# خصوصیات ناشناس بودن (گمنامی) شبکه ی نظیر به نظیر بیتکوین

محسن امجدی – 810896043

## مقدمه)

ارز رمزنگاری شده بیت کوین به دلیل شهرت خود به عنوان یک سیستم مالی با حفظ حریم خصوصی ، استقبال گسترده ای را دیده است. اگرچه در عمل ، بیت کوین چندین آسیب پذیری جدی درباره حریم خصوصی را به نمایش می گذارد.

بیشتر این آسیب پذیری ها به دلیل دو ویژگی اصلی ایجاد می شوند:
1) بیت کوین هر کاربر را با یک اسم مستعار مرتبط می کند.

2) نام مستعار می تواند از طریق دفتر معاملات عمومی ، به نام بلاکچین ، به معاملات مالی مرتبط شود. این بدان معنی است که اگر یک مهاجم بتواند یک نام مستعار را با کاربر انسانی خود مرتبط کند ، مهاجم ممکن است تمام تاریخچه معاملات کاربر را متوجه شود. چنین نشتی بیانگر نقض گسترده حریم خصوصی است .

در این مقاله ، ما به یک آسیب پذیری لایه پایین علاقه مندیم: پشته شبکه . مانند اکثر ارزهای رمزپایه ، نود های بیت کوین از طریق شبکه نظیر به نظیر ارتباط برقرار می کنند. ما به یک جنبه اصلی شبکه نظیر به نظیر بیتکوین علاقه مند هستیم: انتشار تراکنش ها.
هر زمان که یک کاربر (آلیس) معامله ای را ایجاد می کند (به عنوان مثال ، او بیت کوین برای کاربر دیگری ، باب می فرستد) ، ابتدا "پیام معامله" را ایجاد می کند که شامل نام مستعار وی ، نام مستعار باب و مقدار معامله است. متعاقباً آلیس این پیام تراکنش را از طریق شبکه نظیر به نظیر پخش می کند ، که کاربران دیگر را قادر می سازد تراکنش او را تأیید کرده و در بلاک چین جهانی قرار دهند.
پخش تراکنش ها برای حفظ ثبات بلاکچین بسیار مهم است. پخش از طریق جاری کردن معاملات در پیوندهای شبکه نظیر به نظیر ادامه می یابد. به طور خاص ، ما علاقه مندیم به صورت نظری خصوصیات گمنامی پروتکل های ''flooding'' را که توسط شبکه بیت کوین استفاده می شود ، کمی کنیم.
انتشار تراکنش ها راه جدیدی برای حملات حذف گمنامی باز می کند. اگر یک مهاجم بتواند به آدرس IP که یک انتشار تراکنش را آغاز کرده پی ببرد، مهاجم همچنین می تواند آدرس IP را با نام مستعار بیتکوین کاربر مرتبط کند ، از آنجا که آدرس های IP گاهی اوقات می توانند با هویت های انسانی مرتبط شوند (به عنوان مثال ، با کمک یک ISP ) .
این حملات متکی به یک ابر گره ''supernode'' است که به گره های فعال بیت کوین متصل می شود و به ترافیک معاملات منتقل شده توسط گره های صادق گوش می دهد. با استفاده از این روش ، محققان توانستند نام مستعار کاربران بیتکوین را با دقت حداکثر 30٪ به آدرس IP آنها پیوند دهند.

در سال 2015 ، جامعه بیت کوین با تغییر پروتکل های ''flooding'' خود از پروتکل ''gossip-style'' معروف به گسترش قطره ''trickle spreading'' به پروتکل ''diffusion spreading'' که محتوا را با تأخیرهای نمایی مستقل منتشر می کند ، به این حملات پاسخ داد. با این حال ،مشخص نیست که آیا این تغییر در واقع در برابر حملات حذف گمنامی دفاع می کند.

