دانشکده علوم ریاضی و آمار

مدرس: دکتر مجتبی رفیعی

نیمسال اول ۱۴۰۱-۱۴۰۲

پروژه‌ی رمزنگاری

بررسی امکان اثبات اصالت و تمامیت داده برای سیستم‌های انتقال داده‌ی همتابه‌همتا توسط سیستم‌های هش

نگارندگان:

مریم رضائی

محمدحسین فرخی

۲۶ بهمن ۱۴۰۱

فهرست مطالب

۱ مقدمه و هدف

سیستم‌های همتابه‌همتا یا P2P[1] روش‌های جدیدی برای انتشار و اشتراک‌گذاری داده‌های حجیم را معرفی کردند که به علت پخش منابع بر تمامی اعضای شبکه و در نتیجه سرعت بالای آن مورد استقبال بسیاری قرار گرفتند. از آنجا که هر کاربر درون این سیستم‌ها توان بارگذاری و تأمین منابع را دارد، توجه خاصی به تشخیص اصالت داده نیاز است تا نه تنها داده‌ی مورد نظر به کاربر درخواست‌دهنده برسد، بلکه مهاجمی توان تغییر و جعل داده‌ی درست را نداشته باشد و داده در انتقال نیز دچار مشکل نشود؛ این تشخیص تمامیت داده به اصالت‌سنجی داده وابسته است.

متاسفانه در بسیاری از سیستم‌های P2P موجود، مبحث بررسی اصالت و تمامیت داده یا هویت همتایان دچار مشکلات ساختاری است. هدف این تحقیق بررسی امکان تشخیص مناسب اصالت و تمامیت داده توسط هش‌ها در چنین سیستم‌هایی می‌باشد.

۲ معرفی کارکرد سیستم‌های همتابه‌همتا

۱.۲ نمای کلی

برای درک ساختار کلی سیستم‌های P2P، ابتدا با مثالی آغاز می‌کنیم.

مثال

فرض کنید آلیس می‌خواهد به داده‌ای که ندارد دسترسی پیدا کند. وی از شبکه‌ی افرادی که می‌شناسد درمورد داده پرس‌وجو کرده تا افرادی مانند باب و کارول که دارای داده‌ی مورد نظر هستند را پیدا کند. آنگاه باب و کارول هر کدام بخشی از داده را به آلیس ارسال می‌کنند تا تمام حجم و فشار داده بر یکی از آن‌ها تحمیل نشود.

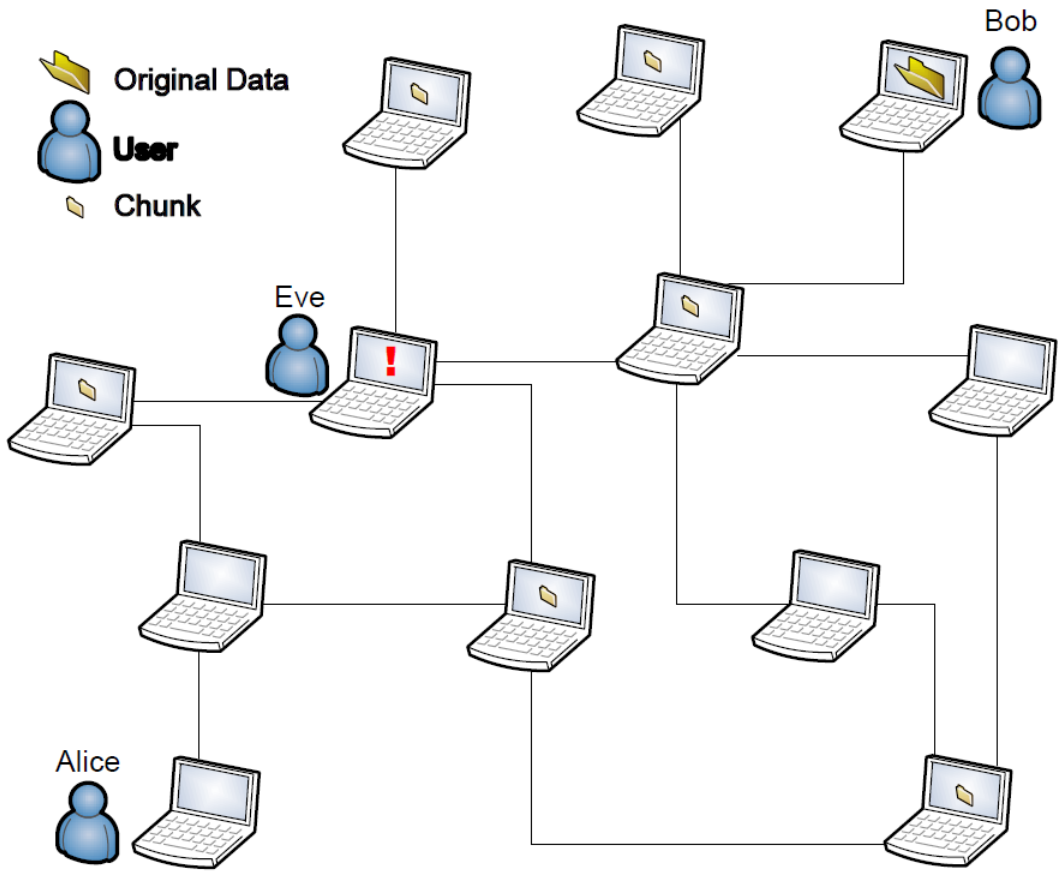
این نمونه‌ای ساده از کارکرد سیستم‌های P2P است. در این سیستم‌ها برخلاف شبکه‌های سنتی، هر کاربر بسته به شرایط هر دو نقش سرور و کلاینت را گرفته و در شبکه‌ای از کاربرهای همتا به اشتراک بخش‌های مختلف داده‌ی درخواستی همتای دیگر می‌پردازند. برای این‌کار درخواست‌دهنده اطلاعاتی خاص از داده‌ی مورد نظر خود را به شبکه ارسال می‌کند. آنگاه الگوریتم‌های هش سیستم به جستجوی هرچند بخش[2] داده در میان همتایان پرداخته و با تایید برابری اطلاعات بخش درخواستی با بخش موجود، به استخراج داده از همتای یافت شده می‌پردازند. به عبارت دیگر هر همتا توان و مسئولیت بارگذاری بخش محدودی از داده‌ی کل را دارد. این امر امکان پخش سنگینی داده، توسعه‌ی دامنه، و بارگذاری موازی و در نتیجه بهبود امکانات، سرعت و بازدهی را فراهم می‌سازد.

