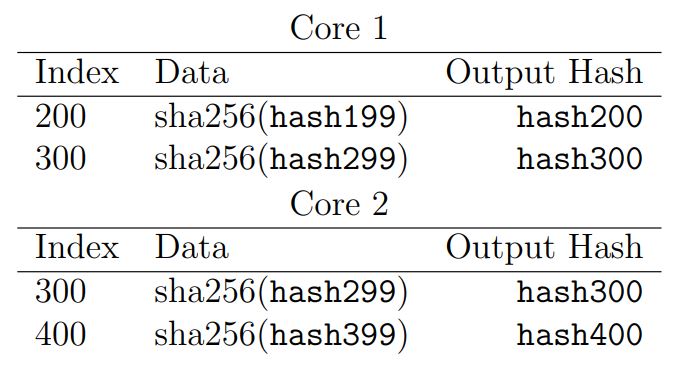
هر نود ای که این دنباله را مشاهده می کند می تواند ترتیب درج همه رویدادها را تعیین کند و زمان واقعی بین درج ها را تخمین بزند.



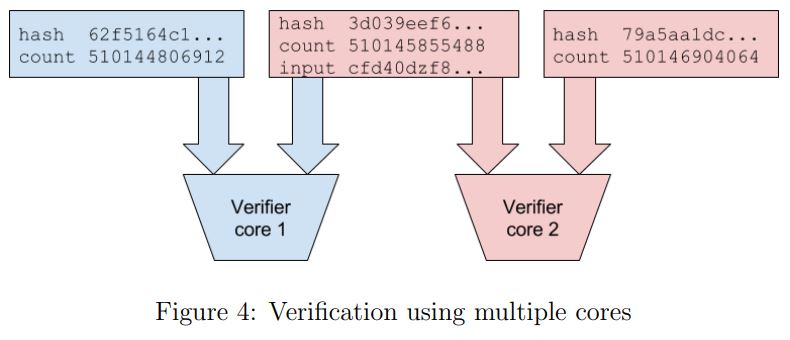
**Verification**

این توالی را می توان با یک رایانه چند هسته ای در زمان بسیار کمتری نسبت به تولید آن تأیید کرد.

به عنوان مثال:



با توجه به تعدادی هسته، مانند یک GPU مدرن با 4000 هسته، تأیید کننده می تواند دنباله هش ها و شاخص های آنها را به 4000 قسمت تقسیم کند، و به طور موازی مطمئن شود که هر قسمت از هش شروع تا آخرین هش درست است. اگر زمان مورد انتظار برای تولید دنباله به صورت زیر باشد:



تعداد کل هش ها

--------------------------------------

هش در ثانیه برای 1 هسته

مدت زمان مورد انتظار برای تأیید صحت دنباله عبارت است از:

تعداد کل هش ها

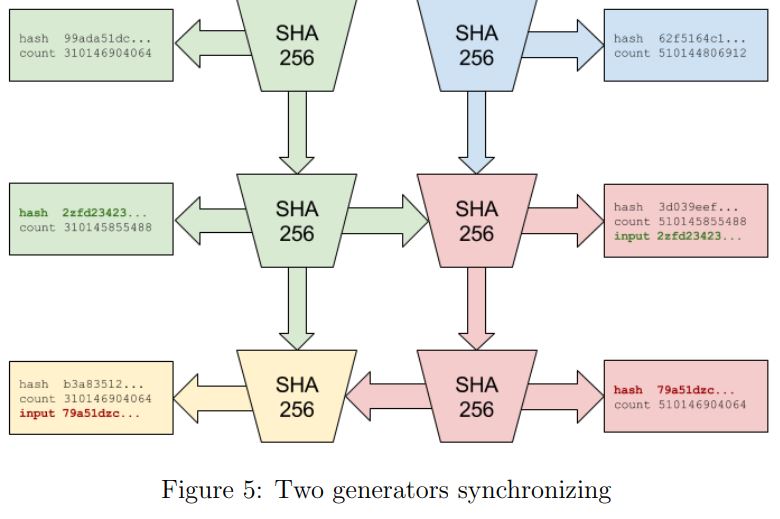
------------------------------------------------------------------------------------------