هدف ما تجزیه و تحلیل خصوصیات ناشناس بودن شبکه نظیر به نظیر بیتکوین است. نکته اصلی مقاله ما این است که نشان دهیم شبکه بیتکوین از ویژگی های بی هویتی ضعیفی برخوردار است و تغییر جامعه به پروتکل به وضعیت کمک نکرد.
برای مشخص کردن ناشناس بودن شبکه نظیر به نظیر بیتکوین ، ما باید سه جنبه اصلی سیستم را مدلسازی کنیم: توپولوژی شبکه ، پروتکل انتشار و توانایی های دشمن.

**مدل شبکه** : شبکه نظیر به نظیر بیتکوین شامل دو گروه نود است: سرورها و سرویس گیرنده ها . کلاینت ها گره هایی هستند که اتصالات ورودی TCP را قبول نمی کنند ، در حالی که سرورها اتصالات ورودی را می پذیرند. ما شبکه نظیر به نظیر سرورها را به صورت نمودار $G(V,E)$ مدل می کنیم ، به طوری که $V$ مجموعه تمام گره های سرور و $E$ مجموعه لبه ها یا اتصالات بین آنها است. در عمل ، هر گره سرور توسط یک زوج ( آدرس IP ، پورت ) نشان داده می شود. در حال حاضر ، حدود 5000 سرور فعال بیت کوین وجود دارد و این تعداد به

طور کلی در بازه زمانی پخش یک تراکنش ثابت هستند.
هر سرور مجاز است حداکثر هشت اتصال خروجی به گره های فعال بیت کوین برقرار کند و حداکثر 125 اتصال فعال را حفظ کند. برای ارتباط بین آلیس و باب ، یک اتصال خروجی (از دیدگاه آلیس) ارتباطی است که توسط آلیس آغاز می شود ، در حالی که یک اتصال ورودی است که توسط باب آغاز شده است. با این حال ، این اتصالات TCP پس از ایجاد دو طرفه هستند. نمودار تصادفی پراکنده بین سرورها می تواند تقریباً به صورت یک نمودار 16 منظم مدل شود. در عمل ، درجه متوسط به دلیل غیرهمگن بودن در سراسر گره ، به 8 نزدیک است. در تجزیه و تحلیل نظری ، ما G را به عنوان یک درخت d-منظم مدل می کنیم.

**پروتکل های انتشار :** هر بار که یک تراکنش انجام می شود ، از طریق شبکه پخش می شود. در این کار ، ما انتشار یک پیام واحد را که از گره منبع $v * \in V$ نشئت می گیرد ، تحلیل می کنیم. بدون از دست دادن کلیت $v*$ را هر زمان که روی نود ها پیمایش میکنیم به عنوان گره 0 برچسب گذاری میکنیم. در این مقاله ، ما هر دو پروتکل ( قبل و بعد از 2015 ) را ارزیابی می کنیم و عملکرد آنها را مقایسه می کنیم.

**پخش قطره** : هر منبع پیام یا رله به طور تصادفی همسایگان خود را که هنوز پیام را ندیده اند ، مرتب میکند. ما به این همسایگان آلوده نشده می گوییم. سپس طبق همان ترتیب ، با تاخیر مداوم 200 میلی ثانیه بین انتقال ها ، پیام را به همسایگان خود منتقل می کند. ما با فرض یک سیستم زمان گسسته ، این پروتکل انتشار را مدل می کنیم. هر منبع یا رله به طور تصادفی همسایگان آلوده نشده خود را مرتب کرده و پیام را در هر مرحله بعدی به یک همسایه منتقل می کند. فرض می کنیم یک گره پس از دریافت پیام ، در مرحله اول شروع به پخش مجدد پیام کند.