کاربرد

شرکت نپستر[3]موفقیت زودهنگام خود در عرصه‌ی اشتراک‌گذاری موسیقی را به تکنولوژی P2P مدیون بود که امکان ارائه‌ی سرویس متنوع و ارزان با تکیه بر دستگاه‌های کاربر نهایی را برای آن ایجاد کرده بود. از دیگر پیاده‌سازی‌های معروف این سیستم‌ها می‌توان به بیت‌تورنت[4]، نوتلا[5]، کازا[6]، و لایم‌وایر[7]اشاره کرد.

[1] Peer-to-Peer
[2] Chunk

شکل ۱ نمایی کلی از یک سیستم P2P ساده است.



شکل ۱: نمایی کلی از یک سیستم P2P ساده

همانطور که در شکل ۱ مشاهده می‌کنیم، سرور بودن تمامی کاربران به آن‌ها اجازه‌ی ارسال هر داده‌ای به عنوان داده‌ی مورد نظر را می‌دهد. این امر باعث امکان بروز خطرهای متعددی از جمله دریافت داده‌ی جعلی یا مخرب می‌شود. بنابراین از نیازهای اصلی سیستم‌های P2P قابل اطمینان بودن همتایان و قوانینی مبنی بر امنیت داده و شبکه است که همواره موضوع تحقیقات بسیاری در سال‌های اخیر بوده است.

در ادامه به بررسی خطرهای سیستم‌های P2P به خصوص در امر اصالت و تمامیت داده خواهیم پرداخت. اما ابتدا مثالی از شکل دقیق یک سیستم P2P پر کاربرد را شرح می‌دهیم که به سیستم‌های P2P کلاسیک اجزای جدیدی اضافه می‌کند.

---

[3] Napster
[4] BitTorrent
[5] Gnutella
[6] KaZaA
[7] LimeWire

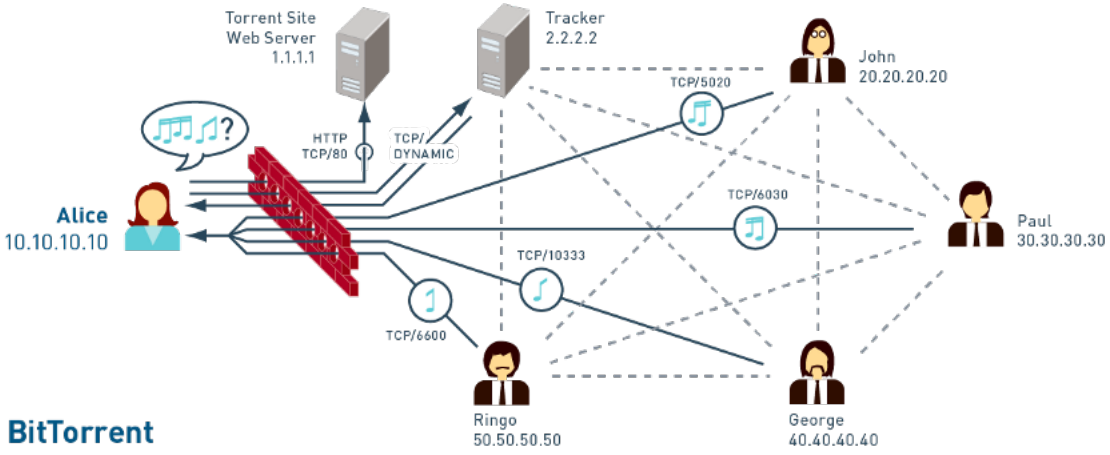
۲.۲ نمونه‌ای از سیستم همتابه‌همتا

بیت‌تورنت پروتکل ارتباطی‌ای بر مبنای P2P است که به کمک فایل‌های تورنت که فراداده‌ای در ارتباط با محتوی مورد نظر هستند، از شبکه‌ای از افراد به طور همزمان به دانلود بخش‌های مختلف داده‌ی مورد نظر می‌پردازد. بعضی اجزای کلی بیت‌تورنت به شکل زیر هستند:

- **تورنت:** فایل‌های torrent. حاوی فراداده‌های فایل موردنظر.
- **کارخواه:**[۸] نرم‌افزاری که فایل‌های torrent. را پردازش کرده و دانلود را انجام می‌دهد.
- **همتا:**[۹] هر کاربر اضافه شده به شبکه که در جابه‌جایی فایل‌ها سهیم می‌شود.
- **انگل:**[۱۰] کاربر بیت‌تورنت که برای دانلود فایل به شبکه پیوسته است.
- **پذرپاش:**[۱۱] کاربر بیت‌تورنت که برای آپلود فایل به شبکه پیوسته است.
- **نسبت اشتراک:** نسبت آپلود به دانلود هر کاربر که میزان کمک او به شبکه‌ی همتابه‌همتا را تعیین می‌کند.
- **ردیاب:**[۱۲] سروری که ترافیک بین بذرپاش‌ها و انگل‌ها را هدایت می‌کند.

نرم‌افزاری که کاربر برای پیوستن به این شبکه استفاده می‌کند کارخواه یا کلاینت است. این نرم‌افزار به کاربر اجازه‌ی باز کردن فایل‌های تورنت را می‌دهد. پس از باز کردن فایل و پردازش فراداده‌ی محتوی مورد نظر، کلاینت درخواست فایل را به ردیاب ارسال می‌کند. ردیاب با الگوریتم‌های تعیین شده به جستجوی داده در همتایان می‌گردد و با تشخیص دسته‌اس از بذرپاش‌ها، ترافیک را از بذپاش‌های به انگل هدایت می‌کند. هرگاه بخشی از فایل وارد دستگاه انگل شود، این کاربر می‌تواند به دسته‌ی بذرپاش‌های این بخش از فایل در شبکه بپیوندد.

هدف بیت‌تورنت تقسیم حجم ترافیک تحمیلی میان کاربران و افزایش دسترسی به انواع فایل‌ها است. به همین علت یکی از تمرکزهای این سیستم بر نسبت اشتراک مناسب است؛ یعنی کاربران همان اندازه که دانلود می‌کنند، به آپلود نیز کمک کنند. به همین علت پاداش آپلود کردن بیشتر، افزایش سرعت دانلود کاربر است. شکل ۲ کارکرد بیت‌تورنت را به تصویر می‌کشد.