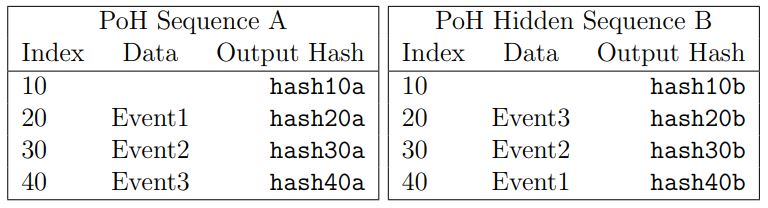
(هش در ثانیه برای هر هسته \* تعداد هسته های موجود برای تأیید)

در مثال شکل 4، هر هسته قادر است هر تکه از دنباله را به صورت موازی تأیید کند. از آنجایی که تمام رشته‌های ورودی با شمارنده و حالتی که به آن اضافه شده‌اند در خروجی ثبت می‌شوند، تأییدکننده‌ها می‌توانند هر برش را به صورت موازی تکرار کنند. هش های قرمز رنگ نشان می دهد که توالی با درج داده تغییر یافته است.

**Consistency**

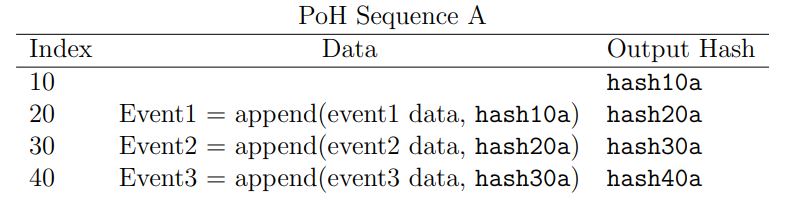
انتظار می رود کاربران بتوانند با درج آخرین خروجی مشاهده شده از دنباله ای که معتبر می دانند در ورودی خود، سازگاری توالی تولید شده را اعمال کنند و آن را در برابر حملات مقاوم کنند.



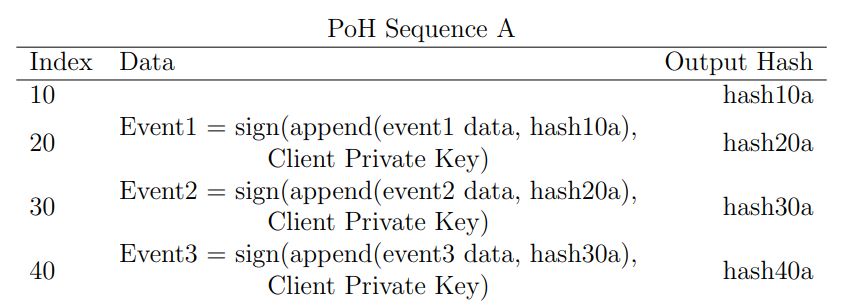


یک ژنراتور مخرب PoH می‌تواند یک دنباله پنهان دوم با رویدادها به ترتیب معکوس ایجاد کند، اگر به همه رویدادها به طور همزمان دسترسی داشته باشد یا بتواند یک دنباله پنهان سریع‌تر ایجاد کند.

برای جلوگیری از این حمله، هر رویداد ایجاد شده توسط سرویس گیرنده باید حاوی آخرین هش باشد که مشتری از آنچه که دنباله‌ای معتبر می‌داند مشاهده کرده است. بنابراین هنگامی که یک کلاینت داده "Event1" را ایجاد می کند، باید آخرین هش را که مشاهده کرده است اضافه کند.



وقتی دنباله منتشر می شود، Event3 به hash30a ارجاع می دهد، و اگر در توالی قبل از این رویداد نباشد، مصرف کنندگان دنباله می دانند که یک دنباله نامعتبر است. سپس حمله مرتب‌سازی مجدد جزئی به تعداد هش‌های تولید شده در زمانی که مشتری یک رویداد را مشاهده کرده و زمانی که رویداد وارد شده است، محدود می‌شود. سپس مشتریان باید بتوانند نرم افزاری بنویسند که ترتیب را برای دوره کوتاه هش بین آخرین هش مشاهده شده و درج شده صحیح فرض نمی کند. برای جلوگیری از بازنویسی هش رویداد مشتری توسط یک مولد PoH مخرب، مشتریان می‌توانند به جای داده‌ها، امضای داده‌های رویداد و آخرین هش مشاهده‌شده را ارسال کنند.