در "diffusion spreading" ، هر منبع یا گره رله با تاخیر مستقل و نمایی ای با نرخ $\lambda$ ، پیام را به هر همسایه آلوده نشده منتقل می کند. ما یک سیستم زمانی پیوسته را فرض می کنیم ، که در آن گره به محض دریافت (یا ایجاد) پیام ، به طور نمایی انتشار را شروع می کند. برای هر دو پروتکل ، ما اجازه می دهیم $Xv$ نشانگر زمانی باشد که در آن گره صادق $v \in V$ یک پیام داده شده را دریافت می کند. توجه داشته باشید که گره های سرور نمی توانند بیش از یک بار آلوده شوند. ما فرض می کنیم که پیام در زمان $t = 0$ باشد ،

بنابراین $Xv * = X0 = 0$ . علاوه بر این ، ما $Gt (Vt,Et)$ را زیرگراف آلوده G در زمان t یا زیر گره گره هایی که پیام را در زمان t دریافت کرده اند (اما لزوما آن را به دشمن اعلام نکرده است) نشان دهد .

**مدل سازی دشمن** : ما یک دشمن را در نظر می گیریم که هدف او پیوند دادن یک پیام با (آدرس IP ، پورت) منشا آن است. در تنظیمات ما ، این به شناسایی گره منبع $v * \in V$ ترجمه می شود. برای این منظور ، ما یک دشمن شنودگر را معرفی می کنیم ، که توانایی های وی بر اساس حملات عملی حذف گمنامی در [8,19] مدل شده است.
این حملات ارزان ، مقیاس پذیر و ساده هستند ، بنابراین تهدیدهای واقعی برای شبکه را نشان می دهند. در حملات [8,19] از یک ابرگره استفاده می شود که به بیشتر سرورهای شبکه بیت کوین متصل می شود. ابرگره می تواند چندین اتصال به هر سرور صادق برقرار کند ، که هر اتصال از یک (آدرس IP و پورت ) متفاوت حاصل می شود. از این رو ، سرور صادق متوجه نمی شود که اتصالات ابرگره همه از یک موجودیت هستند. ابرگره می تواند به دلخواه بسیاری از اتصالات بلااستفاده یک سرور ، تا حد 125 اتصال کل را به خطر بیندازد. ما این فرضیات را با این فرض که دشمن شنود کننده تعداد ثابت $\theta$ از اتصالات به هر سرور، به طوری که $\theta \geq 1$ ایجاد می کند ، مدل می کنیم.
ما این اتصالات خصمانه را در گراف G سرور اصلی قرار نمی دهیم ، بنابراین G یک گراف d–منظم است. هنگامی که ابرگره در [8,19] ایجاد شد ، به سادگی به همه پیام های رله شده در شبکه گوش می دهد ، بدون اینکه محتوایی را پخش یا منتقل کند – به همین دلیل "دشمن شنود کننده " نام گرفته است. با گذشت زمان ، به دلیل پروتکل های انتشار آدرس همتا ، دشمن ساختار شبکه بین سرورها را فرا می گیرد. بنابراین ، فرض می کنیم که $G(V,E)$ برای دشمن شنود کننده شناخته شده باشد.
ابرگره در [8,19] همچنین مهرزمانی را مشاهده می کند که پیام ها از هر سرور صادقانه پخش می شوند. از آنجا که دشمن چندین ارتباط فعال با هر سرور را حفظ می کند ، پیام را از هر سرور چندین بار دریافت می کند. برای سهولت در تحلیل ، ما فرض می کنیم که دشمن شنودگر فقط اولین مهر زمانی را ذخیره می کند. با توجه به این ، $\tau v$ را زمانی در نظر میگیریم که دشمن برای اولین بار پیام را از گره $v \in V$ مشاهده می کند. همچنین $\tau$ مجموعه تمام مهرهای زمانی مشاهده شده را نشان می دهد.

