دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر
گروه مخابرات امن و رمزنگاری

تحلیل امنیت یک شبکه همتابه‌همتا مبتنی بر زنجیره قالبی

پایان‌نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ مهندسی برق
گرایش مخابرات امن و رمزنگاری

محمّدتقی بدخشان

استاد راهنما

دکتر محمّدعلی اخایی

مهر ۱۳۹۹


بسم الله الرحمن الرحيم

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر
گروه مخابرات امن و رمزنگاری

تحلیل امنیت یک شبکه همتابه‌همتا مبتنی بر زنجیره قالبی

پایان‌نامه برای دریافت درجهٔ کارشناسی ارشد در رشتهٔ مهندسی برق
گرایش مخابرات امن و رمزنگاری

محمّدتقی بدخشان

استاد راهنما

دکتر محمّدعلی اخایی

مهر ۱۳۹۹

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده مهندسی برق و کامپیوتر

گواهی دفاع از پایان‌نامه کارشناسی ارشد

هیأت داوران پایان‌نامهٔ کارشناسی ارشد آقای / خانم محمّدتقی بدخشان به شمارهٔ دانشجویی ۸۱۰۱۹۶۳۶۹ در رشتهٔ مهندسی برق - گرایش مخابرات امن و رمزنگاری را در تاریخ ..... با عنوان «تحلیل امنیت یک شبکه همتابه‌همتا مبتنی بر زنجیره قالبی»

	به عدد	به حروف
با نمرهٔ نهایی		

و درجهٔ [ ] ارزیابی کرد.

ردیف	مشخصات هیأت داوران	نام و نام خانوادگی	مرتبهٔ دانشگاهی	دانشگاه یا مؤسسه	امضا
۱	استاد راهنما	دکتر محمّدعلی اخایی	استادیار	دانشگاه تهران	
۲	استاد داور داخلی	دکتر داور داخلی	دانشیار	دانشگاه تهران	
۳	استاد مدعو	دکتر داور خارجی	دانشیار	دانشگاه داور خارجی	
۴	نمایندهٔ تحصیلات تکمیلی دانشکده	دکتر نماینده	دانشیار	دانشگاه تهران	

نام و نام خانوادگی معاون تحصیلات تکمیلی و پژوهشی دانشکده / گروه:

تاریخ و امضا:

نام و نام خانوادگی معاون آموزشی و تحصیلات تکمیلی پردیس دانشکده‌های فنی:

تاریخ و امضا:

تعهدنامهٔ اصالت اثر

باسمه تعالی

اینجانب محمّدتقی بدخشان تأیید می‌کنم که مطالب مندرج در این پایان‌نامه حاصل کار پژوهشی اینجانب است و به دستاوردهای پژوهشی دیگران که در این نوشته از آن‌ها استفاده شده است مطابق مقررات ارجاع گردیده است. این پایان‌نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتری ارائه نشده است.

نام و نام خانوادگی دانشجو: محمّدتقی بدخشان

تاریخ و امضای دانشجو:

تقدیم به:

پدر و مادرم

قدردانی

سپاس خداوندگار حکیم را که با لطف بی‌کران خود، آدمی را به زیور عقل آراست.

در آغاز وظیفه خود می‌دانم از زحمات بی‌دریغ اساتید راهنمای خود، جناب آقای دکتر محمدعلی اخایی، صمیمانه تشکر و قدردانی کنم که در طول انجام این پایان‌نامه با نهایت صبوری همواره راهنما و مشوق من بودند و قطعاً بدون راهنمایی‌های ارزنده ایشان، این مجموعه به انجام نمی‌رسید.

با سپاس بی‌دریغ خدمت دوست گران‌مایه‌ام، آقای حبیب‌الله یاجم در آزمایشگاه مخابرات امن دانشگاه تهران، که با همفکری مرا صمیمانه و مشفقانه یاری داده‌اند.

و در پایان، بوسه می‌زنم بر دستان خداوندگاران مهر و مهربانی، پدر و مادر عزیزم و بعد از خدا، ستایش می‌کنم وجود مقدس‌شان را و تشکر می‌کنم از خانواده عزیزم به پاس عاطفه سرشار و گرمای امیدبخش وجودشان، که بهترین پشتیبان من بودند.

محمّدتقی بدخشان

مهر ۱۳۹۹

چکیده

کاربران سبک[1]، بخش قابل توجهی از شبکه همتابه‌همتا[2] بیت‌کوین[3] را تشکیل می‌دهند. از طرف دیگر این کاربران برای دریافت اطلاعات خود به گره کامل[4] وابسته هستند، در نتیجه بخش قابل توجهی از اطلاعات آن‌ها نزد گره‌های کامل فاش می‌شود. از این رو حفظ حریم خصوصی[5] آن‌ها دارای اهمیت زیادی است. استفاده از فیلتر بلوم[6] که در طرح ۳۷ پیشنهاد بهبود بیت‌کوین، به عنوان اولین راه‌حل حفظ حریم خصوصی کاربران سبک مطرح شد، دارای ایرادات اساسی بسیار زیادی است. در این روش از یک سری آدرس‌های پوششی ، در نتیجهٔ خطای نوع دو[7] فیلتر بلوم، برای پنهان کردن آدرس کاربر سبک استفاده شده است. در این پایان‌نامه ضعف‌های استفاده از فیلتر بلوم در شبکهٔ همتابه‌همتای بیت‌کوین بیان شده است. همچنین مروری بر راه‌حل‌هایی که به عنوان جایگزین فیلتر بلوم یا بهبود دهنده آن مطرح شده‌اند، انجام شده است.

در این پایان‌نامه سعی شده است راه‌حلی ارائه شود که برخلاف فیلتر بلوم که آدرس‌های پوششی به صورت تصادفی انتخاب می‌شدند، آدرس‌ها به صورت هوشمندانه انتخاب می‌شوند. آدرس‌های انتخابی در این روش به نحوی انتخاب شده‌اند که آنتروپی مجموعه آدرس‌های درخواست شده بیشینه شده و گره کامل با ابهام بیشتری مواجه شود. همچنین در این روش امکان تنظیم پهنای باند مصرفی به ازای هر درخواست برای کاربر سبک فراهم شده است.

واژگان کلیدی زنجیرهٔ بلوکی، بیت‌کوین، حفظ حریم خصوصی، کاربر سبک، درستی سنجی پرداخت ساده‌شده

---

[1] Lightweight Client
[2] Peer-to-Peer (P2P) Network
[3] Bitcoin
[4] Full Node
[5] Privacy
[6] Bloom Filter
[7] False Positive

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

با معرفی زنجیره بلوکی[1] بیت‌کوین، به عنوان اولین زنجیرهٔ بلوکی، باب تازه‌ای در کاربردهایی که نیاز به اعتماد به یک طرف سوم دارند گشوده شد. زنجیرهٔ بلوکی بیت‌کوین امکان نگهداری و انتقال دارایی را بدون نیاز به اعتماد به هیچ واسطه‌ای، مانند بانک‌ها، فراهم کرد. پیش از آن اعتماد به بانک‌ها دارای ایرادات فراوانی بود که از آن می‌توان به این موارد اشاره کرد: عدم شفافیت، قطع دسترسی افراد به دارایی‌هایشان، کنترل ناعادلانه تورم، نقض حریم خصوصی افراد و غیره.

رمز ارز بیت‌کوین برای دست‌یابی به چنین امکانی و رفع نواقص بانک‌داری موجود از مجموعه‌ای از مفاهیم و فناوری‌ها مانند یک شبکه همتابه‌همتا، پایگاه داده توزیع‌شده‌ای به اسم زنجیرهٔ بلوکی، الگوریتم‌های رمزنگاری جهت صدور و تصدیق تراکنش‌ها، و الگوریتم اجماع برای آن که تمام اعضای شبکه بر روی یک زنجیرهٔ بلوکی یکتا توافق داشته باشند استفاده می کند.

در نظام بانکی فعلی موجود حساب افراد مستقل از خود دارایی آن‌ها است، ولی در رمز ارز بیت‌کوین هر بیت‌کوین خود دارای ارزش است. به بیان ساده‌تر در نظام بانکی فعلی اگر فردی رمزعبور حسابش را فراموش کند، با احراز هویت حضوری در بانک می‌تواند به رمز عبور جدیدی دسترسی پیدا نماید. همچنین اگر محرز شود که دارایی یک فرد به سرقت رفته است، بانک می‌تواند دارایی فرد متضرر را از سارق پس بگیرد و به حساب اصلی بازگرداند. اما در رمزارز بیت‌کوین، داشتن کلید خصوصی به منزله مالکیت بر دارایی است. اگر کاربر بیت‌کوین کلید خصوصی را گم کند دیگر به دارایی خود دسترسی نخواهد داشت و از طرف دیگر اگر کلید خصوصی وی در دست یک فرد متخاصم قرار بگیرد امکان بازگردانی دارایی وی وجود نخواهد داشت. از این رو امنیت بیت‌کوین دارای چالش‌های بسیار زیاد است.

[1] Blockchain

امنیت در بیت‌کوین از جنبه‌های مختلفی تحلیل می‌شود. در این پژوهش تمرکز اصلی بر روی حریم خصوصی کاربران سبکی است که تمام زنجیره بلوکی را ذخیره نکرده‌اند. عملکرد این کاربران به کاربران دیگری که تمام زنجیرهٔ بلوکی را ذخیره کرده‌اند وابسته است. این وابستگی باعث می‌شود که اطلاعات این کاربران نزد کاربری دیگر فاش شود. فاش شدن اطلاعات می‌تواند تبعاتی به همراه داشته باشد. که می‌توان به افشای هویت کاربر سبک در نتیجهٔ آگاهی یک گره دیگر از آدرس‌هایی که مربوط به آن است اشاره نمود.

اینکه مشخص شود که کدام آدرس‌ها مربوط به کدام کاربر است، می‌تواند باعث فاش شدن تمام فعالیت‌ها و تبادلات مالی آن کاربر شود. علاوه بر این از آن‌جایی که به این طریق می‌توان به دارایی یک فرد پی برد، ممکن است آن فرد در معرض سوء قصد فیزیکی قرار گیرد. چرا که در صورتی که یک فرد بتواند تنها کلیدهای خصوصی قربانی را دریافت نماید، می‌تواند نسبت به تمام دارایی‌های وی در شبکهٔ بیت‌کوین مالکیت داشته باشد. همچنین به خاطر ذات غیر متمرکز بودن این شبکه امکان باز گرداندن دارایی‌های از دست رفته وجود ندارد.

توجه به این نکته ضروری است که با افشای اطلاعات یک کاربر، اطلاعات تمام کاربرانی که با این کاربر مبادله انجام داده‌اند نیز در معرض خطر افشا قرار می‌گیرد. در نتیجه حفظ حریم خصوصی در شبکهٔ بیت‌کوین به جای آن‌که یک امکان باشد، باید به یک الزام تبدیل شود و به صورت ذاتی در پروتکل‌های آن از افشای هویت کاربران جلوگیری شود.

در فصل ۲ به تعریف کاربر سبک، بررسی پروتکل ارتباطی وی در شبکهٔ همتابه‌همتای بیت‌کوین و مرور آسیب‌پذیری‌های موجود در پروتکل ارتباطی فعلی پرداخته شده است. فصل ۳ راه‌حل‌هایی که تاکنون برای حل این مسئله بیان شده‌اند را مرور کرده است. در فصل ۴ به بیان راه‌کاری برای جبران نقص پروتکل فعلی پرداخته ایم. در این راه‌حل برخی از ایراداتی که در راه‌حل‌های جایگزین دیگر وجود دارند برطرف شده است.

فصل ۲

تعاریف، اصول و مبانی نظری

۱.۲ مقدمه

در مقالهٔ[۳۶]، در کنار معرفی بیت‌کوین، روشی به نام درستی سنجی پرداخت ساده‌شده[۱] (SPV) معرفی شده است. در این روش، امکانی به شبکه بیت‌کوین اضافه گشت که دسته‌ای از کاربران، بدون نیاز به راه‌اندازی یک گره کامل، بتوانند با اثبات مرکلی که از یک گره کامل دریافت می‌کنند، تایید کنند که یک تراکنش درون زنجیره بلوکی بیت‌کوین ثبت گردیده است یا خیر. به این کاربران، کاربر سبک و به گره آن‌ها در شبکه بیت‌کوین، گره سبک گفته می‌شود. گره‌های سبک یا به عبارت دیگر گره‌هایی که در وضعیت تایید پرداخت ساده‌شده عمل می‌کنند، نیازی به ذخیرهٔ تمام زنجیره بلوکی موجود ندارند. این گره‌ها تنها سرایند زنجیره بلوکی را از شبکه دریافت و ذخیره می‌کنند. هرچند که در این روش کاربران سبک نیاز به بارگیری تمام زنجیره بلوکی بیت‌کوین ندارند و تنها لازم است که سرایند بلوک‌ها را ذخیره کنند، اما عملکرد صحیح آن‌ها در گرو ارتباط آن‌ها با یک گره کامل درست کار است. اگرچه گره‌های سبک می‌توانند تایید کنند که سرایند بلوک‌هایی که دریافت کرده‌اند اثبات کار صحیحی دارند یا خیر، اما بدون داشتن تمام زنجیره بلوکی نمی‌توانند مطمئن شوند که تمام تراکنش‌های موجود در بلوک‌ها کاملا درست هستند.

آسیب‌پذیری دیگری که گره‌های سبک را تهدید می‌کند، عدم حفظ حریم خصوصی آن‌ها در مقابل گره‌های کاملی است که از آن‌ها درخواست اطلاعات می‌نمایند. یکی از اصلی‌ترین اطلاعاتی که گره‌های سبک از گره‌های کامل درخواست می‌کنند تراکنش‌های مربوط به آدرس(های) گره سبک است. کاربر سبک علاوه بر تراکنش مورد نظر، سرایند بلوکی که تراکنش در آن قرار دارد و همچنین اثبات مرکل وجود آن تراکنش در آن بلوک را دریافت

[1] Simplified Payment Verification (SPV)

می‌کند. در صورتی که گره سبک به صورت فاش اطلاعات آدرس خود را در اختیار گره کامل قرار دهد، گره کامل خواهد توانست اولا، ارتباط آدرس‌های بیت‌کوین گره سبک با آدرس آی‌پی وی را کشف نماید و در نهایت بفهمد که دارنده این آدرس در کدام موقعیت جغرافیایی قرار دارد. این امر می‌تواند باعث افشای هویت آن کاربر شود و حتی می‌تواند تهدیدی جانی برای کاربر سبک باشد اگر حملهٔ فیزیکی به آن فرد برای دزدیدن اطلاعات کیف پولش، که دارایی آن افشا شده است، صورت پذیرد. اگر کاربر سبک از شبکه حافظ گم‌نامی[۲] مثل تور[۳] استفاده نماید این امکان برای گره کامل وجود نخواهد داشت. ثانیا با استفاده یا عدم استفاده از شبکه این امکان به گره کامل داده می‌شود که بتواند از این طریق ارتباط بین آدرس‌های یک شخص را در شبکه بیت‌کوین ساده‌تر کشف کند. کشف آن که کدام آدرس‌های بیت‌کوین مربوط به یک کاربر به خصوص است، می‌تواند به کشف الگوی رفتاری آن کاربر و در نتیجه کشف نسبی هویت آن منجر شود [۴۴]. از این رو فاش شدن هر دوی این اطلاعات حریم خصوصی کاربر سبک را نقض خواهد کرد.

به این ترتیب، گره سبک به خاطر اعتماد به یک یا چند گره کامل و نقض حریم خصوصیش از امنیت کمتری نسبت به گره‌های کامل برخوردار است [۴۵]. از این رو تاکید می‌شود کاربرانی که مقدار زیادی بیت‌کوین را نگهداری یا مبادله می‌کنند، یا کاربرانی که می‌خواهند گم‌نامی آن‌ها حفظ شود از گره کامل استفاده کنند. با این حال لازم است که تلاش شود امنیت، به ویژه گم‌نامی کاربران سبک تا حد امکان تامین گردد. چرا که فاش شدن اطلاعات بخشی از اعضای شبکه می‌تواند منجر به فاش شدن اطلاعات دیگر بخش‌های شبکه گردد.

در پروتکل فعلی بیت‌کوین برای حل مشکل فاش شدن آدرس مربوط به گره سبک نزد گره کامل متخاصم، از فیلتر بلوم استفاده می‌شود [۲۸]. در مقاله [۲۵] توضیح داده شد که استفاده از فیلتر بلوم از امنیت کافی برخوردار نیست. در این فصل پایان‌نامه به معرفی فیلتر بلوم، نحوهٔ استفاده آن در شبکه همتابه‌همتای بیت‌کوین و ضعف‌های آن به عنوان ابزاری جهت حفظ حریم خصوصی کاربران خواهیم پرداخت. در فصل بعد، مروری خواهیم کرد بر راه‌حل‌هایی که برای بهبود امنیت پروتکل فعلی ارائه شده است و همچنین روش‌هایی که جایگزین پروتکل فعلی هستند.

۲.۲ بیت‌کوین

رمزارز بیت‌کوین برای محقق ساختن اهدافی چون تبادل و نگه‌داری دارایی دیجیتال بدون نیاز به یک طرف سوم مورد اعتماد از یک الگورتیم اثبات کار[۴] (PoW) استفاده می‌کند[۳۶]. الگوریتم اجماع اثبات کار بیت‌کوین

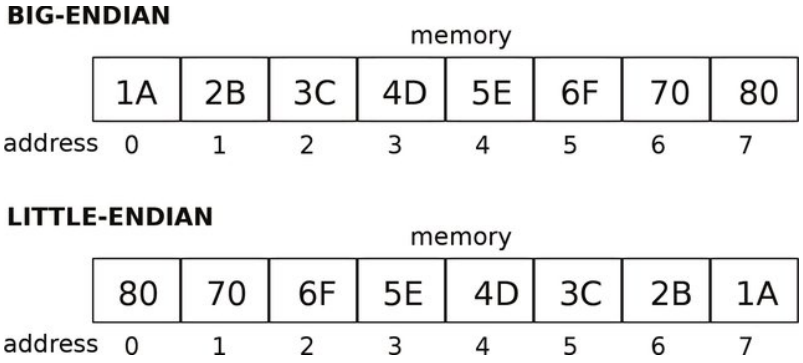
[2] Anonymity Network
[3] Tor
[4] Proof of Work

تضمین می‌کند که بدون احتیاج به وجود یک طرف قابل اعتماد در شبکه، کسی که توان پردازشی آن کمتر از ۵۰ درصد از شبکه باشد نتواند حملهٔ دوبار خرج کردن[۵] را اجرا کند. تراکنش‌ها در بیت‌کوین درون بلوک‌هایی قرار می‌گیرند که این بلوک‌ها طبق الگوریتم اجماع کار تولید می‌شوند. هر بلوک به بلوک قبلی متصل است و تغییر در هر کدام از بلوک‌ها عملا ناشدنی است. در رمزارز بیت‌کوین، دارایی‌های هر کس در اکثر مواقع توسط کلید عمومی و خصوصی او مدیریت می‌شود. به این صورت که اگر یک فرد بخواهد مقداری بیت‌کوین دریافت کند، باید تراکنشی در زنجیرهٔ بلوکی بیت‌کوین قرار بگیرد که خروجی آن شرطی یا نبشته[۶] ای باشد که تنها آن فرد بتواند آن شرط را براورده کند. این شرط می‌تواند اشاره به کلید عمومی آن فرد باشد. مالک بیت‌کوین برای آن که این دارایی را به دیگری منتقل نماید، باید تراکنشی ایجاد کند که در ورودی آن ثابت کند که امکان براورده کردن این شرط را دارد. مثلا می‌تواند با کلید خصوصیش امضای دیجیتال انجام دهد و به این ترتیب ثابت کند که مالک آن کلید عمومی قرار گرفته در تراکنشی است که می‌خواهد آن را خرج نماید. به دلایلی چون خوانایی بهتر، کدگذاری‌های تشخیص خطا و غیره، معمولا به جای کلید عمومی از آدرس‌های بیت‌کوین استفاده می‌شود. این آدرس‌ها چکیدهٔ کلید عمومی هستند که توسط یک روش کدگذاری تشخیص خطا، کدگذاری شده‌اند.

به این ترتیب اگر مستقیما کلید عمومی فرد دریافت کننده در خروجی تراکنش قرار گیرد، به آن استاندارد P2PK[۷] گفته می‌شود. و اگر خروجی آن، آدرس آن فرد باشد که با ۱ شروع شود به آن استاندارد P2PKH[۸] گفته می‌شود. این نوع نبشته بیشترین کاربرد را در بیت‌کوین دارد. جدیدا، با معرفی سِگویت[۹] در بیت‌کوین، نبشته‌ای با نام P2WPKH[۱۰] معرفی شده است که با bc شروع می‌شود. نوع نبشتهٔ دیگری برای شرایط خرج کردن وجود دارد به نام P2SH[۱۱] ، در خروجی تراکنش، چکیدهٔ یک نبشته قرار می‌گیرد که خرج کننده باید خود آن نبشته را ارائه نماید. نبشتهٔ شرط خروجی برای این دسته با ۳ شروع می‌شود.

بعضی از دادگان ذخیره‌شده در تراکنش‌ها و سرایند بلوک‌های بیت‌کوین به صورت اندین کوچک[۱۲] ذخیره شده است، که از آن‌ها می‌توان به چکیده‌ها اشاره نمود. در شکل ۱.۲ مقایسهٔ روش اندین کوچک و روش اندین بزرگ[۱۳] (روشی که انسان‌ها به صورت طبیعی اعداد را می‌نویسند) را برای دادهٔ 0x1A2B3C4D5E6F7080 انجام داده است.

[5] Double-Spend Attack
[6] Script
[7] Pay To Pubkey
[8] Pay To Pubkey Hash
[9] Segwit
[10] Pay to Witness Script Hash
[11] Pay To Script Hash
[12] Little-Endian
[13] Big-Endian



شکل ۱.۲: مقایسهٔ اندین کوچک و اندین بزرگ[۲۶]

۱.۲.۲ گره‌ها

گره‌های بیت‌کوین را می‌توان با توجه به کاری که انجام می‌دهند به انواع مختلفی دسته‌بندی کرد و طبق [۸] هر گره می‌تواند مجموعه‌ای از عملکردهای مسیریابی، پایگاه‌دادهٔ زنجیرهٔ بلوکی، استخراج و کیف پول باشد.

گره کامل در این پایان‌نامه منظور از گره کامل گره‌ای است که تمام زنجیرهٔ بلوکی را ذخیره کرده و قادر به مسیریابی و تبادل اطلاعات در شبکهٔ همتابه‌همتای بیت‌کوین باشد. گره کامل از بالاترین امنیت ممکن برخوردار بوده و قادر به اعتبارسنجی تمام تراکنش‌ها و بلوک‌ها، بدون افشای اطلاعاتش، است. پیاده‌سازی‌های مختلفی برای گره کامل بیت‌کوین وجود دارد که فرقی در عملکرد آن‌ها وجود ندارد. در این پایان‌نامه، گره کامل نرم‌افزار هستهٔ بیت‌کوین[۱۵] را اجرا می‌کند.

گره سبک پیاده‌سازی‌های نرم‌افزاری متعددی برای گره سبک یا به عبارت دیگر، کاربر SPV بیت‌کوین وجود دارد. مانند بیت‌کوین‌جی [۱]، الکترام [۵] [۱۴] و پیکوکوین [۲۴] [۱۵]. اصطلاحا، به نرم‌افزارهای گره سبک، کیف پول [۱۶] نیز گفته می‌شود.

بیت‌کوین‌جی یک کتاب‌خانهٔ کاربر سبک (SPV) به زبان جاوا [۱۷] است. این کیف پول مستقیما با استفاده از پروتکل‌های ارتباطی استاندارد تعریف شده در شبکهٔ همتابه‌همتای بیت‌کوین [۱۳، ۱۴] با گره کامل ارتباط برقرار می‌کند. بیت‌کوین‌جی از اکثر استانداردهای بیت‌کوین،از جمله فیلتر بلوم [۲۸] پشتیبانی می‌کند. در این پایان‌نامه به صورت کلی منظور از گره سبک یا کاربر SPV، بیت‌کوین‌جی است. کاربر سبک بیت‌کوین‌جی به

[14] Electrum
[15] PicoCoin
[16] Wallet
[17] Java

صورت همزمان می‌تواند به چند گره کامل متصل باشد و از طریق آن‌ها اطلاعاتش بروزرسانی گردد. کیف‌پول بیت‌کوین ولت[۱۸]، که برای سیستم‌عامل‌های اندروید و بلک‌بری توسعه پیدا کرده است، مثالی از کیف‌پول‌هایی است که از کتاب‌خانهٔ بیت‌کوین‌جی استفاده می‌کنند.

در پیاده‌سازی الکترام، کاربر سبک مستقیما طبق پروتکل ارتباطی بیت‌کوین با گره کامل مبادلهٔ اطلاعات نمی‌کند. گره کاملی که زنجیره بلوکی بیت‌کوین را ذخیره کرده‌است، لازم است برای ارائه خدمات به کاربران سبکی که از الکترام استفاده می‌کنند، سرور الکترام‌ایکس[۱۹] [۱۸] را در کنار نرم‌افزار گره کامل (هستهٔ بیت‌کوین) راه‌اندازی نماید. در این پیاده‌سازی، برخلاف بیت‌کوین‌جی، گره سبک هم‌زمان به چند سرور الکترام‌ایکس متصل نمی‌شود. بلکه به صورت تصادفی یک سرور را انتخاب می‌کند و به آن متصل می‌گردد. کاربر سبک خودش می‌تواند تعیین کند که به چه سروری متصل گردد. از این رو کاربر سبک می‌تواند اطلاعاتش را تنها از گره کاملی که به آن اعتماد دارد به روز رسانی نماید. همچنین در پروتکل ارتباطی گره سبک الکترام با سرور الکترام‌ایکس از فیلتر بلوم استفاده نمی‌شود. این ویژگی‌ها امنیت این پیاده‌سازی را با ابهام مواجه کرده است [۷].

پیکوکوین، یک کتاب‌خانه بیت‌کوین به زبان سی[۲۰] است. این کتاب‌خانه امکان استفاده به عنوان یک کیف‌پول بیت‌کوین و یک گره کامل را فراهم می‌کند. علاوه بر این، امکان ساخت نرم‌افزارهایی که مرتبط با بیت‌کوین هستند را ممکن می‌کند. این کیف پول از استاندارد ارتباطی بیت‌کوین تبعیت کرده و از فیلتر بلوم استفاده می‌کند، همچنین می‌تواند مستقیما به گره‌های کامل بیت‌کوین، بدون نیاز به راه‌اندازی سروری مجزا در سمت گره کامل، متصل شود.

در این پژوهش هرگاه از گره سبک یا کاربر SPV صحبت می‌شود منظور کیف‌پول بیت‌کوین‌جی [۱] و هر گاه از گره کامل صحبت می‌شود منظور نرم‌افزار هستهٔ بیت‌کوین[۲۱] [۱۵] است.

۲.۲.۲ شبکه همتابه‌همتای بیت‌کوین

گره‌های شبکهٔ بیت‌کوین بر اساس یک پروتکل استاندارد با یکدیگر به تبادل پیام می‌پردازند. گره‌های کامل در شبکه بیت‌کوین بعد از آن‌که بلوک‌ها و تراکنش‌های جدید را تصدیق کردند، آن‌ها را به دیگر گره‌ها ارسال می‌کنند. علاوه بر این، گره‌های سبک می‌توانند از پروتکل ارتباطی بیت‌کوین جهت ارتباط با گره‌های کامل استفاده کنند.

تمام ارتباطات همتابه‌همتا در بیت‌کوین در بستر TCP برقرار می‌شود و تمام پیام‌ها از قالب یکسانی پیروی می‌کنند. رشتهٔ آغازین پیام‌ها و مقدار پیش‌فرض شمارهٔ درگاه با توجه به اینکه پیام در شبکهٔ اصلی، تست یا در حالت تست رگرسیون استفاده می‌شود تفاوت می‌کند. جدول ۱.۲ این مقادیر را نشان می‌دهد. یک گره می‌تواند

[18] Bitcoin Wallet
[19] ElectrumX
[20] C
[21] Bitcoin-core

از شمارهٔ درگاهی متفاوت در یک شبکه استفاده نماید.

جدول ۱.۲: شبکه‌های مختلف بیت‌کوین

شبکه	درگاه پیش‌فرض	رشتهٔ آغازین	توضیحات
اصلی Mainnet	۸۳۳۳	0xf9beb4d9	شبکهٔ اصلی بیت‌کوین. در این شبکه بیت‌کوین دارای ارزش واقعی است.
تست Testnet	۱۸۳۳۳	0x0b110907	شبکهٔ آزمایشی بیت‌کوین برای توسعه دهندگان بهتر و کم هزینه تر است که از شبکهٔ آزمایشی بیت‌کوین استفاده کنند. چرا که بیت‌کوین در آن دارای ارزش واقعی نیست.
تست رگرسیون Regtest	۱۸۴۴۴	0xfabfb5da	حالت تست رگرسیون گاهی در توسعه یک کاربرد نیازی نیست که با گره‌های تصادفی در ارتباط باشیم یا بلوک‌های تصادفی تولید شده را بررسی کنیم. در این شرایط از حالت تست رگرسیون بیت‌کوین استفاده می‌کنیم. در این حالت می‌توان محیط را کنترل کرد و تعیین کرد که چه زمانی یک بلوک جدید ساخته شود.

علاوه بر این تمام پیام‌های شبکهٔ همتابه‌همتای بیت‌کوین شامل سرایندی یکسان هستند که قالب این سرایند مطابق جدول ۲.۲ است.

