نویسنده

حمیدرضا عسگری

h.r.asgari52@gmail.com

منتشر شده توسط

کوین ایران

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

www.coiniran.com

گرافین (Graphene) چیست؟

مقدمه

بلاک های زنجیره بلاکچین به طور معمول دارای یک هدر و مجموعه ای از تراکنش ها، قراردادهای هوشمند و ... هستند. انتقال این بلاک ها در شبکه همتا به همتا (Peer-to-Peer) با میزان بسیار پایین پهنای باند و تاخیر، مزایای بسیاری برای شبکه های رمزارزی دارد.

هرگاه بتوان بلاک ها را با پهنای باند بیشتر و تاخیر انتشار کمتر در چنین شبکه ای انتقال داد این امر موجب افزایش سرعت همگام سازی نقاط (peers synchronization) خواهد شد که از انشعاب یا فورک های ناخواسته جلوگیری خواهد کرد. از سوی دیگر استفاده از پهنای باند کمتر این مزیت را بوجود خواهد آورد که تعداد نقاط بیشتری (در شرایطی که محدودیت پهنای باند ارتباطی وجود دارد) در پردازش و تصمیم گیری در زمان کمتری شرکت کنند.

در نتیجه وجود یک مکانیزم کارا و موثر برای ارسال بلاک ها از یک نود به نود دیگر می‌تواند منجر به افزایش حداکثری اندازه بلاک شود که زمینه را برای ارسال تعداد بسیار بیشتری تراکنش در هر بلاک و رفع معضل مقیاس پذیری فراهم می‌سازد. به عنوان یک مقایسه در نظر بگیرید یک گزینه مثلا ارسال ۸ بایت اول شاخص تراکنش (txnID) در هر بلاک است. اگر بلاک شما در بر دارنده ۲۰۰۰ تراکنش باشد، در اینصورت اندازه این بلاک ۱۶۰۰۰ بایت خواهد شد. بیاید باهم ببینیم برای بهینه کردن این مقدار چه تکنیک هایی داریم:

بلوم فیلترها (Bloom Filters)

بلوم فیلتر ها یک ساختار داده بسیار شگفت انگیز است که می‌تواند وجود مقادیر یا داده های موجود در یک مجموعه مورد نظر را با احتمال خوبی تشخیص دهد. در اینجا مجموعه مذکور همان تراکنش های موجود در بلاک فرستنده (در واقع مجموعه شاخص های تراکنش) و مجموعه شاخص های تراکنش ها در انباره (mempool) گیرنده است.

یک بلوم فیلتر دو خاصیت اصلی دارد:

- اولین خاصیت این است که خطای false-negative ندارد یعنی اگر بلوم فیلتر به ما بگوید این txnID در این مجموعه نیست، پس با قطع و یقین در مجموعه نیست.
- دوم اینکه بلوم فیلتر دارای خطای false-positive است. به عبارت دیگر اگر بلوم فیلتر به ما بگوید این txnID در مجموعه است اما ممکن است واقعا در مجموعه نباشد.

یک مزیت بلوم فیلتر این است که ما می‌توانیم میزان false positive rate (FPR) را بر حسب میزان دلخواه تنظیم کنیم. اما به هر حال در اینجا باید سود و صرفه (trade-off) را نیز لحاظ نمود. اگر میزان دقت بالا را مد نظر قرار دهیم، یعنی به دنبال FPR

پایین هستیم (نرخ پایین خطا) و بنابراین باید تعداد بایت های بلوم فیلتر را افزایش دهیم. اما اگر به تعداد جواب های منفی زیاد اهمیت نمی‌دهیم در اینصورت بلوم فیلتر را می‌توانیم کوچکتر انتخاب کنیم.

حال اگر بخواهیم بلاک ها را با بلوم فیلتر ها انتقال دهیم نتیجه چه خواهد بود؟ برای مثال اگر مقدار FPR را در بلوم فیلتر برابر f=1/m ست کنیم در هر بار که گیرنده تعداد m شاخص یا همان transaction ID را در ممپول خود چک می‌کند باید انتظار داشته باشیم بلوم فیلتر به طور متوسط در هر بلاک به میزان f*m=(1/m)*m=1 وضعیت را اشتباه محاسبه کند که کاملا ناکارآمد است. یعنی ما در این وضعیت نمی‌دانیم کدام تراکنش اشتباه است. برای رفع این مشکل به ناچار باید FPR را در فیلتر کاهش دهیم.