هدف دشمن به شرح زیر است: با توجه به مهر تایم های نویزی مشاهده شده $\tau$ و گراف سرور $G$ ، یک برآوردگر $M(\tau,G)$ پیدا کند که منبع واقعی را به درستی شناسایی کند. معیار موفقیت برای دشمن احتمال شناسایی است. با توجه به یک برآوردگر $M$ ، احتمال تشخیص دشمن $P(M(\tau , G) = v * )$ است. برآوردگر دشمن گونه ای از برآوردگر های به اصطلاح "first-timestamp" است. برآوردگر مهر زمان اول $MFT(\tau , G)$ اولین گره ( قبل از زمان تخمین $t$ ) را برای گزارش پیام به دشمن خروجی میدهد :

$M_{FT}(\tau , G) = \arg\min \tau v$ به طوری که $v \in Vt$

این برآوردگر نیازی به دانش گراف ندارد و از نظر محاسباتی نیز پیاده سازی آن آسان است. همچنین با وجود سادگی خود ، در عمل به نرخ دقت بالایی دست می یابد. ما به خصوص به برآوردگر حداکثر احتمال (ML) علاقه مندیم ، که احتمال شناسایی دشمن را به حداکثر می رساند: $M_{ML}(\tau , G) = \arg\max P(\tau | G, v * = v)$ به طوری که $v \in V$

برآوردگر ML به زمان تخمین $t$ بستگی دارد به حدی که $\tau$ فقط دارای مهر های زمانی تا زمان $t$ است.

بر خلاف برآوردگر "first-timestamp" ، برآوردگر ML در پروتکل های انتشار متفاوت است ، به نمودار گراف بستگی دارد و از نظر محاسباتی غیرقابل حل است. با این وجود ما عملکرد آن را برای هر دو پروتکل های انتشار روی درخت های منظم محدود می کنیم.

هدف اصلی ما این است که بفهمیم آیا حرکت جامعه بیت کوین از انتشار "trickle" به "diffusion" واقعاً ضمانت های ناشناس بودن سیستم را بهبود می بخشد. به همین ترتیب ، مسئله ی اصلی ما توصیف حداکثر احتمال (ML) برای تشخیص دشمن شنودکننده برای فرآیندهای قطره و انتشار در درختان منظم $d$ ، به عنوان تابعی از درجه $d$ ، تعداد اتصالات خراب $\theta$ و زمان تشخیص $t$ است.

**تحلیل "trickle"** : ما با تجزیه و تحلیل احتمال تشخیص انتشار "trickle" شروع می کنیم. ما ابتدا برآوردگر "first-timestamp" و به دنبال آن برآوردگر ML را در نظر می گیریم.

برآوردگر "first-timestamp" : تجزیه و تحلیل انتشار "trickle" به خاطر طبیعت ترکیبی و وابسته به زمان آن پیچیده است. بدین ترتیب ، ما با محدود کردن و پیدا کردن حد پایین احتمال تشخیص برآوردگر "first-timestamp" شروع می کنیم.

ما این کار را با محاسبه احتمال اینکه منبع واقعی پیام را دقیقاً قبل از هر گره دیگری گزارش کند ، انجام می دهیم.

اگر $\tau m = \min(\tau 1, \tau 2, \ldots)$ حداقل مهر زمانی مشاهده شده در بین گره هایی که منبع نیستند را نشان می دهد. سپس محاسبه میکنیم $P(\tau 0 < \tau m)$ یعنی ، احتمال اینکه منبع واقعی پیام را دقیقاً قبل از گره های دیگر به دشمن گزارش دهد. این رویداد (که باعث می شود منبع با احتمال 1 شناسایی شود) شامل مواردی نمی شود که منبع واقعی یکی از $k$ گره ها باشد ($k > 1$) که پیام را به طور همزمان و قبل از هر گره دیگری در سیستم به دشمن گزارش می کند.