شکل ۲: نمایی کلی از سیستم بیت‌تورنت

[8] Client
[9] Peer
[10] Leecher
[11] Seeder
[12] Tracker

۳ مشکلات اصالت و تمامیت سنجی در همتابه‌همتا

شبکه‌های P2P با هدف تقسیم مناسب هزینه‌های محاسباتی سرویس‌ها توسعه یافتند، اما این نکته‌ی قوت آثار منفی نیز دارد. به علت عدم امکان تعیین میزان اطمینان‌پذیری همتایان در سیستمی که قصد آن دسترسی دادن به همگان است، هر کاربر می‌تواند به راحتی مواردی مانند محرمانگی، تمامیت، و دسترس‌پذیری داده را با شنود و اشتراک‌گذاری، دستکاری و دروغ‌گویی درمورد صحت، و مختل کردن نرافیک بر هم بزند. با یک مثال خطر موجود در شکل ۱ را به تصویر می‌کشیم.

مثال

آلیس داده‌ای حجیم را از باب می‌خواهد، اما باب منابع محدود دارد و داده‌ی خود را به کمک یک نرم‌افزار بر مبنای P2P ذخیره کرده است که داده را به بخش‌های کوچکی تقسیم کرده و میان لیستی از کاربران پخش کرده‌است. باب لیست این افراد را به آلیس می‌فرستد و آلیس از این همتایان درون شبکه داده را درخواست می‌کند. ایو می‌خواهد بدافزار خود را منتشر کند و به جای داده‌ی باب، بدافزار را به آلیس می‌فرستد. آلیس هیچ راهی برای تشخیص این مشکل و حذف یا قرنطینه‌ی کرم ندارد و در دام ایو می‌افتد.

چنین خطراتی از مشکل‌های اصلی سیستم‌های بر مبنای P2P است. برای مثال، دو مطالعه در سال ۲۰۰۶ در ۶۸% تمامی محتوای قابل اجرای اشتراک‌گذاشته شده با استفاده از کازا و ۱۵% تمامی فایل‌های روی لایم‌وایر بدافزار پیدا کرد. در بیت‌تورنت نیز که برای تعیین اصالت بخش در دست بذرپاش، هش آن را با هش لیست شده برای بخش درخواستی با استفاده از سیستم هش SHA-1 مقایسه می‌کند، یک همتای بدخواه می‌تواند محتوی را پیش از ارسال به کاربر درخواست‌دهنده تغییر دهد. سایت ردیاب نیز هیچ مکانیزمی برای پیشگیری انتشار این بخش تغییر یافته ندارد.

از آنجا که مزایای سیستم‌های P2P آن‌ها را گزینه‌های مناسبی از نظر کاربرد می‌کند، نیاز است راه‌کارهایی برای این مشکلات ارائه شود. راه‌حل‌های متنوعی برای انواع خطرات سیستم‌های P2P موجود هستند، مانند رمزنگاری داده برای حفظ محرمانگی در زمان انتشار آن به روی کانال‌های ناامن شبکه. تمرکز این تحقیق ارائه‌ی راهکاری برای حل مشکل اثبات اصالت و تمامیت داده می‌باشد.

۴ راه‌کار کلی اثبات اصالت و تمامیت داده در همتابه‌همتا

۱.۴ تعریف مسئله و ارائه‌ی راه‌کار

از آنجا که امکان نقض اصالت و تمامیت داده در هر جای سیستم P2P به انواع شیوه‌ها وجود دارد، حیطه‌ی راه‌کار را به بخش انتهایی یعنی دریافت توسط درخواست‌دهنده محدود می‌کنیم. بنابراین نیاز به پاسخ به سوال زیر داریم.

مسئله

کاربر درخواست‌دهنده چگونه می‌تواند از یکسان بودن داده‌ی دانلودی با داده‌ی مورد نظر اطمینان حاصل کند؟

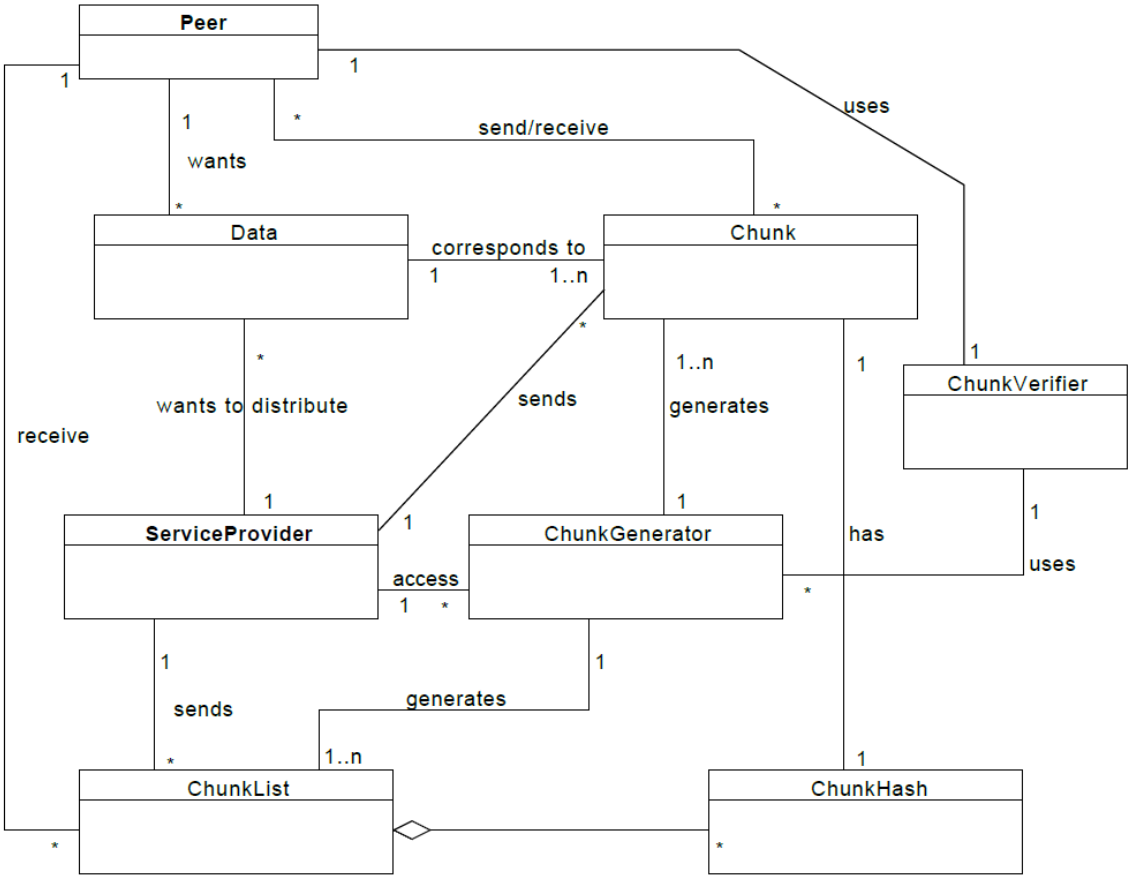
برای این کار نیاز است معیاری برای سنجش درست بودن داده‌ی دانلود شده تعریف کرد که بر هر بخش از داده‌ی درخواستی کلی که به طور کامل وارد دستگاه درخواست‌دهنده شده است تک تک اعمال شود تا در صورت درستی پذیرفته شود و در غیر این صورت از همتایی دیگر دوباره دانلود

شود. لازم به ذکر است که به علت امکان دستکاری بدخواهانه‌ی داده، بایستی پیش از تایید شدن در دستگاه کاربر درون قرنطینه نگهداری شود تا پیش از آن که امنیت آن توسط معیار سنجش تضمین شود توان آسیب به دستگاه را نداشته باشد.