برای مثال اگر میزان FPR را برابر f=1/(144m) ست کنیم می‌توانیم انتظار داشته باشیم که فیلتر در هر ۱۴۴ بلاک تنها یک بار پاسخ اشتباه برگرداند (یعنی مثلا برای بلاکچین بیتکوین هر روز یک بار). به نظر راضی کننده است! اما وجه دیگر آن را نیز در نظر بگیرید که هزینه این میزان دقت، افزایش تعداد بایت های فیلتر است. مثلا در این حالت برای تعداد r آیتم که اضافه شود، مقدار بایت لازم برای بلوم فیلتر برابر r * ln(f)/ln2(2) = r * ln(1/144m)/(8 ln2(2)) بایت خواهد بود. با فرض r=2000 شاخص تراکنش و ممپول m=6000، اندازه بلوم فیلتر برابر خواهد شد با 7,113 bytes که کاملا ناکارآمد است.

جداول جستجوی معکوس بلوم Invertible Bloom Lookup Tables (IBLT)

جداول IBLT نیز یکی دیگر از ساختار داده های احتمالاتی هستند. این جداول به ما اجازه می‌دهند که اختلاف متقارن (Symmetric Difference) مقادیر دو مجموعه را پیدا کنیم. مثلا می‌توانیم یک IBLT از کل شاخص های تراکنش ارسالی یک بلاک از سمت فرستنده و یک IBLT دیگر از تراکنش های ممپول سمت گیرنده ایجاد کنیم. اختلاف IBLT اول از دوم دقیقا به ما می‌گوید که کدام تراکنش های ممپول در بلاک ارسالی نیستند.

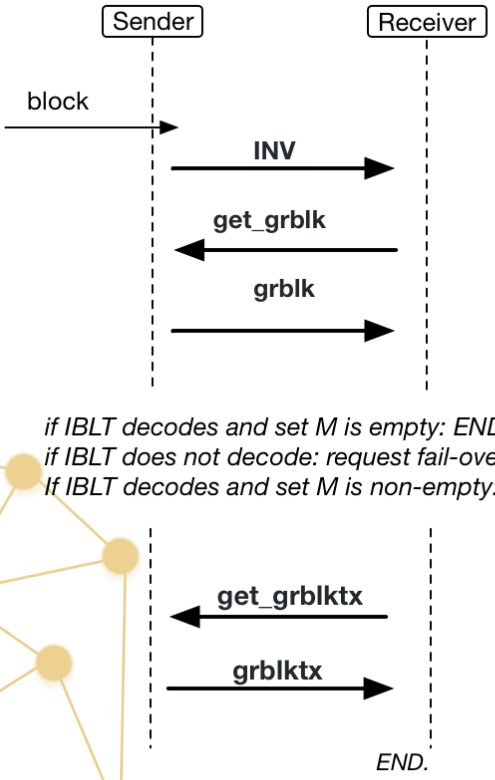
با این قابلیت ما ترغیب می‌شویم که تنها از طریق IBLT بلاک ها را از فرستنده به گیرند ارسال نماییم، اما متاسفانه این یک روش کارا نیست. تعداد بایت های IBLT به طور خطی با میزان اختلاف پوشش داده شده افزایش می‌یابد. یک IBLT در حدود ۱۲ بایت به ازای هر شاخص تراکنش (transaction ID or txnID) استفاده می‌کند که بخشی از همان تفاضل متقارن است و بنابراین میزان سربار (overhead) در اینجا ۱۴۰٪ است. این یعنی اینکه برای یک ممپول با تعداد ۲۰۰۰ تراکنش بیشتر از بلاک ارسالی (تفاضل برابر ۲۰۰۰ است)، IBLT ارسالی برابر با (1.4*2000)*12=33,840 بایت است که انتخاب بهینه و عالی ای نیست.

گرافین (Graphene)

این روش ابداعی حاصل کار تیمی و تلاش افراد زیر است:

Ozisik, Andresen, Bissias, Houmansadr, and Levine

بهترین راه حل، استفاده ترکیبی هر دو ساختار فوق یعنی بلوم فیلتر و IBLT است که همان راه حل گرافین است. ابتدا تمام تراکنش های ممپول گیرنده را از یک بلوم فیلتر بلوک ارسالی فرستنده می‌گذرانیم. طبیعتا در این حالت برای داشتن بلوم فیلتر کوچکتر، مقدار قابل قبولی از false positive را می‌پذیریم. سپس اشتباهات ایجاد شده توسط بلوم فیلتر را با یک IBLT ارسال شده از سمت فرستنده تصحیح می‌کنیم. در این حالت تفاضل متقارن بسیار کوچک خواهد بود و در واقع برابر است با تعداد false positive هایی که در بلوم فیلتر پذیرفتیم. در اینجا هم یک trade-off وجود دارد: ما می‌توانیم بلوم فیلتر را بزرگتر انتخاب کنیم (دقت بالاتر) که منجر به یک IBLT کوچکتر خواهد شد، یا می‌توانیم IBLT بزرگتری بسازیم (تعداد خطای بیشتری تصحیح کنیم) که این نیز منجر به بلوم فیلتر کوچکتری خواهد شد.