قضیه 3.1. پیامی را در نظر بگیرید که انتشار آن مطابق "trickle spreading" روی یک درخت $d$–منظم از سرورهای صادق انجام می شود به طوری که هر گره علاوه بر این دارای $\theta$ اتصال به یک دشمن شنود کننده است. احتمال تشخیص برآوردگر "first-timestamp" در زمان $t = \infty$ برآورده می کند

$$\mathbb{P}(M_{FT}(\tau, G) = v^*) \geq \frac{\theta}{d \log 2}\left[Ei(2^d \log \rho) - Ei(\log \rho)\right] \quad (1)$$

*where* $\rho = \frac{d-1}{d-1+\theta}$, $Ei(\cdot)$ *denotes the exponential integral, defined as*

$$Ei(x) \triangleq -\int_{-x}^{\infty} \frac{e^{-t} dt}{t},$$

*and all logarithms are natural logs.*

ما این حد را با شرطی که وابسته به زمانی است که منبع به دشمن گزارش می دهد و محاسبه احتمال شرطی که همه گره های دیگر بعدا گزارش می کنند ، ثابت می کنیم. سپس اثبات به یک مسئله شمارش ترکیبی تبدیل می شود.

ما می توانیم رفتار مجانبی معادله (1) را برای $d$ های بزرگ با استفاده از انتگرال نمایی بسط تیلور تقریب بزنیم. اول ، توجه داریم که وقتی $d$ بزرگ است ، $Ei(2^{\wedge}d \log \rho) \approx 0$ ، بنابراین داریم:

$$\frac{\theta}{d\log 2}\left[\mathrm{Ei}(2^d\log\rho)-\mathrm{Ei}\,(\log\rho)\right]$$

$$\approx\frac{\theta}{d\log 2}\left(-\gamma-\log|\log\rho|-\sum_{\nu=1}^{\infty}\frac{(\log\rho)^\nu}{\nu\cdot\nu!}\right) \quad (2)$$

$$\approx\frac{\theta}{d\log 2}\left(-\gamma-\log\log\left(1+\frac{\theta}{d}\right)+\log(1+\frac{\theta}{d})\right.$$
$$\left.-\frac{\log^2(1+\frac{\theta}{d})}{4}+\dots\right) \quad (3)$$

$$\approx\frac{\theta}{d\log 2}\left(-\gamma-\log\frac{\theta}{d}+\frac{\theta}{d}-\frac{\theta^2}{4d^2}+\dots\right) \quad (4)$$

به طور خاص ، برای مورد خاص $\theta = 1$ که دشمن فقط یک اتصال در هر سرور برقرار می کند ، خط (4) ساده می شود به :

$$\mathbb{P}(\mathtt{M_{FT}}(\boldsymbol{\tau},G))\approx\frac{\log d}{d\cdot\log 2}+o\left(\frac{\log d}{d}\right). \quad (5)$$

این نشان می دهد که برآوردگر زمان "first-timestamp" احتمال تشخیصی دارد که به صورت مجانبی با log (d) / d به صفر میل میکند . به طور شهودی ، احتمال کشف باید به صفر برسد ، زیرا هرچه درجه درخت بالاتر باشد ، احتمال اینکه گره دیگری غیر از منبع قبل از منبع به دشمن گزارش دهد ، بیشتر است. با این وجود ، (5) فقط یک حد پایین برای احتمال تشخیص "first-timestamp" است ، بنابراین ما می خواهیم بفهمیم که محدودیت محکم چقدر است.

**نتایج شبیه سازی**. برای ارزیابی حد پایین در قضیه 3.1 و تقریب در (5) ، ما برآوردگر "first-timestamp" را روی درختان منظم شبیه سازی می کنیم. شکل 2 نتایج شبیه سازی $\theta = 1$ را در مقایسه با تقریب در (5) نشان می دهد. هر نقطه داده به طور متوسط بیش از 5000 آزمایش است. شکل 2 و تقریب (5) یک راه حل طبیعی برای مشکلات ناشناس بودن شبکه بیت کوین پیشنهاد می کند: افزایش درجه هر گره برای کاهش احتمال تشخیص دشمن. با این حال ، خواهیم دید که برآوردگرهای قوی تر (مثل ، برآورد کننده ML) ممکن است احتمالات بالایی را برای تشخیص حتی برای d بزرگ بدست آورند. بنابراین ، بعید است پروتکل "trickle" به اندازه کافی از ناشناس بودن کاربران محافظت کند.