معیار سنجش مورد استفاده برای تایید اصالت و تمامیت داده را سیستم‌های هش در نظر می‌گیریم که در ادامه مناسب بودن این ابزار و چند نمونه الگوریتم آن را بررسی خواهیم کرد. اما ابتدا به شرح دقیق مراحل سیستم P2P تغییر یافته با استفاده از این راه‌کار می‌پردازیم.

۲.۴ مراحل کارکرد سیستم تغییر یافته

فرض می‌کنیم که سرویس‌دهنده‌ای قصد انتشار داده‌هایی در میان شبکه‌ای از کاربران را دارد. این داده که برای اولین بار به سیستم وارد می‌شود نیاز به بخش‌بندی شدن و تعیین هش هر بخش دارد تا در ادامه همتایان بتوانند آن را از دارنده‌های ابتدایی داده دانلود کنند. مراحل تولید لیست هش بخش‌ها و سپس استفاده از آن برای سنجش اصالت و تمامیت داده در شکل ۳ به تصویر کشیده شده است.



شکل ۳: الگوی امن P2P با استفاده از هش

در ادامه به شرح دقیق مراحل اصلی می‌پردازیم.

۱.۲.۴ تولید اولیه‌ی لیست بخش‌ها

۱. داده به تولید کننده‌ی بخش‌ها ارسال می‌شود تا به بخش‌هایی با اندازه‌ی یکسان (برای مثال ۲۵۶ کیلوبایت) تقسیم شود.
۲. تولید کننده‌ی بخش داده را تقسیم کرده و هش هر کدام را بر اساس الگوریتم تعیین شده محاسبه می‌کند. دقت می‌کنیم الگوریتم بایستی امن باشد تا نه تنها هش تولید شده مناسب تشخیص و یافتن هر بخش خاص باشد، بلکه امن بوده و قابل شکستن و جعل نباشد.
۳. تولید کننده‌ی بخش‌ها لیستی از بخش‌ها و هش‌های تولید شده‌ی آن‌ها را بازمی‌گرداند.
۴. لیست بخش‌ها به طور امن ذخیره و به اشتراک گذاشته می‌شوند تا همتایان با اطمینان توان اثبات تمامیت لیست و بخش‌ها و هویت سرویس‌دهنده و منابع دریافت بخش‌ها را داشته باشند.

بدین شکل بخش‌ها آماده‌ی انتشار هستند و کارخواه‌ها می‌توانند با اتصالی امن لیست بخش‌های داده و منبع تهیه‌ی آن‌ها را درخواست داده و به دانلود بخش‌های تعیین شده (پس از تایید اولیه بر اساس همخوانی هش درون فراداده‌ی بخش با اطلاعات لیست) بپردازند.

۲.۲.۴ استفاده از لیست برای بررسی امنیت بخش

۱. همتایی لیست بخش‌ها را از سرویس‌دهنده درخواست می‌دهد.
۲. با استفاده از الگوریتم‌های سیستم، همتایان دارای این داده‌ها شناسایی و تایید می‌شوند و همتا شروع به دانلود داده از آن‌ها می‌کند.
۳. بخش در حال دانلود تایید نهایی نشده در قرنطینه باقی می‌ماند تا دانلود آن به پایان رسد.
۴. پس از اتمام دانلود، هش بخش دریافت شده‌ی کامل بر اساس الگوریتم تعیین شده در ابتدا محاسبه می‌شود و با هش ذکر شده در لیست بخش‌ها مقایسه می‌شود. دو حالت ممکن است:

- اگر مقادیر برابر بود اصالت و تمامیت بخش تایید شده و از قرنطینه خارج می‌شود تا به دیگر بخش‌ها بپوندند یا با همتایان درخواست‌دهنده‌ی دیگر به اشتراک گذاشته شود.
- در غیر این صورت بخش دریافتی حذف شده و بخش از همتایی دیگر دانلود و مراحل تکرار می‌شود تا بخش سالم شناسایی شود.

در صورت امن بود الگوریتم هش انتخابی و یکسان بودن الگوریتم مورد استفاده توسط سرویس‌دهنده و همتایان، راه‌کار ارائه شده اصالت و تمامیت داده‌ی دریافتی توسط همتایان را تضمین می‌کند. البته سوال‌هایی درمورد انتخاب هش و همچنین ریسک‌های راهکار موجود است که در بخش‌های بعد به آن‌ها پرداخته می‌شود. در زیر مثال پیشین مشکل سیستم‌های P2P را با راه‌کارهای بالا دوباره بررسی می‌کنیم.

مثال

آلیس داده‌ای حجیم را از باب می‌خواهد، اما باب منابع محدود دارد و داده‌ی خود را به کمک یک نرم‌افزار بر مبنای P2P ذخیره کرده است که داده را به بخش‌های کوچکی تقسیم کرده و میان لیستی از کاربران پخش کرده‌است. باب لیست این افراد را با استفاده از کانالی امن به آلیس می‌فرستد و آلیس از این همتایان درون شبکه داده را درخواست می‌کند. ایو می‌خواهد بدافزار خود را منتشر کند و به جای داده‌ی باب، بدافزار را به آلیس می‌فرستد. پس از دانلود بخش که در قرنطینه نگهداری می‌شود، آلیس هش فایل را محاسبه می‌کند و عدم برابری آن با هش لیست شده را مشاهده می‌کند. پس بخش را به دور می‌اندازد و از همتایی دیگر یه دانلود بخش می‌پردازد. بنابراین در دام ایو نمی‌افتد و در نهایت با به هم چسباندن بخش‌ها فایلی که از باب می‌خواست را بدون نقضی در تمامیت آن در دستگاه خود ذخیره می‌کند.