راستی‌آزمایی این داده‌ها به تأیید امضا، و جستجوی هش در توالی هش‌های قبل از این نیاز دارد.

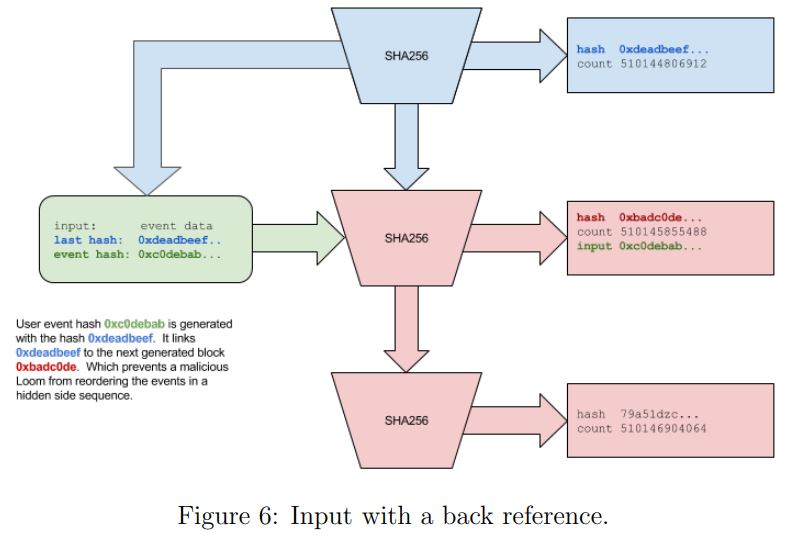
تأیید کنید:

(Signature, PublicKey, hash30a, event3 data) = Event3

Verify(Signature, PublicKey, Event3)

Lookup(hash30a, PoHSequence)

در شکل 6، ورودی ارائه شده توسط کاربر وابسته به هش 0xdeadbeef... است که در توالی تولید شده قبل از درج آن وجود دارد. فلش بالای سمت چپ آبی نشان می دهد که مشتری به هش قبلا تولید شده ارجاع می دهد. پیام مشتری فقط در دنباله‌ای معتبر است که حاوی هش 0xdeadbeef.... باشد رنگ قرمز در دنباله نشان می‌دهد که توالی توسط داده‌های کلاینت اصلاح شده است.