جدول ۲.۲: قالب سرایند تمام پیام‌ها در شبکهٔ همتابه‌همتای بیت‌کوین

نام	توضیحات
start string	بایت‌هایی که در جدول ۱.۲ توضیح داده شد که نشان دهندهٔ شبکه‌ای است که این پیام در آن تولید شده است.

command name	رشته‌ای در استاندارد اسکی[۲۲] است که مشخص می‌کند چه نوع پیامی در پایه‌بار[۲۳] قرار گرفته است. اندازهٔ این قسمت ۱۲ کاراکتر است و بایت‌های بعد از نام پیام برابر صفر (0x00) خواهند بود. به عنوان مثال برای پیام Version خواهیم داشت: version\0\0\0\0\0.
payload size	اندازه بایت‌های پیام داخل پایه‌بار را مشخص می‌کند. حداکثر تعداد بایت مجاز در پایه‌بار ۳۲ مگابایت (”MAX_SIZE“) است. پیام‌های بزرگ‌تر از این مقدار دورانداخته می‌شوند. پیام‌هایی مانند VerAck بدون پایه‌بار هستند.
checksum	چهار بایت اول حاصل SHA256(SHA256(payload)) است. اگر پایه‌بار خالی باشد، مانند پیام‌های VerAck و GetAddr، مقدار این بخش برابر 0x5df6e0e2 بوده که معادل SHA256(SHA256(رشتهٔ خالی)) است.

۱.۲.۲.۲ یافتن همتا

اولین گامی که هر گره در شبکهٔ همتابه‌همتای بیت‌کوین انجام می‌دهد، یافتن گره‌های (همتاهای) دیگر و اتصال به آن‌ها است. از آن‌جایی که یک گره در زمان راه اندازی، آدرس آی‌پی گره‌های کامل فعال را ندارد، از یک یا چند سرور DNS[۲۴] که آدرس آن‌ها در کد بیت‌کوین‌جی از پیش قرارگرفته است پرسمان انجام می‌دهد. پاسخ دریافت شده شامل آدرس یک یا چند گره کامل است که ارتباطات ورودی را قبول می‌کنند. علاوه بر این تعدادی آدرس گره کامل در هر ورژن از کدهای بیت‌کوین‌جی قرار دارد که در زمانی که آن ورژن مشخص منتشر می‌شده فعال بوده‌اند.

۲.۲.۲.۲ اتصال به همتا

بعد از آن‌که کاربر جدید آدرس آی‌پی یک یا چند گره کامل را بدست آورد، برای آن گره(ها) پیام version را ارسال می‌کند. این پیام برای ایجاد ارتباط ارسال می‌شود و شامل اطلاعاتی از گره ارسال کننده است. این

[22] ASCII
[23] Payload
[24] Domain Name System

اطلاعات در جدول ۳.۲ توضیح داده شده است. گره دریافت کننده نیز یک پیام `version` را که شامل اطلاعات خودش است، ارسال می‌کند. هر دو گره به محض دریافت پیام `version` پیام `verack` را برای گره مقابل ارسال می‌نماید. پیام `verack` بدون پایه‌بار است و به گره دریافت کننده اطلاع می‌دهد که آماده دریافت پیام‌های بعدی است.

جدول ۳.۲: قسمت‌های پیام `version` در شبکه همتابه‌همتای بیت‌کوین

نام	توضیحات
version	بالاترین نسخهٔ پروتکلی که توسط گره ارسال کننده شناخته می‌شود. در زمان نگارش این پایان‌نامه، بالاترین نسخه پروتکل بیت‌کوین ۷۰۰۱۵ است که در سال ۲۰۱۷ منتشر شده است.
services	خدماتی که گره ارسال‌کننده پشتیبانی می‌کند را مشخص می‌کند. برای گره‌های سبکی مثل بیت‌کوین‌جی، مقدار آن برابر 0x00 است.
timestamp	ساعت یونیکس[۲۵] با توجه به ساعت گره ارسال کننده در زمان ارسال پیام.
addr_recv services	سرویس‌هایی که از دید گره ارسال‌کننده، توسط گره گیرنده پشتیبانی می‌شود. فرمت نمایش آن مانند قسمت services است. اگر گره ارسال‌کننده، بیت‌کوین‌جی باشد، همیشه به صورت پیش‌فرض مقدار این قسمت را برابر 0x00 قرار می‌دهد.
addr_recv port	شماره پورت گره گیرنده از دید گره ارسال‌کننده.
addr_trans services	خدماتی که گره ارسال‌کننده پشتیبانی می‌کند را مشخص می‌کند. یکسان با قسمت services باید باشد.
addr_trans IP address	آدرس آی‌پی گره ارسال کننده.
addr_trans port	شماره پورت گره ارسال کننده.

[25] Unix time

nonce	تک‌شمار، یک عدد تصادفی است که اگر یک گره، یک پیام با تک‌شماری مشابه با تک‌شمار ارسالی دریافت کرد، ارتباط را قطع نماید. (قسمت تک‌شمار در نسخهٔ 0.1.6 بیت‌کوین اضافه شده و هدفش آن است که گره متوجه شود که به خودش متصل نشده باشد) [۲۶].
user_agent bytes	تعداد بایت‌هایی که پیام قسمت user_agent (قسمت بعدی) استفاده کرده است.
user_agent	نوع برنامه کاربر را معین می‌کند. مثلا: ۱. بیت‌کوین‌جی: /bitcoinj:1.0/MultiBit:1.0(Windows)/ ۲. هستهٔ بیت‌کوین (گره کامل): /Satoshi:0.20.0/(70015)/
start_height	ارتفاع بهترین زنجیره بلوکی گره ارسال کننده در این قسمت قرار گرفته می‌شود. در صورتی که کاربر SPV باشد، ارتفاع بهترین زنجیره سرایند بلوک‌ها قرار داده می‌شود.
relay	قرار دادن این بخش در پیام اختیاری است. این بخش در [۲۸] به همراه پیشنهاد استفاده از فیلتر بلوم در بیت‌کوین معرفی شده است. مقدار آن صحیح (0x01) یا غلط (0x00) است. در صورتی که صحیح باشد، یا از آن استفاده نشود، تغییری در پروتکل ایجاد نمی‌شود. ولی در صورتی که غلط باشد، قبل از آن‌که کاربر ارسال کننده، پیام‌های filterload و filterclear را ارسال کرده باشد، هیچ پیام inv یا tx به آن ارسال نمی‌شود. این کار باعث می‌شود که در فاصلهٔ زمانی انجام دستداد[۲۷] (ارسال پیام version) و فرستادن فیلتر بلوم، کاربر سبک تحت سیل پیام‌های گره‌کامل قرار نگیرد.

زمانی که اتصال با یک گره کامل برقرار شد، پیام getaddr برای گره کامل فرستاده می‌شود تا آدرس آی‌پی گره‌های کامل فعالی که گره دریافت‌کننده به آن‌ها متصل است در قالب پیام addr برای گره فرستنده ارسال شود. گره فرستنده همتاهای فعال خودش را نیز در قالب پیام addr برای گره کامل گیرنده ارسال می‌کند.

[26] https://github.com/bitcoin/bitcoin/commit/cc0b4c3b62367a2aebe5fc1f4d0ed4b97e9c2ac9
[27] Handshake

۳.۲.۲.۲ هم‌گام سازی گره سبک

از این قسمت به بعد تنها به بررسی فعالیت‌های گره سبک در شبکه می‌پردازیم و هم‌گام‌سازی دیگر گره‌های شبکه مورد بررسی قرار نمی‌گیرند. کاربر سبک بعد از اتصال اولیه به یک گره کامل، نیاز دارد که سرایند بلوک‌های زنجیرهٔ بلوکی را دریافت نماید به این کار هم‌گام‌سازی[۲۸] گفته می‌شود. همان‌طور که گفته شد کاربر سبک به جای ذخیره‌سازی و تصدیق تمام زنجیرهٔ بلوکی، تنها سرایند آن را ذخیره می‌کند. حجم سرایند یک بلوک ۸۰ بایت است. در گره‌های کامل، که می‌خواهند تمام زنجیره بلوکی را دریافت نمایند، این فرایند به دو صورت «ابتدا-بلوک[۲۹]» یا «ابتدا-سرایند[۳۰]» قابل انجام است که در این‌جا به توضیح آن‌ها پرداخته نمی‌شود. گره سبک در گام اول هم‌گام‌سازی لازم است که بهترین سراید زنجیرهٔ بلوکی[۳۱] را دانلود کند. سرایند زنجیرهٔ بلوکی، زنجیره‌ای از سرایند بلوک‌ها است که هر کدام از سرایندها به سرایند بلوک قبل خود اشاره می‌کند. بهترین سرایند زنجیرهٔ بلوکی، زنجیره‌ای است که دشوارترین بازآفرینی را داشته باشد.

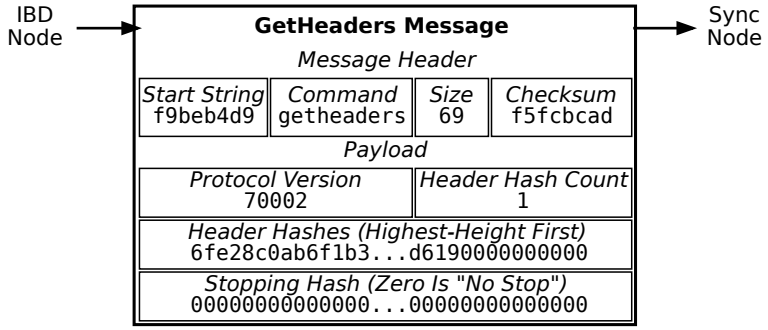
گره سبک برای دریافت سرایند زنجیرهٔ بلوکی، پیام `getheaders` را برای گره کاملی (گره هم‌گام‌ساز) که می‌خواهد با آن همگام شود ارسال می‌کند. جدول ۴.۲ بخش‌های مختلف این پیام را توضیح می‌دهد و شکل ۲.۲ مثالی از یک پیام `getheaders` است که گره سبک برای اولین‌بار برای گره هم‌گام‌ساز ارسال می‌کند.

جدول ۴.۲: قسمت‌های پیام `getheaders` در شبکه همتابه‌همتای بیت‌کوین

نام	توضیحات
version	شمارهٔ نسخهٔ پروتکل. شبیه آنچه در پیام `version` ارسال شد.
hash count	تعداد چکیده‌هایی که در بخش بعدی پیام قرار می‌گیرند، در این قسمت تعیین می‌شوند. محدودیتی در تعداد چکیده‌های ارسالی نیست. اما اندازه کل پیام باید کمتر از "MAX_SIZE" (۳۲ مگابایت) باشد.

[28] Synchronization
[29] Blocks-First
[30] Headers-First
[31] Best header chain

block header hashes	چکیدهٔ یک یا چند سرایند بلوکی که گره ارسال کننده آن‌ها را در حافظهٔ خود دارد. ترتیب چکیده‌ها از بالاترین ارتفاع بلوک (جدیدترین) به پایین‌ترین ارتفاع است. به این ترتیب به گره دریافت‌کنندهٔ پیام این امکان داده می‌شود که جدیدترین چکیدهٔ سرایندی که با هم مشترک هستند را پیدا کند. اگر گره سبکش تازه راه‌اندازی شده باشد در این قسمت، چکیدهٔ بلوک جنسیس (6fe2...0000) را که در نرم‌افزارش از ابتدا وجود داشته است، قرار می‌دهد.
stop hash	این قسمت چکیدهٔ آخرین بلوکی است که گره ارسال‌کننده می‌خواهد دریافت کند. با صفر قراردادن آن، طولانی‌ترین پاسخ ممکن از گره کامل تقاضا می‌شود. حداکثر تعداد سرایندی که گره کامل دریافت کنندهٔ این پیام پاسخ می‌دهد، ۲۰۰۰ سرایند است. برای دریافت بیشتر از این مقدار، این پیام در چند نوبت ارسال می‌شود



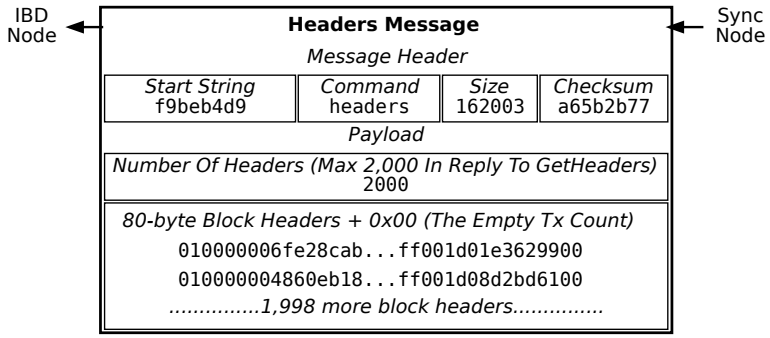
شکل ۲.۲: مثالی از پیام getheaders در همگام‌سازی اولیهٔ یک گره جدید

گره هم‌گام‌ساز در پاسخ به پیام getheaders در شکل ۲.۲ دنبال بلوکی با چکیده مشخص شده می‌گردد و می‌یابد که این بلوک برابر بلوک شمارهٔ صفر (بلوک جنسیس) است. به این ترتیب سرایند بلوک را که از بلوک شمارهٔ یک آغاز می‌شوند در قالب پیام headers برای گره درخواست دهنده ارسال می‌کند. قالب این پیام در جدول ۵.۲ مشخص شده است. شکل ۳.۲ مثالی از پیام بازگردانده شده توسط گره هم‌گام‌ساز است.

وقتی گره سبک پاسخ شکل ۳.۲ را دریافت کرد، فورا صحت آن را بررسی کرده و مجددا پیام getheaders جدیدی برای گره همگام‌ساز برای گرفته باقیمانده سرایندها ارسال می‌کند. این فرایند تا گرفتن کامل سرایندها

جدول ۵.۲: قسمت‌های پیام `headers` در شبکه همتابه‌همتای بیت‌کوین

نام	توضیحات
count	تعداد سرایندهای بلوک قرار گرفته در بخش بعدی این پیام. (حداکثر ۲۰۰۰)
headers	سرایند بلوک‌ها در این قسمت قرار می‌گیرند.



شکل ۳.۲: مثالی از پیام `headers` در همگام‌سازی اولیهٔ یک گره جدید

ادامه پیدا می‌کند. در زمان نوشتن این پایان‌نامه، حجم تمام سرایندهای زنجیرهٔ بلوکی ۵۰ مگابایت است. پس از اتمام دانلود سرایندهای زنجیرهٔ بلوکی، گره سبک آخرین پیام `getheaders` را برای چند همتای دیگر ارسال کرده و پاسخ آن‌ها را با پاسخ گره هم‌گام‌ساز ابتدایی مقایسه می‌کند. به این ترتیب مطمئن می‌شود که بهترین سرایند زنجیرهٔ بلوکی را دریافت کرده است.

۴.۲.۲.۲ انتشار

زمانی که گره کامل یک بلوک جدید را دریافت می‌کند، پیام `inv` را برای همهٔ همتاهایش (چه گره کامل چه گره سبک) ارسال می‌کند. پیام ارسال شده دارای یک مدخل فهرست[۳۲] مربوط به بلوک جدید است. یک مدخل فهرست، شامل یک علامت نوع داده و یک چکیده داده به عنوان مشخص‌کنندهٔ آن است. داده می‌تواند انواع مختلفی داشته باشد، به عنوان نمونه، علامت تراکنش "MSG_TX" و علامت بلوک "MSG_BLOCK" است. به صورت کلی مدخل فهرست به وجود تراکنش‌ها یا بلوک‌هایی برای دانلود اشاره می‌کند. جدول ۶.۲ قسمت‌های مختلف پیام `inv` را شرح می‌دهد.

[32] Inventory

جدول ۶.۲: قسمت‌های پیام inv در شبکه همتابه‌همتای بیت‌کوین

نام	توضیحات
compactSize uint	تعداد مدخل‌های فهرست.
inventory	یک یا چند مدخل فهرست. حداکثر تعداد آن می‌تواند ۵۰۰۰۰ باشد. به عنوان مثال محتوای این قسمت از پیام برای اطلاع‌رسانی بلوک ارتفاع ۶۴۵۷۴۷[۳۳] به گره‌های همتا به این صورت است: علامت نوع داده: MSG_BLOCK مشخص‌کنندهٔ داده (چکیده): 0x333ab9f10d...0000000000

گره سبک بعد از دریافت این پیام، یک پیام getdata برای گره کامل می‌فرستد. در این پیام درخواست می‌کند که با توجه به فیلتر بلومی که پیش‌تر در اختیار گره کامل گذاشته بوده، تراکنش‌هایی از بلوک جدید را، که در آن فیلتر صدق می‌کنند برای اون بفرستد. ساختار پیام getdata شبیه inv است. با این تفاوت که علامت نوع داده، اطلاعاتی است که گره ارسال کننده این پیام از گره دریافت‌کننده درخواست می‌کند. در این کاربرد، گره سبک علامت "MSG_FILTERED_BLOCK" را در کنار چکیدهٔ بلوک مورد نظر در پیام قرار می‌دهد و برای گره کامل ارسال می‌کند. به این ترتیب گره کامل تراکنش‌هایی که حداقل یک آدرس آن‌ها در فیلتر بلوم صدق می‌کنند را در کنار اثبات مرکل آن‌ها برای گره سبک ارسال می‌کند. پاسخ در قالب یک پیام merkleblock که شامل اثبات مرکل وجود تراکنش‌های مرتبط در بلوک است و تعداد صفر یا چند پیام tx را که خود تراکنش‌ها هستند خواهد بود.

به خاطر ماهیت فیلتر بلوم، پاسخ گره کامل شامل تراکنش‌هایی می‌شود که مورد توجه گره سبک نیستند. این اتفاق منجر به گمراه شدن گره کامل در شناخت تراکنش‌های مرتبط با گره سبک می‌شود. هدف از این کار حفظ گم‌نامی کاربر سبک و فاش نشدن آدرس وی نزد گره کامل است. در قسمت ۳.۲ علاوه بر توضیح فیلتر بلوم، نحوه استفاده از آن در شبکهٔ همتابه‌همتا، مثل ارسال آن برای گره کامل از طریق ارسال پیام filterload و نحوهٔ تولید پیام merkleblock توسط گره کامل و ساختار آن توضیح داده می‌شود. همچنین، در این قسمت در مورد آسیب‌پذیری‌های فیلتر بلوم و ناتوانی آن در حفظ حریم خصوصی کاربران بحث خواهد شد.

[33] https://blockchair.com/bitcoin/block/645747

۳.۲ فیلتر بلوم

فیلتر بلوم را نخستین بار برتون بلوم در [۱۷] معرفی کرد. هدف این فیلتر امتحان سریع وجود یک عضو در یک مجموعه است. فیلتر بلوم کاربرد گسترده‌ای در پایگاه‌های داده، شبکه و حتی موتورهای جست‌وجو دارد. فیلتر بلوم آرایه‌ای از n بیت $b[i]$ است که i از ۰ تا $n-۱$ است. به صورت پیش‌فرض تمام بیت‌ها مقدار صفر دارند. اگر بخواهیم عضو x را (مثلا یک رشته) درون مجموعه آن قرار دهیم، آن عضو را در ورودی k تابع چکیده‌ساز مستقل $H_۱(.), H_۲(.), ..., H_k(.)$ قرار می‌دهیم. خروجی هر تابع چکیده ساز یک عدد صحیح بین ۰ تا $n-۱$ است. از این رو هر تابع چکیده ساز، یک عنصر ورودی را به یکی از n بیت فیلتر بلوم نگاشت می‌کند. برای قرار دادن آن رشته در مجموعه مربوط به فیلتر بلوم، بیت متناظر عدد حاصل را برابر با یک قرار می‌دهیم:

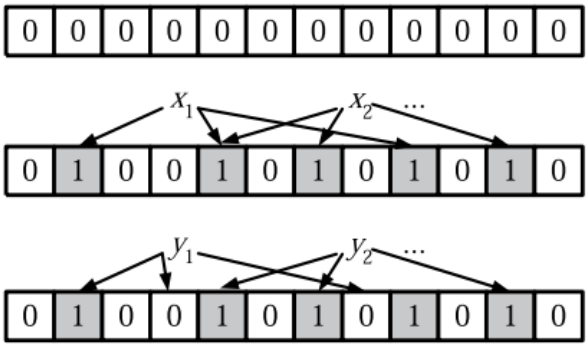
$$\forall j \in \{۱..k\}, b[H_j(x)] \leftarrow ۱.$$

به همین ترتیب اگر بخواهیم بررسی کنیم که یک رشته در مجموعه قرار دارد، چکیده آن رشته را توسط همان k تابع چکیده‌ساز حساب نموده و بررسی می‌کنیم که آیا مقدار ذخیره شده در تمام k جایگاه بدست آمده برابر یک است یا خیر. اگر برابر با یک باشد، آن رشته را عضو احتمالی آن مجموعه در نظر می‌گیریم. به آن عضو احتمالی گفته می‌شود چرا که ممکن است عناصری عضو مجموعه نباشند و به جایگاه‌هایی که مقدار بیت آن‌ها برابر با یک است نگاشت شوند. به این ترتیب امکان بروز خطای نوع دو وجود دارد. مجموعهٔ تمامی عناصر با $\mathcal{U}$ اعضایی که درون فیلتر بلوم قرار گرفته‌اند با $\mathcal{S}$ و مجموعهٔ عناصری که در نتیجه خطای نوع دو عضو فیلتر بلوم در نظر گرفته می‌شوند با $\mathcal{V}$ نمایش داده می‌شوند. به صورت کلی می‌توان گفت هرگاه لیست یا مجموعه‌ای مورد استفاده قرار گرفت، هزینه فضای ذخیره‌سازی و دسترسی به اعضای مجموعه قابل توجه بود و خطای نوع دو خسارت و هزینه چندانی به سامانه تحمیل نکند، استفاده از فیلتر بلوم مفید خواهد بود. فیلتر بلوم امکان انجام مصالحه بین فضای استفاده شده، زمان پاسخ‌گویی و احتمال خطای قابل قبول را فراهم می‌کند[۱۷]. با توجه به ساختار فیلتر بلوم روشن است که امکان بروز خطای نوع یک، یا به عبارت دیگر امکان آنکه عضو مجموعه را غیر عضو تشخیص دهد، وجود ندارد.

در فیلتر بلوم برای تنظیم نرخ قابل قبول خطای نوع دوم (P_t)، با توجه به حداکثر تعداد عناصری که در فیلتر قرار خواهند گرفت(M)، اندازه فیلتر(n) و تعداد توابع چکیده‌ساز(k) تعیین می‌شوند. جدول ۷.۲ نشانه‌گذاری‌های مربوط به فیلتر بلوم را نشان می‌دهد.

برای فیلتر بلوم $B(M, P_t)$ اندازه فیلتر به صورت زیر محاسبه می‌شود[۲۵]:

$$n = -\frac{M \ln(P_t)}{(\ln(۲))^۲} \qquad (۱.۲)$$



شکل ۲.۴: فیلتر مجموعه بدون عضو متشکل از یک آرایه از بیت‌ها با مقدار صفر است. k دفعه چکیده هر عضو مجموعه x_i محاسبه می‌شود که حاصل هر چکیده موقعیت یک بیت است. که مقدار این بیت‌ها ۱ می‌شود. حال برای آنکه بررسی کنیم که y_i درون این مجموعه است به تعداد k بار از آن چکیده می‌گیریم و بیت‌های مرتبط را بررسی می‌کنیم. عنصر y_1 نمی‌تواند عضو مجموعه باشد چرا که یکی از بیت‌هایی که به آن اشاره می‌کند صفر است. عنصر y_2 یا عضو مجموعه است یا اینکه به خاطر خطای نوع دو فیلتر، عضو مجموعه تشخیص داده شده است.[۱۹]

و تعداد توابع چکیده‌ساز به صورت زیر محاسبه می‌گردد[۲۵]:

$$k = \ln(2)\frac{n}{M} \qquad (۲.۲)$$

احتمال خطای نوع دو فیلتر بلوم $B(M, P_t)$ ، در صورتی که m عنصر در آن قرار دهیم که $m < M$ ، کمتر از مقدار هدف آن (P_t) می‌شود. اگر بیشتر از ظرفیت یک فیلتر در آن عنصر قرار داده شود، نرخ خطای نوع دو آن از از احتمال هدف بیشتر می‌گردد.

احتمال خطای نوع دو در یک فیلتر با توجه به عناصری که در آن قرار داده شده است، (m) با دقت‌های متفاوتی محاسبه شده است. مقاله [۱۷]، که فیلتر بلوم را معرفی کرده است، احتمال خطای نوع دو را برای فیلتر بلوم محاسبه کرده است. این مقاله با فرض این‌که بعد از قرار دادن m عضو در فیلتر بلوم نسبت بیت‌هایی که مقدار آن‌ها صفر مانده است به کل بیت‌ها برابر $(1 - k/n)^m$ باشد، احتمال خطای نوع دو را به صورت زیر محاسبه کرده است:

$$P_f(m) = \left(1 - \left(1 - \frac{k}{n}\right)^m\right)^k \qquad (۳.۲)$$

در مقاله [۳۵] محاسبه دقیق‌تری از احتمال خطای نوع دو به دست آمده است. در این مقاله، احتمال آن که

جدول ۷.۲: قرارداد نشانه‌گذاری برای فیلتر بلوم

نشانه‌گذاری	معنا
$\mathcal{S}$	مجموعه عناصری که عضو فیلتر شده‌اند
M	حداکثر تعداد عناصر فیلتر
$m = \|\mathcal{S}\|$	تعداد عناصر قرار داده شده در فیلتر
n	اندازه (تعداد بیت‌های) فیلتر
k	تعداد توابع چکیده‌ساز
$\mathcal{U}$	مجموعهٔ تمام عناصر، $\|\mathcal{U}\| = N_u$
$\mathcal{V}$	مجموعهٔ پنهان‌سازی (عناصر خطای نوع دو)، $\|\mathcal{V}\| = N_v$
P_t	نرخ (احتمال) خطای نوع دوی هدف (ایده‌آل)
P_f	نرخ (احتمال) خطای نوع دوی واقعی
$B(M, P_t)$	فیلتر بلوم با حداکثر ظرفیت M و نرخ خطای نوع دو هدف P_t

یک بیت دلخواه بعد از مقدار دهی k بیت مقدارش عوض نشود، $(۱ - ۱/n)^k$ محاسبه شده است. پس به این ترتیب بعد از قرار دادن m عضو در فیلتر، احتمال آن‌که مقدار یک بیت تغییر نکند، برابر $(۱ - ۱/n)^{km}$ خواهد بود. در نتیجه احتمال آن‌که مقدار یک بیت تغییر کند به صورت $p_{set} = ۱ - (۱ - ۱/n)^{km}$ محاسبه می‌شود. پس احتمال خطای نوع دو برابر است با احتمال آن‌که تمام بیت‌های انتخابی حاصل از kتا چکیدهٔ عنصری که عضو فیلتر بلوم مورد نظر نیست، از قبل مقدار یک گرفته باشند. به این ترتیب احتمال خطای نوع دو طبق اثبات [۳۵]، به صورت زیر محاسبه می‌شود.

$$P_f(m) = \left(۱ - \left(۱ - \frac{۱}{n}\right)^{km}\right)^k \approx \left(۱ - e^{-\frac{mk}{n}}\right)^k \qquad (۴.۲)$$

برای $n \gg k$، مقادیر معادله‌های (۴.۲) و (۳.۲) به هم نزدیک خواهند بود. مقاله [۲۱] به فرمولی با دقت بیشتر از دو مقاله قبلی برای محاسبه احتمال خطای نوع دوی فیلتر بلوم دست پیدا کرده است که به شرح زیر است:

$$P_f(m) = \frac{n!}{n^{k(m+1)}} \sum_{i=1}^{n} \sum_{j=1}^{i} (-1)^{i-j} \frac{j^{km} i^k}{(n-i)!j!(i-j)!} \qquad (۵.۲)$$

اثبات فرمول (۵.۲) خارج از بحث این پایان‌نامه است. اگر تعداد پیام‌های قرارداده شده در فیلتر بلوم برابر با M باشد، در آن صورت $P_f(M) = P_t$.

کاربردهای متعددی برای فیلتر بلوم وجود دارد و در ادامه یکی از آن‌ها را مرور خواهیم کرد. در وبسایت‌هایی که خدمات کوتاه‌کردن لینک را ارائه می‌کنند (مانند [۱۶])، معمولا لیست سیاهی از آدرس‌های غیر امن نگهداری می‌شود و به کاربر استفاده کننده از لینک‌های کوتاه‌شده اطمینان می‌دهد که آدرسی که به آن هدایت خواهد شد یک آدرس امن است (در لیست سیاه آدرس‌های ناامن قرار ندارد). جست‌وجو کردن لیست سیاه آدرس‌های ناامن برای هر درخواست امری زمان‌بر است. از این رو، مجموعهٔ تمام آدرس‌های ناامن در یک فیلتر بلوم نگهداری می‌شود. اگر پاسخ فیلتر بلوم برای یک آدرس درخواست داده شده منفی باشد (عضو مجموعه نباشد) می‌توانیم صددرصد مطمئن باشیم که آدرس درخواست داده شده یک آدرس امن است و اگر پاسخ مثبت باشد، جهت جبران خطای نوع دو، پایگاه داده لیست سیاه آدرس‌های ناامن را جست‌وجو می‌کند[۹].

کاربرد فیلتر بلوم مورد نظر در این پایان‌نامه، استفاده از آن در گره‌های سبک برای حفظ گم‌نامی این گره‌ها است [۲۸]. در بخش ۱.۳.۲ به نحوهٔ استفاده از این فیلتر در ارتباط بین گره‌های سبک و گره‌های کامل پرداخته می‌شود و در بخش ۵.۱.۳.۲ به ضعف‌ها و آسیب‌پذیری‌های استفاده از این فیلتر در شبکه بیت‌کوین خواهیم پرداخت.