۳.۴ نمونه‌هایی از استفاده‌ی راه‌کار

برخی از پیاده‌سازی‌های این الگو شامل Freenet، CFS، Kadmila و Chordd می‌باشند. Freenet و CFS فایل‌سیستم‌های P2P هستند که از هش‌ها برای شناسایی هر فایل استفاده می‌کنند. پیاده‌سازی جدول هش تولید شده یا DHT[۱۳] توسط Kadmila نیز برای بیت‌تورنت سازگار شده بود تا در آن سیستم برای یافتن و تایید بخش‌های قابل دانلود استفاده شود. اشکال متفاوتی از DHTها در ساختارهایی مانند Can، Pastry و Tapestry نیز موجودند که در خصوصیاتی مانند طول مسیر یا تعداد همتایان شناخته شده با یکدیگر تفاوت دارند.

۴.۴ مزایا و معایب راه‌کار

راه‌کار ارائه شده با استفاده از سیستم‌های هش نتایجی درمورد کارکرد سیستم کلی دارد که نیازمند بررسی برای تعیین میزان مناسب بودن آن و حیطه‌های بهبودی می‌باشد. این موارد را به شکل زیر داریم.

مزایا:

- به سرویس‌دهنده‌ی اصلی تنها زمان دریافت لیست یا در شرایطی که هیچ همتایی بخشی را ذخیره نداشته باشد مراجعه می‌شود.
- تمامیت بخش دانلودی با مقایسه‌ی هش آن با هش ذکر شده در لیست تعیین می‌شود و برای تعیین و چشباندن بخش‌ها نیازی به سرویس‌دهنده یا همتایی دیگر نیست.
- از هش هر بخش نه تنها برای تایید تمامیت بلکه برای یافتن و تایید اصالت ابتدایی بخش درخواستی استفاده می‌شود.
- مانند تمامی سیستم‌های P2P، دانلود موازی بخش‌ها از چندین همتا منجر به افزایش سرعت و پخش فشار تحمیلی می‌شود.

معایب:

- لیست هش بخش‌ها می‌تواند مورد حمله و سوءاستفاده قرار بگیرد.
- بسته به الگوریتم هش انتخابی، امکان جعل و حمله می‌تواند ممکن باشد.
- به دیگر ضعف‌های سیستم‌های P2P در دیگر حیطه‌ها مانند محرمانگی توجهی نشده است.

همانطور که مشاهده می‌کنیم، انتخاب الگوریتم هش مناسب بحث اصلی درستی و امکان کارایی راه‌کار ارائه شده است. پس نیاز است به بررسی سیستم‌های هش و مقایسه‌ی الگوریتم‌های موجود بپردازیم.

۵ معرفی کارکرد هش و امنیت استفاده‌ی آن

۱.۵ نمای کلی

با یک مثال به شرح کارکرد هش می‌پردازیم.

[13] Distributed Hash Table

مثال

آلیس با این ادعا که توانسته است راه‌حل درستی برای یک مسئله‌ی خیلی سخت پیدا کند پیش باب می‌رود. باب می‌خواهد خود نیز برای حل مسئله تلاش کند اما می‌خواهد مطمئن شود آلیس دروغ نمی‌گوید. به این دلیل آلیس پاسخ خود را به شکل بیتی در آورده و تابعی را به روی آن اعمال می‌کند که داده را به عددی خاص تبدیل می‌کند، آنگاه حاصل را با تابع برای باب می‌فرستد. سپس باب شروع به حل مسئله می‌کند و پس از چند روز به پاسخ درست می‌رسد. آلیس با راه‌حل خود پیش باب می‌آید و باب خود دوباره تابع را به روی پاسخ آلیس اعمال می‌کند و حاصل را با عدد اولیه که از آلیس دریافت کرده بود مقایسه می‌کند. بدین شکل باب می‌تواند مطمئن شود جواب مسئله از قبل وجود داشته است.

به عبارت دیگر، سیستم‌های هش[۱۴] الگوریتم‌هایی هستند که برای نگاشت هر داده به عددی با اندازه‌ی ثابت استفاده می‌شوند. برای مثال می‌توان یک فایل چند بایت تا چند گیگابایتی را با عددی به طول ۲۵۶ بیتی نشان داد. این ویژگی باعث می‌شود هش‌ها کاربردهای متعددی داشته باشند، مانند جستجوی سریع، شمارش و مقایسه‌ی داده‌ها، و درصورت امنیت الگوریتم (تابع هش رمزنگارانه[۱۵] بودن) در حیطه‌ی امنیت اطلاعات مانند بررسی گذرواژه، امضاهای دیجیتال و فرم‌های مختلف احراز هویت. بنابراین سیستم‌های هش از پرکاربردترین ابزارها برای سنجش داده هستند.

کاربرد

بستر VirusTotal از مثال‌های کاربردی هش برای تعیین اصالت داده است که در آن می‌توان با بارگذاری فایل، ویروسی بودن یا سلامت داده را متوجه شد. این بستر از بررسی هش فایل برای تعیین امنیت آن کمک می‌گیرد.

همان‌طور که بیان شد در سیستم‌های P2P اطلاعی از امن و قابل اطمینان بودن همتایان وجود ندارد و ممکن است کاربران قصد ایجاد اختلالی در داده‌ها را داشته باشند، به این علت راه‌کاری برای بررسی اصالت و تمامیت داده با هش ارائه کردیم. اما سوالی که بایستی به آن پاسخ دهیم امنیت این سیستم است. یعنی آیا یک همتا می‌تواند تغییراتی را در بخشی از فایل ایجاد کند اما بعد از اعمال تابع هش به روی داده‌ی جدید همچنان مقدار هش به دست آمده با مقدار هش داده‌ی اصلی برابر باشد؟ این واقعه در توابع هش همان برخورد[۱۶] نام دارد.

۲.۵ برخورد و اهمیت آن در راه‌کار

گفتیم توابع هش همواره برای هر ورودی دارای خروجی‌ای با طول ثابت هستند. برای مثال خروجی الگوریتم SHA-1 یک رشته‌ی ۱۶۰ بیتی است، یعنی نهایتا می‌تواند 2^{160} هش متفاوت تولید کند، این درحالیست که ورودی‌های ممکن برای تابع هش نامحدود است، پس قطعا می‌توان دو داده‌ی متفاوت با یک هش یکسان، یعنی یک برخورد پیدا کرد.