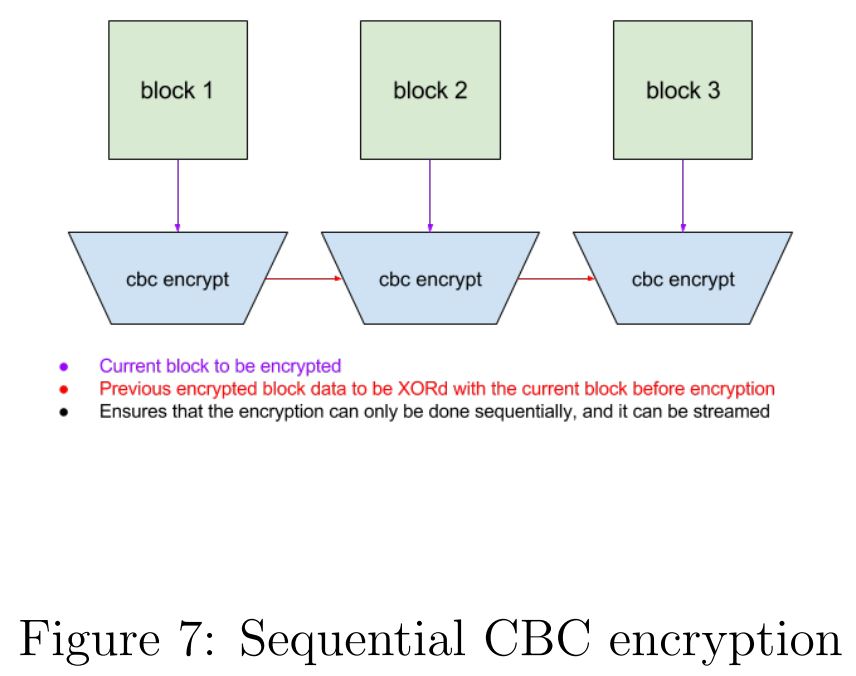


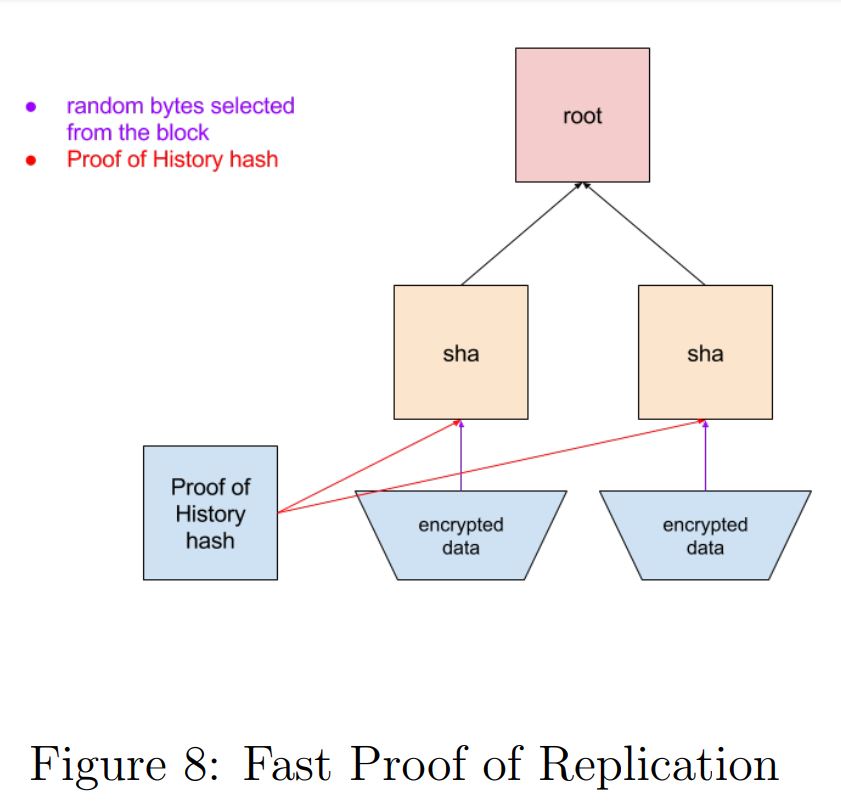
فلش سمت چپ بالا نشان می دهد که مشتری به هش قبلا تولید شده ارجاع می دهد. پیام مشتری فقط در دنباله‌ای معتبر است که حاوی هش 0xdeadbeef.. باشد رنگ قرمز در دنباله نشان می‌دهد که توالی توسط داده‌های کلاینت اصلاح شده است.

**Algorithm**

همانطور که در شکل 7 نشان داده شده است، رمزگذاری CBC هر بلوک داده را به ترتیب رمزگذاری می کند، با استفاده از بلوک رمزگذاری شده قبلی برای XOR داده های ورودی. هر هویت تکراری با امضای یک هش که دنباله اثبات تاریخچه ایجاد شده است، یک کلید ایجاد می کند. این کلید را به هویت شبیه سازها، و به دنباله ای خاص از اثبات تاریخ گره می زند. فقط هش های خاصی را می توان انتخاب کرد. مجموعه داده کاملاً بلوک به بلوک رمزگذاری شده است. سپس برای ایجاد یک اثبات، از کلید برای ایجاد یک مولد اعداد شبه تصادفی استفاده می شود که 32 بایت تصادفی را از هر بلوک انتخاب می کند. هش مرکل با هش PoH انتخاب شده که به هر برش اضافه شده است، محاسبه می شود.