۱.۳.۲ فیلتر بلوم در شبکهٔ همتابه‌همتای بیت‌کوین

امکان استفاده از فیلتر بلوم در ارتباط بین گره سبک و گره کامل به دنبال معرفی آن در طرح پیشنهادی بهبود بیت‌کوین شمارهٔ ۳۷ (BIP37) [۲۸] در سال ۲۰۱۳ فراهم شد. گره‌های سبک برای حفظ گم‌نامی خود، به جای آن‌که آدرس‌های مربوط به خودشان را صورت فاش در اختیار یک گره کامل قراردهند، آدرس‌های خود و دیگر اطلاعات مورد نیازشان را در یک فیلتر بلوم با نرخ خطای نوع دوی معین قرار می‌دهند. گره کامل با تطابق داده‌های داخل تراکنش‌ها با فیلتر بلوم بررسی می‌کند آن داده درون فیلتر بلوم صدق می‌کند یا خیر. اگر یک داده درون فیلتر بلوم صدق کرد، گره کامل آن داده را برای گره سبک ارسال می‌کند. به صورت کلی اطلاعاتی که می‌توانند درون فیلتر بلوم قرار بگیرند و توسط گره کامل با فیلتر بلوم تطابق داده شوند به صورت زیر است:

۱. چکیدهٔ تراکنش (TXID)

۲. برای هر خروجی تراکنش، داده‌های نبشته خروجی بررسی می‌شوند. این داده‌ها نظیر pubKeyHash یا pubKey هستند. زمانی که یکی از این داده‌ها با فیلتر بلوم تطابق پیدا کنند، گره کامل، در صورت درخواست کاربر، می‌تواند دادهٔ COutPoint را به فیلتر اضافه نماید. به این‌ترتیب فیلتر را به‌روزرسانی کند.

۳. برای هر ورودی، COutPoint بررسی می‌شود.

۴. برای هر ورودی، داده‌های نبشتهٔ ورودی بررسی می‌شوند. این داده‌ها نظیر pubKey یا sig هستند.

اگر گره‌کامل بتواند در یک تراکنش مطابقتی بین هر کدام از موارد بالا و فیلتر بلوم پیدا کند آن تراکنش را برای گره سبک ارسال می‌کند. در غیر این صورت چیزی برای گره سبک ارسال نمی‌شود. در ادامه، به بررسی پروتکل ارتباطی گره‌های سبک با گره‌های کامل و گرفتن اثبات مرکل برای تراکنش‌های مورد نظر کاربر سبک با بهره‌گیری از فیتلر بلوم پرداخته خواهد شد.

در قسمت ۲.۲.۲، نحوهٔ اتصال یک گره سبک به گره‌های فعال شبکهٔ همتابه‌همتای بیت‌کوین توضیح داده شد. در این قسمت نحوهٔ ساخت و فرستادن فیلتر بلوم به یک گره کامل و دریافت تراکنش‌های موجود در یک بلوک که در آن فیلتر صدق می‌کنند پرداخته خواهد شد.

۱.۱.۳.۲ ساخت فیلتر بلوم

همان‌طور که در بخش ۳.۲ توضیح داده‌شد، فیلتر بلوم دو پارامتر تعیین‌کننده دارد: اندازهٔ (تعداد بیت‌های) فیلتر (n) و تعداد توابع چکیده‌ساز فیلتر (k). قطعه کد زیر از فایل BloomFilter.java از منبع کد بیت‌کوین‌جی [۲] نحوهٔ اختصاص‌دهی مقادیر n و k را که به ترتیب با متغیرهای `size` و `hashFuncs` مشخص شده‌اند و طبق فرمول‌های (۱.۲) و (۲.۲) محاسبه شده‌اند نشان می‌دهد:

```
int size = (int)(-1/(pow(log(2),2))*elements*log(falsePositiveRate));
size = max(1,min(size,(int)MAX_FILTER_SIZE*8)/8);
hashFuncs = (int)(data.length*8/(double)elements*log(2));
hashFuncs = max(1,min(hashFuncs,MAX_HASH_FUNCS));
```

اندازه فیلتر بلوم حداکثر می‌تواند ۳۶۰۰۰ بایت (“MAX_FILTER_SIZE”) و تعداد توابع چکیده‌ساز حداکثر می‌تواند ۵۰ (“MAX_HASH_FUNCS”) باشد. در فیلتر بلوم بیت‌کوین از نسخهٔ ۳ تابع چکیده‌ساز

۳۲ بیتی مورمور [۳۴] استفاده می‌شود[۲۸]. برای دستیابی به k تابع چکیده‌ساز متفاوت، از مقدار بذر[۳۵] متفاوتی برای هرکدام از توابع استفاده می‌شود. بذر هر تابع چکیده‌ساز مطابق فرمول (۶.۲) محاسبه می‌شود.

$$SEED_{(nHashNum)} = nHashNum \times 0xfba4c795 + nTweak \qquad (۶.۲)$$

که در آن nHashNum، شمارهٔ ترتیب تابع چکیده‌ساز است. مقدار آن برای اولین تابع چکیده‌ساز صفر و برای آخرین تابع $k - ۱$ است. عدد 0xfba4c795 یک عدد ثابت بهینه‌شده است تا اختلاف مقدار بذر توابع مختلف را زیاد نماید. nTweak به ازای هر فیلتر بلوم مقدار متفاوتی دارد که توسط کاربر سبک انتخاب می‌شود.

سپس برای تعیین بیت‌هایی که باید در فیلتر بلوم مقدار آن‌ها به یک تغییر کند، چکیدهٔ هر کدام از آدرس‌های مورد نظر را توسط هر k تابع چکیده‌ساز حساب کرده و باقیمانده حاصل را به اندازهٔ فیلتر بلوم می‌سنجیم. حاصل شماره بیتی است که باید اندازهٔ آن به یک تغییر کند. دستور محاسبهٔ چکیده در فایل bloom.cpp هستهٔ بیت‌کوین در کد منبع آن [۱۵]به صورت زیر است:

```
MurmurHash3(nHashNum*0xFBA4C795+nTweak, vDataToHash)%(vData.size()*8)
```

۲.۱.۳.۲ فرستادن فیلتر بلوم برای گره کامل

بعد از آن‌که گره سبک باتوجه به مقادیر مورد نظرش فیلتر بلوم را تولید کرد، لازم است که آن را از طریق پیام `filterload` برای گره کامل ارسال نماید. به این ترتیب کاربر سبک می‌تواند تراکنش‌هایی که مربوط به کیف پولش هستند به علاوهٔ تعدادی تراکنش حاصل از خطای نوع دو دریافت نماید تا مانع اطلاع گره کامل از آدرس‌های مربوط به گره سبک شود. جدول ۸.۲ قسمت‌های مختلف پیام `filterload` را توضیح می‌دهد.

جدول ۸.۲: قسمت‌های پیام `filterload` در شبکه همتابه‌همتای بیت‌کوین

نام	توضیحات
nFilterBytes	تعداد بایت‌های فیلتری که در قسمت بعدی قرار گرفته است.
filter	آرایه از بیت‌ها که همان فیلتر بلوم است. حداکثر اندازهٔ آن می‌تواند ۳۶۰۰۰ باشد.

[34] MurmurHash3 (x86_32)
[35] Seed

nHashFuncs	تعداد توابع چکیده‌ساز به‌کار گرفته‌شده در فیلتر بلوم. حداکثر تعداد آن می‌تواند ۵۰ باشد.
nTweak	یک مقدار دلخواه برای اضافه کردن بذر به توابع چکیده‌ساز استفاده شده در فیلتر بلوم. عملا گره دریافت‌کنندهٔ این پیام می‌تواند با استفاده از این مقدار تمام توابع چکیده‌ساز مورد نیاز را ایجاد نماید.
nFlags	این بخش می‌تواند یکی از مقادیر زیر را داشته باشد. هر کدام از این مقادیر به گره کامل می‌گوید که در آینده چه تغییراتی در فیلتر بلوم ارسال شده ایجاد نماید. ۱- صفر (BLOOM_UPDATE_NONE): گره کامل نباید تغییری در فیلتر بلومی که در اختیار دارد ایجاد نماید. ۲- یک (BLOOM_UPDATE_ALL): اگر فیلتر با هر یک از داده‌های نبشتهٔ خروجی تطابق پیدا کند، گره کامل COutPoint را به فیلتر اضافه نماید و فیلتر را به‌روزرسانی کند. ۳- دو (BLOOM_UPDATE_P2PUBKEY_ONLY): اگر فیلتر با هر یک از داده‌های نبشتهٔ خروجی تطابق پیدا کند، تنها اگر نبشته از نوع P2PK یا P2SH باشد، گره کامل COutPoint را به فیلتر اضافه نماید و فیلتر را به‌روزرسانی کند. از آن‌جایی که گره کامل با توجه به تطبیق‌های اشتباهی که به خاطر خطای نوع دو انجام شده است نیز فیلتر بلوم را به روزرسانی می‌کند، عناصر موجود در فیلتر بلوم بسیار سریع زیاد خواهد شد و به خاطر بالا رفتن نرخ خطای نوع دو خیلی زود فیلتر بلااستفاده خواهد شد.

گره سبک می‌تواند با فرستادن پیام `filterclear` به گره دریافت‌کننده بگوید که فیلتر بلومی که قبل‌تر برایش ارسال شده است را پاک کند. پیام `filterclear` هیچ پایه‌باری ندارد و برای آن‌که گره سبک یک فیلتر بلوم جدید ارسال نماید، نیاز نیست که فیلتر بلوم قبلی را حذف کند. گره سبک همچنین می‌تواند با فرستادن پیام `filteradd` به گره دریافت کننده، داده‌ای را به فیلتر بلومی که پیش‌تر برایش ارسال کرده بوده اضافه نماید. بدون آن که نیازی باشد که یک فیلتر بلوم جدید را برای او ارسال کند. به این ترتیب، از آنجایی که عنصر جدید مستقیما به گره دریافت کننده ارسال می‌شود، حریم خصوصی کاربر حفظ نمی‌شود. از این رو کاربر برای حفظ نسبی حریم خصوصیش باید مجددا فیلتر بلوم جدید را محاسبه کند و به وسیلهٔ پیام `filterload` برای گره کامل ارسال نماید.

به این ترتیب گره سبک سعی می‌کند که به جای ارسال مستقیم آدرس‌هایش به یک گره کامل، آدرس‌هایش را درون یک فیلتر بلوم قرار دهد و این فیلتر با پیام `filterload` برای یک گره کامل ارسال نماید. گره کامل عناصر

متفاوتی از یک تراکنش را در فیلتر بلوم ارسال شده ارزیابی می‌کند و همچنین می‌تواند در صورت اجازهٔ گره سبک آن را به روزرسانی نماید.

۳.۱.۳.۲ اطلاعات دریافتی به ازای هر تراکنش منطبق شده

همان‌طور که در بخش ۲.۲.۲ شرح داده شد، گره سبک بعد از آن‌که برای بار اول فیلتر بلوم را با گره(های) هم‌گام‌ساز به اشتراک گذاشت، به ازای هر بلوک جدیدی که از شبکهٔ همتابه‌همتا به گره(های) هم‌گام‌ساز می‌رسید، از طرف آن(ها) به گره سبک یک پیام `inv` ارسال می‌شد. گره سبک بعد از دریافت این پیام، یک پیام `getdata` برای آن‌ها ارسال می‌کرد و به این طریق از آن‌ها می‌خواست که داده‌های تراکنش‌ها را با فیلتر بلوم ارسالی ارزیابی کنند و اگر داده‌ای از یک تراکنش با آن فیلتر مطابق شد، آن تراکنش را به علاوهٔ اثبات مرکل برای آن گره سبک ارسال کنند.

گره‌های کامل تراکنش‌های منطبق شده را در قالب پیام `tx`، که پایه‌بار آن یک تراکنش خام است، برای گره سبک ارسال می‌کنند. علاوه بر آن پیام `merkleblock` که شامل TXIDهای تراکنش‌ها و هر بخشی از درخت مرکل که نیاز است که این تراکنش‌ها را به ریشهٔ مرکل موجود در سرایند بلوک مرتبط کند، است. بخش‌های پیام `merkleblock` در جدول ۹.۲ شرح داده شده‌اند.

جدول ۹.۲: قسمت‌های پیام `merkleblock` در شبکه همتابه‌همتای بیت‌کوین

نام	توضیحات
block header	سرایند بلوکی که تراکنش‌ها از آن انتخاب شده‌اند و اثبات مرکل مرتبط در این پیام قرار داده شده است.
transaction count	تعداد کل تراکنش‌های موجود در بلوک انتخابی.
hash count	تعداد چکیده‌های موجود در قسمت بعدی.
hashes	شامل چکیدهٔ تراکنش‌ها (TXID) و گره‌های درخت مرکل است.
flag byte count	تعداد بایت‌های پرچمی که در قسمت بعدی آمده است.

flags	مجموعه‌ای از بیت‌ها که که هرکدام از چکیده‌ها را به یک گره در درخت مرکل اختصاص می‌دهد. نحوهٔ عملکرد آن در مثالی در متن آورده شده است. تعداد بیت‌های آن باید به هشت (اندازهٔ یک بایت) بخش‌پذیر باشد. برای این منظور می‌شود از لایی‌گذاری صفر استفاده کرد.

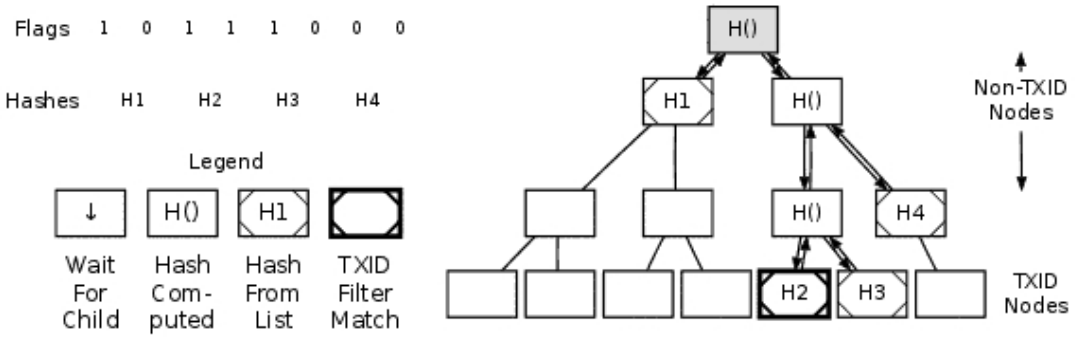
مثال در این مثال فرض کنید که پیام merkleblock بدون سرایند مطابق زیر باشد [۱۴]:

```
01000000 ........................... Block version: 1
82bb869cf3a793432a66e826e05a6fc3
7469f8efb7421dc88067010000000000 ... Hash of previous block's header
7f16c5962e8bd963659c793ce370d95f
093bc7e367117b3c30c1f8fdd0d97287 ... Merkle root
76381b4d ........................... Time: 1293629558
4c86041b ........................... nBits: 0x04864c * 256**(0x1b-3)
554b8529 ........................... Nonce

07000000 ........................... Transaction count: 7
04 ................................. Hash count: 4

3612262624047ee87660be1a707519a4
43b1c1ce3d248cbfc6c15870f6c5daa2 ... Hash #1
019f5b01d4195ecbc9398fbf3c3b1fa9
bb3183301d7a1fb3bd174fcfa40a2b65 ... Hash #2
41ed70551dd7e841883ab8f0b16bf041
76b7d1480e4f0af9f3d4c3595768d068 ... Hash #3
20d2a7bc994987302e5b1ac80fc425fe
25f8b63169ea78e68fbaaefa59379bbf ... Hash #4

01 ................................. Flag bytes: 1
```



شکل ۵.۲: شکل مثال تحلیل پیام merkleblock در سمت کاربر سبک.[۱۴]

```
1d ................................ Flags: 1 0 1 1 1 0 0 0
```

با استفاده از تعداد تراکنش‌ها گره سبک می‌تواند یک درخت مرکل خالی را ایجاد نماید. در این مثال که تعداد تراکنش‌ها هفت است، درخت مرکل سه لایه خواهد داشت. در اثبات مرکل اگر گره کامل چکیدهٔ یک گره مرکل را در اختیار کاربر سبک قرار دهد، کاربر سبک می داند که دیگر از گره‌ها یا TXIDهای زیردستی آن مقداری در اختیار وی قرار نداده است. ترتیب چکیده‌ها و بیت‌های flags یکی است. شروع حرکت از ریشهٔ درخت مرکل است و برای حرکت به سمت گره‌های بچه، ابتدا گره چپ را انتخاب می‌کنیم. اطلاعات زیر را می‌توانیم با توجه به flags راجع به جایگاه مقادیر چکیده در درخت مرکل بدست بیاوریم:

۱. **مقدار صفر:** به این معنی است که اولین مقدار چکیدهٔ استفاده نشده را به عنوان مقدار این گره استفاده کن و گره‌های پایین دستی این گره را رها کن. به اولین گرهی برو که مقدار آن محاسبه نشده است.

۲. **مقدار یک:** مقدار چکیدهٔ این گره نیاز به محاسبه شدن دارد. برای این منظور گره بعدی را گره بچهٔ سمت چپی قرار بده اما اگر مقدار چکیده در گره بچهٔ سمت چپ محاسبه شده است به گره بچهٔ سمت راست برو.

با توجه به توضیحات بالا، گام‌های محاسبهٔ اثبات مرکل با توجه به پیام merkleblock دریافت شده در مثال به صورت زیر خواهد بود:

۱. باتوجه به اینکه تعداد تراکنش‌ها هفت عدد است، یک درخت مرکل سه لایه، مطابق شکل ۵.۲ ایجاد می‌کنیم که در ابتدا تمام گره‌های آن خالی باشد.

۲. از ریشهٔ مرکل شروع می‌کنیم. مقدار اولین بیت flags یک است. به این ترتیب مقدار گره ریشه بعدا و با

توجه به مقدار بچه‌هایش مشخص می‌شود. گره بعدی مورد بررسی را بچهٔ سمت چپ ریشهٔ مرکل قرار می‌دهیم. برای شماره گذاری، گره ریشه را گره شمارهٔ یک می‌نامیم.

۳. در این مرحله، گره مورد بررسی گره بچهٔ سمت چپ ریشهٔ مرکل است. با توجه به اینکه بیت بعدی flags برابر صفر است، اولین چکیدهٔ استفاده نشده (Hash #1) را در این گره قرار می‌دهیم و دیگر کاری با گره‌های زیر دستی آن داریم. این گره را گره شمارهٔ دو می‌نامیم. به این ترتیب مطابق شکل ۵.۲ مقدار H1 در این گره قرار می‌گیرد. به گره بالاتر (ریشهٔ مرکل) برگشته و بچهٔ راستی آن را انتخاب می‌کنیم.

۴. شماره این گره را سه می‌گذاریم. مقدار بیت بعدی (سوم) flags برابر ۱ است، پس مقدار این گره باید توسط گره‌های زیردستی آن محاسبه شود. به این ترتیب، بچهٔ سمت چپی آن انتخاب می‌شود.

۵. شمارهٔ این گره را چهار می‌گذاریم. در این مرحله هم مانند مرحلهٔ قبل، چون بیت چهارم flags نیز یک است گره بچهٔ سمت چپی انتخاب می‌شود.

۶. شمارهٔ این گره را پنج می‌نامیم. در این مرحله یک گره TXID انتخاب شده است. از آن‌جا که گره‌های مربوط به تراکنش‌ها زیرگره‌ای ندارند، مقدار آن‌ها حتما باید توسط گره کامل در اختیار گره سبک قرار گیرد. به این ترتیب مقدار چکیدهٔ استفاده نشدهٔ بعدی، یعنی Hash #2 را در این گره قرار می‌دهیم. مقدار بیت پنجم flags که متناظر با این گره است برابر یک بوده که این معنا را می‌رساند که این TXID برای تراکنشی است که یکی از عناصر آن با فیلتر بلوم منطبق شده‌اند پس به گره سبک دریافت کننده مربوط است.

۷. در مرحلهٔ بعدی، به گره پدر (چهارم) برگشته و گره بچهٔ سمت راستی انتخاب می‌شود. این گره، گره ششم است. باز هم چون این گره، بچه‌ای ندارد و مربوط به یک TXID است، مقدار Hash #3 را در آن قرار می‌دهیم. بیت ششم flags صفر بوده به این معنا است که این تراکنش با فیلتر منطبق نشده است. ارسال آن صرفا برای اثبات مرکل نیاز است.

۸. به گره پدر (گره چهارم) بر می‌گردیم. از آن‌جایی که اطلاعات لازم برای محاسبهٔ چکیدهٔ این گره را از دو مقدار TXID داده شده داریم، مقدار چکیدهٔ را محاسبه کرده و سپس گره بالاتر را انتخاب می‌کنیم.

۹. در این مرحله وارد بچهٔ راست سومین گره می‌شویم. چون مقدار بیت هفتم flags صفر است، چکیدهٔ Hash #4 را درون آن قرار می‌دهیم. در این مرحله دیگر گره‌ای نیست که گره کامل مقدار آن را فرستاده باشد و بیت آخر flags به خاطر لایی گذاری مقدار صفر دارد.

۱۰. در مرحلهٔ آخر مقدار چکیدهٔ گره سوم و به تبع آن مقدار ریشهٔ درخت مرکل را محاسبه می‌کنیم. بررسی

می‌شود که مقدار ریشهٔ محاسبه شده با مقدار ریشهٔ مرکلی که در سرایند بلوک قرار داشته است یکسان باشد.

۴.۱.۳.۲ ملاحظات پیاده‌سازی

بیت‌کوین‌جی[۱] به صورت پیش‌فرض، نرخ خطای نوع دو را برابر ۰٫۱ درصد قرار می‌دهد. هرچند که بالا بردن نرخ خطای نوع دو می‌تواند به حفظ بهتر حریم خصوصی کاربران سبک بیانجامد، اما نه تنها باعث افزایش پهنای باند مصرفی کاربر سبک می‌شود، بلکه احتمال آن که آدرس‌های پر استفاده‌ای مثل ساتوشی‌دایس[۳۶]، که یک سایت شرط‌بندی مبتنی بر زنجیرهٔ بلوکی است، منطبق با فیلتر بلوم شود بیشتر خواهد شد. در این صورت اطلاعات به شدت زیادی برای کاربر سبک ارسال خواهد شد و اگر کاربر سبک به گره کامل هم‌گام‌ساز اجازه دهد که فیلتر را به‌روز رسانی کند، به سرعت فیلتر اشباع و بلااستفاده خواهد شد.

در کتاب‌خانهٔ بیت‌کوین‌جی، اگر کاربر بخواهد m عنصر را در فیلتر بلوم قرار دهد، مقادیر اندازه فیلتر (n) و تعداد توابع چکیده‌ساز آن (k) با توجه به ۱۰۰ عنصر اضافه‌تر طبق فرمول‌های (۱.۲) و (۲.۲) تعیین می‌شوند $M = m + ۱۰۰$ [۲۵]. هدف از این کار آن است که امکان اضافه شدن عناصر جدید به فیلتر بلوم توسط خود کاربر یا توسط گره کامل، بدون نیاز به به‌روزرسانی آن مطابق آن‌چه قبل‌تر توضیح داده شد، فراهم باشد به نحوی که فیلتر بلوم سریع پر نشود و نرخ خطای نوع دو آن به قدری زیاد نشود که فیلتر بلوم عملا بلااستفاده شود. در حالی که به مرور زمان به تعداد عناصر فیلتر بلوم یک کاربر SPV اضافه می‌شود، قاعدتا، مقادیر k و n تغییری نمی‌کنند. اما اگر کاربر SPV نیاز به راه‌اندازی مجدد کیف‌پولش داشته باشد، در راه‌اندازی دوباره، نرم‌افزار بیت‌کوین‌جی با توجه به m، جدید، اقدام به محاسبهٔ $M = m + ۱۰۰$ می‌نماید و به این ترتیب مقادیر k و n برای فیلتر جدید متفاوت خواهند بود.

همچنین در پیاده‌سازی گره سبک بیت‌کوین‌جِی برای هر آدرس، کلید عمومی (PubKey) و چکیدهٔ کلید عمومی (PubKeyHash) گذاشته می‌شود. پس به ازای یک آدرس بیت‌کوین، دو عنصر در فیلتر بلوم قرار می‌گیرند. به بیان دیگر، در ازای قرار دادن N آدرس در فیلتر بلوم، $m = ۲N$ عنصر در آن قرار می‌گیرد پس با توجه به آن‌چه در بالا گفته شد، می‌توان نوشت $M = ۲N + ۱۰۰$. قرار دادن کلید عمومی و چکیدهٔ آن یک آسیب‌پذیری در فیلتر بلوم ایجاد خواهد کرد که به گره متخاصم این امکان را می‌دهد که در صورتی که متوجه شود یک PubKey در فیلتر بلوم قرار دارد، مقدار چکیده (PubKeyHash) آن را نیز امتحان می‌کند. اگر مقدار چکیده هم در فیلتر بلوم قرار داشت، با اطمینان بیشتری می‌تواند مطمئن شود که این آدرس، یکی از آدرس‌های

[36] Satoshi Dice (https://satoshidice.com/) -
!

کاربر سبکِ استفاده کننده از فیلتر بلوم است. قطعه کد زیر بخشی از پیاده‌سازی فیلتر بلوم در کد بیت‌کوین‌جِی است [۲] که این مسئله را به خوبی نشان می‌دهد.

```
/** Inserts the given key and equivalent hashed form (for the address).
*/
public synchronized void insert(ECKey key) {
insert(key.getPubKey());
insert(key.getPubKeyHash());
}
```

در پایان‌نامهٔ[۳۷]، با هدف پیدا کردن آدرس‌های نهفته شده در این فیلتر بلوم با استفاده از این آسیب‌پذیری، در بازهٔ تاریخی ۱۲ دسامبر ۲۰۱۴ الی ۱۰ فوریه ۲۰۱۵، یک گره کامل راه‌اندازی شده و شروع به جمع‌آوری ۷۰, ۰۷۸ فیلتر بلوم از کاربران سبک کرده است. همچنین، در این پایان‌نامه، مجموعه‌ای از تمام کلید عمومی‌ها (PubKey) و چکیدهٔ کلید عمومی (PubKeyHash) متناظر آن‌ها که در زنجیرهٔ بلوکی مورد استفاده قرار گرفته‌اند جمع‌آوری شده است. در نهایت همهٔ آن‌ها را با تمام فیلترهای بلوم جمع شده تطبیق داده است. اگر هر جفت کلید عمومی و چکیدهٔ آن بر فیلتر منطبق بود، نتیجه گرفته است که آن آدرس در آن فیلتر قرار دارد. در نهایت این پایان‌نامه توانسته است به ۵۵, ۱۱۱ جفتِ کلید عمومی و چکیدهٔ آن برسد که هردو در یک فیلتر بلوم منطبق هستند.

هرچند که این ایراد به نظر ایرادی می‌آید که به سادگی قابل حل شدن باشد، اما در صورتی که بیت‌کوین‌جی کلید عمومی‌ها را در فیلتر قرار ندهد، کیف پول‌هایی که از آن کتاب‌خانه استفاده می‌کنند نخواهند توانست از تراکنش‌هایی که خروجی آن‌ها P2PK است مطلع شود. در حالی که، بیت‌کوین‌جی می‌خواهد از تمام انواع تراکنش‌ها پشتیبانی کند. به خاطر همین، باتوجه به آگاهی به وجود این مشکل، اقدامی برای برطرف کردن آن انجام نشده است.

در کنار مشکلات ذکر شده، استفاده از فیلتر بلوم در شبکهٔ همتابه‌همتای بیت‌کوین با آسیب‌پذیری‌ها و چالش‌های بیشتری مواجه است که عملا این ابزار را برای حفظ حریم خصوصی کاربران سبک بلااستفاده کرده است. در بخش ۵.۱.۳.۲ به بررسی این ضعف‌ها پرداخته شده است.

۵.۱.۳.۲ آسیب‌پذیری‌ها

مقالهٔ [۲۵] به طور مفصل به بررسی آسیب‌پذیری‌های موجود در فیلتر بلوم استفاده شده در شبکهٔ همتابه‌همتای بیت‌کوین پرداخته است. در این مقاله توضیح داده شده است که فیلتر بلوم نشت اطلاعاتی بسیار زیادی دارد که

این نشت به تعداد آدرس‌هایی که یک کاربر دارد وابسته است. اگر کاربر تعداد متوسطی، مثلا ۱۰ آدرس، را در فیلتر بلومی قرار دهد، مهاجم می‌تواند با احتمال خوبی آدرس‌های قرار گرفته شده در فیلتر بلوم را حدس بزند. به عنوان مثال احتمال درست حدس زدن آدرس‌های فیلتر بلوم با ۱۰ آدرس برابر ۰/۹۹ است.

علاوه بر این حتی اگر تعداد آدرس‌ها در فیلتر بلوم افزایش پیدا کند، در حالی که مهاجم بتواند به دو فیلتر بلوم مربوط به یک کاربر سبک دست پیدا کند، قادر خواهد بود که با دقت بالایی آدرس‌های مربوط به کاربر سبک را تشخیص دهد. چرا که اگر یک گره کامل متخاصم دو فیلتر بلوم متفاوت از یک کیف پول را در دست داشته باشد، می‌تواند با وارد کردن عناصر به هر دو فیلتر، تا حد قابل ملاحظه‌ای خطاهای نوع دوم را برطرف نماید[۳۷]. لازم به ذکر است که در پیاده‌سازی‌های فعلی با راه‌اندازی مجدد گره سبک، چون از مقدار تصادفی nTweak متفاوتی استفاده خواهد شد، فیلتر بلوم تغییر می‌کند و به گره کامل متخاصم شانس دسترسی به فیلتری‌های بلوم متعددی از یک کاربر سبک را می‌دهد[۲۵] .