اما مسئله‌ی موجود این است که یافتن برخورد با استفاده از کامپیوترهایی با توان پردازشی محدود بسیار دشوار است؛ برای مثال یافتن برخورد در توابع هش SHA-256 و Keccak-256 عملا غیرممکن است و این برخوردتابی[۱۷] این توابع را معنی می‌دهد. در توابعی نیز مانند MD5 که یافتن برخورد تضمین شده و ممکن است، استفاده از این برخورد برای حمله‌ی بدخواهانه‌ای با منظور خاص سخت خواهد بود.

[14] Hash systems
[15] Cryptographic hash function
[16] Collision
[17] Collision resistance

در بررسی تمامیت داده به طور کلی حتی توابع هش با برخوردهای محدود مانند MD5 در کاربرد مشکلی ایجاد نمی‌کنند. اما از آنجا که راه‌کار ارائه شده تمرکزی بر تضمین امنیت داده دارد، حساسیت موضوع توابع هش رمزنگارانه دارای شرایط خاص امنیتی را می‌طلبد. پس اکنون بایستی بررسی کنیم که کدام توابع هش قابلیت استفاده برای اطمینان از اصالت و تمامیت داده‌های مختلف در سیستم‌های انتقال داده‌ی P2P را دارند.

۶ بررسی امکان استفاده‌ی چند سیستم هش

از آنجا که الگوریتم‌های متنوعی برای محاسبه‌ی هش موجودند، سه مورد از معروف‌ترین و پراستفاده‌ترین الگوریتم‌ها یعنی SHA-1، SHA-2 و SHA-3 را مورد بررسی قرار می‌دهیم تا با درک کارکرد و ویژگی‌های آن‌ها و در انتها مقایسه‌ای نهایی، امکان استفاده‌ی هر کدام برای راه‌کار ارائه شده و ساخت سیستمی امن را پاسخ دهیم.

۱.۶ الگوریتم SHA-1

الگوریتم SHA-1 که توسط NSA[18] ارائه شده است عضوی از خانواده الگوریتم‌های هش امن می‌باشد، الگوریتمی تک‌سو است که هر پیام با هر طولی را به شکل یک هش ۱۶۰ـبیتی نگاشت می‌کند که توسط یک عدد ۴۰ بر مبنای ۱۶ نمایش داده می‌شود.

ساختار آن گونه‌ای است که با استفاده از پدینگ[19] طول ورودی را به اندازه‌ای می‌کند که باقی‌مانده‌ی آن بر ۵۱۲ برابر ۴۴۸ بشود، سپس طول متن را درقالب یک رشته‌ی ۶۴ـبیتی به انتهای آن اضافه می‌کند تا طول رشته‌ی نهایی مضربی از ۵۱۲ شود. سپس توسط ۸۰ عمل محاسباتی که هر کدام شامل ۲۰ مرحله از تبدیلات بیتی است رشته‌ی اولیه به یک هش ۱۶۰ـبیتی نگاشت می‌شود.

با وجود این که در سال ۲۰۰۵ به طور تئوری شکسته شد، در سال ۲۰۱۱ از لیست هش‌های استاندارد NIST[20] خارج شد و همچنین در سال ۲۰۱۷ در پروژه‌ی SHAttered به طور رسمی و عملی شکسته شد، این الگوریتم همچنان به عنوان یک الگوریتم به طور جامع کاربردی به حساب می‌آید و از آن همچنان در ذخیره‌سازی رمزهای کاربران آنلاین استفاده می‌شود. زیرا برای تایید رمز هر کاربر صرفا نیاز به داشتن یک هش از رمزی است که برای دفعه‌ی اول وارد می‌شود تا برای بررسی و تایید رمز در دفعات بعدی که وارد می‌شود استفاده شود.

مزایا:

- کند بودن نسبی الگوریتم باعث می‌شود احتمال دستیابی به هش‌های مشابه فایل هدف بسیار کند پی برود.
- می‌توان از آن برای تغییرات ناآگاهنه‌ی ایجاد شده در فایل‌ها و کدها استفاده کرد.
- می‌توان از آن برای جلوگیری از عدم سازگاری با سیستم‌های قدیمی‌تر به جای الگوریتم‌های هش جدید استفاده کرد.

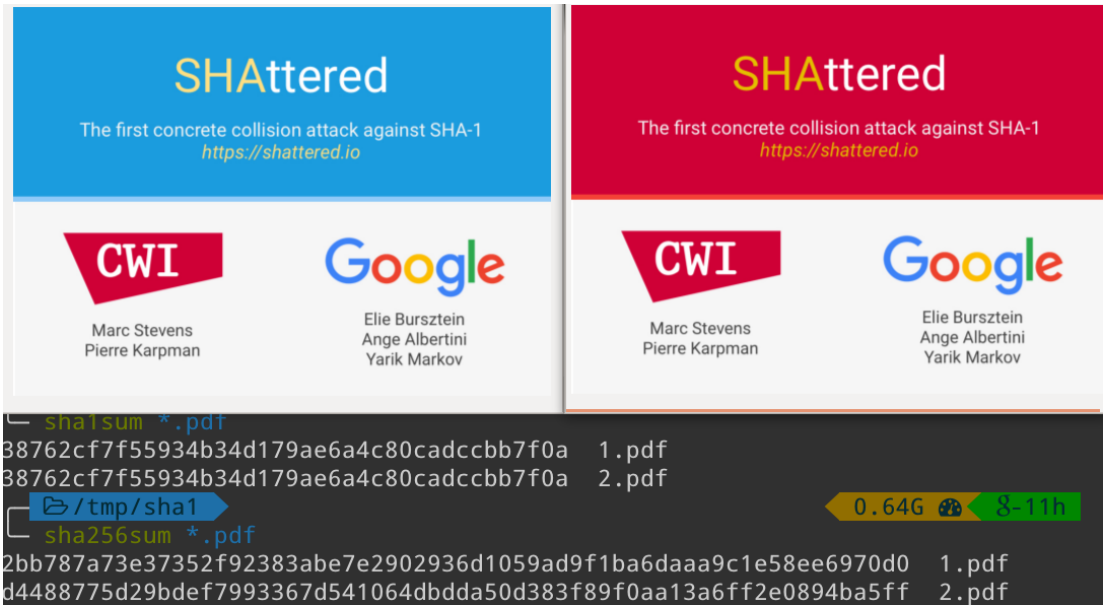
معایب:

- کند بودن الگوریتم باعث می‌شود که کاربردهای آن بسیار محدود شود.
- امنیت پایینی دارد زیرا نقاط ضعف بیشتری در آن در طول زمان پیدا شده است.
- پیدا کردن برخورد در آن در طول زمان راحت‌تر و با قدرت محاسباتی کمتر میسر می‌شود.