ریشه به همراه کلید و هش انتخابی که تولید شده است منتشر می شود. گره تکثیر باید اثبات دیگری را در N هش منتشر کند زیرا آنها توسط مولد Proof of History تولید می شوند، جایی که N تقریباً 12 زمانی است که برای رمزگذاری داده ها طول می کشد. مولد Proof of History هش های خاصی را برای Proof of Replication در دوره های از پیش تعریف شده منتشر می کند. گره replicator باید هش منتشر شده بعدی را برای تولید اثبات انتخاب کند. دوباره، هش امضا می‌شود و برش‌های تصادفی از بلوک‌ها برای ایجاد ریشه merkle انتخاب می‌شوند. پس از یک دوره N اثبات، داده ها با یک کلید CBC جدید دوباره رمزگذاری می شوند.

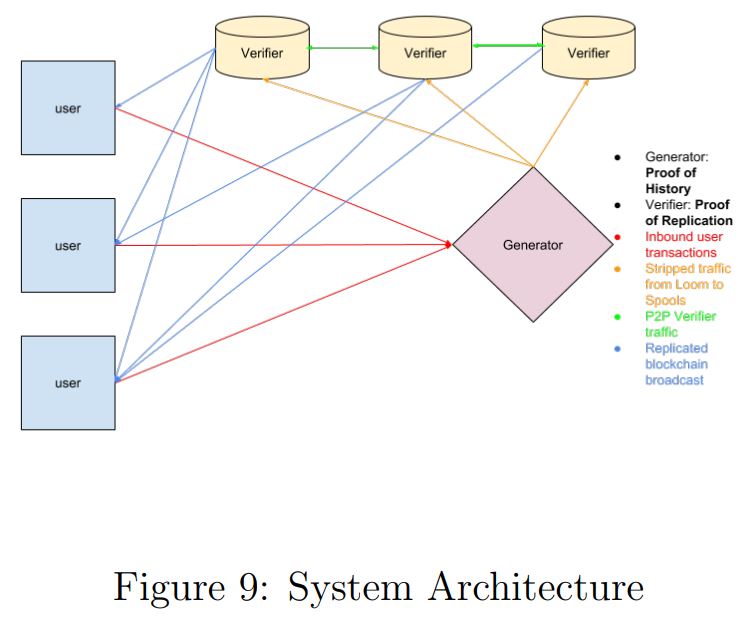




در N هش همانطور که توسط Proof of History مولد تولید می شوند، که در آن N تقریباً 12 زمان لازم برای رمزگذاری داده ها است. مولد Proof of History هش های خاصی را برای Proof of Replication در دوره های از پیش تعریف شده منتشر می کند. گره replicator باید هش منتشر شده بعدی را برای تولید اثبات انتخاب کند. دوباره، هش امضا می‌شود و برش‌های تصادفی از بلوک‌ها برای ایجاد ریشه merkle انتخاب می‌شوند. پس از یک دوره N اثبات، داده ها با یک کلید CBC جدید دوباره رمزگذاری می شوند.

**Tragedy of Commons**

تایید کننده های PoS می توانند به سادگی PoRep را بدون انجام کاری تایید کنند. انگیزه‌های اقتصادی باید با تأییدکننده‌های PoS برای انجام کار، مانند تقسیم پرداخت ماینینگ بین تأییدکننده‌های PoS و گره‌های تکرار PoRep، ردیف شوند. برای اجتناب بیشتر از این سناریو، تأیید کننده های PoRep می توانند در درصد کمی از مواقع شواهد نادرست ارائه دهند. آنها می توانند با ارائه تابعی که داده های نادرست را ایجاد کرده، نادرست بودن اثبات را ثابت کنند. هر تأییدکننده PoS که اثبات نادرست را تأیید کند، کاهش می یابد.