برآورد گر حداکثر احتمال "Maximum-Likelihood Estimator"

در این بخش ، ما یک برآوردگر منبع ML را برای زمان تشخیص محدود t بررسی می کنیم ، که از ساختار گراف و مهر زمان برای پی بردن به منبع استفاده می کند. ما محدودیت احتمال تشخیص آن را وقتی t به سمت بینهایت می رود بررسی می کنیم و نشان میدهیم که حد پایین احتمال تشخیص آن بدون توجه به درجه درخت به ½ می رسد . این مورد ضعف انتشار قطره را هنگامی نشان می دهد که دشمن گراف را بداند. همنگامی که پیام منتشر می شود به برخی نود ها قبل از برخی دیگر می رسد. در هر زمان t اگر کسی مهر زمانی Xv را بداند می تواند نود های زیر گراف آلوده شده ی Gt را به همان ترتیب که پیام را دریافت کرده اند مرتب کند. ما به چنین چیدمانی یک مرتب سازی از نود ها میگوییم.

از آنجایی که انتشار "trickle" مبتنی بر سیستم زمانی گسسته است ، چندین نود ممکن است که پیام را همزمان دریافت کنند که در این صورت آنها در ترتیب بندی با هم مثل یک توده می شوند. البته ترتیب واقعی نود ها توسط دشمن قابل مشاهده نیست . اما مهرهای زمانی مشاهده شده یعنی ($\tau$) مجموعه ترتیب بندی های محتمل را محدود میکند. یک ترتیب محتمل ترتیبی است که متناسب با قوانین انتشار "trickle" روی گراف و همچنین مهر زمانی های مشاهده شده است. ما از $\tau$ برای اشاره به تمام مهرهای زمانی مشاهده شده توسط دشمن استفاده می کنیم ، و این نشان دهنده ی اولین مهر زمانی از هر سرور نیست. بنابراین اگر دشمن با هر سرور $\theta$ اتصال داشته باشد ، $\tau$ شامل مهر زمان های $\theta$ برای هر سرور صادق است. ما یک برآوردگر به نام "timestamp rumor centrality" پیشنهاد می کنیم ، که تعداد ترتیب بندی های ممکن از هر منبع کاندیدا را محاسبه می کند. نامزدی با بیشترین ترتیب بندی ممکن به عنوان خروجی برآوردگر انتخاب می شود. این برآوردگر شبیه "rumor centrality" است ، برآوردگری که برای تصاویر حملات دشمن در [34] ساخته شده است. با این وجود ، وجود مهر زمان و عدم آگاهی از زیرگراف آلوده ، امور را پیچیده می کند. ما در ابتدا "timestamp rumor centrality" را ایجاد میکنیم. سپس نشان میدهیم که این برآوردگر منبع ML برای "trickle spreading" تحت حملات یک دشمن شنودگر است.

قضیه 3.2. فرایند قطره ای "trickle" را روی گراف d–منظمی در نظر بگیرید ، به طوری که هر گره دارای $\theta$ اتصال به دشمن شنودکننده است. هر ترتیب ممکن o1 و o2 با توجه به مهر زمان های مشاهده شده $\tau$ و نمودار G احتمال یکسانی دارند.

این ادعا با استدلال استقرایی قابل اثبات است ، که نشان می دهد در هر زمان مشخص ، تعداد نود ها با درجه ی آلوده نشده مشخص یا تعداد همسایگان آلوده نشده مشخص ، قطعی است – یعنی به ترتیب بندی اصلی بستگی ندارد. علاوه بر این ، احتمال یک ترتیب داده شده اکیدا تابعی از درجه نود های آلوده نشده در هر مرحله ی زمانی است ، بنابراین همه ی ترتیب های ممکن احتمال یکسانی دارند.