[18] National Security Agency
[19] Padding
[20] National Institute of Standards and Technology

نکته

پروژه‌ی SHAttered همکاری محققان CWI و گوگل است که در سال ۲۰۱۷ پس از دو سال تلاش موفق به شکستن عملی الگوریتم SHA-1 شدند. حمله‌ی خاص آن‌ها به SHA-1 نیروی پردازشی معادل ۱۱۰ سال GPU یک هسته‌ای را نیاز دارد، که صد هزار بار سریع‌تر از حمله‌ی بروت فورس است که با تکیه بر پارادوکس تولد به 12,000,000 سال GPU نیازمند است.



شکل ۴: نمونه حمله‌ی پروژه‌ی SHAttered

۲.۶ الگوریتم SHA-2

خانواده الگوریتم‌های SHA-2 الگوریتمی مدرن‌تر و در تکامل الگوریتم نسبتا قدیمی SHA-1 است که توسط NIST نیز برای کاربردهای امنیتی پیشنهاد می‌شود. این خانواده الگوریتم شامل شش ساخت مختلف است:

- **SHA-224**: نسخه‌ی کوتاه شده از SHA-256 است که با داده‌های درونی متفاوتی محاسبه می‌شود.
- **SHA-256**
- **SHA-384**: نسخه‌ی کوتاه شده از SHA-512 است که با داده‌های درونی متفاوتی محاسبه می‌شود.
- **SHA-512**
- **SHA-512/224**: نسخه‌ی کوتاه شده از SHA-512 است.
- **SHA-512/256**: نسخه‌ی کوتاه شده از SHA-512 است.

الگوریتم SHA-256 پراستفاده‌ترین عضو این خانواده است، به خصوص توسط آژانس‌های مختلف دولت آمریکا برای بخش‌های امنیتی داده‌های حساس استفاده می‌شود. این الگوریتم بسیار قوی‌تر از الگوریتم SHA-1 است و در زمان عرضه امن‌ترین الگوریتم هش موجود بود. همچنین یکی از کاربردهای بسیار مهم و روزمره‌ی SHA-256 استفاده از آن در رمزارز بیت‌کوین است.

این الگوریتم به دلیل سرعت تولید هش بالایی که دارد، امنیت زیادی نسبت به الگوریتم‌های هش دیگر دارد، توسط اکثر سیستم عامل‌ها و بسترهای حال حاضر پشتیبانی می‌شود، قابلیت کاربرد برای محیط‌های کاربری مدرن و یا نسبتا قدیمی را داراست، و می‌تواند گزینه‌ای ایده‌آل برای کاربردهایی در زندگی روزمره باشد. در ادامه بر بررسی الگوریتم SHA-256 از میان الگوریتم‌های SHA-2 تمرکز می‌کنیم.

مزایا:

- حتی در برابر کامپیوترهای کوانتومب برخوردتاب است و خواص امنیتی PR[۲۱] و SPR[۲۲] را داراست.
- به نقاط ضعف SHA-1 رسیدگی کرده و آنها را برطرف کرده است.
- توسط اکثر نرم‌افزارهای روز و سیستم‌های عامل و سامانه‌های انتقال پیام پشتیبانی می‌شود.
- از آنجایی که طول هش تولید شده بیشتر است سطح امنیتی بسیار بالاتری را ارائه می‌دهد که برای کارهای امنیتی مناسب است.

معایب:

- نسخه‌ی SHA-256 نسبت به دیگر الگوریتم‌های پیشین خود کندتر است.
- بعضی نرم‌افزارها ممکن است نیاز به به‌روز رسانی داشته باشند.

۳.۶ الگوریتم SHA-3

این خانواده الگوریتم جدیدترین عضو از مجموعه‌ی Secure Hash Algorithmها است که توسط یک مسابقه‌ی عمومی ترتیب داده شده توسط NIST توسعه داده شد. با وجود اینکه SHA-3 دارای استانداردهای یکسانی با SHA-1، SHA-2و MD5 است اما از نظر ساختاری کاملا متفاوت است.

این الگوریتم بر اساس رویکرد رمزنگاری جدیدی که ساختار اسفنجی[۲۳] نامیده می‌شود و توسط تیم کِچکKeccak Team استفاده می‌شود توسعه داده شده است. نام آن به این دلیل ساختار اسفنجی گذاشته شده است که طوری به نظر می‌رسد که داده را جذب می‌کند و سپس مانند اسفنجی که آب را جذب کرده بعد از فشرده شدن به بیرون می‌دهد.

سازمان استاندارد امنیتی آمریکا این خانواده الگوریتم را صرفا به عنوان تقویت کننده خانواده الگوریتم‌های SHA-2 می‌بیند و نه یک جایگزین کاملا برتر نسبت به آن. این خانواده الگوریتم دارای ۴ عضو اصلی و دو تابع خروجی دهنده‌ی متفاوت است که برای تولید هش رندوم، رمزنگاری جریان داده و ساخت MAC آدرس‌ها کاربرد دارند. این اعضا عبارتند از:

- **SHA-224**
- **SHA-256**
- **SHA-384**
- **SHA-512**
- **SHAKE-128** (تابع خروجی)
- **SHAKE-256** (تابع خروجی)

[21] Preimage Resistance
[22] Second Preimage Resistance
[23] Sponge Construction

این خانواده از الگوریتم‌ها را به مانند قبل بررسی می‌کنیم.

مزایا:

- در زمینه‌ی گرفتن ورودی و خروجی‌هایی با اندازه‌های متفاوت انعطاف‌پذیر است.
- اگر رمزنگاری به طور سخت‌افزاری صورت بگیرد بسیار سریع‌تر از دیگر توابع هش می‌تواند محاسبات را انجام دهد.
- نمونه‌های آن از یک جایگشت برای تمامی توان‌های امنیتی استفاده می‌کنند که باعث کاهش هزینه‌ها می‌شود.
- می‌تواند با توجه به نیاز و کاربردی که دارد به طور متناسب میزان امنیت یا سرعت پردازش آن را تغییر داد.
- نسبت به حمله‌های افزایش طول مقاوم است.

معایب:

- به دلیل مشکلات نرم‌افزاری و نه صرفا نیاز به توان پردازی بیشتر از SHA-2 کندتر است.
- به علت تازگی با کمبود پشتیبانی توسط نرم‌افزارها و سخت‌افزارهای مختلف مخصوصا موارد قدیمی‌تر روبه‌روست.