M is the set of transactions that are in the block but missing from the receiver mempool. If M *is too large, then the IBLT will not decode; in this case a fail-over block must be requested.*

تکنیک گرافین پارامترها را از هر دو ساختار طوری انتخاب می‌کند که سرجمع دو اندازه خیلی کوچک شود. مثلا برای یک بلاک با اندازه n=2000 و یک ممپول با اندازه m=6000 فرستنده IBLT را طوری محاسبه می‌کند که بتواند ۲۷ مقدار (شاخص تراکنش) را بازیابی کند و با یک بلوم فیلتر از n مقدار، عدد نهایی FPR برابر f=0.00675 است که مینیمال است. در یک آزمایش انجام شده برای مقدار IBLT=715 bytes و Bloom filter=2601 bytes به نتیجه ۳۳۱۶ بایت رسیدیم که در حدود یک پنجم اندازه وقتی است که فقط به ازای هر تراکنش ۸ بایت ارسال می شد (یعنی کاهش اندازه بلاک ارسالی به ۲۰٪). اما اگر دنباله تراکنش های ارسالی کانونیکال نباشد نیاز است که توضیحات نحوه ordering تراکنش ها نیز ارسال شود که در نتیجه مقدار نهایی از ۲۰٪ به ۳۸٪ افزایش پیدا خواهد کرد. در این شرایط IBLT تنها هر ۲۴۰ بلاک یک بار موفق به دیکود اطلاعات نمی‌شود (یعنی در هر ۲۴۰ بار، یک بار بلاک دوباره باید ارسال شود).

گرافین قابلیت خودش را با افزایش اندازه بلاک بهبود می‌بخشد. به عنوان مثال برای یک بلاک با تعداد n=10,000 تراکنش، اندازه این بلاک برای لیست کردن این تعداد تراکنش در حالت ۸ بایت برای هر txnID برابر 80,000 bytes خواهد شد. با یک ممپول به اندازه m=30,000 تراکنش در آن، با استفاده از گرافین، تعداد بایت ارسالی برای هر بلاک 14,482 bytes خواهد شد (و یا ۳۱,۰۹۱ بایت در حالت غیر کانونیکال).

مقایسه روشهای کاربردی مرتبط

فرض کنید که یک بلاک حاوی ۲۰۰۰ تراکنش است و نود گیرنده نیز یک ممپول به اندازه ۴۰۰۰ تراکنش دیگر دارد. بنابراین n=2000 و m=2000+4000=6000 است. با استفاده از تکنیک گرافین، ارسال کننده ابتدا یک سیگنال INV می‌فرستد. دریافت کننده سپس سیگنال GETDATA و مقدار m را در پاسخ ارسال می‌کند. ارسال کننده IBLT را محاسبه و توانایی بازیابی a=27 مقدار را دارد. در اینجا یک بلوم فیلتر برای n ایتم برابر خواهد شد با f=a/(m-n)=0.00675 که کمترین است. در نتیجه مقدار بایت ارسالی در یک بلاک برابر با ۳۳۱۶ بایت، بر اساس ۷۱۵ بایت IBLT و ۲۶۰۱ بابت بلوم فیلتر خواهد شد. اگر ترتیب تراکنش ها کانونیکال نباشد، عدد ۳۳۱۶ به ۶۰۶۶ افزایش خواهد یافت.

در روش **Tschipper's XTreme Thin Blocks** گیرنده با ارسال یک بلوم فیلتر ۳۹۵۶ بایتی از یک ممپول با f=1/m=1/2000=0.0005 و ۸ بایت برای هر تراکنش و با n=2000 تراکنش شروع می‌کند. در این روش سرجمع بایت های ارسالی هر بلاک برابر 3956+8*2000=19956 می‌شود.

در روش **Corallo's Compact Blocks** عدد سرجمع نهایی برابر 8*2000=16000 است.

بنابراین از سه روش فوق گرافین دارای کارایی بالاتری است. رشد گرافین بصورت خطی است اما با آهستگی بسیار (در مقایسه با دو روش دیگر) است.

آیا در نظر گرفتن ۸ بایت شاخص برای هر تراکنش از collision (وجود شاخص یکسان برای دو تراکنش متفاوت)جلوگیری می‌کند؟

با txnID ۸ بایتی (۶۴ بیتی) احتمال رخداد collision بسیار پایین است. اگر فرض کنیم که b اندازه تعداد بیت های txnID باشد، احتمال بروز collision در شرایط Birthday Attack با تخمین خیلی خوبی در حدود 1-exp(-m(m-1)/2^(b+1)) است. به عنوان مثال برای ممپول m=10,000,000 تراکنش، و b=64 bits احتمال مذکور در حدود 0.00000271050 است که کاملا قابل چشمپوشی است.