در مقالهٔ [۲۵]، به معرفی یک معیار برای سنجش حریم خصوصی فیلتر بلوم پرداخته است. این معیار اینطور تعریف می‌شود که $P_{h_{(j)}}$ برابر است با احتمال آن‌که یک گره متخاصم، j عنصری که واقعا در فیلتر بلوم قرار گرفته‌اند و فرد متخاصم اطلاعاتی در مورد آن‌ها نداشته است را درست حدس بزند. محاسبهٔ $P_{h_{(j)}}$ به صورت زیر است:

$$P_{h_{(j)}} = \prod_{k=۰}^{j-۱} \frac{N-k}{N+N_v-k} = \frac{N}{N+N_v} \cdot \frac{N-۱}{N+N_v-۱} \cdots \qquad (۷.۲)$$

که در آن N تعداد آدرس‌هایی است که در بیت‌کوین قرار داده شده است. از آن‌جایی که هم PubKey و هم PubKeyHash درون فیلتر بلوم قرار می‌گیرند، تعداد عناصر قرار گرفته در فیلتر بلوم برابر $m = ۲N$ است. N_v هم تعداد اعضای مجموعهٔ عناصری است که به خاطر خطای نوع دو با فیلتر بلوم منطبق می‌شوند. از نظر شهودی، معادلهٔ (۷.۲) به معنی احتمال آن است که j آدرس انتخاب شده از بین تمام آدرس‌هایی که منطبق با فیلتر بلوم می‌شوند، جزء آدرس‌های اصلی فیلتر باشند.

با توجه به معادلهٔ (۷.۲) احتمال آن‌که کاربر متخاصم تمام آدرس‌هایی که در حقیقت درون فیلتر بلوم B هستند را به درستی حدس بزند، برابر $P_{h_{(N)}} = \prod_{k=۰}^{N-۱} \frac{N-k}{N+N_v-k} = \frac{N!N_v!}{(N+N_v)!}$ خواهد بود[۲۵]. بدیهی است که هر چه مقدار $P_{h_{(.)}}$ بیش‌تر باشد، فیلتر بلوم حریم خصوصی را کمتر حفظ می‌کند. علاوه بر این گره کامل می‌تواند تعداد عناصر قرارداده شده در فیلتر را نیز حدس بزند که این خود می‌تواند به برملاء شدن اطلاعات کاربر کمک نماید. در این مقاله با فرض این‌که گره کامل متخاصم تنها بتواند به یک فیلتر بلوم مربوط به یک کیف پول دسترسی پیدا کند، می‌تواند تخمینی از تعداد عناصر موجود در یک فیلتر بلوم را با توجه به اندازهٔ فیلتر، توابع چکیده‌ساز و تعداد بیت‌هایی از فیلتر که یک شده‌اند انجام دهد. این مقاله این تخمین را با بهره‌گیری از ایدهٔ مقالهٔ

[۴۷] محاسبه کرده است که به شرح زیر است:

$$m \approx -n\frac{\ln\left(۱-\frac{X}{n}\right)}{k} \tag{۸.۲}$$

که در آن X تعداد بیت‌های فیلتر بلوم مورد نظر است. از طرف دیگر اگر $\mathcal{B}_i$ تمام آدرس‌هایی در شبکهٔ بیت‌کوین باشد که در فیلتر بلوم صدق می‌کنند مقدار آن برابر $|\mathcal{B}_i| = N + N_v$ خواهد بود. پس:

$$N_v = |\mathcal{B}_i| - N \approx |N_u - N|P_f(۲N) \tag{۹.۲}$$

که در آن N_u تعداد کل آدرس‌های بیت‌کوین و $P_f(۲N)$ احتمال خطای نوع دو فیلتر به ازای قرار دادن $۲N$ عنصر در آن است. به این ترتیب می‌توان فرمول (۷.۲) را به صورت زیر نوشت:

$$P_{h_{(j)}} = \prod_{k=۰}^{j-۱}\frac{N-k}{N+N_v-k} \approx \prod_{k=۰}^{j-۱}\frac{N-k}{N+|N_u-N|P_f(۲N)-k} \tag{۱۰.۲}$$

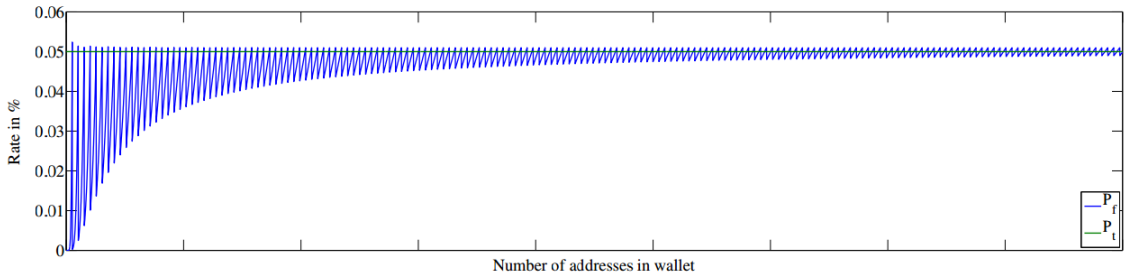
که در این معادله N_u تعداد تمام آدرس‌های استفاده شده در شبکهٔ بیت‌کوین بوده که مقدار آن در زمان نگارش این پایان‌نامه ۳۰٫۴ میلیون آدرس است[۳۷].

همان‌طور که قبل‌تر گفته شده بود طبق [۲۵] در زمان ساختن ابتدائی فیلتر بلوم در بیت‌کوین‌جی، مقادیر n و k با توجه به ۱۰۰ عنصر بیشتر از تعداد عناصری که کاربر می‌خواهد وارد کند انتخاب می‌شوند ($M = m+۱۰۰$). به این‌ترتیب مقدار $P_f(m)$ بسیار کمتر از حالتی است که تمام M عنصر در فیلتر بلوم قرار گرفته باشد ($P_t = P_f(M)$). شکل ۶.۲ تفاوت P_t و P_f را با توجه به تعداد آدرس‌های قرار گرفته در فیلتر بلوم نشان می‌دهد.

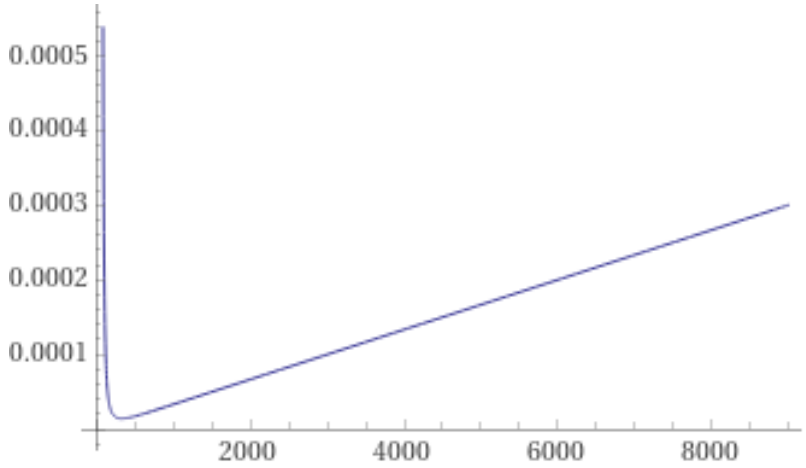
کوچک بودن $P_f(۲N)$ نسبت به P_t باعث می‌شود که امکان فاش شدن آدرس‌های اصلی فیلتر بلوم، در زمان راه‌اندازی فیلتر وقتی تعداد آدرس‌های آن کم باشد، بسیار بالاتر از حد انتظار باشد. به عنوان مثال، در یک فیلتر بلوم با ۱۵ آدرس و به تبع آن ۳۰ عنصر، مقدار $M = ۱۳۰$ خواهد بود. در نتیجه، با توجه به فرمول‌های (۱.۲)، (۲.۲) و (۴.۲) برای نرخ خطای دوی هدف $P_t = ۰٫۰۰۱$ خواهیم داشت: $n = ۱۸۶۹$، $k = ۱۰$ و $P_f(۳۰) = ۵٫۱ \times ۱۰^{-۹}$.

حال با توجه به فرمول (۱۰.۲) احتمال آن‌که گره کامل متخاصم بتواند یکی از آدرس‌های قرارگرفته در فیلتر

[37] https://rb.gy/3o6nrm



شکل ۶.۲: مقادیر محاسبه شده برای P_f و P_t با توجه به تعداد آدرس‌های (N) قرار داده‌شده در فیلتر بلوم. محور افقی این نمودار، نسبت m فعلی فیلتر به M انتخاب شده در زمان راه‌اندازی است. [۲۵]



شکل ۷.۲: احتمال حدس درست یک آدرس اصلی فیلتر بلوم ($P_{h_{(۱)}}$) با توجه به تعداد آدرس‌های آن (N) در راه‌اندازی اولیه.

بلوم را حدس بزند برابر $P_{h_{(۱)}} = ۰/۹۹$ خواهد بود. به همین ترتیب گره کامل متخاصم می‌تواند با احتمال $P_{h_{(۱۵)}} = ۰/۸$ تمام آدرس‌های اصلی داخل فیلتر بلوم را حدس بزند. جدول ۱۰.۲ از مقالهٔ [۲۵] مقایسه‌ای بین تعداد آدرس‌های قرارگرفته در فیلتر بلوم در زمان راه‌اندازی و احتمال حدس زدن آدرس‌های آن توسط گره متخاصم را نشان داده است. توجه شود که مقادیر $P_{h_{(.)}}$ نزولی اکید نیستند. از نظر شهودی نیز انتظار می‌رود که هرچه تعداد آدرس‌های یک فیلتر بلوم، در مقایسه با تمام آدرس‌های بیت‌کوین، افزایش پیدا کند، احتمال آن‌که آدرسی که در فیلتر بلوم منطبق است جزء آدرس‌های اصلی آن باشد بیش‌تر می‌شود. این ویژگی در مقادیر $P_{h_{(۱)}}$ در جدول ۱۰.۲ قابل مشاهده است. شکل ۷.۲ نموداری از احتمال حدس درست یک آدرس اصلی فیلتر بلوم با توجه به تعداد آدرس‌های آن در راه‌اندازی اولیه است.

در مقالهٔ [۲۵] همچنین اثبات کرده است که اگر یک کاربر سبک دو فیلتر بلوم با اعداد تصادفی (nTweak) متفاوت اما با اعضای دارای اشتراک تولید کند، احتمال آن‌که یک گره متخاصم j عنصری که واقعا در فیلتر بلوم

جدول ۱۰.۲: مقادیر $P_{h_{(.)}}$ با توجه به N ($P_t = ٪۰/۱$). [۲۵].

N	۱	۱۹	۴۹	۵۴	۸, ۹۹۹
$P_{h_{(۱)}}$	۱(±۰)	۰/۴۲(±۰/۰۳)	۰/۰۰۲۱(±۰/۰۰۰۱۹)	۰/۱۴(±۰/۰۰۵۹)	۰/۲۱(±۰/۰۰۰۷۵)
$P_{h_{([N/۲])}}$	–	۰/۰۰۰۰۲۶	۰	۰	۰
$P_{h_{([N])}}$	۱	۰	۰	۰	۰

قرار دارند حدس بزند به صورت زیر محاسبه می‌شود:

$$P_{h_{(j)}} \approx \prod_{k=۰}^{j-۱} \frac{N_۱ - k}{N_۱ + P_f(m_۱)P_f(m_۲)N_u - k} \qquad (۱۱.۲)$$

که $P_{h_{(j)}}$ بدست آمده، به طور قابل ملاحظه‌ای، بیشتر از زمانی است که گره متخاصم تنها به یک فیلتر بلوم دسترسی داشته باشد (۷.۲).

امکان جبران حدودی آسیب‌پذیری‌های گفته شده تا اینجا با اصلاح رفتار کاربر سبک وجود دارد. در قسمت ۲.۳ مروری بر کارهایی که کاربر سبکی که از بیت‌کوین‌جی استفاده می‌نماید می‌تواند انجام دهد تا بتواند تا حد ممکن آدرس‌هایش را از گره کاملی که از آن خدمات دریافت می‌کند حفظ نماید، انجام شده است. با این حال آسیب‌پذیری‌های دیگری برای این روش وجود دارد که نیاز به توجه بیشتر دارد.

آسیب‌پذیری دیگر آن است که، به صورت کلی، در کاربردهای حفظ حریم خصوصی با استفاده از فیلتر بلوم، لازم است که به این مسئله توجه شود که اگر با قرار دادن آدرس x در فیلتر بلوم، تعدادی بیت یک شود آیا هر کدام از این بیت‌ها از طریق قرار دادن یک یا چند عنصر پنهان‌سازی (خطای نوع دو) یک می‌شوند؟ به بیان دیگر اگر $b[i]$ فیلتر بلوم تنها توسط عنصر x یک شود و نتوان آن بیت را با قرار دادن عناصری غیر عضو ولی منطبق با فیلتر بلوم یک نمود، امکان حاشا کردن آنکه x در آدرس‌های مطلوب کاربر سبک قرار دارد، ممکن نخواهد بود. در نتیجه، اگر گره کامل متوجه شود که فقط به ازای یک آدرس x خاص، خروجی توابع چکیده‌ساز به یک یا چند بیت مشخص نگاشت می‌شوند، می‌فهمد که حتما آدرس x جزء آدرس‌های اصلی قرار گرفته در فیلتر بلوم بوده است و گره سبک نمی‌تواند وجود آن آدرس را «حاشا» کند. مقاله [۱۲] ضمن اشاره به این آسیب‌پذیری، معیاری برای سنجش حریم خصوصی فیلتر بلوم با توجه به احتمال آنکه بیت‌های یک شده در فیلتر بلوم توسط عناصر غیر عضو پوشش داده شوند، ارائه کرده است که در بخش ۳.۳ به آن پرداخته شده است.

یک مشکل اساسی دیگر روش استفاده از فیلتر بلوم [۲۸]، بار پردازشی بسیار زیاد آن بر روی گره کامل ارائه دهندهٔ این سرویس است چرا که به ازای هر بلوک جدید باید تک‌تک عناصر مهم همهٔ تراکنش‌های بلوک را با

تمامی فیلترهای بلومی که کاربران سبک با او به اشتراک گذاشته‌اند بررسی کند. هر بار بررسی وجود یک عنصر در یک فیلتر بلوم نیاز به چند مرتبه (حداکثر ۵۰ مرتبه) اجرای توابع چکیده‌ساز را دارد. از این رو گره کامل می‌تواند مورد حملهٔ منع خدمت[۳۸] قرار گیرد. کد منبع [۴۹] یک کد پیاده‌سازی این حمله به زبان پایتون[۳۹] است. با این حال تحلیل این آسیب‌پذیری نیاز به توجه بیشتری دارد.

آسیب‌پذیری دیگری که در ارتباط با پرسمان گره سبک از گره کامل وجود دارد، تحلیل بسامد پرسمان یک آدرس و مقایسهٔ پدیدار شدن آن در زنجیرهٔ بلوکی است. مسئلهٔ دیگر مقایسهٔ بازه‌های زمانی فعالیت یک گره سبک و آدرس‌هایی که در آن زمان در زنجیرهٔ بلوکی قرار گرفته و طبق فیلتر بلوم کاربر برای وی ارسال می‌شود، است. در این پایان‌نامه به صورت خاص بر روی این دو موضوع تمرکز شده است.

در حال حاضر بسیاری از کاربران سبک بیت‌کوین هستند که جز سرمایه‌گذاری و نگه‌داری طولانی‌مدت بیت‌کوین فعالیت اقتصادی دیگری با آن انجام نمی‌دهند. این کاربران، خواه از کیف‌پول‌های سرد استفاده نمایند خواه نه، احتمال آن که همواره کیف پولشان در حال اجرا و هم‌گام سازی با شبکه باشد بسیار پایین است. در تلفن‌های همراه، به خاطر حفظ طول عمر باتری، در نتیجهٔ استفاده نشدن طولانی مدت از نرم‌افزار کیف پول، فعالیت‌های پس‌زمنیه‌ای کیف‌پول‌ها متوقف می‌شود.

هرچند که متاسفانه تا کنون جمع‌آوری اطلاعاتی راجع به زمان‌های فعالیت گره‌های سبک و اتصال آن‌ها به گره‌های کامل انجام نشده است، اما همچنان دور از ذهن نیست که فرض کنیم نرم‌افزار کیف پول کاربران کم فعالیت، اکثرا فقط زمان‌هایی به شبکه متصل می‌شوند که بخواهند از قرارگیری تراکنشِ به تازگی منتشر شدهٔ خود در زنجیرهٔ بلوکی مطلع شوند، یا اینکه بررسی کنند که تراکنشی که از طریق دیگری انتظار دریافتش را داشته باشند در زنجیرهٔ بلوکی ثبت شده باشد.

به این ترتیب اگر فرض کنیم که گره سبک کم فعالیت l_i در هر بار اتصال به یک گره کامل مشخص (مثلا f_j) در شبکهٔ بیت‌کوین متصل شود، و تنها در زمان‌هایی که انتظار ثبت تراکنش مربوط به خودش را داشت، با شبکه همگام شود و در بازهٔ زمانی اطراف آن به پیام‌های `inv` از طرف گره کامل f_j پاسخ `getdata` را ارسال نماید، احتمال آن‌که تراکنش‌های منطبق شده با فیلتر بلوم کاربر که برایش ارسال می‌شوند، واقعا مربوط به کاربر سبک باشد، بسیار بیشتر خواهد بود. چرا که در آن بازه، از آن‌جایی که تعداد تراکنش‌های به نسب کمتری در فیلتر بلوم آزموده می‌شوند، تعداد تراکنش‌های حاصل از خطای نوع دو به نسبت بسیار کم خواهند بود. به این ترتیب گره f_j با احتمال بیشتری می‌تواند مطمئن باشد که تراکنش‌هایی که به کاربری که به تازگی متصل شده است ارسال می‌شوند، مربوط به خودش است. این آسیب‌پذیری در روش‌های بعدی که در فصل ۳ به آن‌ها پرداخته خواهد شود نیز وجود دارد. هرچند در ابتدا فرض اتصال همیشگی به یک گره کامل یکسان ناشدنی به نظر بیاید، اما یک

[38] Denial of Service Attack
[39] Python

گره کامل متخاصم می‌تواند بدون نیاز به پرداخت هزینه‌ای، با اجرای حملهٔ سیبیل[۴۰] هویت‌های جعلی زیادی در شبکه ایجاد نماید تا شانس ارتباطش با یک گره سبک را در به‌روزرسانی‌های او بالا ببرد.

حملهٔ دیگری که می‌شود تعریف کرد که نسبت به حملهٔ قبلی شدنی‌تر باشد، آن است که گره کامل، سابقهٔ پرسمان‌های انجام شده از یک کیف پول به خصوص را ذخیره نماید. تشخیص این‌که پرسمان‌های صورت گرفته مربوط به یک کیف پول است می‌تواند از روی فیلترهای بلوم یکسانی که ارسال می‌شود تشخیص داده شود. همچنین اگر کاربر سبک فیلتر بلوم خود را عوض نماید اما از آدرس‌های یکسانی در فیلتر بلوم جدید هم استفاده کند، گره کامل می‌تواند با مقایسهٔ آدرس‌های مشترک، به متعلق بودن هر دو فیلتر بلوم به یک کیف پول پی ببرد.

گره کامل متخاصم می‌تواند با تحلیل بسامدی که از یک کیف پول درخواست دریافت کرده است (با همان فرض قبلی که گره‌های سبک کم فعالیت ارتباطشان را به طور مداوم با گره کامل حفظ نمی‌کنند)، و مقایسهٔ آن با بسامد قرار گرفتن آدرس‌های منطبق شده بر فیلتر بلوم در زنجیرهٔ بلوکی، در مورد آدرس‌های اصلی قرار گفته شده در فیلتر بلوم اطلاعات کسب نماید. به عنوان مثال اگر یک کاربر سبک حدودا هر سه ماه یک‌بار به یک گره کامل متصل شود، گره کامل می‌تواند آدرس‌هایی که با فیلتر بلوم وی منطبق شده و روزانه در زنجیرهٔ بلوکی ظاهر شده‌اند را حذف نماید. از طرف دیگر کاربری که درخواست‌های زیادی انجام داده است، احتمال این‌که مالک آدرس‌های پر استفاده از فیلتر بلومش باشد بیشتر است.

از طرف دیگر در روش فیلتر بلوم، گره کامل می‌تواند به بررسی روابط بین آدرس‌های منطبق شده با روش‌هایی مثل [۳۴] در یک فیلتر بلوم بپردازد. به این ترتیب می‌تواند احتمال بدهد که آدرس‌هایی که با فیلتر بلوم منطبق شده‌اند اما با آدرس‌های دیگر ارتباطی ندارند، جزء آدرس‌های پوششی فیلتر بلوم هستند[۲۵]. انتخاب تصادفی آدرس‌های خطای نوع دو می‌تواند شامل مشکلات دیگری نیز باشد، مثلا با توجه به [۲۵] از آن‌جایی که فیلتر بلوم تازه در نیمهٔ دوم سال ۲۰۱۱ معرفی شده است، اگر آدرسی که به عنوان خطای نوع دو با فیلتر بلوم منطبق شود مربوط به زمانی قبل‌تر از آن باشد (۲۰۰۹ تا ۲۰۱۱)، گره کامل می‌تواند آن آدرس‌ها را با احتمال بالایی به عنوان خطای نوع دو فیلتر بلوم حساب نماید.

در این پایان‌نامه قصد داریم روشی را ارائه دهیم که کاربر سبک بتواند به صورت هوشمندانه آدرس‌های نامرتبط با خودش را به نحوی انتخاب نماید که بسامد استفاده از این آدرس‌ها برابر با نرخ استفاده از آدرس خودش باشد. همچنین در این روش کاربر سبک مجبور نخواهد بود که آدرس‌های مربوط به خودش را در یک ساختار دادهٔ واحدی مثل فیلتر بلوم قرار دهد که گره کامل بتواند با بررسی روابط بین آدرس‌های آن و آدرس‌های خطای نوع دو، به آدرس‌های اصلی پی ببرد. بلکه چون گره سبک برای درخواست به روزرسانی هر آدرسش از یک سری مجموعه آدرس‌های پوششی نا مرتبط استفاده می‌کند و بقیهٔ آدرس‌های خودش را در آن‌ها قرار نمی‌دهد، گره کامل نمی‌تواند به ارتباط بین آدرس‌های مربوط به آن کاربر SPV پی ببرد. همچنین گره سبک می‌تواند به سادگی

[40] Sybil Attack

مجموعهٔ آدرس‌های خود را به روزرسانی کند، بدون آن‌که حریم خصوصیش از این بابت نقض شود.

فصل ۳

مروری بر کارهای انجام شده

۱.۳ مقدمه

مایک هرن[1]، نویسندهٔ BIP37 [۲۸] در پست [۲۷] خودش اعلام می‌کند که فیلتر بلوم از امنیت کافی برخوردار نیست. او در این پست به برخی از ایراداتی که در بخش ۵.۱.۳.۲ به آن‌ها اشاره شده، پرداخته است. همچنین مروری بر راه‌حل‌های جایگزین از جمله استفاده از روش‌های بازیابی اطلاعات خصوصی[2] (PIR)، رمزنگاری ارتباط همتابه‌همتا برای جلوگیری از افشا شدن اطلاعات نزد طرف‌های متخاصم، از جمله سازمان‌های اطلاعاتی، اشاره می‌کند. هرن همچنین توضیح می‌دهد که حل این مسئله ساده نخواهد بود و به دشواری‌ها و چالش‌های آن اشاره کرده است[۲۷].

علاوه بر این، افراد دیگری پیشنهادهای زیادی در تغییر شیوهٔ موجود ارائه کرده‌اند که در این قسمت به بررسی پیشنهادها و راه‌حل‌هایی که تا کنون برای بهبود حریم خصوصی کاربران سبک منتشر است پرداخته می‌شود.

۲.۳ اصلاح رفتار کاربر سبک فعلی جهت حفظ حریم خصوصیش

در مقالهٔ [۲۵] در کنار تحلیل امنیت و بیان ضعف‌های استفاده از فیلتر بلوم در ارتباط بین گره سبک با گره کامل [۲۸]، به بیان چند رویه پرداخته است که اگر گره سبک از این رویه‌ها پیروی کند، می‌تواند در عین این که از

[1] Mike Hearn
[2] Private Information Retrieval (PIR)

پروتکل فعلی استفاده می‌کند، تا حدی حریم خصوصی خودش را حفظ کند. در ادامه به بیان این موارد پرداخته می‌شود.

همان‌طور که در ۵.۱.۳.۲ نشان داده شد، نرخ خطای نوع دوی فیلتر بلوم به طور قابل ملاحظه‌ای تحت تاثیر تعداد عناصر قرار گرفته در یک فیلتر بلوم است. همچنین باید از ایجاد چند فیلتر بلوم با عناصر مشترک پرهیز شود. در نتیجه پیشنهاد می‌شود هر گره سبک در ابتدا یک فیلتر بلوم با N آدرس ایجاد کند به طوری که $M = N$. به این ترتیب، فیلتر بلوم با نرخ خطای نوع دوی هدف، P_t، ساخته می‌شود. همچنین پیشنهاد شده است که $M = m$، به این معنی که تنها یکی از مقادیر PubKey یا PubKeyHash در فیلتر بلوم قرار بگیرد. مقالهٔ [۲۵] بررسی کرده است که برای تقریبا ٪۹۹ از آدرس‌های بیت‌کوین قرار دادن یکی از این دو مقدار در فیلتر بلوم کفایت می‌کند تا تمام تراکنش‌های مرتبط با خودشان را دریافت نمایند.

گره سبک می‌تواند به مرور که به آدرس‌های بیشتری نیاز پیدا کرد، از N آدرسی که پیش‌پیش در فیلتر بلوم قرار گرفته است، استفاده نماید. زمانی که از تمام این N آدرس استفاده کرد، یک فیلتر بلوم جدید تولید نماید که این هم شامل N آدرس جدید باشد و آدرس مشترکی با فیلتر بلوم قبلی نداشته باشد. گره سبک می‌تواند این که هر آدرسش در کدام فیلتر بلوم قرار گرفته است را تحت اطلاعاتی جانبی، در کنار آدرس‌هایش، ذخیره نماید. در این صورت گره متخاصم نمی‌تواند با در دست داشتن فیلترهای بلوم مربوط به یک کیف پول به اطلاعات اضافه‌ای دست پیدا کند. به این ترتیب گره سبک باید همزمان از چند فیلتر بلوم استفاده نماید و آن‌ها را برای گره‌های مختلف ارسال کند. البته خود مقالهٔ [۲۵] اذعان داشته است که در صورتی که گره سبک برای اولین بار از یک آدرس از پیش ذخیره شده در فیلتر بلوم استفاده نماید، از آن‌جایی که این آدرس تا الان در زنجیره بلوکی استفاده نشده است و در اولین استفاده‌اش با فیلتر بلوم این کاربر منطبق شده است، می‌تواند گره کامل را مطمئن سازد که این آدرس جزء آدرس‌های اصلی این فیلتر بلوم است.

برای حفظ بیشتر حریم خصوصی کاربر و رفع ضعف ذکر شده، مقالهٔ [۲۵] پیشنهاد داده است که گره سبک با توجه به آدرس‌های موجود در زنجیرهٔ بلوکی، که مربوط به خودش نیستند، دست به ایجاد یک فیلتر بلوم بزند. سپس سعی کند برای هر آدرسی که احتیاج دارد، با تلاش‌ها و آزمون و خطاهای مکرر آدرس جدید را طوری ایجاد نماید که در فیلتر بلوم تولید شده قرار گیرد. در نتیجه در صورت قرار گرفتن آدرس تازه ساخته شدهٔ این گره در زنجیرهٔ بلوکی، احتمال بیشتری وجود خواهد داشت که گره کامل آن تراکنش آن را برای گره‌های دیگری نیز ارسال نماید. اما این روش بار پردازشی زیادی را بر روی گره سبک بابت تولید آدرس جدید تحمیل می‌کند.

در هر بار راه‌اندازی یک کیف پول در یک گره سبک، نرم‌افزار کیف پول شروع به محاسبهٔ مجدد فیلتر بلوم با استفاده از آدرس‌هایش می‌نماید؛ چرا که فیلتر بلوم تولید شده‌اش را در حافظه دائمی ذخیره نمی‌کند. در نتیجه، این موضوع می‌تواند باعث شود که فیلترهای بلوم متعددی با $nTweak$های متفاوت اما عناصر یکسان در دست یک گره کامل بیافتد. مقالهٔ [۲۵] پیشنهاد داده است که گره سبک فیلتر بلومش و اطلاعات جانبی آن مانند آدرس‌هایی

که در آن قرار گرفته است و غیره را در یک حافظهٔ دائمی ذخیره کند. این مقاله تخمین زده است که هر گره سبک نیاز خواهد داشت که برای هر فیلتر بلوم، چیزی در حدود ۲۲۰ بایت ذخیره نماید که سربار قابل توجهی به نرم‌افزار کیف پول اضافه نمی‌کند.

روش‌های پیشنهاد شده در این قسمت، هر چند تا حدودی توانسته بودند ضعف‌های اساسی [۲۸] را جبران نمایند، اما به طور کامل نتوانسته بودند که ایرادات آن را برطرف کنند. علاوه بر این، روش‌های پیشنهاد شده نسبت به حملهٔ تحلیل بسامد پرسمان و استفاده، آسیب پذیر است. همچنین راه حل مشخصی پیشنهاد نشده است که جلوی گره کامل متخاصم گرفته شود تا نتواند از روی روابط بین آدرس‌های یک فیلتر بلوم به آدرس‌های اصلی پی ببرد. باید به این نکته نیز توجه کرد که یکی از دلایلی که در پیاده‌سازی گره سبک، المان‌های فیلتر تازه ساخته شده باتوجه به $M = m + ۱۰۰$ بوده است آن است که اجازهٔ به روزرسانی فیلتر با توجه به تراکنش‌های منتشر شده در شبکه، طبق جدول ۸.۲، به گره کامل داده شود و در عین حال جلوی اشباع زودهنگام فیلتر گرفته شود. اما با توجه به راه حل [۲۵]، که پیشنهاد داده است که در همان ابتدا $M = m$ باشد، امکان به روزرسانی فیلتر با توجه به تراکنش‌های جدید سلب می‌شود.