۴.۶ مقایسه‌ی کلی

خلاصه‌ای از مقایسه‌ی این ۳ نوع الگوریتم در جدول ۱ قابل مشاهده است. همانطور که می‌بینیم از میان انواع الگوریتم‌های SHA، الگوریتم SHA-2 دارای بهترین ترکیب پشتیبانی، امنیت و سرعت است که آن را گزینه‌ی مناسبی برای استفاده‌های وسیع در حیطه‌های رمزنگاری کرده است. بنابراین به عنوان تابع هش رمزنگارانه‌ی مورد نیاز در راه‌کار ارائه شده نیز گزینه‌ی کاربردی‌ای می‌باشد که راه‌کار را ممکن می‌کند.

الگوریتم	SHA-1	SHA-2 (SHA-256)	SHA-3
سال در دسترس قرار گرفتن	۱۹۹۵	۲۰۰۲	۲۰۰۸
ساختار	Merkle-Damgard	Merkle-Damgard	Sponge (Keccak)
سطح یافتن برخورد	نسبتا زیاد – به طور تئوری و عملی شکسته شده است	کم	کم
سرعت Intel 2nd Gen Core cycle per byte: cpb	6.55 - 7.79 cpb	SHA-256: 15.31 - 18.62 cpb SHA-512: 9.71 - 11.82 cpb	20 cpb
کاربردها	برای HMAC و اطمینان از اصالت و تمامیت داده در مقابل تغییرات ناخواسته	در پروتکل‌های امنیتی مانند TLS، SSL، PGP، SSH، تایید انتقال ارزهای دیجیتال، تصدیق کننده سندهای دیجتال، و دیگر نرم‌افزارهای کاربردی	صرفا برای جایگزینی SHA-2 در مواردی که عملکرد بهتری می‌تواند داشته باشد

جدول ۱: مقایسه‌ی انواع الگوریتم‌های SHA

۷ نتیجه‌گیری و بحث

بر این اساس می‌توانیم بگوییم که حتی با منابع محدود می‌توان داده‌های حجیم را میان شبکه‌ای بزرگ از کاربران ناشناحته و غیرقابل اطمینان به اشتراک گذاشت در حالی که اصالت و تمامیت داده حفظ شود. بدین شکل که داده به بخش‌هایی تقسیم شده و برای هر کدام هشی محاسبه و ذخیره می‌شود، آنگاه سرویس‌دهنده تنها کافیست لیست هش بخش‌های اصلی و نسخه‌ی ابتدایی داده را به اشتراک بگذارد تا همتایان به دانلود مستقیم بخش‌ها از هم بپردازند. امنیت چنین سیستمی با بررسی هش بخش پیش از پذیرش و باز کردن آن در دستگاه کاربر تضمین می‌شود.

البته انتخاب هش مناسب برای تضمین این امنیت موضوع حساسیست و الگوریتم هش انتخابی حتما بایستی برخوردتاب و سریع باشد. برای مثال، الگوریتم SHA-1 ناامن و کند است و الگوریتم SHA-3 بسیار امن اما جدید و بدون پشتیبانی مناسب است. اما الگوریتم SHA-2 سریع و امن (حتی امن در مقابل کامپیوترهای کوانتومی) بوده و گزینه‌ای ایده‌آل برای طرح ارائه شده است. بنابراین استفاده از سیستم‌های هش برای اثبات اصالت و تمامیت داده در سیستم‌های P2P ممکن است.

در نهایت سوالاتی باقی می‌مانند که برای تکمیل مبحث امنیت چنین شبکه‌هایی نیاز به پاسخ دارند اما در این تحقیق پوشش داده نشده‌اند:

- چگونه می‌توان تشخیص داد داده‌ای که هش آن با هش اصلی برابر نبوده است دارای بدافزار بوده و از همتایی جدید باید خواسته شود یا به علت اختلالات در حین انتقال ناخواسته تغییر یافته است و مشکلی در درخواست دوباره از این همتا نیست؟
- آیا داده را به اشتراک گذاشته یا دانلود می‌کنیم و بعد عواقب آن را بررسی می‌کنیم؟ یعنی بر مبنای ریسک عمل می‌کنیم؟ یا از اینکار پیشگیری می‌کنیم؟ اگر پیشگیری استراتژی بهتری است، چگونه می‌توان آن را میسر کرد؟
- آیا تمام داده را با افراد غیرقابل اطمینان به اشتراک می‌گذاریم یا بخشی از داده را؟ اگر تنها بخشی از آن را، چگونه این بخش را تعیین می‌کنیم؟ چگونه افراد غیرقابل اطمینان را شناسایی می‌کنیم؟

۸ منابع

1. Zhang, X., Chen, S., & Sandhu, R. (2005). Enhancing data authenticity and integrity in p2p systems. IEEE Internet computing, 9(6), 42-49.
2. Schleinzer, B., & Yoshioka, N. (2010, October). A security pattern for data integrity in p2p systems. In Proceedings of the 17th Conference on Pattern Languages of Programs (pp. 1-5).
3. Hamlen, K. W., & Thuraisingham, B. (2007, November). Secure peer-to-peer networks for trusted collaboration. In 2007 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom 2007) (pp. 58-63). IEEE.
4. BitTorrent: How it works and how to use it. (2015, February 22). Doug Vitale Tech Blog. $https://dougvitale.wordpress.com/2012/02/01/bittorrent-how-it-works-and-how-to-use-it$
5. Wikipedia contributors. (2023, January 10). Cryptographic hash function. Wikipedia. $https://en.wikipedia.org/wiki/Cryptographic_hash_function$
6. Hunt, T. (2022, July 5). Understanding Have I Been Pwned's Use of SHA-1 and k-Anonymity. Troy Hunt. $https://www.troyhunt.com/understanding-have-i-been-pwneds-use-of-sha-1-and-k-anonymity$
7. CodeSigningStore. (2022, March 23). Hash Algorithm Comparison: MD5, SHA-1, SHA-2 & SHA-3. Code Signing Store. $https://codesigningstore.com/hash-algorithm-comparison$
8. SHAttered. (n.d.). $https://shattered.io$
9. Gueron, S. (2012, April). Speeding Up SHA-1, SHA-256 and SHA-512 on the 2nd Generation Intel® Core™ Processors. In 2012 Ninth International Conference on Information Technology-New Generations (pp. 824-826). IEEE.
10. Wikipedia contributors. (2022, December 24). SHA-3. Wikipedia. $https://en.wikipedia.org/wiki/SHA-3$
11. RFC ft-eastlake-sha2b: US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF). (2011, May 6). IETF Datatracker. $https://datatracker.ietf.org/doc/html/rfc6234$
12. Dang, Q. H. (2015). Secure Hash Standard (SHS). Federal Information Processing Standards Publication, FIPS PUB 180-4. $https://doi.org/10.6028/nist.fips.180-4$