**System Architecture**

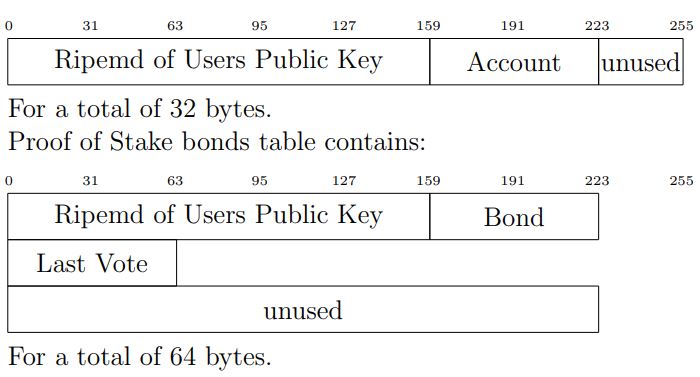
**Components**

Leader, Proof of History generator

لیدر منتخب ژنراتور اثبات تاریخ است. تراکنش‌های دلخواه کاربر را مصرف می‌کند و دنباله‌ای از تاریخچه اثبات تمام تراکنش‌ها را ارائه می‌کند که نظم جهانی منحصربه‌فرد را در سیستم تضمین می‌کند. پس از هر دسته از تراکنش‌ها، رهبر امضای حالتی را که نتیجه اجرای تراکنش‌ها به آن ترتیب است، صادر می‌کند. این امضا با هویت رهبر امضا شده است.

**State**

یک جدول هش ساده که توسط آدرس کاربران نمایه شده است. هر سلول شامل آدرس کامل کاربران و حافظه مورد نیاز برای این محاسبه است. به عنوان مثال جدول معاملات شامل:



**Verifier, State Replication**

نودهای Verifier حالت بلاک‌چین را تکرار می کنند و در دسترس بودن بالایی از حالت بلاک‌چین را فراهم می‌کنند. هدف تکرار توسط الگوریتم اجماع انتخاب می‌شود و اعتبار‌دهنده‌ها در الگوریتم اجماع نود‌های Proof of Replication را که بر اساس معیارهای تعریف‌شده خارج از زنجیره تأیید می‌کنند، انتخاب کرده و رأی می‌دهند.

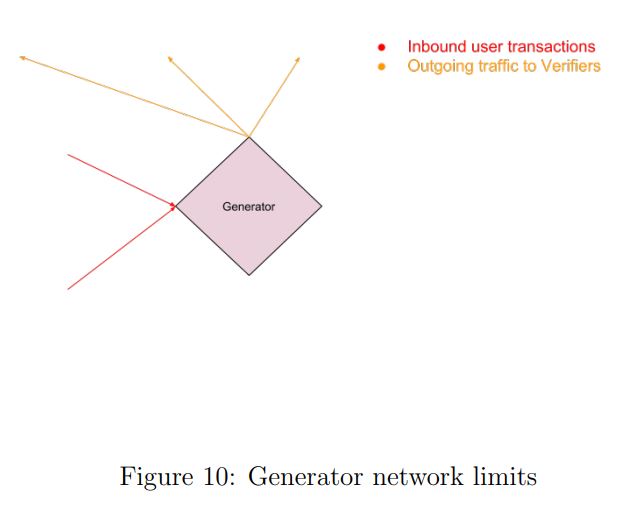
شبکه را می توان با حداقل اندازه زنجیره اثبات سهام و نیاز به هویت تکثیرکننده در هر پیوند پیکربندی کرد.

**Validators**

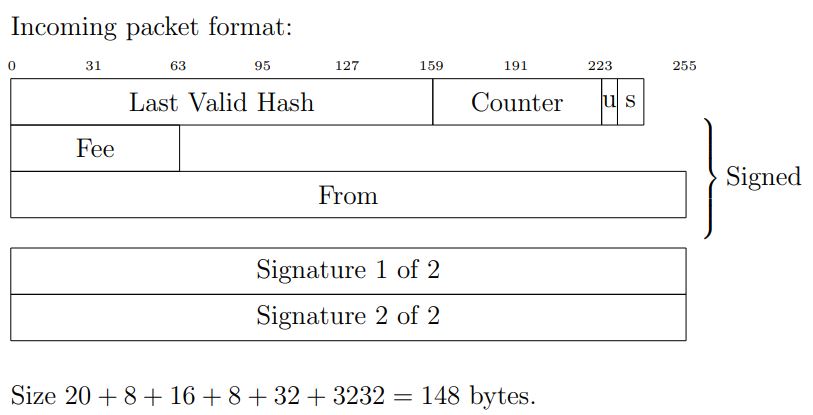
این نودها پهنای باند را از Verifiers مصرف می کنند. آن‌ها نودهای مجازی هستند و می‌توانند روی همان ماشین‌هایی مانند Verifiers یا Leader یا روی ماشین‌های جداگانه‌ای که مختص الگوریتم اجماع پیکربندی شده برای این شبکه هستند اجرا شوند.