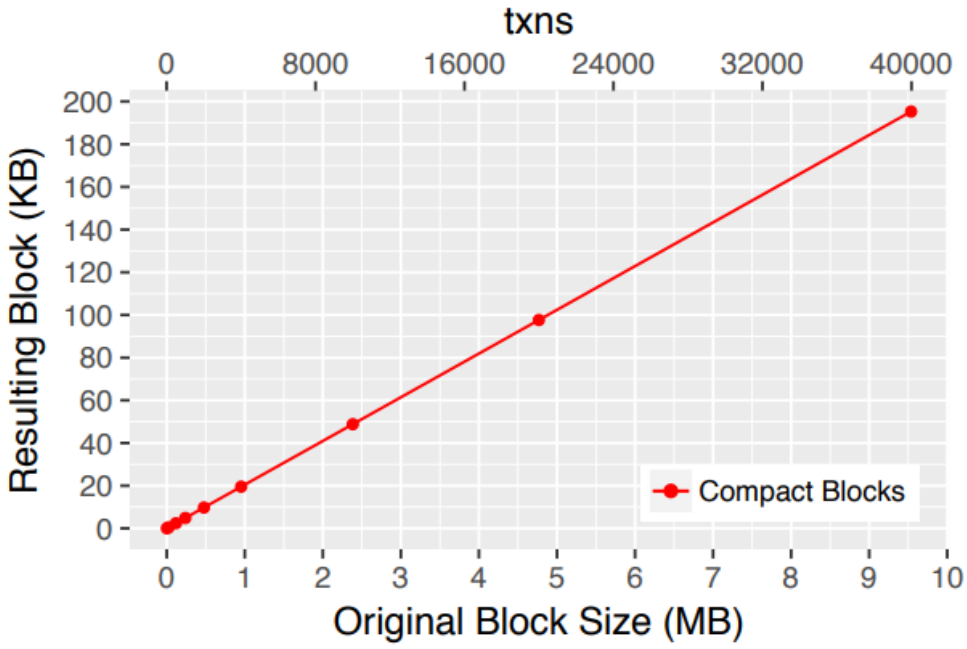
پیاده سازی عملی متد گرافین در بلاکچین بیتکوین کش (BC)

این پیاده سازی که توسط افراد نامبرده در ابتدای مقاله انجام شده کارا بودن این روش و تاثیر آن در مقیاس پذیری را به اثبات رسانده است. بنابر این پیاده سازی اندازه بلاک ارسالی در مقایسه با سایر روش ها به 1/10 کاهش یافته است و این در حالی است که تاخیر انتشار نیز افزایش نیافته و اثر چندانی هم بر افزایش بار پردازنده و فضای ذخیره سازی نداشته است. در ادامه مقایسه گرافین با روش های دیگر که در بلاکچین بیتکوین کش پیاده سازی شده بیان می‌شود.

پروتکل ۱: Compact Blocks

در این پروتکل، نیازی به ارسال تمام اطلاعات تراکنش ها نیست بلکه تنها ۶۴ بایت txnID ها ارسال می‌شود (2Xsha256) و تنها نیاز به ۵ یا ۶ بایت اول است. در این حالت احتمال بروز اشتباه ۱ در میلیارد خواهد بود.

در این روش با داشتن بلاک های 1MB به جای ارسال کل بلاک تنها 80+4200*5=21KB ارسال می‌شود. این عدد برای بلاک های 8MB برابر 80+4200*8*5=164KB است.



پروتکل ۲: بلوم فیلتر Bloom Filter

اگر در پروتکل ۱ دقت کنید در روش کامپکت بلاک، اگر گیرنده بخش قبل توجهی از این txnID ها را از قبل داشته باشد، در اینصورت آیا بهتر نیست تنها آنهایی را که ندارد ارسال کنیم؟ پاسخ به این سوال در بلوم فیلتر نهفته است.

در فرستنده با هش کردن تراکنش ها نهایتا یک عدد حاصل می‌شود که مکان و ارزش متناظر با آن تراکنش را در بلوم فیلتر نشان می‌دهد. در هر مکانی که مقدار ۱ شده است بیانگر وجود تراکنش در بلاک فرستنده است. (به دو خاصیت بلوم فیلتر که در ابتدای این مقاله آمده است دقت نمایید.)

Bloom Filter: Insertion

[0]	[1]	[2]	[3]	[4]	[5]	[6]
1	1	0	0	1	0	0

insert: txn_1
$H_1(txn_1) = 1$
$H_2(txn_1) = 4$

insert: txn_2
$H_1(txn_2) = 0$
$H_2(txn_2) = 4$

در سمت گیرنده مقادیر بلوم فیلتر چک و اطلاعات بازیابی می‌شود.

Bloom Filters: Check

[0]	[1]	[2]	[3]	[4]	[5]	[6]
1	1	0	0	1	0	0

False Negatives are not possible.