۳.۳ معیار حاشاپذیری-γ برای سنجش حریم خصوصی فیلتر بلوم

در مقاله [۱۲] معیاری کمّی، بر اساس مدل گمنامی-K [۴۸]، برای اندازه‌گیری حریم خصوصی فیلتر بلوم معرفی شده است.این معیار حاشاپذیری[۳]-γ نام دارد. در این مقاله بیان شده است که احتمال خطای نوع دو (P_f) به تنهایی معیار مناسبی برای سنجش حریم خصوصی فیلتر بلوم نیست. بلکه باید تعداد عناصر خطای نوع دو (N_v) مورد بررسی قرار گیرد. واضح است که اندازهٔ N_v علاوه بر P_f وابسته به تعداد کل عناصر (N_u) است: $N_v = (N_u - m) \times P_f$. با توجه به این موضوع، مقاله [۱۲] با بهره‌برداری از نسخهٔ احتمالاتیِ مدل گمنامی-K [۳۲]، یک معیار سنجش گم‌نامی مناسب فیلتر بلوم ارائه داده است. عنوان این معیار «حاشاپذیری-γ» است.

در فیلتر بلوم، داده به صورت تجزیه ناپذیر ذخیره می‌شود و در گمنامی-K داده به صورت ساختاریافته و دارای ویژگی‌های مشخصی هست. اما می‌توان شباهت‌های نزدیکی بین آن‌ها در نظر گرفت. به طور شهودی می‌توان این گونه تعبیر کرد که بیت‌های فیلتر ($b[i]$ و $i \in [۰, n-۱]$) «ویژگی‌های» عنصر x هستند. یعنی، عنصر x دارای ویژگی $b[i]$ است، اگر و تنها اگر به ازای حداقل یک $j \in [۱, k]$ داشته باشیم $H_j(x) = i$. به این ترتیب می‌توانیم از تعریف گمنامی-K در فیلتر بلوم استفاده نماییم. عنصر x، قرار گرفته در فیلتر، گمنام-K یقینی است اگر به ازای تمام بیت‌های $b[i]$ که توسط این عنصر یک شده‌اند، حداقل $K-۱$ عنصر خطای نوع دو

[3] Deniability

وجود داشته باشد که به همان بیت‌ها نگاشت شوند.

می‌توان از این تعریف فهمید که برقراری شرایط گمنامی-K یقینی همیشه امکان‌پذیر نیست. از این رو، استفاده از تعمیم احتمالاتی گمنامی-K [۳۲] برای فیلتر بلوم مناسب‌تر است. به این ترتیب مقالهٔ [۱۲] برای عنصری که به فیلتر بلوم اضافه شده است، از صفت «حاشاپذیر» استفاده کرده است. به این معنی که آیا دارنده فیتلر می‌تواند وجود آن عنصر در فیلتر را انکار نماید یا خیر. به این ترتیب می‌گوییم عنصر $x \in \mathcal{S}$ حاشاپذیر است اگر به ازای $\forall i \in \{۱..k\}$، حداقل یک عنصر از مجموعه پنهان‌سازی $v \in \mathcal{V}$ (خطای نوع دو) وجود داشته باشد به گونه‌ای که $\exists j \in \{۱..k\}$ به شرطی که $H_i(x) = H_j(v)$. به بیان ساده‌تر یک عنصر حاشاپذیر است اگر بتوان بدون تغییر بیت‌های فیلتر، آن عنصر را توسط عناصری که عضو فیلتر نیستند جایگذاری کرد.

فیلتر بلوم B، حاشاپذیر-γ است (یا دارای ویژگی حاشاپذیری-γ) است، هر گاه یک عنصر تصادفی آن $x \in \mathcal{S}$ با احتمال γ حاشاپذیر باشد. احتمال تقریبی حاشاپذیری-γ فیلتر B به صورت معادله (۱.۳) محاسبه می‌شود [۱۲].

$$\gamma(B) \approx \left(۱ - exp\left(-\frac{N_v k}{n\left(۱ - e^{-km/n}\right)}\right)\right)^k \tag{۱.۳}$$

که در آن $N_v = (N_u - m) \times P_f$. هرچه مقدار γ به یک نزدیک‌تر باشد، سطح بهتری از حریم خصوصی مُهیا شده است. شکل ۱.۳ مثالی را نشان می‌دهد که در آن $\mathcal{S} = \{x_۱, x_۲, x_۳\}$ مجموعه عضو فیلتر بلوم است. مجموعهٔ پنهان‌سازی (خطای نوع دو)، شامل عناصر $\mathcal{V} = \{v_۱, v_۲, v_۳\}$ می‌شود. عنصر $x_۱$ حاشاپذیر است چرا که بیت‌های مرتبط با آن، یعنی $b[۰]$، $b[۲]$ و $b[۷]$، توسط عناصر $v_۱$ و $v_۲$ پوشانده شده است. به همین ترتیب می‌توان نشان داد که عنصر $x_۲$ نیز حاشاپذیر است. اما عنصر $x_۳$ حاشاپذیر نیست. چرا که بیت $b[۸]$ توسط هیچ‌کدام از عناصر مجموعهٔ پنهان‌سازی پوشانده نشده است. به این ترتیب، این فیلتر به صورت کلی، حاشاپذیر-۰٫۶۶ است.

در [۲۹] پیشنهاد شده است که فیلتر بلوم استفاده شده در پروتکل بیت‌کوین با توجه به معیار حاشاپذیری-γ [۱۲]، ساخته شود. زیرا نرخ خطای نوع دو (P_t) به تنهایی برای سنجش حریم خصوصی فیلتر بلوم ساخته شده کافی نیست. به این ترتیب لازم است که طبق معادله (۱.۳) در هر لحظه باتوجه به تعداد آدرس‌های یکتایی که از نقطه بررسی تا آخرین بلوک استخراج شده در زنجیره بلوکی نمایان شده‌اند (N_u) و γ، مقدار P_t تعیین گردد. از آن‌جایی که محاسبه N_u برای گره سبک غیرممکن است، در [۲۹] پیشنهاد شده است که از تکنیک رگرسیون خطی برای تخمین N_u استفاده شود. ضرایب مدل رگرسیون خطی، باید بصورت متناوب (مثلا به صورت هفتگی) محاسبه گردد. این محاسبه می‌تواند به توسعه دهندگان نرم‌افزار که طرح ارائه شده در [۲۹] را پیاده‌سازی می‌کنند،

	b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]
x_1	1	0	1	0	0	0	0	1	0
x_2	0	0	1	1	0	0	0	1	0
x_3	0	0	0	1	0	1	0	0	1
B(S)	1	0	1	1	0	1	0	1	1
v_1	1	0	1	1	0	0	0	0	0
v_2	0	0	1	0	0	1	0	1	0
v_3	1	0	0	0	0	1	0	1	0

شکل ۱.۳: یک فیلتر بلوم تشکیل شده از عناصر $\{x_۱, x_۲, x_۳\}$ که سه عنصر $\{v_۱, v_۲, v_۳\}$ را به عنوان خطای نوع دو می‌پذیرد[۱۲].

سپرده شود. به این ترتیب گره سبک می‌تواند مقدار P_t را به نحوی تعیین کند که از امنیت فیلتر بلوم مطمئن گردد.

روش ارائه شده در [۲۹] دارای اشکالاتی است. یکی از اصلی‌ترین این اشکالات به روزرسانی متناوب فیلتر بلوم باتوجه به تخمین حاصل از N_u است. طبق مقاله [۲۵]، اگر گره کامل متخاصم به دو فیتلر بلوم که مربوط به یک گره سبک هستند دست پیدا کند، می‌تواند با دقت بیش‌تری آدرس‌های مربوط به گره سبک را حدس بزند. از این رو تولید متناوب فیلتر بلوم می‌تواند حریم خصوصی کاربر سبک را به خطر بیاندازد. از ایرادات دیگر این روش می‌توان به افزایش P_t در نتیجهٔ به کار گیری از این طرح اشاره نمود. به این ترتیب، پهنای باند مورد نیاز زیادتر می‌شود.

۴.۳ فیلترکردن بلوک

در [۴۰] پیشنهاد شده است که بر خلاف آن‌که گره سبک فیلتر بلوم را تولید کند و برای گره کامل ارسال نماید، گره کامل یک فیلتر از روی تمام دادگان یک بلوک ایجاد کند. گره سبک به ازای هر بلوک جدید، فیلتر مربوطه را از گره کامل دریافت کرده و خودش بررسی می‌کند که آیا داده مورد نظرش در آن قرار دارد یا نه. اگر داده مورد نظر گره سبک در آن فیلتر قرار داشت، تمام بلوک را از گره کامل دریافت می‌کند. این ایده برای اولین بار در ایمیل آدام بک[۴]، از دانشمندان حوزهٔ بیت‌کوین، بیان شده است[۱۰]. به فیلتر استفاده شده در این روش فیلتر بلوک[۵] گفته

[4] Adam Back
[5] Block Filter

می‌شود. کیف پول نوترینو[۶] در حال حاضر از این روش پشتیبانی می‌کند.

از آن‌جایی که مقدار ساخته شده برای فیلترها یقینی هستند، نیاز است تنها یک مرتبه ساخته شده و ذخیره شوند. بر خلاف فیلتر بلوم، در این روش از یک عدد تصادفی برای ساخت فیلتر استفاده نمی‌شود. از این رو گره کامل از خطر حملات منع خدمت در امان است. در این روش برای هر فیلتر بلوک یک سرایند مرتبط وجود دارد که اندازهٔ این سرایند ۳۲ بایت بوده و سرایند شامل چکیدهٔ مقدار حاصل از الحاق[۷] چکیدهٔ فیلتر بلوک و سرایند فیلتر بلوک قبلی است. سرایند فیلتر بلوک برای هر بلوک زنجیرهٔ قالبی می‌تواند به عنوان یک خروجی OP_RETURN در تراکنش کوین‌بِیس[۸] قرار بگیرد.

در این روش، هر فیلتر بلوک به ازای هر تراکنش بلوک شامل نبشتهٔ‌های خروجی قبلی که در هر ورودی آن خرج شده است می‌شود همچنین تمام scriptPubKeyهای هر خروجی تمام تراکنش‌ها نیز در آن قرار می‌گیرد. در این روش نیز اگر عنصری در فیلتر قرار گرفته باشد، با احتمال ۱ در آن صدق می‌کند اگر قرار نداشته باشد با احتمال $\frac{۱}{M}$ با آن منطبق می‌شود. مراحل ساخت فیلتر بلوک با N عضو، به شرح زیر است (توجه شود که $N, M < ۲^{۳۲}$)

۱. چکیدهٔ تمام اعضای فیلتر بلوک با استفاده از تابع چکیده‌ساز SipHash محاسبه می‌شود. سپس خروجی تابع چکیده‌ساز به صورت یکنواخت در بازهٔ $[۰, N \times M)$ نگاشت می‌شود.

۲. مقادیر خروجی مرحلهٔ قبل با توجه به مقدارشان مرتب می‌شوند و اختلاف هر دو مقدار متوالی محاسبه می‌شود. برای کوچک‌ترین مقدار، اختلاف آن با صفر محاسبه می‌شود که برابر با خودش است.

۳. مقادیر اختلاف‌ها که از مرحلهٔ قبل بدست آمده پشت سر هم نوشته می‌شوند و به وسیلهٔ کدگذاری گلومب-رایس[۹] فشرده می‌شوند.

از آن‌جا که خروجی مرحلهٔ یک دارای یک توزیع یکنواخت[۱۰] است، اختلاف آن‌ها دارای یک توزیع هندسی[۱۱] خواهد بود. روش کدگذاری گلومب-رایس در فشرده‌سازی داده‌هایی با توزیع هندسی بهینه عمل می‌کند[۳۹]. برای کدگذاری گلومب-رایس، پارامتر P تعریف می‌شود که طول کد باقیمانده را تعیین می‌کند. این کدگذاری به این صورت است که هر مقداری (در اینجا اختلاف بین دو چکیده) بر ۲^P تقسیم شده و خروجی آن دو قسمت خارج قسمت q و باقیماندهٔ r خواهد بود. سپس، q با روش کدگذاری یگانی[۱۲] که به صورت رشته‌ای خواهد بود

[6] Neutrino
[7] Concatenate
[8] Coinbase
[9] Golomb-Rice Coding
[10] Uniform Distribution
[11] Geometric Distribution
[12] Unary Coding

که با تعداد q یک به همراه یک ۰ نوشته می‌شود. مقدار r هم در P بیت با استاندارد اندین بزرگ نوشته می‌شود. به عنوان مثال کدگذاری عدد ۹ با $P = ۲$ به صورت 01 110 خواهد بود. که در آن $q = ۲$ و $r = ۱$ است.

در این روش امکان استفاده از فیلترهای مختلف وجود دارد اما در فیلتر اولیهٔ این روش، مقدار $M = ۷۸۴۹۳۱$ و مقدار $P = ۱۹$ است. حال در این پایان‌نامه، برای آن‌که تخمینی از سربار پهنای باندی برای گره سبک در این روش داشته باشیم، به این ترتیب عمل می‌کنیم:

- در زمان نگارش این پایان‌نامه، تعداد روزانهٔ هرکدام از ورودی‌ها و خروجی‌های P2PKH حدودا برابر ۳۵۰,۰۰۰ عدد است[۱۳] که در مجموع می‌شود ۷۰۰,۰۰۰ عدد در روز. همچنین برای P2SH، تعداد خروجی‌ها برابر ۳۱۰,۰۰۰ و تعداد ورودی‌ها برابر ۲۲,۰۰۰عدد است[۱۴] که در مجموع تعداد آن برابر ۳۳۲,۰۰۰ عدد در روز خواهد بود. به این ترتیب برای هر هر فیلتر بلوک در زمان نگارش پایان‌نامه می‌توان ۷۱۶۷ عضو متصور شد($N = ۷۱۶۷$)

- با فرض اینکه از فیلتر اولیه استفاده شود، $M = ۷۸۴۹۳۱$ و $P = ۱۹$ خواهد بود. به این ترتیب از آن‌جا که خروجی چکیدهٔ اعضای فیلتر بلوک، در بازهٔ $[۰, N \times M)$ نگاشت می‌شوند، نتایج در بازهٔ صفر تا $N \times M = ۵,۶۲۵,۶۰۰,۴۷۷$ به صورت یکنواخت توزیع خواهد شد.

$$۰ \leq h_۱ \leq h_۲ \leq \cdots \leq h_N < ۵/۶۲۵ \times ۱۰^۹ \qquad (۲.۳)$$

که در آن h_iها خروجی تابع چکیده‌ساز بعد از نگاشت به بازهٔ گفته شده بوده که به ترتیب اندازهٔ آن‌ها مرتب شده‌اند.

- با توجه به روش گفته شده تفاضل بین h_iها را به صورت زیر محاسبه می‌کنیم:

$$\delta_i = h_i - h_{i-۱}, \quad ۱ < i \leq N; \qquad \delta_۱ = h_۱ \qquad (۳.۳)$$

- حال باید بر روی مقادیر δ_i کدگذاری گلومب-رایس اعمال شود و بیت‌های حاصل به ترتیب در کنار هم قرار بگیرند. تعداد بیت‌های خروجی برای یک فیلتر بلوک (L) از فرمول زیر محاسبه می‌شود.

[13] https://transactionfee.info/charts/inputs-and-outputs-p2pkh/
[14] https://transactionfee.info/charts/inputs-and-outputs-p2sh/

$$L = \sum_{i=1}^{N} \left(\left[\frac{\delta_i}{2^P} \right] + P + 1 \right) < \left[\frac{\sum_{i=1}^{N} \delta_i}{2^P} \right] + NP + N < \left[\frac{MN}{2^P} \right] + N(P+1) \quad (۴.۳)$$

که در آن $\frac{\delta_i}{2^P}$ تعداد یک‌های حاصل از کدگذاری هر کدام از δ_iها است و به ازای هر کدام از آن‌ها یک بیت صفر و P بیت شامل باقیمانده قرار داده می‌شود. مجموع تفاضل‌های δ_i برابر با h_N می‌شود و با توجه به (۲.۳)، این مقدار می‌تواند حداکثر MN باشد.

- با توجه به مقادیر N، M و P داریم:

$$L < \left[\frac{5/625 \times 10^9}{2^{19}} \right] + 7168 \times 19 = 146921\text{b} = 18366\text{B} = 18\text{KB} \quad (۵.۳)$$

به این ترتیب می‌توان گفت که اندازهٔ هر فیلتر بلوک از ۱۸ کیلوبایت کوچک‌تر است. با توجه به زیاد بودن اندازهٔ M، احتمال آن‌که یک بلوک به عنوان خطای نوع دو انتخاب شود بسیار کم خواهد بود. هرچند که کم بودن نرخ خطای نوع دو باعث کاهش پهنای باند مصرفی گره سبک می‌گردد، اما از طرف دیگر می‌تواند حریم خصوصی کاربر سبک را با خطر مواجه کند.

اگر گره سبک از آدرس‌های محدودی استفاده کند به گره کامل متخاصم این امکان را می‌دهد که بتواند با در نظر گرفتن آدرس‌های مشترک بین بلوک‌های درخواست شده توسط آن کاربر، آدرس کاربر را در مجموعه محدودتری جست‌وجو نماید. گره کامل با استفاده از گراف تراکنش‌ها حتی می‌تواند به نتایج دقیق‌تری دست پیدا کند[۶].

مشکل دیگر این روش، کاربرد آن برای گره‌های سبکی است که تراکنش‌های نسبتا زیادی در شبکه ارسال می‌کنند. حریم خصوصی این گره‌ها نه تنها بیش‌تر در معرض نقض شدن قرار دارد، بلکه، آن‌ها برای هم‌گام سازی با شبکه نیاز است که پهنای باند زیادی را مصرف نمایند. چرا که لازم است برای هر تراکنش، یک بلوک کامل را دانلود نمایند.

گره کامل متخاصم می‌تواند با تحلیل بسامد درخواست‌ها و آدرس‌های بلوک درخواست داده شده تعدادی از آدرس‌های پوششی بلوک‌های درخواست داده شده را کنار بگذارد و در مجموعهٔ کوچک‌تری به جست‌وجوی

آدرس‌های کاربر سبک بپردازد.

۵.۳ بازیابی اطلاعات خصوصی

در مقاله [۴۱] از روش بازیابی اطلاعات خصوصی (PIR) جهت دریافت اطلاعات تراکنش‌ها از گره کامل استفاده کرده است. بازیابی اطلاعات خصوصی به کاربران این امکان را می‌دهد که از یک پایگاه داده یا مجموعه‌ای از آن‌ها یک پرسمان انجام دهند، به گونه‌ای که سرور پایگاه داده نتواند اطلاعاتی راجع به کاربران درخواست دهنده و درخواست آن‌ها کسب نماید. در مقاله [۴۱] از ترکیبی از دو رده بازیابی اطلاعات خصوصی، یعنی بازیابی اطلاعات خصوصی نظریه اطلاعاتی (IT-PIR) و محاسباتی (C-PIR) استفاده کرده است. این ترکیب در مقاله [۲۳] معرفی شده است. در C-PIR، پرسمان توسط کاربر به نحوی کدگذاری می‌شود که پایگاه داده پاسخ مناسب را در اختیار کاربر قرار دهد، اما چیزی از پرسمان و اطلاعات ذخیره شده متوجه نشود. تضمین این حریم خصوصی بر مبنای این فرض است که با اختیار داشتن توان پردازشی محدود، حل برخی مسئله‌ها غیر ممکن یا سخت خواهد بود [۲۳].

رده IT-PIR وابسته به فرض سخت بودن حل الگوریتم‌های پایه رمز نگاری با منابع محاسباتی محدود نیست. پروتکل‌های رده IT-PIR از چند سرور به صورت همزمان استفاده می‌کند. تا زمانی که سرورهایی که تبانی نمی‌کنند از یک تعدادی بیش‌تر باشد، حریم خصوصی کاربر تضمین می‌شود [۲۳].

یکی از نقص‌های IT-PIR آن است که در عمل راه حلی وجود ندارد که بتوان حداقل تعداد سرورهایی که تبانی نکنند را تامین کرد. به ویژه که یک سرور می‌تواند در شبکه حملهٔ سیبیل[۱۵] را انجام دهد. از طرف دیگر یکی از نقص‌های اساسی C-PIR آن است که به خاطر آن‌که تنها وابسته به یک سرور است، امکان تشخیص پاسخ‌های ناقص یا غیر صحیح از طرف سرور پایگاه داده وجود ندارد [۴۱]. به بیان ساده‌تر، در کاربرد فعلی سروری که قرار است اطلاعات مربوط به زنجیره بلوکی را در اختیار کاربران سبک قرار دهد، می‌تواند از انشعابی نامعتبر از زنجیره بلوکی استفاده نماید. چون گره سبک با گره‌های کامل دیگر ارتباط ندارد، نمی‌تواند متوجه این مشکل شود.

مقاله [۴۱] با استفاده از از روشی که در [۲۳] معرفی شده، از هر دوی IT-PIR و C-PIR استفاده کرده است. از این طریق به نقاط قوت هر دو روش دست پیدا کرده و تا حدی نقاط ضعف آن‌ها را برطرف کرده است. روش‌های بازیابی اطلاعات خصوصی عموما سرعت پایین و پیچیدگی محاسباتی بالا و همچنین مصرف پهنای باند بالایی دارند. در روش ارائه شده [۴۱] برای رفع این مشکل، پایگاه‌های داده در سه دسته هفتگی، ماهانه (احتمالا ۳۰ روزه) و تمام-مدت نگهداری می‌شوند. از این طریق تاخیر و پهنای باند مصرفی برای گره‌های

[15] Sybil Attack

سبکی که نیاز به دریافت و ارزیابی تراکنش‌های جدید دارند، کاهش می‌یابد. در این روش به ازای اضافه شدن هر بلوک جدید به زنجیره بلوکی، اطلاعات بلوک جدید به دسته هفتگی اضافه می‌شود. بعد از پایان یک هفته (اضافه شدن ۱۰۰۸ بلوک)، دسته هفتگی خالی شده و تمام اطلاعات آن به دسته ماهانه اضافه می‌شود. بعد از آنکه دسته ماهانه تکمیل شد (اضافه شدن ۴۳۲۰ بلوک برای ۳۰ روز) اطلاعات آن به دسته تمام-مدت اضافه می‌شود.

روش ارائه شده در [۴۱] مشکلاتی به همراه دارد، اول از همه آن‌که این روش نسبت به روش فیلتر بلوم [۲۸] به صورت قابل ملاحظه‌ای پهنای باند بیشتری مصرف می‌کند. به عنوان مثال برای آنکه یک کاربر بخواهد اطلاعات یک تراکنش را که در دسته تمام-مدت قرار دارد، دریافت کند، لازم است ۶۴٫۵۳ مگابایت پهنای باند مصرف نماید؛ در حالی که در صورتی که از روش مرسوم فیلتر بلوم استفاده نماید، لازم است که ۶۹٫۳۲ کیلوبایت پهنای باند مصرف کند. البته لازم به ذکر است که هر چه تعداد تراکنش‌های درخواستی افزایش پیدا کند و از دسته‌های جدیدتر پرسمان صورت گیرد، اختلاف پهنای باند مصرفی نسبت به روش فیلتر بلوم کمتر می‌شود. مثلا، برای دریافت ۱۰۰ تراکنش از دسته هفتگی، لازم است مجموعا ۳۳ مگابایت اطلاعات دریافت شود و در روش مرسوم فیلتر بلوم این مقدار برابر ۱۰٫۰۹ مگابایت است.

دوم، آن که برای انجام بازیابی اطلاعات خصوصی، سرور پایگاه داده برای هر جدول مربوط هر دسته یک فایل مانیفست ایجاد می‌کند. این فایل مانیفست شامل ابعاد پایگاه‌داده و موقعیت هر داده است. این فایل در اختیار کاربر قرار داده می‌شود. کاربر با توجه به این مانیفست می‌تواند پرسمان‌هایی ایجاد نماید به طوری که اطلاعاتی از او نزد سرور فاش نشود. با به‌روز شدن هر دسته، حتی با اضافه شدن هر اطلاعات جدیدی از زنجیره بلوکی به دسته هفتگی، نیاز است که فایل مانیفست مربوط به آن دسته به‌روز شود. به این ترتیب نیاز است که کاربر مانیفست جدید را دریافت کند. اندازه فایل مانیفست برای پرسمان از پایگاه داده‌ای که تنها شامل بایت تراکنش‌ها باشد و پرسمان از طریق TXID تراکنش صورت بگیرد، به این صورت است: هفتگی: ۷۲٫۴۵ مگابایت، ماهانه: ۲۱۸٫۶۸ مگابایت و تمام-مدت ۳٫۳۰ گیگابایت. البته لازم به ذکر است که نویسندگان مقاله [۴۱] می‌خواهند بعدا ساز و کاری به روش ارائه شده اضافه نمایند که کاربر سبک بدون نیاز به بارگیری فایل مانیفست، برای آنکه اطلاعات مشخصی را استخراج نماید، بتواند بدون از بین رفتن محرمانگی درخواستش، اطلاعات مورد نیازش را از مانیفست ذخیره شده در گره کامل دریافت نماید.

ایراد سوم این روش آن است که روشن است پرسمان از دسته تمام-مدت همچنان زمان‌بر است. از این رو در این مقاله پیشنهاد شده است که دسته تمام مدت به زیر دسته‌هایی تقسیم شود. پرسمان کاربر سبک از زیر دسته‌های کوچک‌تر می‌تواند برای گره کامل متخاصم حاوی اطلاعاتی باشد. مثلا با تحلیل زیردسته‌هایی که از آن‌ها پرسمان انجام شده است، و همچنین کشف ارتباط بین آدرس‌ها با توجه به تراکنش‌های بیت‌کوین، به بخشی از آدرس‌های مربوط به یک کاربر سبک پی ببرد. علاوه بر این، می‌توان به این نکته اشاره کرد که آدرس‌های یک زیر دسته قاعدتا همگی نرخ استفاده یکسانی ندارند. می‌توان فرض کرد که آدرس‌های پراستفاده‌تر احتمال پرسمان بیش‌تری از

طرف کاربر سبک مالک آن داشته باشند. از این رو احتمال پرسمان آدرس‌های یک زیر دسته برابر نیست و این اطلاعاتی جانبی برای حدس آدرس درخواست شده محسوب می‌شود [۳۸]. در [۴۱] اشاره شده است که اگر این زیردسته‌ها به اندازه کافی بزرگ باشند، مثلا به اندازه دسته ماهانه، کار را برای گره متخاصم برای یافتن الگویی در پرسمان‌های کاربر سبک سخت‌تر می‌کنند. از طرف دیگر خود تقسیم‌بندی زمانی نیز باعث می‌شود که گره کامل متخاصم بتواند با توجه به دسته‌های زمانی‌ای که کاربر از آن‌ها درخواست می‌دهد به اطلاعات جانبی از کاربر سبک دست پیدا کند.

آخرین ضعفی که می‌توان برای این روش [۴۱] نام برد، آن است که در این روش زمانی که بلوک‌های ظرفیت هر دسته تکمیل شد، مثلا برای دسته هفتگی ۱۰۰۸ بلوک، آن دسته خالی شده و مقادیر آن به دسته دیگر، مثلا ماهانه، منتقل می‌شود. این معماری می‌تواند مشکلاتی به همراه داشته باشد. مثلا، کاربرانی که تراکنش‌های مربوط به آن‌ها در بلوک‌های پایانی هفته در زنجیره بلوکی ثبت می‌شود، خیلی زود تراکنش آنها وارد دسته ماهانه می‌شود. در نتیجه لازم است برای دستیابی به اطلاعات تراکنش مربوط به خود، هر چند که مدت زمان زیادی از آن نگذشته است، از دسته ماهانه پرسمان انجام دهد و به تبع آن پهنای باند زیادی مصرف کنند. به همین ترتیب برای تراکنش‌هایی که در بلوک‌های پایانی یک ماه ثبت می‌شوند می‌توان این مشکل را متصور شد. از طرفی دیگر اگر معماری به نحوی تغییر پیدا کند که به عنوان مثال دسته هفتگی شامل ۱۰۰۸ عدد از آخرین بلوک‌هایی باشد که استخراج شده‌اند و به ازای اضافه شدن هر بلوک جدید، قدیمی‌ترین بلوک این دسته را وارد دسته ماهانه شود، باعث می‌شود که بروز رسانی دسته‌های ماهانه و به همین ترتیب دسته تمام-مدت هر ۱۰ دقیقه انجام شود که نه تنها سربار پردازشی بسیار زیادی برای گره کامل به وجود خواهد آورد، بلکه همه فایل‌های مانیفستی که مربوط به سه دسته هستند و نزد کاربر سبک است پس از ده دقیقه منقضی می‌شوند که با توجه به اندازهٔ آن‌ها، به روزرسانی مداوم آن‌ها مقرون به صرفه نخواهد بود.