قضیه 3.2 به این معنی است که در هر زمان ثابت ، احتمال مشاهده $\tau$ در صورت داده شدن یک منبع کاندید متناسب با تعداد ترتیب های ممکن ناشی از آن منبع نامزد است. بنابراین، یک برآوردگر ML ، که آن را "timestamp rumor centrality" می نامیم ، تعداد ترتیب های ممکن در زمان تخمین متناهی t می شمارد.

"timestamp rumor centrality" یک الگوریتم ارسال پیام است که به شرح زیر پیش می رود: برای هر منبع کاندیدا ، با توجه به مهر زمانهای مشاهده شده ، مجموعه ای از زمانهای ممکن را که ممکن است هر گره آلوده شده باشد ، به صورت بازگشتی تعیین میکند. این امر با عبور مجموعه ای از "زمان های امکان پذیر دریافت" از منبع کاندیدا به برگهای بزرگترین زیر درخت ممکن که منبع نامزد ریشه آن است ، حاصل می شود. در هر مرحله ، گره ها هر زمان از دریافت را که با مهر زمان مشاهده شده آنها مغایرت داشته باشد ، هرس می کنند. در مرحله بعدی ، با توجه به مجموعه زمان های امکان پذیر دریافت هر گره ، آنها تعداد ترتیب های ممکن را که از قوانین انتشار قطره "trickle" پیروی می کنند ، محاسبه می کنند. این امر با عبور مجموعه هایی از ترتیب های جزئی از برگها به منبع مورد نظر و هرس ترتیب های غیرممکن تحقق حاصل می شود.

**تحلیل "Diffusion"** : می خواهیم بدانیم که آیا "Diffusion" ویژگی های گمنامی بهتری از انتشار "trickle" دارد. اگرچه ، چالش اصلی این است که حتی در گراف های ساده مانند یک خط ، محاسبه احتمال دقیق منبع انتشار نامشخص است از آنجا که مجموعه گره های آلوده رشد می کنند. این به این دلیل است که احتمال منبع بودن یک گره به زمان های مشاهده نشده ای که در آن هر نود آلوده شده بستگی دارد یک فضای حالتی که در اندازه گراف به صورت نمایی رشد می کند.

برآوردگر "first-timestamp" و تخمین گر ابتکاری خلقت خودمان که آن را برآوردگر "reporting center" می نامیم. خواهیم دید که این دو برآوردگر مکمل یکدیگر هستند ، به این معنا که منبع را برای حالت های مختلف درجه d به خوبی تشخیص می دهند .علاوه بر این ، آنها احتمال تشخیص نظیر همان گسترش قطره را می دهند.

"first-timestamp": اگرچه این برآوردگر از دانش گراف زیرساخت استفاده نمی کند ، عملکرد آن به شدت به ساختار گراف زیرین بستگی دارد. قضیه زیر دقیقاً احتمال نهایی آن را برای شناسایی روی یک درخت منظم مشخص می کند.

قضیه 4.1) فرایند انتشار "Diffusion" ای را با نرخ $\lambda = 1$ روی درخت d–منظمی در نظر بگیرید. d>2

فرض کنید یک دشمن زمان آلودگی هر نود را با یک نرخ تاخیر نمایی مستقل $\lambda 2 = \theta, \theta \geq 1$ ببیند. در این صورت عبارت زیر احتمال تشخیص برآوردگر "first-timestamp" را در زمان $t = \infty$ توصیف میکند:

$$\mathbb{P}(M_{FT}(\tau, G) = v^*) = \frac{\theta}{d-2} \log\left(\frac{d+\theta-2}{\theta}\right) \qquad (8)$$