Is txn1 in the set?
$H_1(txn_1) = 1$, $H_2(txn_1) = 4$
cell 1 = 1
cell 4 = 1
Yes!
True Positive

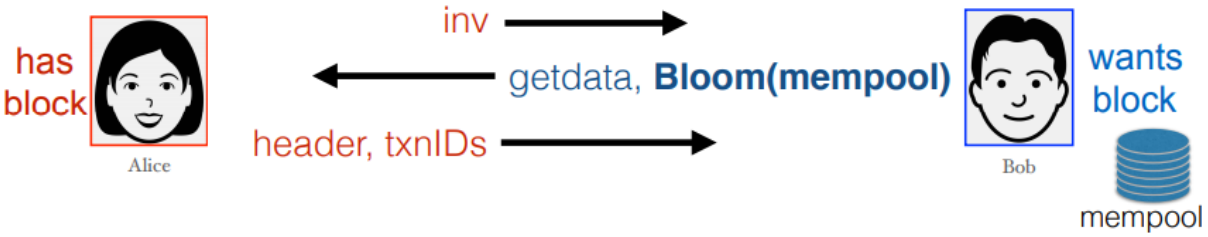
Is txn3 in the set?
$H_1(txn_3) = 1$, $H_2(txn_3) = 5$
cell 1 = 1
cell 5 = 0
No!
True Negative

Is txn4 in the set?
$H_1(txn_4) = 0$, $H_2(txn_4) = 1$
cell 0 = 1
cell 1 = 1
Yes!
False Positive

The False Positive Rate is tunable: More bits will lower the FPR.

پروتکل ۳: Xtreme Thinblocks

در این روش تمام txnID ها و همچنین بلوم فیلتر ارسال می‌گردد. مقدار داده ارسالی در شبکه نسبت به روش کامپکت بلاک بیشتر است.

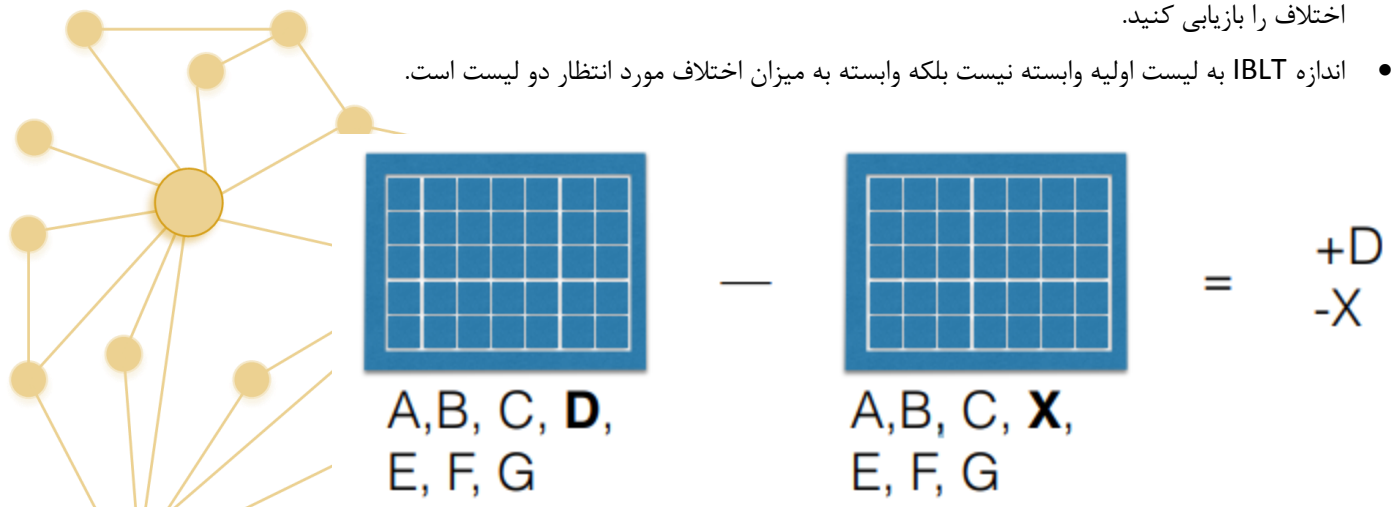


پروتکل ۴: Invertible Bloom Lookup Tables (IBLTs)