۶.۳ محیط اجرای قابل اعتماد

روش BITE [۳۳] ، از یک محیط اجرای قابل اعتماد (مانند SGX[16] [۲۲]) برای حفظ حریم خصوصی کاربران سبک بهره‌گیری می‌کند. محیط اجرای قابل اعتماد SGX در گره‌های کامل قرار گرفته و وظیفه پاسخ دهی به درخواستِ تایید تراکنش از طرف کاربر سبک را دارد. SGX از نرم‌افزارهایی که در خارج از آن اجرا می‌شوند (حتی سیستم‌عامل) مجزا و منزوی است و می‌تواند یکپارچگی و محرمانگی داده‌ها را در مقابل گره کامل متخاصم دارنده آن حفظ نماید. در نتیجه قادر است در حفظ حریم خصوصی کاربران سبک و صحت (یک‌پارچگی) پاسخ

[16] Software Guard Extensions

به آن‌ها مفید باشد. به طوری که نه تنها باعث جلوگیری از فاش شدن اطلاعات گره سبک در برابر گره کامل دارنده آن می‌گردد بلکه می‌تواند گره سبک را مطمئن کند که اطلاعات دریافتی صحیح و کامل هستند. با این حال گره کامل می‌تواند با بررسی الگوی دسترسی SGX به یک حافظه خارجی، مانند پایگاه دادهٔ تراکنش‌ها، آدرس کاربر درخواست دهنده را حدس بزند. همچنین SGX نسبت به حملات کانال جانبی متعددی آسیب‌پذیر است. در مقاله [۳۳] سعی شده است با بهره‌گیری از روش بازیابی اطلاعات خصوصی و تکنیک‌های حفاظت از کانال جانبی، امنیت روش پیشنهاد شده را افزایش دهد.

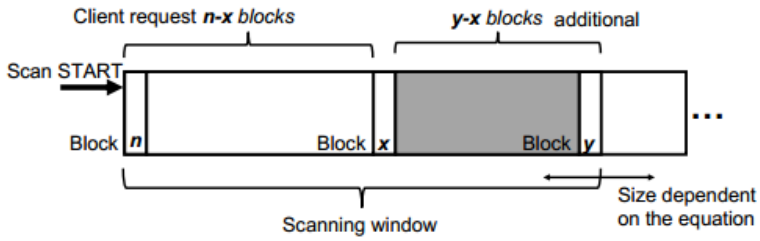
مقاله [۳۳] دو نوع راه حل ارائه داده است. راه حل اول پنجره پویش (Scanning Window) و راه حل دوم پایگاه داده ناآگاهانه (Oblivious Database) نام دارد. در هر دو روش، تصدیق از راه دور صورت می‌گیرید و یک ارتباط امن در لایه انتقال(TLS[۱۷]) مابین کاربر سبک و SGX برقرار می‌شود. کاربر سبک آدرس مورد نظرش را برای SGX می‌فرستد و SGX با توجه به زنجیرهٔ بلوکی تمام اطلاعات مورد نیاز جهت درستی سنجی وجود تراکنش در زنجیره بلوکی را بدست آورده و برای کاربر سبک درخواست دهنده می‌فرستد.

در روش پنجره پویش، برای نرمالیزه کردن رابطه بین اندازه پاسخ و اطلاعاتی که در واقع به آن‌ها دسترسی صورت گرفته است از یک روش پویش خاص استفاده می‌شود. همان‌طور که گفته شد گره کامل متخاصم می‌تواند با بررسی الگو دسترسی SGX به حافظه، آدرس(های) درخواست داده شده را حدس بزند، در این روش قرار است از حفظ حریم خصوصی کاربر سبک از طریق پنهان‌سازی الگوهای دسترسی به داده یا بلوک اطمینان حاصل شود. هدف اصلی این روش پنهان‌سازی کامل نسبت اندازه پاسخ (نشان‌دهنده تعداد تراکنش‌های بازگردانده شده به کاربر) و تعداد بلوک‌های پویش شده است. زیرا گره کامل متخاصم می‌تواند با مقایسه اندازه پاسخ تولید شده توسط گره کامل و همچنین تعداد بلوک‌های پویش شده توسط آن به بسامد تراکنش‌هایی که مربوط به آن آدرس هستند دست پیدا کند؛ در نتجیه آدرس مورد نظر گره سبک درخواست دهنده را حدس بزند.

در شکل ۲.۳ جزئیات روش پنجره پویش را، که در آن نسبت اندازه پاسخ و تعداد بلوک‌های پویش شده ثابت می‌ماند، نشان داده می‌شود. در این روش بعضا بلوک‌های بیشتری پویش می‌شوند تا نسبت اندازه پاسخ با بلوک‌های پویش شده ثابت بماند. گره کامل متخاصم تنها می‌تواند بلوک‌هایی که به آن‌ها دسترسی صورت گرفته است را شناسایی کند و چیزی درمورد آدرسی که از طرف کاربر سبک ارسال شده است و یا تراکنش‌های بازگردانده شده نمی‌داند. در این روش برای آن‌که جلوی حمله زمانی به الگوریتم گرفته شود، می‌توان اثبات مرکل را برای تمام تراکنش‌های موجود در بلوک‌های پویش شده محاسبه کرده و به محاسبهٔ اثبات مرکل، تنها برای تراکنش‌های مورد نظر کاربر درخواست دهنده، بسنده نکرد. به این ترتیب این روش بار پردازشی بسیار زیادی را متحمل خواهد شد. از طرف دیگر اگر که گره کامل متخاصم بتواند ملات کانال جانبی دیجیتال دانه‌بندی زیاد[۱۸] را اجرا نماید به

[17] Transport Layer Security
[18] High-Granularity Digital Side-Channel Attacks

طوری که بتواند مسیر اجرای برنامه را با دانه‌بندی سطح دستورات مشاهده کند، می‌تواند تراکنش‌هایی که انتخاب شده‌اند را تشخیص دهد. در این مقاله، برای مقابله با این حملات از روشی مبتنی بر [۴۲] بهره می‌گیرد که بار پردازشی الگوریتم را افزایش می‌دهد.



شکل ۲.۳: پنجره پویش. مطابق با تعداد بلوک‌های درخواست داده شده (x) و تعداد تراکنش‌های منطبق شده با درخواست مشتری در آن‌ها، احتمالا بلوک‌های بیشتری (y) از حافظه خوانده می‌شود تا نسبت بین بلوک‌های خوانده شده و اندازه پاسخ ثابت بماند[۳۳].

روش دوم ارائه شده در [۳۳] پایگاه داده ناآگاهانه نام دارد. در این روش کاربر سبک آدرس‌های مورد نظر خودش را، از طریق یک کانال محرمانه، برای SGX ارسال می‌کند و مستقیما اطلاعات مربوط به خروجی‌های خرج نشده را دریافت می‌نماید. در این روش برخلاف روش‌های پیشین و همچنین روش پنجره پویش، نیاز نیست که کاربر سبک سرایند بلوک‌ها و مسیر (اثبات) درخت مرکل را دریافت و بررسی کند. در این روش کاربر به صحت عملکرد SGX و پاسخ آن اعتماد کامل دارد. برای آنکه SGX بتواند کاربر را از صحت عملکرد خودش مطمئن سازد، تمام اقدامات و مقداردهی‌های اولیه به عنوان حالت اولیه ثبت می‌شود. با استفاده از آن کاربر می‌تواند مطمئن شود که کد صحیحی بر روی سامانه در حال اجرا است. به این فرایند تصدیق از راه دور[۱۹] گفته می‌شود. تصدیق ایجاد شده، که شامل حالت اولیه است، امضا شده و برای کاربر ارسال می‌شود. کاربر می‌تواند توسط سرویس تصدیق برخطی که توسط اینتل ارائه می‌شود [۳]، امضا را بررسی نماید.

در روش پایگاه داده ناآگاهانه، SGX اطلاعات مربوط به خروجی خرج نشدهٔ تراکنش‌ها (UTXO) را در یک پایگاه داده رمزنگاری شده نگهداری می‌کند. همچنین از ماشین دسترسی تصادفی ناآگاهانه[۲۰] معرفی شده در [۴۶] برای جلوگیری از نشت اطلاعات در هنگام دسترسی به حافظه استفاده می‌کند. به این ترتیب گره کامل متخاصم نمی‌تواند الگویی از دسترسی SGX به حافظه پیدا نماید. از طرف دیگر در این روش طول درخواست‌ها و پاسخ‌ها همواره یک مقدار ثابت است. اگر اندازه آن‌ها از آن مقدار ثابت کوتاه‌تر باشد، با لایی‌گذاری و اگر طولانی‌تر بود با تکه‌تکه کردن، به اندازه‌های ثابت تبدیل می‌شوند. در این روش SGX به تمام زنجیره بلوکی

[19] Remote Attestation
[20] Oblivious Random Access Machine (ORAM)

دسترسی ندارد و تنها دادهٔ UTXO را نگهداری کرده و به ازای اضافه شدن هر بلوک جدید، بعد از آن که آن بلوک را از جنبه اثبات کار و درخت مرکل درستی سنجی کرد، آن را به روزرسانی می‌کند. از آن‌جایی که UTXO در حافظه ORAM ذخیره می‌گردد، به روزرسانی آن امری نسبتا زمان‌بر، چیزی در حدود ۷۸٫۵ ثانیه، خواهد بود.

در دو روش ارائه شده در [۳۳] بار پردازشی چندانی بر روی گره سبک قرار نخواهد گرفت. همچنین از آن‌جایی که دیگر لازم نیست برای حفظ حریم خصوصی کاربر تراکنش‌هایی مازاد به خاطر خطای نوع دو نیز دریافت شوند، پهنای باند به طور قابل ملاحظه‌ای در این دو روش نسبت به روش فیلتر بلوم کاهش پیدا می‌کند. از طرف دیگر در روش دوم (پایگاه داده ناآگاهانه) نیاز نیست که پاسخ گره کامل با اثبات‌های مرکل همراه باشد و به عبارتی گره سبک به عملکرد صحیح SGX اعتماد دارد. در نتیجه در این روش پهنای باند مصرفی بسیار کاهش پیدا می‌کند. علاوه بر مزایای ذکر شده، این روش ایراداتی نیز دارد که در ادامه به بیان آن خواهیم پرداخت.

اول از همه آنکه زمان تولید جواب در روش پنجره پویش، در صورتی که اقدامت مورد نیاز جهت جبران حمله کانال جانبی انجام شود، بسیار زمان‌بر است. به عنوان مثال برای پردازش ۱۰۰ بلوک در این روش چیزی در حدود ۷۳ ثانیه زمان نیاز است. این زمان برای روش فیلتر بلوم با نرخ خطای نوع دوی ۰٫۵ درصد، حدود ۱٫۱ ثانیه است [۳۳]. هر چند که تولید پاسخ در روش پایگاه داده ناآگاهانه بسیار سریع‌تر انجام می‌شود، اما برای به روز رسانی داده خروجی خرج نشده تراکنش‌ها نیاز به ۷۸٫۵ ثانیه زمان دارد. به عبارتی می‌توان اینطور گفت که هر ده دقیقه یک‌بار (زمان مورد نیاز برای استخراج یک بلوک جدید)، حدود یک دقیقه و هجده ثانیه، صرف به روز رسانی شده و امکان پاسخ‌گویی به کاربران سبک را ندارد. مقاله [۳۳] برای افزایش دسترس‌پذیری سیستم در شرایط به روز رسانی، پیشنهاد استفاده از دو سیستم موازی را داده است. در این شرایط نیز، سیستم ارائه دهنده خدمات از وضعیت فعلی شبکه حداکثر حدود ۷۸٫۵ ثانیه عقب‌تر خواهد بود.

مشکل دیگری که روش [۳۳] دارد، حملات فیزیکی کانال جانبی مدرنی است که SGX نسبت به آن‌ها آسیب‌پذیر است. مثلا حملات اسپکتر[۲۱] [۳۰]، ملت‌داون[۲۲] [۳۱] و حمله [۲۰] که به تازگی کشف شده است، می‌توانند برای استخراج کلیدهای تصدیق از SGX مورد استفاده قرار گیرند. در صورتی که گره کامل متخاصم از چنین حمله‌ای بهره‌بردای کند، می‌تواند در روش پنجره پویش، حریم خصوصی کاربران سبک درخواست دهنده را نقض نماید؛ همچنین در روش پایگاه دادهٔ ناآگاهانه علاوه بر نقض حریم خصوصی کاربر سبک می‌تواند اطلاعات اشتباهی را در اختیار وی قرار دهد.

علاوه بر مشکلات ذکر شده در بالا، می‌توان به این مسئله نیز اشاره نمود که برای آنکه یک گره کامل بخواهد خدمات پیشنهاد شده در [۳۳] را به گره‌های سبک ارائه دهد، نه تنها نیاز است که یک محیط اجرای قابل اطمینان تهیه و راه‌اندازی نماید، بلکه لازم است که منابع پردازشی قابل توجهی را برای این منظور اختصاص دهد. در نتیجه

[21] Spectre
[22] Meltdown

گره‌های کاملی که بتوانند چنین خدماتی ارائه دهند، محدود خواهند بود. به تبع آن کاربران سبک مجبور خواهند بود که بین گره‌های کامل محدودتری انتخاب کنند که این مسئله انگیزه این گره‌های کامل را برای انجام اقدامات خصمانه بیشتر خواهد کرد. از این اقدامات می‌توان به ایجاد و دنبال کردن یک انشعاب ناصحیح از زنجیره بلوکی بیت‌کوین اشاره نمود. در حالت عادی که تعداد گره‌های کامل زیاد هستند، گره سبک می‌تواند با دریافت خدمات از گره‌های کامل متعدد از صحت اطلاعات دریافتی مطمئن گردد.

از طرف دیگر، شرکت‌های محدودی مانند اینتل، تجهیزات مربوط به یک محیط اجرای قابل اطمینان را تولید و به فروش می‌رسانند. همچنین نیاز است که برای تصدیق از راه دور عملکرد آن‌ها به سرویس‌هایی مثل [۳] وابسته بود. به بیان دیگر می‌توان این طور گفت که برای آنکه بتوان از روش [۳۳] بهره‌برداری کرد، لازم است به شرکت‌های محدودی اعتماد شود که این خود بر خلاف ذات شبکه‌های همتابه‌همتایی مثل بیت کوین است.

۷.۳ مقایسه

در این فصل، ضمن آشنایی با ساز و کار فعلی شبکهٔ همتابه‌همتای بیت‌کوین در ارسال اطلاعات مربوط به تراکنش‌های یک گره سبک با هدف حفظ حریم خصوصی وی [۲۸]، توضیح داده شد که روش فعلی بسیار آسیب‌پذیر است و در صورتی که کاربر سبک خود اقداماتی را جهت حفظ بیشتر حریم خصوصیش انجام ندهد، عملا حریم خصوصی وی اصلا حفظ نمی‌شود.

علاوه بر این، در این فصل به بیان مفصل راه‌حل‌هایی که تا کنون برای رفع این مشکل ارائه شده‌اند، پرداخته شده است. در این قسمت این راه‌حل‌ها از سه جنبهٔ امنیت، پهنای باند مصرفی، بار پردازشی سمت گره کامل با هم مقایسه می‌شوند. که نتیجهٔ این مقایسه در جدول‌های زیر آورده شده است.

جدول ۱.۳: مقایسهٔ امنیت روش‌های بحث شده.

روش	آسیب‌پذیری‌ها
فیلتر بلوم [۲۸]	نرخ خطای نوع دوی عملا خیلی پایین، دسترسی به چند فیلتر بلوم از یک کاربر، کشف اولین استفاده از آدرس، تحلیل گراف تراکنش‌ها و کشف آدرس‌های مرتبط، تحلیل بسامد استفاده از آدرس، تحلیل زمان درخواست

اصلاح رفتار گره سبک [۲۵]	کشف اولین استفاده از آدرس، تحلیل گراف تراکنش‌ها و کشف آدرس‌های مرتبط، تحلیل بسامد استفاده از آدرس، تحلیل زمان درخواست
معیار حاشاپذیری-γ [۲۹]	دسترسی به چند فیلتر بلوم از یک کاربر، کشف اولین استفاده از آدرس، تحلیل گراف تراکنش‌ها و کشف آدرس‌های مرتبط، تحلیل بسامد استفاده از آدرس، تحلیل زمان درخواست
فیلتر بلوک [۴۰]	تحلیل گراف تراکنش‌ها و کشف آدرس‌های مرتبط، تحلیل بسامد استفاده از آدرس، تحلیل زمان درخواست
بازیابی اطلاعات خصوصی [۴۱]	تبانی گره‌های کامل، تحلیل زیردسته‌های دستهٔ تمام-مدت مورد پرسمان واقع شده، تحلیل بسامد استفاده در زیردسته‌ها، سخت بودن راه‌اندازی یک گره کامل در نتیجه نیاز به اعتماد به گره‌های اندک موجود
پنجرهٔ پویش (SGX) [۳۳]	اعتماد به سازنده‌های سخت‌افزار محیط‌های قابل اعتماد، افشای اطلاعات در صورت حملات کانال جانبی، سخت بودن راه‌اندازی یک گره کامل در نتیجه نیاز به اعتماد به گره‌های اندک موجود
پایگاه دادهٔ ناآگاهانه (SGX) [۳۳]	اعتماد به سازنده‌های سخت‌افزار محیط‌های قابل اعتماد، افشای اطلاعات در صورت حملات کانال جانبی، ارسال اطلاعات نادرست در صورت حملات کانال جانبی، سخت بودن راه‌اندازی یک گره کامل در نتیجه نیاز به اعتماد به گره‌های اندک موجود

جدول ۲.۳: مقایسهٔ پهنای باند مصرفی در روش‌های بحث شده.

روش	**پهنای باند**
فیلتر بلوم [۲۸]	به خاطر به‌روز رسانی فیلتر توسط گره کامل، از خیلی کم به زیاد تغییر می‌کند. شامل تراکنش‌ها و اثبات مرکل
اصلاح رفتار گره سبک [۲۵]	متوسط - نرخ خطای نوع دو بالاتر از روش [۲۸]. شامل تراکنش‌ها و اثبات مرکل.

معیار حاشاپذیری-γ [۲۹]	کم
فیلتر بلوک [۴۰]	برای گره‌های مختلف با تعداد تراکنش‌های مختلف متفاوت است.
بازیابی اطلاعات خصوصی [۴۱]	خیلی زیاد. اندازهٔ فایل مانیفست تمام-مدت ۳۰٫۳ گیگابایت
پنجرهٔ پویش (SGX) [۳۳]	خیلی کم. شامل تراکنش‌های مرتبط و اثبات مرکل. بدون خطای نوع دو.
پایگاه دادهٔ ناآگاهانه (SGX) [۳۳]	ناچیز. شامل تراکنش‌های مرتبط بدون نیاز به اثبات مرکل و بدون خطای نوع دو.

جدول ۳.۳: مقایسهٔ پردازش سمت گره کامل در روش‌های بحث شده.

روش	**پهنای باند**
فیلتر بلوم [۲۸]	زیاد - برای هر فیلتر بلوم باید چیکدهٔ تمام داده‌های تراکنش‌های یک بلوک k بار حساب شود.
اصلاح رفتار گره سبک [۲۵]	زیاد - برای هر فیلتر بلوم باید چیکدهٔ تمام داده‌های تراکنش‌های یک بلوک k بار حساب شود.
معیار حاشاپذیری-γ [۲۹]	زیاد - برای هر فیلتر بلوم باید چیکدهٔ تمام داده‌های تراکنش‌های یک بلوک k بار حساب شود.
فیلتر بلوک [۴۰]	کم - فقط یک بار باید چکیدهٔ تمام داده‌های تراکنش‌های یک بلوک حساب شده و برای فشرده‌سازی، کدگذاری شوند.
بازیابی اطلاعات خصوصی [۴۱]	زیاد. به ازای استخراج یک بلوک جدید باید دستهٔ هفتگی به روز رسانی شود.
پنجرهٔ پویش (SGX) [۳۳]	خیلی خیلی زیاد. به ازای درخواست هر کاربر سبک باید تمام اثبات‌های مرکل تمام تراکنش‌های چند بلوک را حساب کند. تعداد بلوک‌های محاسبه شده با توجه به درخواست کاربر تعیین می‌شود.
پایگاه دادهٔ ناآگاهانه (SGX) [۳۳]	زیاد. به ازای استخراج هر بلوک جدید، باید پایگاه‌داده ناآگاهانه را به روزرسانی کند.

فصل ۴

ارائهٔ روش

۱.۴ مقدمه

روش گمنامی-k اولین بار در مقاله [۴۸] معرفی شده است. با این روش می‌توان اطلاعات مرتبط با افرادی را که می‌خواهیم گم‌نامی آن‌ها حفظ شود، منتشر نمود. از این روش در سرویس‌های مبتنی بر مکان نیز استفاده می‌شود [۳۸] . در این سرویس‌ها عموما لازم است کاربر برای دریافت خدمات مرتبط با موقعیت جغرافیایی فعلی خود، اطلاعات مکانی خود را در اختیار ارائه دهنده این خدمات قرار دهد. در [۳۸] شرح داده شده است که در صورتی که کاربر اطلاعات مکانی خود را با اطلاعاتی مصنوعی که نشان‌دهنده مکان‌های دیگری هستند ترکیب کند و مجموعه آن‌ها را به سرویس‌دهنده ارسال نماید، می‌تواند تا حدی اطلاعات مکانی خود را پنهان کند. مقاله [۳۸] با استفاده از معیار آنتروپی توضیح داده که در صورتی که کاربر اطلاعات مصنوعی را به نحوی انتخاب نماید که نرخ (احتمال) پرسمان اطلاعات مربوط به آن مکان‌ها با نرخ پرسمان اطلاعات مکانی خود یکسان باشد، آنتروپی تشخیص مکان کاربر توسط سرویس‌دهنده بیشتر می‌شود. به بیان دیگر سرویس‌دهنده در مورد آن که کدام یکی از موقعیت‌های درخواست داده شده حقیقی و مربوط به کاربر هستند با ابهام بیشتری مواجه خواهد شد.

در این فصل سعی داریم برخلاف روش مبتنی بر فیلتر بلوم که آدرس‌های مصنوعی به صورت کورکورانه و تصادفی اتخاذ می‌شدند، آدرس‌های مصنوعی به نحوی اتخاذ شوند که دارای نرخ پرسمان تقریبا یکسانی با آدرس کاربر درخواست دهنده باشد. در ابتدا تعاریف ریاضی مسئله را بیان کرده و سپس مروری می‌کنیم بر ویژگی‌هایی که لازم است پروتکل ارائه شده از آن‌ها برخوردار باشد.

۲.۴ تعریف‌های ریاضی

در این مقاله کاربر سبک iام را با l_i نمایش می‌دهیم. مجموعه تمام گره‌های سبک در شبکه به صورت $LW = \{l_1, ..., l_M\}$ است که M تعداد کل این گره‌ها است. گره کامل jام به صورت f_j و مجموعه تمام گره‌های کامل به صورت $FN = \{f_1, ..., f_W\}$ و تعداد کل گره‌های کامل با W نمایش داده می‌شود. تعداد کل آدرس‌های بیت‌کوین N است و $A = \{a_1, ..., a_N\}$ مجموعه تمام آدرس‌ها است. هر آدرس شبکه متعلق به یک گره کامل یا یک گره سبک است. همچنین هر گره می‌تواند مالک چند آدرس باشد. آدرس‌های مربوط به گره سبک l_i، که مالک C_{l_i} عدد آدرس است، مجموعه $A_{l_i} = \{a_{l_{i1}}, ..., a_{l_{iC_{l_i}}}\}$ را تشکیل می‌دهد. به همین ترتیب آدرس‌های مربوط به هر کدام از گره‌های کامل تعریف می‌شوند.

کاربران سبک اطلاعات جدید مربوط به آدرس‌هایشان را از گره‌های کامل پرسمان می‌کنند. متغیر تصادفی $X_{a_{nj}}$ به این ترتیب تعریف می‌شود که گره کامل f_j آخرین درخواستی که دریافت می‌کند مربوط به آدرس a_n باشد. با توجه به قانون اعداد بزرگ در احتمال، امید ریاضی $X_{a_{nj}}$ برابر با تعداد دفعات درخواست‌های مربوط به آدرس a_n به تعداد کل درخواست‌هایی است که f_j ، به طوری که $(1 < j < W)$ ، دریافت کرده است.

$$E\{X_{a_{nj}}\} = \frac{Q_{a_{nj}}}{Q_{Tj}} \tag{۱.۴}$$

در معامله (۱.۴)، $E\{X_{a_{nj}}\}$ امید ریاضی پرسمان آدرس a_n از گره کامل f_j است و $Q_{a_{nj}}$ و Q_{Tj} به ترتیب تعداد دفعات پرسمان آدرس a_n و تعداد دفعات کل پرسمان‌ها از گره کامل f_j هستند. همچنین می‌توان با استفاده از قانون اعداد بزرگ بورل، احتمال رخداد پیشامد پرسمان a_n در آخرین درخواست انجام شده از گره کامل f_j را برابر امید ریاضی $E\{X_{a_{nj}}\}$ قرار داد. با توجه به اینکه کاربران سبک در هر نوبت پرسمان به صورت تصادفی یک گره کامل را انتخاب می‌کنند، می‌توان فرض کرد که احتمال پرسمان آدرس‌ها در تمام گره‌های کامل با هم برابر است.

$$Pr\{a_n\} = Pr\{a_{nj}\} = E\{X_{a_{nj}}\}, \quad 0 < j < M \tag{۲.۴}$$

در معادله (۲.۴)، $Pr\{a_{nj}\}$ احتمال پرسمان آدرس a_n از گره کامل f_j است. لازم به ذکر است که احتمال درخواست آدرس‌های مربوط به هر گره کاملی برابر صفر است. چرا که این گره‌ها خودشان وضعیت کامل زنجیره

بلوکی را ذخیره کرده‌اند و از گره کامل دیگری در مورد اطلاعات مربوط به آدرس‌هایشان پرسمان انجام نمی‌دهند.

$$Pr\{a_n\} = \begin{cases} ۰, & \text{if } a_n \in A_{f_j},\ \forall_j : ۱ < j < W \\ Pr\{a_{l_{ic}}\}, & \text{if } a_n = a_{l_{ic}} \in A_{l_i},\ \forall_{i,c} : ۱ < i < M,\ ۱ < c < C_{l_i} \end{cases} \quad (۳.۴)$$

۳.۴ ملزومات پروتکل

پیش از پرداختن به ساختار طراحی پروتکل لازم است ویژگی‌هایی که پروتکل باید از آن‌ها برخوردار باشد بررسی شوند. با توجه به هدف پروتکل، که حفظ حریم خصوصی کاربران دارای گره‌های سبک است، لازم است که به مصون ماندن پروتکل نسبت به حملاتی که حریم خصوصی افراد را نقض می‌کنند، توجه ویژه داشت. از طرف دیگر از آن‌جا که کاربران دارای گره‌های سبک امکان پردازش و ذخیره‌سازی حجم بالای اطلاعات را ندارند، همچنین پهنای باند آن‌ها محدود و پر هزینه است، لازم است پروتکل طراحی شده، کمترین میزان بار محاسباتی، مصرف حافظه و پهنای باند را در سمت کاربران سبک داشته باشد. لازم به ذکر است، در صورت وجود فرایندهای پیچیده در پروتکل، آسیب‌پذیری‌ها و حفره‌های امنیتی زیادی در پروتکل و پیاده‌سازی‌های آن وجود خواهد داشت. از این رو در این پروتکل سعی شده است که تا جای ممکن از فرایندهای ساده‌ای که پیش از این در کاربردهای مختلف آزموده شده‌اند استفاده گردد.

علاوه بر این نباید تغییرات عمده‌ای در ساز و کار گره‌های کامل اعمال کرد و آن‌ها را ملزم به استفاده از ابزارهایی سخت‌افزاری، غیر از سخت افزار مورد نیاز یک گره کامل در شرایط فعلی، نمود. همچنین نباید از نرم‌افزارهایی با مالکیت اختصاصی و متن بسته استفاده شود. این دو کار نه تنها به خاطر دشوار کردن و هزینه‌بر کردن راه‌اندازی یک گره‌کامل، تعداد آن‌ها را در شبکه کمتر می‌کند و منجر به متمرکز شدن شبکه می‌شود، بلکه پروتکل بیت‌کوین را که یک پروتکل بی‌نیاز به اعتماد به یک طرف سوم است، ملزم به اعتماد به شرکت‌های تولید سخت‌افزار و نرم‌افزار به خصوصی می‌کند که وجود درهای پشتی در محصولات آن‌ها اجتناب ناپذیر خواهد بود.

در ادامهٔ این بخش محدودیت‌ها و ضوابطی را برای این پروتکل بیان می‌کنیم تا حریم خصوصی کاربر حفظ شده و همچنین بار محاسباتی، مصرف حافظه و پهنای باند چندانی به طرفین پروتکل اضافه نشود.

نخست، برای آنکه گره کامل $f_j \in FN$ نتواند آدرس $a_{l_{ic}} \in A_{l_i}$ مربوط به گره سبک درخواست دهنده l_i را تشخیص دهد، گره سبک l_i باید اطلاعات مربوط به آدرس خود را به همراه اطلاعت مربوط به $k - ۱$ عدد از آدرس‌های مصنوعی را همزمان درخواست دهد، به طوری که $a_n \in A \backslash \{a_{l_{ic}}\}$. اگر آدرس‌های مصنوعی به نحوی انتخاب شوند که نرخ پرسمان آن‌ها، در زمان درخواست، کمتر از آدرس اصلی باشند، یا اصلا وجود

نداشته باشند ($a_n \notin A$)، گره کامل می‌تواند با احتمال بالاتری آدرس درخواست دهنده را در میان آدرس‌های مصنوعی درخواست داده شده حدس بزند. از این رو باید کاربر سبک آدرس‌های مصنوعی را به نحوی اتخاذ نماید که احتمال درخواست تقریبا برابری با آدرس حقیقی خود داشته باشند، تا گم‌نامی کاربر سبک درخواست دهنده بیشتر حفظ شود.

دوم، واضح است که محاسبه و پیدا کردن حداقل $k-1$ آدرسی که دارای احتمالی برابر با آدرس کاربر درخواست دهنده باشند، بدون دسترسی به اطلاعات تراکنش‌ها و تناوب درخواست آن‌ها از گره‌های کامل، امری غیر ممکن است. از طرف دیگر اگر کاربر سبک بخواهد این اطلاعات را به طور مستقیم از گره کامل، که دارای تمام این اطلاعات است، دریافت نماید، مشکلاتی اساسی پدید خواهد آمد. در سناریوی پیش رو به دو مورد از آن‌ها اشاره خواهیم نمود.