**Network Limits**

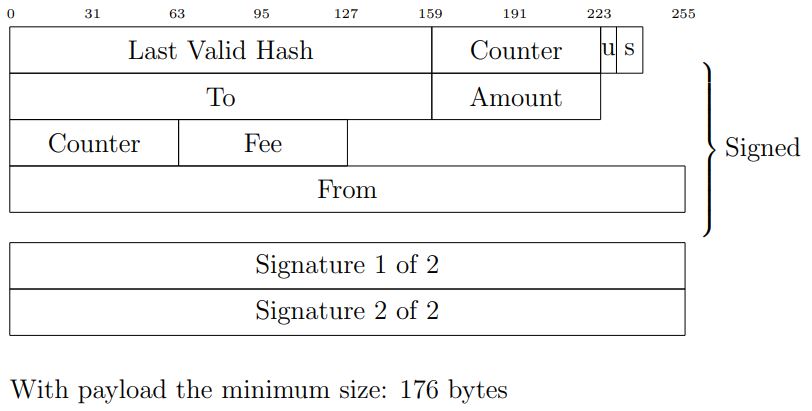
انتظار می رود که Leader بتواند بسته‌های کاربر ورودی را بگیرد، آنها را به کارآمدترین روش ممکن سفارش دهد و آنها را در یک توالی اثبات تاریخچه که برای تأییدکنندگان پایین دستی منتشر می شود، ترتیب دهد. کارایی بر اساس الگوهای دسترسی به حافظه تراکنش ها است، بنابراین تراکنش ها برای به حداقل رساندن خطاها و به حداکثر رساندن واکشی اولیه ترتیب داده می شوند.



قالب بسته ورودی:

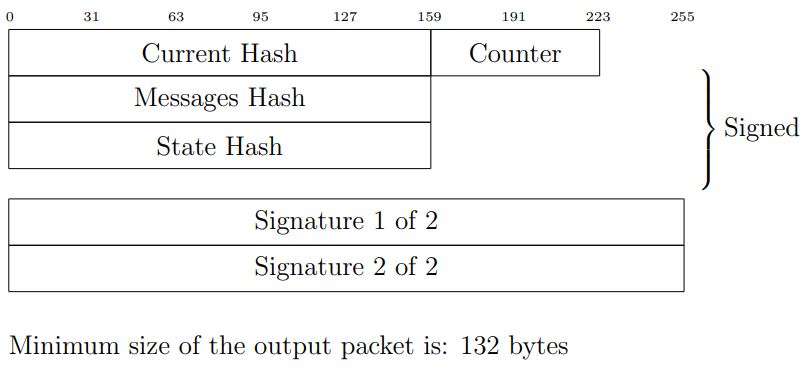


حداقل بار قابل پشتیبانی 1 حساب مقصد خواهد بود. با محموله:



بسته توالی Proof of History شامل هش، شمارنده و هش تمام پیام‌های جدید اضافه شده به دنباله PoH و امضای حالت پس از پردازش همه پیام‌ها است. این بسته یکبار هر N پیام پخش می شود.

بسته اثبات تاریخچه:



در اتصال شبکه 1 گیگابیت بر ثانیه حداکثر تعداد تراکنش های ممکن 1 گیگابیت در ثانیه تقسیم بر 176 بایت = حداکثر 710 هزار تراکنش در ثانیه است. مقداری از دست دادن 1-4٪ به دلیل چارچوب اترنت انتظار می رود. ظرفیت اضافی بیش از مقدار مورد نظر برای شبکه را می توان با کدگذاری خروجی با کدهای Reed-Solomon و جدا کردن آن به تأیید کننده های پایین دستی موجود برای افزایش در دسترس بودن استفاده کرد.