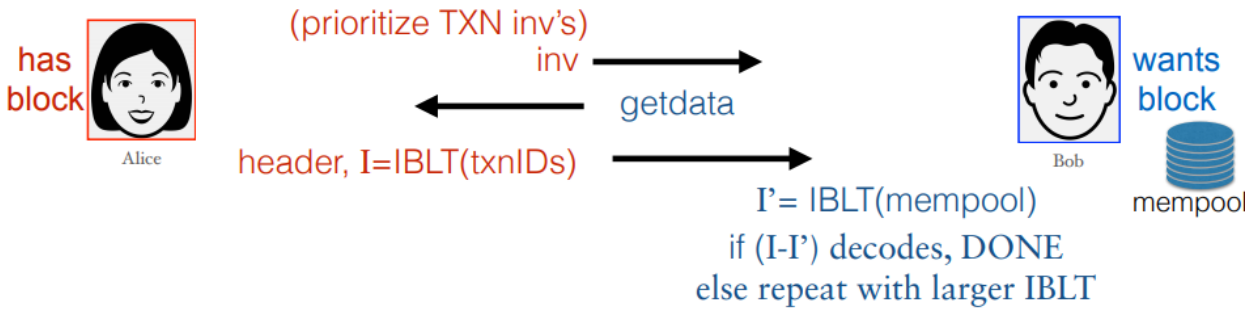
جداول IBLT جداول تعمیم یافته ای از بلوم فیلتر ها هستند که در اینجا هر سلول بجای نگهداری یک بیت، شامل تعداد و مقدار واقعی هستند.

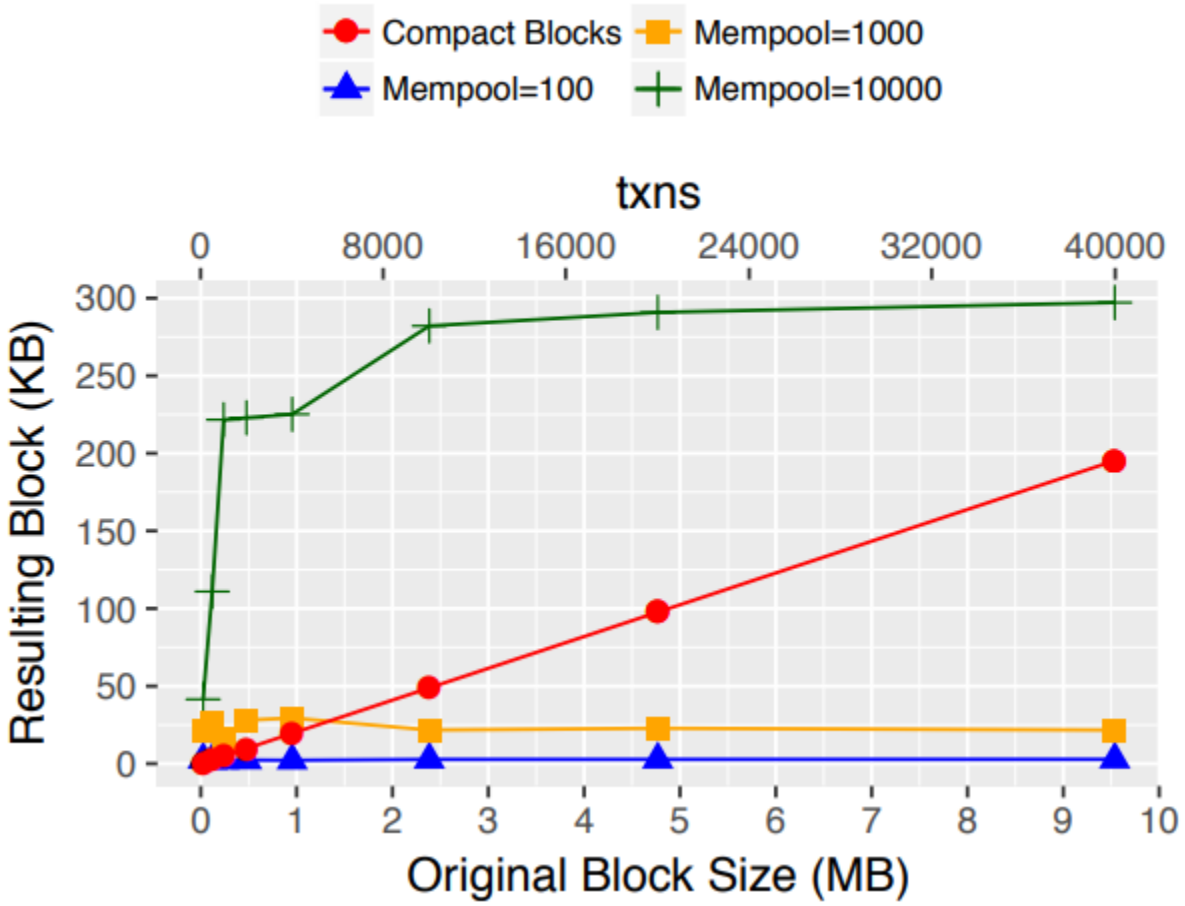
دو خاصیت مهم IBLT:

- اگر دو لیست داشته باشید که اختلاف آنها بیش از ۱۵٪ نباشد شما می‌توانید با مقایسه IBLT هر لیست، آیتم های مورد اختلاف را بازیابی کنید.
- اندازه IBLT به لیست اولیه وابسته نیست بلکه وابسته به میزان اختلاف مورد انتظار دو لیست است.