فرض کنید، l_i در مرحله اول احتمال درخواست آدرس cام خود، یعنی $a_{l_{ic}}$، را حساب نماید. بعدا توضیح داده خواهد شد که این محاسبه بدون نیاز به افشای آدرس به شخص سومی و صرفا بر اساس سوابق کیف پول کاربر قابل انجام است. سپس، احتمال به دست آمده را (بدون ذکر $a_{l_{ic}}$) به گره کامل f_{j_1} ارسال کرده و درخواست $k-1$ آدرس با احتمال پرسمان برابر $Pr\{a_{l_{ic}}\}$ بدهد. در مرحله دوم، کاربر آدرس خودش را در میان آدرس‌های مصنوعی هم احتمال قرار داده و اطلاعات مربوط به مجموعه k آدرس حاصل را از یک گره کامل دیگر f_{j_2} یا همان گره کامل پیشین درخواست نماید. در این سناریو اگر کاربر اطلاعات مجموعه آدرس‌ها را از همان f_{j_1} درخواست نماید، f_{j_1} با توجه به سوابق آدرس‌هایی که قبل‌تر ارسال کرده بوده، متوجه آدرس کاربر، که آدرس جدیدیست و در سوابق اخیرش موجود نیست، می‌شود و آدرس کاربر نزد گره کامل فاش می‌گردد.

کاربر برای حفظ گم‌نامیش می‌تواند از گره دیگری، مثل f_{j_2} اطلاعات مربوط به مجموعه آدرس‌های هم احتمال را درخواست نماید. در این حالت نیز، در صورتی که f_{j_1} با f_{j_2} تبانی نموده و سوابقشان را با هم به اشتراک بگذارند، طبق روندی که پیش‌تر گفته شد، آدرس فرد درخواست دهنده قابل تشخیص خواهد بود. برای رفع این مشکل کاربر می‌تواند در مرحله اول آدرس‌های هم احتمال را از چند گره متفاوت دریافت نماید و در مرحله دوم زیرمجموعه‌ای تصادفی از تمام آن‌ها را انتخاب و حاصل را به چند بخش تقسیم کرده و هر قسمت را از گره‌ای مجزا درخواست نماید. در این حالت احتمال تشخیص آدرس توسط گره‌هایی که تبانی کرده‌اند کاهش می‌یابد. با این حال، در این روش کاربر سبک باید در صحت هم احتمال بودن آدرس‌های دریافتی به گره(ها)ی کامل اعتماد نماید چرا که تصدیق و صحت‌سنجی آن‌ها بدون دسترسی به اصل داده‌ها امکان‌پذیر نیست. به عنوان مثال، در مرحله اول یک گره کامل می‌تواند با ارسال آدرس‌هایی که نرخ پرسمان یکسان، اما متفاوت با نرخ پرسمان آدرس درخواست داده‌شده، داشته باشند، منجر به افشای آدرس کاربر گردد.

از این حیث، لازم است که پروتکل طراحی شده به کاربر سبک اجازه دهد بدون نیاز به فاش کردن اطلاعاتش و همچنین اعتماد به اطلاعات ارسال شده از یک طرف سوم، آدرس‌هایی هم احتمال با آدرس خودش را اتخاذ

نموده و مجموعه آن‌ها را از گره کامل پرسمان نماید.

سوم، باید این امر مهم را در نظر بگیریم که کاربران سبک بسیار زیادی هستند که برای به روز رسانی اطلاعاتشان، از طریقی غیر از شبکه‌های حافظ گم‌نامی (مانند تور[۱] [۴]، کراودز[۲] [۴۳] و غیره)، با گره‌های کامل تبادل اطلاعات انجام می‌دهند. در این صورت مبداء ارسال پرسمان‌های یک کاربر سبک با تقریب خوبی یکسان خواهد ماند. گره کامل متخاصم می‌تواند سابقه‌ای را از پرسمان‌های یک کاربر سبک l_i تشکیل دهد. در این صورت اگر l_i آدرس‌های خودش را، یعنی $a_{l_{ic}}$ به طوری که $۱ < c < C_{l_i}$، مدام در مجموعه‌ای متفاوت از آدرس‌های هم احتمال قرار دهد، گره متخاصم می‌تواند با اشتراک گیری بین سوابق پرسمان کاربر سبک، به مجموعه‌ای محدودتر از آدرس‌هایی دست پیدا کند که در تمام یا اکثر پرسمانهای آن کاربر وجود داشته‌اند. در نتیجه با احتمال بیشتری می‌تواند آدرس کاربر را تشخیص دهد. برای حفظ گم‌نامی کاربر سبک در برابر این حمله، کاربر سبک باید از مجموعه هم احتمال و تا حد امکان ثابتی استفاده نماید.

در این بخش ملزوماتی برای پروتکل طراحی شده مطرح شدند که علاوه بر آنکه این پروتکل اثر نامطلوبی بر توزیع‌شدگی و امنیت بیت‌کوین نداشته باشد، در برابر حملاتی که ممکن است منجر به فاش شدن آدرس‌های مربوط به یک کاربر سبک گردد مقاوم باشد.

۴.۴ ساختار پروتکل

در این قسمت به توصیف پروتکل پرداخته می‌شود. پروتکل ارائه شده به دو بخش تقسیم می‌گردد. بخش اول شامل فرایندی است که در گره‌های کامل انجام می‌گردد. در این بخش، احتمال پرس‌وجوی تمام آدرس‌های بیت‌کوین (A) محاسبه شده و به آدرس‌هایی با احتمال نزدیک به هم تقسیم می‌شوند. به هرکدام از این قسمت‌ها، تکه (Chunk) گفته می‌شود. بخش دوم شامل فرایندی است که گره سبک به صورت آفلاین و بدون نیاز به طرف سومی انجام می‌دهد تا بفهمد که آدرس مربوط به آن در کدام تکه قرار دارد.

در طراحی این پروتکل فرض می‌کنیم که احتمال پرسمان اطلاعات مربوط به هر آدرس a_n، به شرطی که مربوط به یک گره کامل نباشد، متناسب است با احتمال استفاده از آن آدرس در شبکه بیت‌کوین. یعنی فرض شده است که هرچه یک کاربر سبک از یک آدرس بیشتر استفاده نماید، بیشتر تمایل دارد اطلاعاتش را در مورد آن

[1] Tor
[2] Crowds

آدرس از طریق پرسمان از گره‌های کامل به روزرسانی نماید. به بیان دیگر می‌توانیم بنویسیم:

$$Pr\{a_n\} \propto p(a_n) \triangleq \frac{NT_{a_n}}{\sum_{m=1}^{N} NT_{a_m}}; \forall i \; a_n \in A_{l_i} \qquad (۴.۴)$$

در معادله (۴.۴)، NT_{a_n} تعداد تراکنش‌هایی هستند که در آن‌ها از آدرس a_n به عنوان ورودی یا خروجی استفاده گردیده و در زنجیره بلوکی بیت‌کوین ثبت شده است. $p(a_n)$ احتمال استفاده از آدرس a_n در شبکه بیت‌کوین تعریف می‌شود. با استفاده از تعریف (۴.۴) می‌توانیم به تعریفی قابل اجماع از احتمال پرسمان یک آدرس دست پیدا کنیم. دستیابی به این تعریف با توجه به توزیع شدگی و شفافیت و همچنین یکتا بودن وضعیت زنجیره بلوکی در میان تمام گره‌های شبکه قابل انجام است.

۱.۴.۴ محاسبه مستقل از دیگر آدرس‌ها

کاربران سبک باید بتوانند بدون نیاز به هر درخواست اطلاعات اضافه‌ای از گره کامل، محاسبه نمایند که آدرس‌های آن‌ها در کدام تکه قرار گرفته است. فرایند انجام این محاسبه به طور کامل در بخش ۳.۴.۴ توضیح داده شده است. در این بخش می‌خواهیم پروتکل را به نحوی طراحی کنیم که گره‌های سبک بدون نیاز به پرسیدن چیزی از گره دیگری، بتوانند تکه مربوط به خود را مشخص نماید. در گام اول لازم است به این موضوع اشاره شود که گره‌های سبک امکان محاسبه NT_{a_n} مربوط به آدرس خود را به صورت آفلاین و با توجه به سوابق تراکنش‌هایشان دارند. به خاطر یکسان بودن $\sum_{m=1}^{N} NT_{a_m}$ نیازی نیست که کاربران سبک از آن اطلاع داشته باشند.

مسئله دیگری که لازم است مورد توجه قرار گیرد، ارائه روشی برای محاسبه آدرس‌های هر تکه به صورتی است که نه تنها با تغییر عمده نرخ پرسمان یک آدرس، آدرس در تکه‌ای جدا و متناسب با نرخ جدید قرار بگیرد و در تکه قبلی نماند، بلکه با توجه به آنچه در بخش ۳.۴ بحث شده بود، لازم است که تکه‌ها نسبت که تغییرات اندک نرخ پرسمان آدرس‌ها مقاوم باشند. تغییر خیلی کند تکه‌ها باعث می‌شود که آنتروپی هر تکه کاسته شود و از طرف دیگر تغییر سریع تکه‌ها باعث می‌شود که گره کامل با اشتراک گیری درخواست‌هایی که از یک منبع ثابت ارسال می‌شوند، به تعداد محدودتری از آدرس‌های محتمل برای گره سبک درخواست دهنده دست یابد.

برای رسیدن به این هدف ابتدا تعریفی از امتیاز هر آدرس در زمان t_0 به صورت $s_{a_n}^{t_0}$ تعریف می‌کنیم. مجموعه تمام این امتیازها در یک زمان خاص، حالت سیستم در آن زمان نامیده می‌گردد.

$$\forall \, a_n \text{ in } A \text{ at } t_0 : S^{t_0} = [s_{a_1}^{t_0}, ..., s_{a_n}^{t_0}]^T \qquad (۵.۴)$$

که S^{t_0} حالت سیستم در پنجره زمانی t_0 است. پنجره زمانی با W مشخص شده و به این صورت تعریف می‌شود: پارامتر W برابر با تعداد بلوک‌هایی است که نشان‌دهنده یک واحد زمانی هستند. بعد از استخراج W بلوک و ثبت آن در زنجیره بلوکی بیت‌کوین، حالت سیستم با توجه به W بلوک اخیر استخراج شده به روزرسانی می‌گردد. به این ترتیب امتیاز آدرس a_n در پنجره زمانی t_0 (W^{t_0})به صورت $s_{a_n}^{t_0}$ نمایش داده شده و به صورت معادله (۶.۴) تعریف می‌شود.

$$s_{a_n}^{t_0} = \beta NT_{a_n}^{t_0} + (1-\beta)s_{a_n}^{t-1} \tag{۶.۴}$$

که $0 < \beta < 1$ و $NT_{a_n}^{t_0}$ تعداد تراکنش‌هایی است که شامل آدرس a_n در ورودی یا خروجیشان بوده‌اند و در بلوک‌های موجود در پنجره زمانی W^{t_0} ظاهر شده‌اند است. به این ترتیب حالت کلی سیستم (S^{t_0}) به صورت معادله (۷.۴) به روزرسانی می‌شود.

$$S^{t_0} = \beta NT_A^{t_0} + (1-\beta)S^{t-1} \tag{۷.۴}$$

که $NT_A^{t_0} = [NT_{a_1}^{t_0}, ..., NT_{a_N}^{t_0}]^T$ است. به این ترتیب توانستیم امتیاز هر کدام از آدرس‌ها را مستقل از آدرس‌های دیگر محاسبه نموده و با تنظیم پارامتر β و اندازه W سرعت تغییرات تکه‌ها را تنظیم نماییم. اندازه پنجره‌های زمانی نیز بر اساس تعداد بلوک‌های استخراج شده تعیین می‌شوند. به خاطر آنکه زمان بین استخراج دو بلوک تقریبا زمان ثابتی و برابر با ۱۰ دقیقه فرض می‌شود، زمان به روزرسانی امتیاز هر کدام از آدرس‌ها تقریبا ثابت خواهد بود. به عنوان مثال اگر اندازه پنجره زمانی ۱۴۴ بلوک در نظر گرفته شود، تقریبا هر ۲۴ ساعت یک بار امتیاز آدرس‌ها به روز رسانی می‌گردد.

۲.۴.۴ تعیین و انتشار تکه‌ها

در قسمت قبل، برای هر آدرس امتیازی در نظر گرفته شد و توضیح داده شد که چطور در هر پنجره زمانی وضعیت امتیاز تمام آدرس‌ها به روز رسانی می‌گردد. در این بخش با توجه به چگونگی توزیع امتیاز آدرس‌ها، که در شکل ۱.۴ قابل مشاهده است، مرز بین تکه‌ها و تعداد اعضا هر تکه را به نحوی انتخاب می‌کنیم که نه تنها باعث کاهش امنیت و گم‌نامی گره‌های سبک نگردد، بلکه پهنای باند مصرف شده جهت پرسمان تمام آدرس‌های یک تکه مقرون به صرفه باشد.

با توجه به معادله (۳.۴) احتمال پرسمان آدرس a_n به شرطی که برای یک گره کامل نباشد برابر $Pr\{a_n\}$

و مخالف صفر است. اگر گره سبک l_i بخواهد اطلاعات مربوط به آدرس $a_{l_{ic}}$ را که مربوط به خودش است درخواست نماید، لازم است اطلاعات مربوط به آدرس‌های هم احتمال با خودش را که یک تکه را تشکیل می‌دهند (مثلا تکه شماره γ) دریافت کند. تعداد اعضای این تکه را با K_γ نمایش می‌دهیم. از منظر گره کامل هر کدام از K_γ آدرس موجود در تکه درخواست داده شده ممکن است برای درخواست دهنده ($a_{l_{ic}}$) باشد. احتمال آنکه هر کدام از آدرس‌های تکه، آدرس مورد نظر باشد را با q_k نشان می‌دهیم که $k = (۱, ۲, ..., K_\gamma)$. به این ترتیب q_k به صورت زیر تعریف می‌شود.

$$q_k = \frac{Pr\{a_{\gamma k}\}}{\sum_{m=۱}^{K_\gamma} Pr\{a_{\gamma m}\}}; \quad \sum_{k=۱}^{K_\gamma} q_k = ۱ \tag{۸.۴}$$

که $Pr\{a_{\gamma m}\}$ احتمال پرسمان آدرس kام مربوط به تکه γ است. به این ترتیب آنتروپی تشخیص آدرس مورد نظر از میان آدرس‌های تکه γ به صورت معادله (۹.۴) قابل محاسبه خواهد بود:

$$H = -\sum_{k=۱}^{K_\gamma} q_k . \log_۲ q_k \tag{۹.۴}$$

که هر چه آنتروپی (H) بزرگ‌تر باشد به معنی حفظ بیشتر گم‌نامی آدرس مورد نظر است است. زمانی این مقدار ماکزیمم است که تمام K_γ عضو تکه، احتمالی برابر داشته باشند. این مقدار ماکزیمم برابر $H_{max} = \log_۲ K_\gamma$ است.

با فرض اینکه احتمال پرسمان یک آدرس مربوط به گره سبک متناسب است با احتمال قرارگیری این آدرس در تراکنش‌های موجود در زنجیره بلوکی بیت‌کوین (معادله (۴.۴)) می‌توانیم برای تعیین تکه‌ها و ساخت آن‌ها با توجه به امتیاز آدرس‌ها (مطابق فرمول (۵.۴)) عمل نماییم. جهت نحوه مرزبندی تکه‌ها لازم است به نکات زیر توجه شود:

۱. هرچه مقدار K_γ برای تکه γ بیش‌تر باشد، آنتروپی آن بیشتر خواهد بود. بالا رفتن K_γ باعث می‌شود که کاربر سبک به ازای هر درخواست، اطلاعات اضافه بیشتری را دریافت نماید. در نتیجه پهنای باند و ترافیک زیادی مصرف نماید.

۲. با توجه به شکل ۱.۴ مشاهده می‌شود که اکثر آدرس‌های استفاده شده در بیت‌کوین دارای امتیازی کمتر از یک هستند و آدرس‌هایی که امتیاز آن‌ها بیشتر از یک است بسیار کم و پراکنده هستند. از این رو اگر

مرز بین تکه‌ها به صورت توان‌های دو $(\ldots, ۲^{۲}, ۲^{۱}, ۲^{۰}, ۲^{-۱}, ۲^{-۲}, \ldots)$ انتخاب شوند، در آدرس‌هایی با امتیاز کمتر از ۱، مقدار آدرس‌های تقریبا یکسانی در هر تکه قرار خواهند گرفت.

۳. این شیوه قسمت‌بندی، امنیت و گم‌نامی زیادی برای آدرس‌های پر استفاده ایجاد نمی‌کند. زیرا اولا، تعداد اعضای تکه‌های آدرس‌های پر استفاده کمتر است. ثانیا، مرز این تکه‌ها گسترده‌تر بوده که باعث می‌شود که آدرس‌هایی با رنج وسیعی از احتمال پرسمان را در خود جایی دهد. با این حال می‌توان استدلال کرد که آدرس‌های پر استفاده، به منظور حفظ بیشتر امنیت، مربوط به گره‌های کامل هستند یا اینکه بهتر است دارندگان این آدرس‌ها به جای استفاده از یک گره سبک، یک گره کامل راه‌اندازی نمایند.

۴. با توجه به اینکه تکه‌های مربوط به آدرس‌های کم استفاده (با امتیاز کمتر از ۱) تعداد اعضای زیادی خواهند داشت و بارگیری آن‌ها توسط گره سبک مصرف زیاد پهنای باند و ترافیک شبکه را به همراه خواهد داشت، می‌توان از دو رویه برای آن‌ها استفاده نمود. اول اینکه از توان‌های اعشاری برای جداسازی تکه‌های آدرس‌های کم استفاده بهره برد. دوم اینکه آدرس‌های مربوط به یک تکه با اعضای زیاد را به ترتیب الفبا مرتب نمود و با توجه به حروف الفبا آن‌ها را تقسیم بندی نمود. به عنوان مثال آدرس‌هایی مربوط به یک تکه که با bc1qaaa آغاز می‌شوند درون یک زیر تکه قرار بگیرند و آدرس‌هایی که با bc1qaac شروع می‌شوند در یک زیر تکه دیگر قرار گیرند (در استاندارد آدرس بک-۳۲ [۳] از کاراکتر b استفاده نمی‌شود [۵۰]). روش دوم از روش اول برتری دارد. از آن جهت که در روش اول تغییرات تکه‌ها افزوده خواهد شد. در بخش ۳.۴ توضیح داده شد که تغییرات زیاد تکه‌ها باعث کاهش گم‌نامی خواهد شد.

در این بخش به شیوه بهینه تشکیل تکه‌ها و اهمیتی که بر حفظ گم‌نامی کاربران سبک ایفا می‌کند پرداخته شد. تکه‌های تشکیل شده در این روش در تمام شبکه، که یک زنجیره بلوکی واحد را به اشتراک می‌گذارند، یکتا است. به این ترتیب هر آدرس در زمان $t_۰$ تنها در یک تکه به خصوص قرار خواهد داشت. تشخیص این که گره کاربر سبک در کدام تکه قرار دارد، به صورت آفلاین و بدون نیاز به درخواست اطلاعات اضافه‌ای، توسط خود کاربر قابل انجام است. از این رو، کاربر می‌تواند تنها با ارسال شماره تکه (γ) و زیرتکه (جداسازی الفبایی) به صورت امن به اطلاعات مربوط به آدرس خودش دست پیدا کند.

۳.۴.۴ محاسبه آفلاین در سمت کاربر سبک

تا اینجا، نحوه تکه‌بندی آدرس‌ها با توجه به امتیاز هر آدرس، که مطابق معادله (۶.۴) محاسبه گردید، توضیح داده شد. طراحی پروتکل به نحوی انجام شد که کاربران سبک بتوانند بدون نیاز به ارسال هیچ اطلاعات اضافه‌ای، از

[۳] Bech-32

شکل ۱.۴: امتیاز آدرس‌های بیت‌کوین در مقیاس لگاریتمی از تاریخ ۲۶ ژوئن ۲۰۱۹ الی ۰۲ دسامبر ۲۰۱۹

تکه‌ای که مربوط به خودشان است آگاه شوند. از این رو محاسبه آفلاین تکه در سمت کاربر به امری ساده تبدیل شد. در این بخش به نحوه انجام این محاسبه خواهیم پرداخت.

کاربر سبک همگام با شبکه، سرایند تمام بلوک‌های زنجیره بلوکی را دارد و همچنین می‌داند که تمام تراکنش‌هایی که در آن‌ها از آدرس وی استفاده شده است، در کدام بلوک‌ها قرار دارند. از این رو کاربر سبک می‌تواند به راحتی با توجه به سوابقی که در اختیار دارد و با استفاده از فرمول (۶.۴) از شماره تکه‌ای که شامل آدرس خودش است مطلع شود. کاربر سبک همچنین می‌تواند به راحتی تشخیص دهد که آدرسش، با توجه به تقسیم‌بندی الفبایی، در کدام زیر تکه قرار دارد.

اگر کاربر سبک تمام سوابق خودش را از دست دهد، می‌تواند زیر تکه متناسب با آدرسش را از تمام تکه‌های موجود درخواست دهد. استفاده از روش چینش الفبایی کمک می‌کند که کاربر سبک زیرتکه‌های کمتری را امتحان کرده و درنتیجه ترافیک شبکه کمتری مصرف نماید.

از آن‌جایی که محاسبهٔ امتیاز هر آدرس با توجه به زنجیره بلوکی تعیین می‌شود، مقدار آن قابل اجماع است. از این رو گره‌های کامل می‌توانند به صورت روزانه (به ازای هر ۱۴۴ بلوک) امتیازها را بروز رسانی نموده و لیست آدرس‌های جدید همراه با امتیاز آن‌ها را در شبکه قرار دهند. همچنین، می‌توان درخت مرکل تمام آدرس‌ها را تولید کرده و در تراکنش کوین‌بیس قرار دهند. به این ترتیب گره سبکی که تاریخچه اطلاعاتش را از دست داده می‌تواند ساده‌تر به دنبال آدرس خودش بگردد و از صحت امتیاز آدرسش مطمئن شود.

۵.۴ پیاده‌سازی و شبیه‌سازی

پروتکل معرفی شده در این فصل با زبان پایتون پیاده‌سازی شده است. کد منبع مربوط به این پیاده‌سازی در [۱۱] قابل دسترسی است. رایانه‌ای که شبیه‌سازی‌ها بر روی آن اجرا شده است دارای یک پردازندهٔ Intel(R) Core(TM) i7-4770K CPU @ 3.50GHz و حافظهٔ موقت ۱۶ گیگ DDR3 است. در کنار نرم‌افزار پیاده‌شده در این سیستم، نرم‌افزار هستهٔ بیت‌کوین نیز اجرا می‌شود. این هستهٔ بیت‌کوین با استفاده از شبکهٔ تور و با راه‌اندازی یک سرور مخفی[۴] از طریق یک آدرس onion. اختصاصی به دیگر گره‌های شبکه شناخته شده و با آن‌ها ارتباط برقرار می‌کند.

به منظور استقلال کامل این پیاده‌سازی از نرم‌افزار هستهٔ بیت‌کوین، دادهٔ مربوط به بلوک‌ها مستقیما از فایل‌های blk.dat خوانده می‌شود. اطلاعات ذخیره شده در این فایل‌ها به صورت خام بوده و برای خواندن آن به پیاده‌سازی کدی که بتواند دادهٔ باینری را تجزیه[۵] نماید وجود دارد. در این پروتکل امکان تجزیه تمام انواع نبشته‌ها، یعنی P2PK، P2SH، P2PKH و P2WPKH، در تابع script_decoder فراهم شده است. پروتکل از آن جهت مستقل از نرم‌افزار هستهٔ بیت‌کوین ساخته شده است و از واسط برنامه‌نویسی کاربردی[۶] ارائه شده توسط آن استفاده نمی‌کند تا اگر کاربر گره کامل بخواهد نرم‌افزار هستهٔ بیت‌کوین را در یک محیط ایزوله اجرا نماید تا جهت حفظ امنیت آن ارتباطی با سایر نرم‌افزارهای سیستم‌عامل میسر نباشد، در اجرای این پروتکل با مشکل مواجه نشود. نرم‌افزار پیاده‌شده می‌تواند تنها دسترسی فقط خواندنی[۷] به فایل‌های blk.dat داشته باشد.

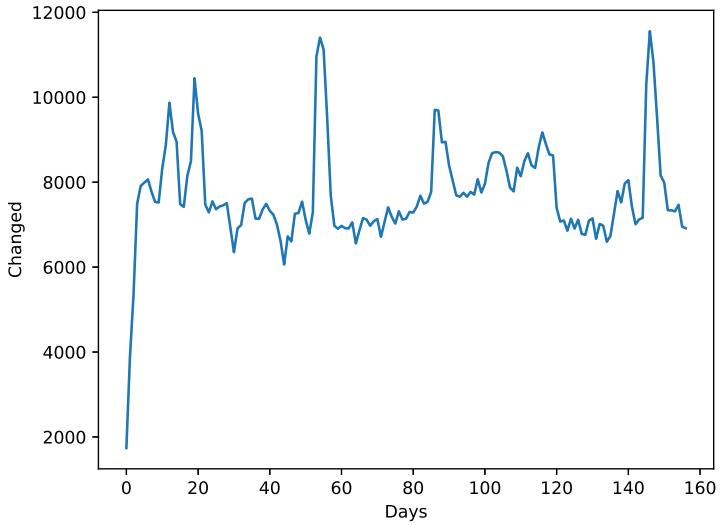
در پیاده‌سازی انجام شده، از پایگاه‌دادهٔ ردیس[۸] برای ذخیره‌سازی وضعیت هر آدرس و امتیاز آن استفاده شده است. این پایگاه داده با مدیریت مناسب حافظهٔ RAM، به بهبود عملکرد نرم‌افزار کمک می‌کند. برای ذخیرهٔ امتیاز تمام ۴۶۰ میلیون آدرس با استفاده از این پایگاه داده به چیزی در حدود ۳۶ گیگ حافظهٔ دائمی نیاز است. اکثر بخش‌های این شبیه‌سازی بر روی اطلاعات زنجیرهٔ بلوکی از تاریخ ۲۶ ژوئن ۲۰۱۹ الی ۰۲ دسامبر ۲۰۱۹ صورت گرفته است. برای نشان‌دادن بهتر وضعیت آدرس‌ها، تعداد تکه‌ها زیاد انتخاب شده است به این ترتیب که بازهٔ آن‌ها به صورت $\{(0, 2^{-20}), (2^{-20}, 2^{-19}), (2^{-19}, 2^{-18}), \ldots, (2^{20}, \infty)\}$ است. در عمل این بازه‌ها متفاوت خواهد بود.

یکی از اصلی‌ترین معیارهای سنجش عملکرد صحیح پروتکل، عدم تغییر سریع آدرس‌ها در میان تکه‌های مختلف است همچنین مقدار تغییر آدرس‌ها باید در یک ناحیه‌ای محدود باشد و با گذشت زمان صعودی نباشد. با تنظیم پارامتر β، که در معادله (۶.۴) تعریف شد، و همچین بازهٔ تکه‌ها می‌توان سرعت تغییر آدرس‌ها در میان

[4] Hidden Server
[5] Parse
[6] Application Programming Interface (API)
[7] Read-Only Access
[8] Redis

تکه‌ها را کنترل نمود.

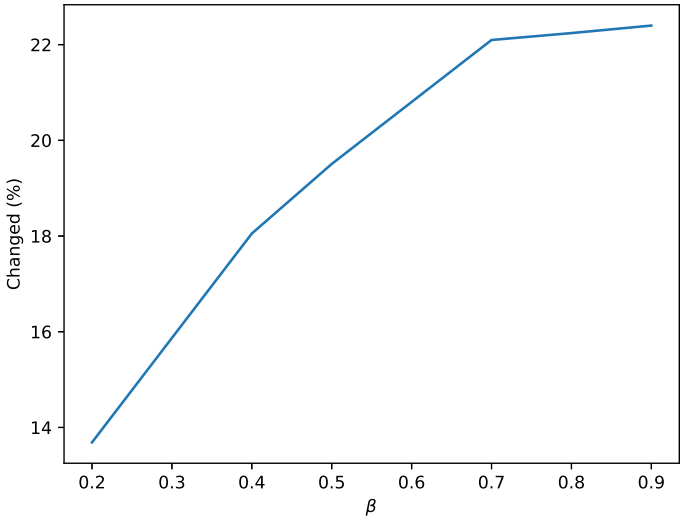
همان‌طور که در شکل ۲.۴ نمایش داده شده است، تغییرات کلی آدرس‌ها، به ازای $\beta = ۰/۳$، در حول مقدار ثابتی تغییر می‌کند. شکل ۳.۴ درصد تغییرات آدرس‌ها در بین تکه‌های مختلف نسبت به کل آدرس‌ها با توجه به مقدار β را نشان می‌دهد. هر چه β به ۱ نزدیک‌تر باشد، امتیاز آدرس‌ها سریع‌تر تغییر می‌کند.