این روش در حالتی که اندازه ممپول گیرنده بسیار بزرگتر از بلاک ارسالی است خیلی خوب کار می‌کند. اندازه IBLT وابسته به تفاضل متقارن بلاک و ممپول است. اما ما مقدار آن را نمی‌دانیم و نمی‌خواهیم باعث اتلاف زمان ارسال داده ها شویم.





پروتکل ۵: Grahpene

ساختن بلوم فیلتر و IBLT هنگامی که میزان تفاضل متقارن بالا باشد هزینه بالایی دارد (زمان، پردازش و ...). راه حل چیست؟

- استفاده از یک بلوم فیلتر برای کاهش تفاضل متقارن (Symmetric difference) بین بلاک و ممپول
- استفاده از IBLT برای بازیابی خطاهای کوچک رخ داده در بلوم فیلتر

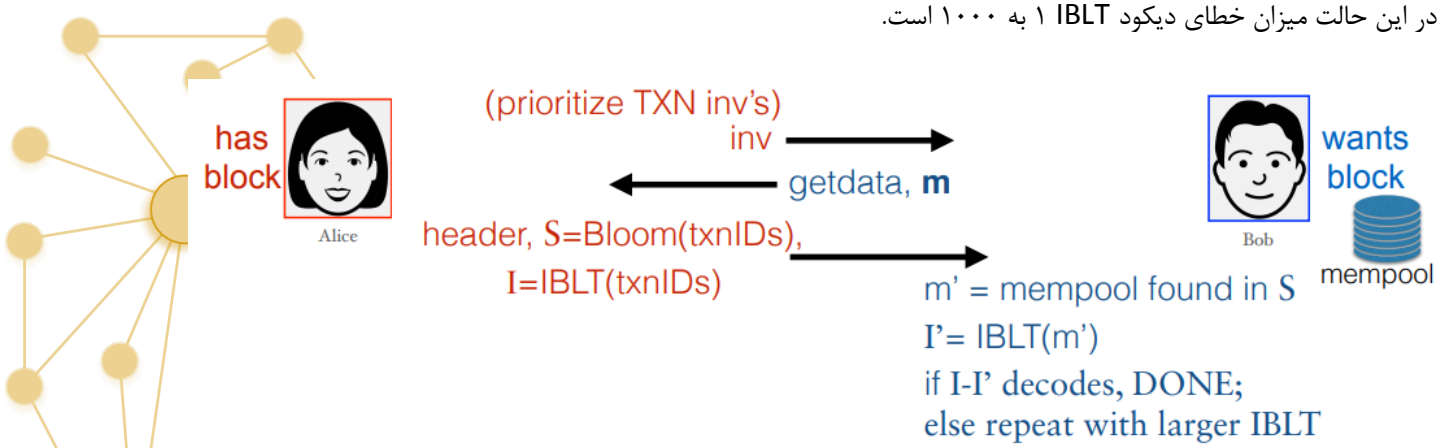
در تکنیک گرافین از هر دو روش بلوم فیلتر و IBLT به طور همزمان بهره می‌گیریم.

بنابراین به FPR خیلی پایینی در بلوم فیلتر نیاز نداریم زیرا IBLT به ما در بازیابی کمک می‌کند. یادآوری می‌شود که سایز IBLT تنها بر اساس اختلاف (تفاضل) دو لیست است.

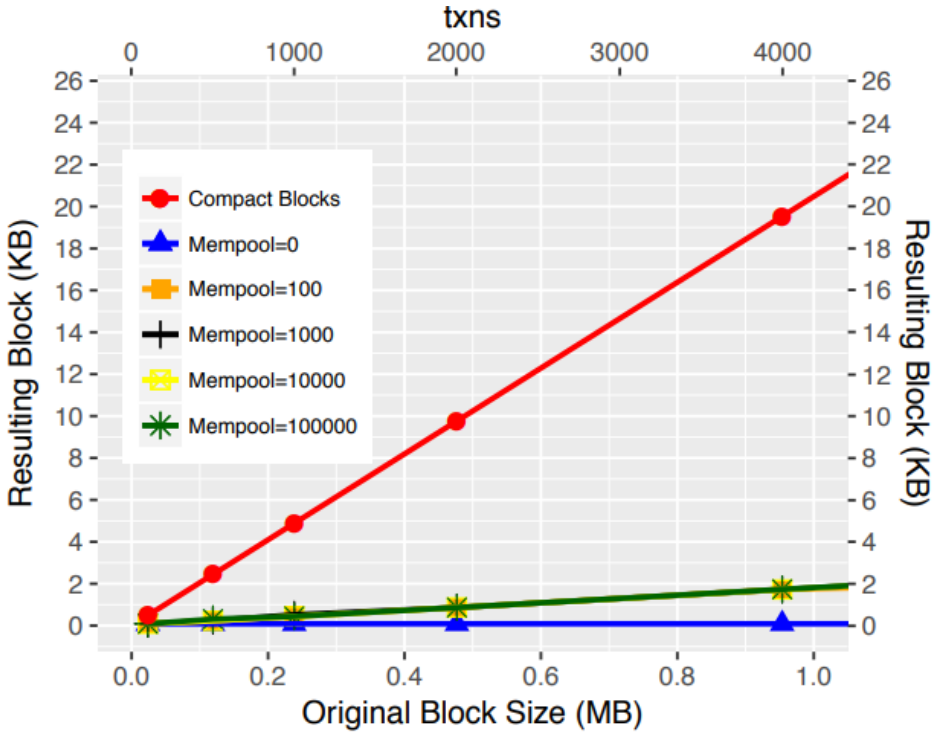
- بلوم فیلتر را به FPR=1/m کوچک می‌کنیم.
- انتظارمان یک false positive است. بنابراین IBLT را با این فرض یعنی تنها یک اختلاف می‌سازیم که بسیار کوچک خواهد بود. خروجی مقایسه دو IBLT دقیقا مشخص می‌کند کدام txnID دارای false positive است.
- حال پارامترهای FPR و IBLT را باهم طوری تغییر می‌دهیم که مجموع بایت ها بصورت بهینه ای کوچک باشد. برای یک بلاک با تعداد n تراکنش و ممپول با m تراکنش، مقدار بهینه FPR از رابطه زیر بدست می‌آید:

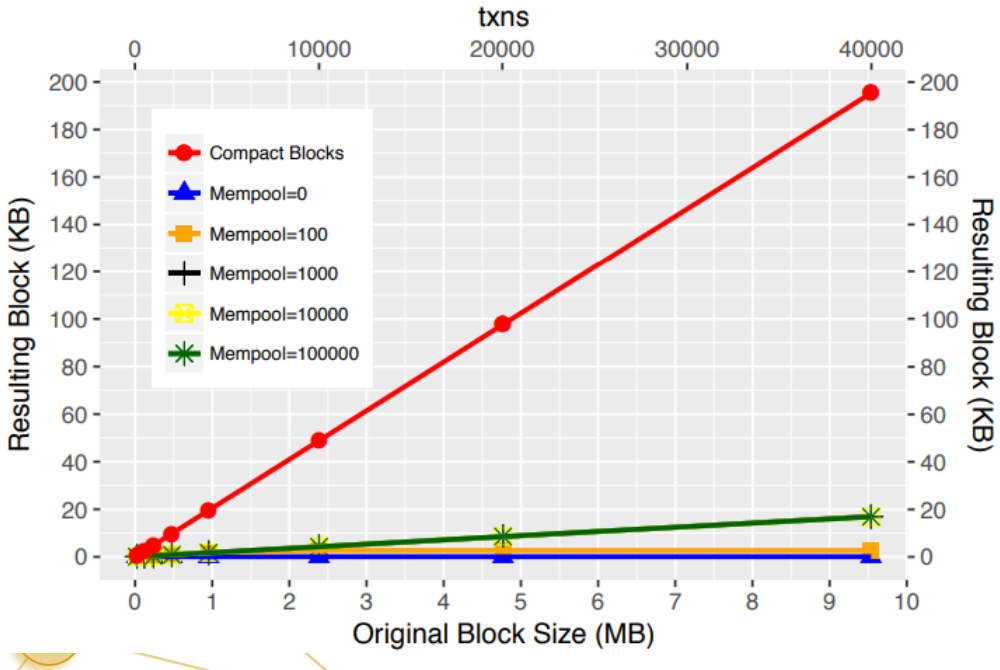
$$\text{FPR} = \frac{n}{132 \cdot (m-n)\ln^2(2)}$$

در این حالت میزان خطای دیکود IBLT ۱ به ۱۰۰۰ است.



کارایی متد گرافین در گراف های زیر نشان داده شده است:





نتیجه گیری:

- بلاک های **Graphene** برای ارسال در شبکه برابر **1/10** سایر روش ها است.
- در اکثر مواقع تنها در یک پکت **IP** قابل ارسال است. (کمتر از ۱۵۰۰ بایت می‌شود).
- نسبت به کامپکت بلاک زمان تاخیر انتشار بیشتری ندارد.
- اثر قابل توجهی بر منابع پردازشی و ذخیره سازی ندارد.
- این تکنیک نتیجه ترکیب خلاقانه دو روش موجود یعنی **Bloom Filter** و **IBLT** است.

منابع:

http://cryptoeconomics.cs.umass.edu/graphene/explained.html

https://github.com/BitcoinUnlimited/BUIP/blob/f3943189ab2a561c92a1cbca670bde20ee4e010d/093.mediawiki#ref1

http://cryptoeconomics.cs.umass.edu/graphene/graphene.BC-scaling.2017.key.pdf

https://bitcoincore.org/en/2016/06/07/compact-blocks-faq/