شکل ۲.۴: نمودار جابه‌جایی آدرس‌ها در میان تکه‌های مختلف از تاریخ ۲۶ ژوئن ۲۰۱۹ الی ۰۲ دسامبر ۲۰۱۹. ($\beta = ۰/۳$)

پیاده‌سازی به این صورت است که به ازای هر روز، آدرس‌های موجود در تمام بلوک‌های استخراج شده در آن روز به همراه امتیاز آن ذخیره می‌شود و با توجه به این امتیازها و وضعیت قبلی ($S^{t-۱}$)، وضعیت جدید محاسبه می‌شود. این امر مستلزم به‌روزرسانی روزانهٔ امتیاز تمام آدرس‌های بیت‌کوین است. تعداد کل آدرس‌های ساخته شدهٔ بیت‌کوین تا اکنون ۴۶۰ میلیون آدرس است که به روزرسانی امتیاز تمام آن‌ها در روز به ۲۵ دقیقه زمان نیاز دارد. از طرف دیگر بخش بیش‌تری از این آدرس‌ها عملا بدون استفاده و با موجودی صفر هستند. از این تعداد آدرس، تنها ۳۰ میلیون آدرس موجودی غیر صفر دارند. به روزرسانی امتیاز آدرس‌های بلااستفاده، عملا سربار اضافه‌ای برای این روش است که با حذف آن و محاسبهٔ امتیاز فقط ۳۰ میلیون آدرس فعال، زمان محاسبهٔ روزانه به حدود ۲دقیقه در روز کاهش پیدا می‌کند.

اگر گره کاملی به تازگی بخواهد شروع به ارائهٔ سرویس معرفی شده در این پایان‌نامه نماید، دو راه دارد. اول آن‌که گره تازه اضافه شده، لیست آخرین امتیازهای به روز رسانی شده را از چند گره متفاوت دریافت نماید و بدون درستی‌سنجی، آن را قبول کند. این کار مستلزم آن است که به این گره‌ها اعتماد داشته باشد. اما راه دوم استفاده از امکانی است که در پیاده‌سازی این پروتکل در نظر گرفته شده و اجازه می‌دهد که گره کامل تمام امتیازات فعلی

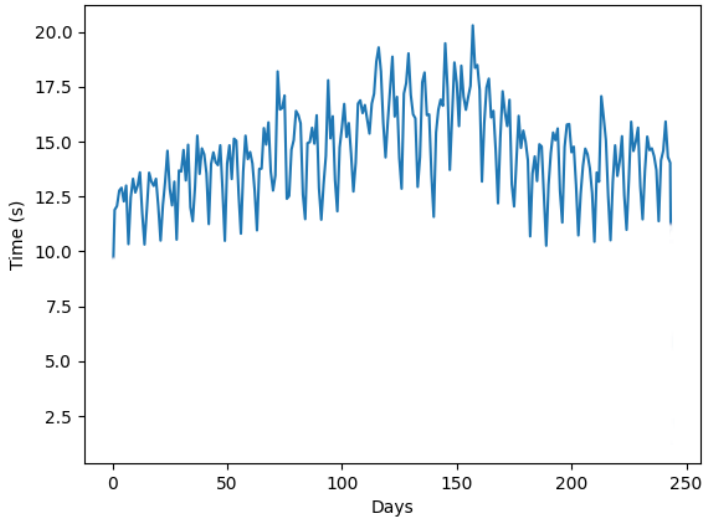


شکل ۳.۴: درصد آدرس‌های تغییر کرده در بین تکه‌های مختلف به تعداد کل آدرس‌ها با توجه به اندازهٔ β

هر آدرس را با توجه به سوابق آن، از ابتدا تا کنون، در زمان کمی محاسبه نماید. بدون استفاده از این امکان و به صورت معمول، با توجه به آن‌چه در بالا گفته شد، اگر گره کامل جدید بخواهد حدود ۲ دقیقه زمان صرف محاسبهٔ امتیازهای هر روز نماید، فرایند هم‌گام‌سازی وی بسیار طول خواهد کشید (با فرض اینکه سوابق آدرس‌های بلااستفاده حال حاضر را بررسی نکند). البته در ابتدای زنجیرهٔ بلوکی تعداد آدرس‌ها به مراتب کمتر بوده، پس زمان کل بسیار کمتر خواهد شد، اما با در نظر گرفتن آیندهٔ زنجیرهٔ بلوکی و افزایش تعداد آدرس‌ها، باز هم محاسبهٔ این‌چنینی امتیاز آدرس‌ها زمان‌بر خواهد بود.

برای رفع این مشکل در پیاده‌سازی این پروتکل، در همگام‌سازی اولیه تنها امتیاز آدرس‌هایی که در روز مورد بررسی استفاده شده‌اند توسط گره کامل به‌روزرسانی می‌شود. به این صورت که برای هر آدرس در پایگاه داده دو مقدار امتیاز و آخرین زمان به‌روزرسانی ثبت می‌شود. وقتی یک آدرس در بلوک‌های مربوط به یک روز مشاهده می‌شود، از تعداد روزهای مابین آخرین به‌روزرسانی تا روز مورد نظر و امتیاز قبلی آن برای محاسبهٔ امتیاز جدید استفاده می‌شود. زمان مصرف شده به ازای به روزرسانی روزانهٔ آدرس‌ها با این روش در شکل ۴.۴ نمایش داده شده است. همان‌طور که مشاهده می‌شود، این روش سربار پردازشی زیادی بر روی گره‌های کامل اعمال نمی‌کند. به این ترتیب در آخر همگام‌سازی گره کامل، با توجه به آخرین زمان هم‌گام‌سازی، امتیاز جدید همه آدرس‌ها محاسبه می‌شود.

همان‌طور گفته شد که می‌توان برای سادگی بیشتر به جای به روز رسانی تمام ۴۶۰ میلیون آدرس ایجاد شده در زنجیرهٔ بلوکی تا کنون، از تنها ۳۰ میلیون آدرسی که موجودی غیر صفر دارند، استفاده کرد. برای پیاده‌سازی سادهٔ



شکل ۴.۴: زمان به روز رسانی وضعیت امتیازها برای هر روز از تاریخ ۱۹ ژانویه ۲۰۱۹ الی ۲۱ سپتامبر ۲۰۱۹.

این موضوع می‌توان این طور گفت که گره کامل تنها امتیاز آدرس‌هایی را نگهداری نماید که در زمان تصمیم‌گیری حداقل یک تراکنش از آن‌ها در مجموعهٔ خروجی‌های خرج نشده تراکنش‌ها[9] وجود داشته باشد.

۶.۴ بحث و مقایسه

مقدار β باید پیش از راه‌اندازی پروتکل تعیین شود. هر چه این مقدار بیش‌تر باشد، تغییرات آدرس‌ها در تکه‌ها سریع‌تر بوده و به گره کامل متخاصم امکان می‌دهد که با اشتراک‌گیری، آدرس مورد نظر کاربر را تشخیص دهد. از طرف دیگر اگر مقدار آن کم باشد باعث می‌شود که آدرس‌هایی که در یک تکه قرار گرفته‌اند در حال حاضر مقدار امتیاز مشابهی نداشته باشند در نتیجه، طبق فرمول (۹.۴) آنتروپی آدرس‌ها در یک تکه کاهش پیدا می‌کند. در نتیجه باید بررسی‌های بیش‌تری برای انتخاب β انجام گیرد.

علاوه بر این، تعداد تکه‌ها و مرزبندی بهینهٔ بین آن‌ها قابل تعیین شدن است. هرچند که از آن‌جایی که این مرزبندی‌ها تاثیر مستقیمی بر روی پروتکل ندارند و در حین ارتباط بین گره سبک و گره کامل قابل تعیین است، الزامی نیست که تمام اعضای شبکه از مرزبندی‌های یکسانی استفاده کنند. به عنوان مثال، چنان‌چه گره سبک اطلاعات مربوط به تمام آدرس‌هایی را که در بازهٔ امتیازی دلخواه $[s_1, s_2]$ قرار داشته و آدرس آن‌ها با bc1qaa

[9] Unspent Transaction Output (UTXO)

شروع می‌شوند، از یک گره کامل درخواست دهد، درخواست وی در سمت گره کامل قابل پردازش و پاسخگویی خواهد بود و از هر گره پاسخ یکسانی دریافت خواهد کرد. چرا که آدرس‌ها در تمام گره‌های کامل شبکه امتیازی یکسان دارند.

روش ارائه شده در این پایان‌نامه نسبت به روش BIP37 (فیلتر بلوم) از مزایای زیادی برخوردار بوده و از روش‌های دیگر ارائه شده، برتری‌هایی دارد. مقایسهٔ این روش با دیگر روش‌ها از سه جنبهٔ امنیت، پهنای باند مصرفی و بار پردازشی قابل انجام است. که در ادامه به این مقایسه خواهیم پرداخت.

۱.۶.۴ مقایسهٔ امنیت

امنیت این روش نسبت به روش فیلتر بلوم به خاطر آن‌که آدرس‌های پوششی به طور هوشمندانه‌ای انتخاب می‌شوند، نسبت به حملهٔ تحلیل بسامد پرسمان آدرس‌ها مقاوم‌تر است. همچنین انتخاب غیرهوشمندانهٔ آدرس‌های پوششی در روش فیلتر بلوم باعث می‌شد که گاهی آدرس‌هایی به فیلتر بلوم اضافه شوند که کنار گذاشتن آن‌ها به خاطر قدیمی‌بودن یا شناخته شده بودن از فیلتر بلوم توسط گره کامل امکان‌پذیر باشد.

از آن‌جایی که در این روش از فیلتر بلوم استفاده نمی‌شود، امکان کشف ارتباط بین آدرس‌های گره سبک برای گره کامل وجود نخواهد داشت. همچنین، در روش BIP37 [۲۸] گره کامل با داشتن چند فیلتر بلوم از یک کیف پول قادر بود که با احتمال بالاتری آدرس‌های اصلی آن را تشخیص دهد، که در این روش این امکان نیز برای گره کامل متخاصم وجود ندارد.

در روش ارائه شده نیاز است که کاربر سبک از شبکه‌های حافظ گم‌نامی برای پنهان کردن آدرس IP خودش بهره‌گیری نماید. همچنین تا جای امکان سعی کند از گره‌های کامل مختلفی پرسمان انجام دهد. در غیر این صورت امکان کنار هم قرار دادن درخواست‌های یک کاربر و تحلیل آن‌ها برای یک گره کامل متخاصم فراهم خواهد بود. هر چند که این مشکل در روش‌هایی که از فیلتر بلوم استفاده می‌کردند [۲۸، ۲۹] و همچنین روش فیلتر بلوک [۴۰] نیز وجود دارد. اما روش‌هایی که از ابزار بازیابی اطلاعات خصوصی [۴۱] و محیط اجرایی قابل اعتماد[10] [۳۳] استفاده کرداند فاقد این مشکل هستند.

در روش ارئه شده از هیچ سخت‌افزار و نرم‌افزار خاصی استفاده نشده است که آسیب‌پذیری‌های به‌خصوصی داشته باشد. به عنوان مثال در [۳۳] نه تنها از SGX استفاده می‌کرد و این سخت‌افزار آسیب‌پذیری‌های زیادی دارد، بلکه استفاده از آن نیازمند اعتماد کامل به سرویس‌هایی متمرکزی چون [۳] است. این روش در هر رایانه‌ای با سخت‌افزاری معمولی قابل اجرا است.

[10] Trusted Execution Environment

۲.۶.۴ مقایسهٔ پهنای باند

نوع اطلاعات مبادله شده در روش ارائه شده، همانند روش فیلتر بلوم [۲۸] خواهد بود. به این صورت که در هر دو روش گره سبک لازم است که سرایند زنجیرهٔ بلوکی را ذخیره نموده و به ازای هر تراکنشی که از گره کامل دریافت می‌کند، اثبات مرکل را نیز دریافت نماید. اما تفاوت اصلی روش ارائه شده آن است که در این روش در هر لحظه گره سبک می‌تواند تعداد آدرس‌های پوششی را معین نماید. به این ترتیب کنترل پهنای باند مصرفی کاملا در هر لحظه در دست گره سبک است. همچنین فیلتر بلوم بعد از اضافه شدن آدرس‌های جدید به آن توسط گره کامل به خاطر بالا رفتن نرخ خطای نوع دو عملا بلااستفاده می‌شد که این مشکل در روش ارائه شده وجود ندارد.

روش پیشنهاد شده نسبت به روش [۴۱] که از ابزار بازیابی اطلاعات خصوصی برای حفظ گم‌نامی استفاده می‌کرد، به خاطر عدم نیاز به دانلود فایل‌های مانیفست حجیم، به طور فاحشی پهنای باند کمتری مصرف می‌کند. اما روش [۳۳] به خاطر عدم نیاز به استفاده از آدرس‌های پوششی و حتی عدم نیاز به دریافت اثبات مرکل در استفاده از پایگاه دادهٔ ناآگاهانه، از همهٔ روش‌ها، از جمله روش ارائه شده در این پایان‌نامه، پهنای باند کمتری مصرف می‌نماید.

در روش فیلتر بلوک [۴۰] در صورتی که کاربر سبک تراکنش‌های زیادی داشته باشد، نیاز است که بسیاری از بلوک‌های زنجیرهٔ بلوکی را دانلود نماید، در نتیجه، در این روش پهنای باند به مراتب بیشتری نسبت به روش پیشنهاد شده مصرف خواهد شد.

۳.۶.۴ مقایسهٔ میزان پردازش سمت گره کامل

در روش فیلتر بلوم[۲۸]، گره کامل مجبور بود چندین بار (k بار) چکیدهٔ اطلاعات تمام تراکنش‌های هر بلوک را به ازای هر فیلتر بلومی که در اختیارش هست، محاسبه نماید. چرا که هر فیلتر بلوم توابع چکیده‌ساز مخصوص به خودش را دارد. به خاطر همین این روش به حملات منع خدمت آسیب پذیر است [۴۹]. در روش فیلتر بلوک [۴۰]، محاسبه در سمک گره کامل به محاسبهٔ یک فیلتر برای هر بلوک کاهش پیدا کرده است. در روش [۴۱] و همچنین روش پایگاه دادهٔ ناآگاهانه [۳۳] پردازش نسبتا زیادی برای به‌روزرسانی دستهٔ هفتگی به ازای استخراج هر بلوک انجام می‌شود. در روش پنجرهٔ پویش [۳۳] نیز برای جلوگیری از حملات کانال جانبی، نیاز به پردازش بسیار بالایی است.

در روش ارائه شده همان‌طور که در بخش قبل شرح داده شد، به‌روزرسانی امتیازها روزانه انجام شده و زمان زیادی صرف این بروزرسانی نخواهد شد.

۴.۶.۴ جمع‌بندی

در جدول ۱.۴ به بررسی امنیت، پهنای باند و میزان پردازش سمت گره کامل پرداخته شده است. این نتایج با جدول‌های ۱.۳، ۲.۳ و ۳.۳ قابل مقایسه است.

جدول ۱.۴: بررسی امنیت، پهنای باند و پردازش سمت گره کامل برای روش ارائه شده

معیار	توضیحات
امنیت	امکان تعیین آدرس‌های پوششی به صورت هوشمندانه و به مقدار دلخواه، استقلال بین درخواست‌های کاربر و دشواری ایجاد پیوند بین درخواست‌های وی، مقاوم در برابر تحلیل بسامد استفاده از آدرس، عدم نیاز به تجهیزات سخت‌افزاری و نرم‌افزاری پیچیده و امکان راه‌اندازی سادهٔ یک گره کامل و در نتیجه کاهش نیاز به اعتماد به گره‌های محدود جهت دریافت خدمات
پهنای باند مصرفی	پهنای باند مصرفی قابل تنظیم که تنها شامل تراکنش‌های اصلی به علاوه تراکنش‌های پوششی و اثبات مرکل آن‌ها است.
پردازش سمت کاربر کامل	با توجه به شکل ۴.۴ به ازای به روز رسانی هر روزانه، زمان زیادی صرف نمی‌شود.

فصل ۵

کارهای آینده

۱.۵ مقدمه

در روش ارائه شده سعی شده است که بدون نیاز به استفاده از پروتکل و ابزارهای پیچیده، با حذف فیلتر بلوم و بهره‌گیری از معیار K-گمنامی و قرار دادن آدرس‌هایی با احتمال درخواست یکسان، در یک دسته به سطح بالاتری از امنیت نسبت به فیلتر بلوم دست پیدا شود. همچنین در این روش تعیین پهنای باند مصرفی کاملا در اختیار کاربر سبک است و کاربر سبک می‌تواند با سطح امنیت مورد نظرش این مقدار را تعیین کند.

در این پایان‌نامه با فرض این‌که احتمال پرسمان اطلاعات مربوط به هر آدرس a_n، به شرطی که مربوط به یک گره کامل نباشد، متناسب است با احتمال استفاده از آن آدرس در شبکه بیت‌کوین به طراحی پرتکل ارائه شده پرداخته شده است. هرچند که این فرض، دور از ذهن نیست اما لازم است که صحت آن از طریق شبیه‌سازی سنجیده شود. به این منظور، در این پژوهش سعی شد که یک نرم افزار هستهٔ بیت‌کوین راه‌اندازی شده تا درخواست‌های گره‌های سبک، که از فیلتر بلوم استفاده می‌کنند، ثبت شود. همچنین سعی شد که با بهره‌گیری از ایدهٔ پایان‌نامهٔ [۳۷]، که برای کشف آدرس‌های اصلی فیلتر بلوم PubKey و PubKeyHash آدرس‌های بیت‌کوین را در فیلترهای بلوم آزمایش کرده بود، به آدرس‌های اصلی فیلترهای دریافت پی برده شود. در کنار این، یک سرور الکترام‌ایکس[1] نیز راه‌اندازی شد تا به صورت مستقیم، بدون استفاده از فیلتر بلوم، نرخ درخواست از طرف کیف‌پول‌های بیت‌کوین الکرترام ثبت شود. اما متاسفانه، به خاطر محدود بودن پهنای باند و الزام استفاده از شبکهٔ تور به خاطر نداشتن آدرس IP استاتیک، فرصت نشد که گره کامل راه‌اندازی شده به صورت کامل به شبکه شناسانده شود و تعداد درخواست‌های قابل توجهی دریافت نماید. از این رو لازم است که برای درستی

[1] ElectrumX

سنجی فرض انجام شده در آینده شبیه‌سازی گسترده طولانی مدتی انجام گیرد.

پارامتر β در این پروتکل و هچنین مرز بین تکه‌ها در امنیت پروتکل تاثیر مستقیمی دارد. لازم است که در آینده مقدار مشخصی برای این پارامتر محاسبه شود. روش محاسبهٔ امتیازها در این پایان‌نامه یک روش ساده[۲] محسوب می‌شود که امکان اجماع همه گره‌ها بر روی امتیاز نهایی را فراهم نموده است. به نظر می‌رسد که با بررسی بیشتر امکان بهره‌گیری از ابزارها و متدهای پیشرفته‌تری برای محاسبهٔ امتیازها وجود خواهد داشت.

[2] Naive

مراجع

[1] bitcoinj. https://bitcoinj.github.io/.

[2] bitcoinj/BloomFilter.java at master · bitcoinj/bitcoinj. https://github.com/bitcoinj/bitcoinj/blob/master/core/src/main/java/org/bitcoinj/core/BloomFilter.java.

[3] Intel® SGX Attestation Service Utilizing Enhanced Privacy ID (EPID). https://api.portal.trustedservices.intel.com/EPID-attestation.

[4] Tor Project: Anonymity Online. https://www.torproject.org/.

[5] Electrum Bitcoin Wallet, 2016. https://electrum.org/#home.

[6] Client-side block filtering - Bitcoin Wiki, feb 2019. https://en.bitcoin.it/wiki/Client-side_block_filtering.

[7] Alison, Bob. Electrum security/privacy model?, sep 2014. https://www.reddit.com/r/Bitcoin/comments/2feox9/electrum_securityprivacy_model/.

[8] Antonopoulos, A. M. *Mastering Bitcoin*, volume 50. 2016.

[9] Azar, Erik and Alebicto, Mario Eguiluz. *Swift Data Structure and Algorithms*. Packt Publishing, 2016.

[10] Back, Adam. Bloom Filtering, Privacy, feb 2015. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2015-February/007500.html.

[11] Badakhshan, Mohammadtaghi. mtbadakhshan/Anonymous-Lightweight-Node-Query: A privacy preserving protocol for lightweight nodes and full nodes communication in the Bitcoin network.

[12] Bianchi, Giuseppe, Bracciale, Lorenzo, and Loreti, Pierpaolo. Better than nothing privacy with bloom filters: To what extent? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7556 LNCS:348–363, 2012.

[13] Bitcoin. P2P Network (Developer Guide) — Bitcoin. https://developer.bitcoin.org/devguide/p2p_network.html.

[14] Bitcoin. P2P Network (Reference) — Bitcoin. https://developer.bitcoin.org/reference/p2p_networking.html#getblocks.

[15] Bitcoincore.org. bitcoin/bitcoin: Bitcoin Core integration/staging tree. https://github.com/bitcoin/bitcoin.

[16] Bitly.com Team. Bitly | Custom URL Shortener, Link Management & Branded Links, 2020. https://bitly.com/.

[17] Bloom, Burton H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.

[18] Booth, Neil, Bauer, Johann, and Jegutanis, John. ElectrumX — Lightweight Electrum Server in Python. https://electrumx.readthedocs.io/en/latest/.

[19] Broder, Andrei and Mitzenmacher, Michael. Network applications of bloom filters: A survey. *Internet Mathematics*, 1(4):485–509, 2004.

[20] Bulck, Jo Van, Moghimi, Daniel, Schwarz, Michael, Lipp, Moritz, Minkin, Marina, Genkin, Daniel, Yarom, Yuval, Sunar, Berk, Gruss, Daniel, and Piessens, Frank. LVI : Hijacking Transient Execution through Microarchitectural Load Value Injection. *IEEE S&P 2020*, pages 54–72, 2020.

[21] Christensen, Ken, Roginsky, Allen, and Jimeno, Miguel. A new analysis of the false positive rate of a Bloom filter. *Information Processing Letters*, 110(21):944–949, 2010. http://dx.doi.org/10.1016/j.ipl.2010.07.024.

[22] Costan, Victor and Devadas, Srinivas. Intel sgx explained. Cryptology ePrint Archive, Report 2016/086, 2016. https://eprint.iacr.org/2016/086.

[23] Devet, Casey and Goldberg, Ian. The best of both worlds: Combining information-theoretic and computational PIR for communication efficiency. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8555 LNCS:63–82, 2014.

[24] Garzik, Jeff. jgarzik/picocoin: A bitcoin library in C, SPV wallet & more. https://github.com/jgarzik/picocoin/#readme.

[25] Gervais, Arthur, Karame, Ghassan O., Gruber, Damian, and Capkun, Srdjan. On the privacy provisions of bloom filters in lightweight bitcoin clients. *ACM International Conference Proceeding Series*, 2014-Decem(December):326–335, 2014.

[26] Grochowski, Konrad, Breiter, Michał, and Nowak, Robert. Serialization in Object-Oriented Programming Languages. In *Introduction to Data Science and Machine Learning*. IntechOpen, mar 2020.

[27] Hearn, Mike. Bloom filter privacy and thoughts on a newer protocol, feb 2015. https://groups.google.com/g/bitcoinj/c/Ys13qkTwcNg/m/9qxnhwnkeoIJ.

[28] Hearn, Mike and Corallo, Matt. BIP 0037, 2013. https://en.bitcoin.it/wiki/BIP_0037.

[29] Kanemura, Kota, Toyoda, Kentaroh, and Ohtsuki, Tomoaki. Design of privacy-preserving mobile bitcoin client based on γ-deniability enabled bloom filter. *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2017-October:1–6, 2017.

[30] Kocher, Paul, Horn, Jann, Fogh, Anders, Genkin, Daniel, Gruss, Daniel, Haas, Werner, Hamburg, Mike, Lipp, Moritz, Mangard, Stefan, Prescher, Thomas, Schwarz, Michael, and Yarom, Yuval. Spectre attacks: Exploiting speculative execution. In *Proceedings - IEEE Symposium on Security and Privacy*, volume 2019-May, pages 1–19. Institute of Electrical and Electronics Engineers Inc., may 2019.

[31] Lipp, Moritz, Schwarz, Michael, Gruss, Daniel, Prescher, Thomas, Haas, Werner, Horn, Jann, Mangard, Stefan, Kocher, Paul, Genkin, Daniel, Yarom, Yuval, Hamburg, Mike, and Strackx, Raoul. Meltdown: Reading Kernel Memory from User Space. *Communications of the ACM*, 63(6):46–56, 2020. https://www.usenix.org/conference/usenixsecurity18/presentation/lipp.

[32] Lodha, Sachin and Thomas, Dilys. Probabilistic Anonymity. In Bonchi, Francesco, Ferrari, Elena, Malin, Bradley, and Saygin, Yücel, editors, *Privacy, Security, and Trust in KDD*, pages 56–79, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[33] Matetic, Sinisa, Kostiainen, Kari, Wüst, Karl, Karame, Ghassan, Schneider, Moritz, and Capkun, Srdjan. BITE: Bitcoin lightweight client privacy using trusted execution. *Proceedings of the 28th USENIX Security Symposium*, pages 783–800, 2019.

[34] Meiklejohn, Sarah, Pomarole, Marjori, Jordan, Grant, Levchenko, Kirill, McCoy, Damon, Voelker, Geoffrey M., and Savage, Stefan. A fistful of bitcoins: Characterizing payments among men with no names. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, page 127–140, New York, NY, USA, 2013. Association for Computing Machinery. https://doi.org/10.1145/2504730.2504747.

[35] Mullin, James K. A second look at bloom filters. *Communications of the ACM*, 26(8):570–571, aug 1983.

[36] Nakamoto, Satoshi. Bitcoin: A Peer-to-Peer Electronic Cash System. *SSRN Electronic Journal*, 2009. https://bitcoin.org/en/bitcoin-paper.

[37] Nick, Jonas David. Data-Driven De-Anonymization in Bitcoin. *ETH Zurich*, pages 1–32, 2015. https://www.research-collection.ethz.ch/handle/20.500.11850/155286.

[38] Niu, Ben, Li, Qinghua, Zhu, Xiaoyan, Cao, Guohong, and Li, Hui. Enhancing privacy through caching in location-based services. *Proceedings - IEEE INFOCOM*, 26:1017–1025, 2015.

[39] Osuntokun, Olaoluwa and Akselrod, Alex. bips/bip-0158.mediawiki at master · bitcoin/bips - Compact Block Filters for Light Clients, 2017. https://github.com/bitcoin/bips/blob/master/bip-0158.mediawiki#cite_ref-4-0.

[40] Osuntokun, Olaoluwa, Akselrod, Alex, and Posen, Jim. bips/bip-0157.mediawiki at master · bitcoin/bips - Client Side Block Filtering, 2017. https://github.com/bitcoin/bips/blob/master/bip-0157.mediawiki.

[41] Qin, Kaihua, Hadass, Henryk, Gervais, Arthur, and Reardon, Joel. Applying private information retrieval to lightweight bitcoin clients. In *Proceedings - 2019 Crypto Valley Conference on Blockchain Technology, CVCBT 2019*, pages 60–72. Institute of Electrical and Electronics Engineers Inc., jun 2019.

[42] Rane, Ashay, Lin, Calvin, and Tiwari, Mohit. Raccoon: Closing digital side-channels through obfuscated execution. In *Proceedings of the 24th USENIX Security Symposium*, pages 431–446, 2015. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/rane.

[43] Reiter, Michael K and Rubin, Aviel D. Crowds: Anonymity for web transactions. *ACM transactions on information and system security (TISSEC)*, 1(1):66–92, 1998.

[44] Ron, Dorit and Shamir, Adi. Quantitative analysis of the full Bitcoin transaction graph. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7859 LNCS:6–24, 2013.

[45] Sompolinsky, Yonatan and Zohar, Aviv. Bitcoin's Security Model Revisited. may 2016. http://arxiv.org/abs/1605.09193.

[46] Stefanov, Emil, Van Dijk, Marten, Shi, Elaine, Fletcher, Christopher, Ren, Ling, Yu, Xiangyao, and Devadas, Srinivas. Path ORAM: An extremely simple oblivious RAM protocol. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 299–310, New York, New York, USA, 2013. ACM Press. http://dl.acm.org/citation.cfm?doid=2508859.2516660.

[47] Swamidass, S. Joshua and Baldi, Pierre. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of Chemical Information and Modeling*, 47(3):952–964, may 2007. https://pubs.acs.org/doi/abs/10.1021/ci600526a.

[48] Sweeney, Latanya. K-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002. https://doi.org/10.1142/S0218488502001648.

[49] Todd, Peter. petertodd/bloom-io-attack. https://github.com/petertodd/bloom-io-attack.

[50] Wuille, Pieter. BIP 0173 - Bitcoin Wiki, 2017. https://en.bitcoin.it/wiki/BIP_0173.

Abstract

Lightweight users make up a significant portion of Bitcoin's peer-to-peer network. On the other hand, these users depend on the full node to receive their information, so a significant part of their information is revealed to the full nodes. Therefore, their privacy is very important. The use of the Bloom filter, which was proposed as the first way to protect the privacy of light users in the 37th Bitcoin improvement Proposal (BIP) , has many major drawbacks. In this method, a series of cover addresses, as a result of Bloom filter's false positive error, is used to hide the SPV client's addresses. This thesis outlines the weaknesses of using the Bloom filter in the Bitcoin peer-to-peer network. It also reviews solutions that have been proposed to replace or improve the Bloom filter.

In this Thesis, an attempt has been made to provide a solution that, unlike the Bloom filter, in which cover addresses were chosen randomly and blindly, these addresses were selected wisely. The selected addresses are selected in such a way that the entropy of the set of requested addresses is maximized and the full node faces more ambiguous. This method also allows the SPV clients to adjust the bandwidth consumed per request.

Keywords Blockchain, Bitcoin, Privacy Provision, Lightweight Client, Simplified Payment Verification(SPV)

University of Tehran
College of Engineering
Faculty of Electrical and
Computer Engineering
Secure Communication and
Cryptography

Security Analysis of a Blockchain Based Peer-to-Peer Network

A Thesis submitted to the Graduate Studies Office
In partial fulfillment of the requirements for
The degree of Master of Science
in Electrical Engineering - Secure Communication and Cryptography

By:

Mohammadtaghi Badakhshan

Supervisor:

Dr. Mohammadali Akhaee

September 2020