این عبارت چند نکته را برجسته می کند: 1- برای یک درجه ثابت d ، احتمال تشخیص اکیدا مثبت است زمانی که $t \to \infty$ . 2- قانون کاهش بازده ای با توجه به $\theta$ وجود دارد : برای یک درجه ثابت d با افزایش $\theta$ ، نرخ رشد معادله 8 کاهش می یابد. از آنجایی که $\theta$ تعداد اتصالات دشمن به هر نود صادق را نشان می دهد ، دشمن بیشترین سود را از اولین اتصالات ایجاد شده در هر گره بدست می آورد. 3- وقتی $\theta = 1$ ، یعنی دشمن فقط یک اتصال با هر نود دارد ، احتمال تشخیص به $\log(d)/d$ نزدیک میشود. توجه کنید که این مقدار در صورتی که $d \to \infty$ به صفر میل میکند.

قضیه 4.1 نشان میدهد که انتقال جامعه بیت کوین از انتشار قطره به انتشار "Diffusion" هیچگونه دستاوردی برای ناشناخته بودن (به صورت مجانبی در درجه گراف) ، حداقل برای دشمن "first-timestamp" فراهم نمی کند. در مرحله بعدی ، ما می خواهیم ببینیم که آیا این مورد برای برآوردگرهایی که از ساختار گراف استفاده می کنند نیز صادق است.

"Centrality-Based Estimators" : ما یک حد پایین متفاوتی را برای احتمال ML تشخیص با تحلیل یک برآوردگر مبتنی بر مرکزیت محاسبه میکنیم. برخلاف برآوردگر "first-timestamp" ، این برآوردگر "reporting centrality" با انتخاب منبع کاندیدایی که نزدیک به مرکز (روی گراف) مهر زمان های مشاهده شده است ، از ساختار زیرگراف آلوده استفاده می کند. اگرچه ، به صراحت از مهرهای زمانی مشاهده شده استفاده نمی کند. همچنین بر خلاف برآوردگر "first-timestamp" ، این برآوردگر مبتنی بر مرکزیت با افزایش درجه d درخت زیرین بهبود می یابد. درواقع ، وقتی $d \longrightarrow \infty$ احتمال تشخیص اکیدا مثبتی دارد ، این بدان معنی است که دشمن شنود کننده احتمال ML تشخیصی را دارد که به عنوان $\Theta(1)$ در d مقیاس می یابد.

مرکزیت گزارشگری به شرح زیر کار میکند: برای هر منبع نامزد v ، برآوردگر تعداد گره هایی را شمارش میکند که از هر یک از زیردرخت های مجاور v به دشمن گزارش داده است. این یک منبع نامزد را انتخاب می کند که تعداد گره های گزارش شده آن تقریبا در هر زیر درخت برابراست.

برای هر منبع نامزد v ، ما d همسایه آن را در نظر می گیریم ، که مجموعه $N(v)$ را تشکیل می دهد. به این معنی که مرکزیت گزارشگری گره 1 است اگر و فقط اگر هر یک از زیردرخت های مجاور آن کاملاً کمتر از $Y(t)/2$ گره گزارشگر باشد. می گوییم گره ای مرکز گزارشگری "reporting center" است اگر مرکزیت گزارشگری آن 1 باشد .برآوردگر گره v ای را انتخاب می کند که بصورت یکنواخت از مجموعه مراکز گزارشگری انتخاب شده باشد .

به عنوان مثال در شکل 5 ، فقط یک مرکز گزارش دهی وجود دارد ، $v^*$ . توجه داشته باشید که مرکزیت گزارش از مهرهای زمانی مشاهده شده توسط دشمن استفاده نمی کند. او فقط تعداد گره های گزارشگری در هر یک از زیر درخت های مجاور گره را محاسبه می کند. این برآوردگر از "rumor centrality" الهام گرفته است . یک برآوردگر ML برای منبع فرآیند انتشار "diffusion" تحت حمله ی "snapshot" . حمله ی "snapshot" زیرگراف آلوده Gt را در زمان t میبیند اما هیچ اطلاعاتی در مورد مهرهای زمانی را یاد نمی گیرد . مرکزیت گزارشگری ویژگی کلیدی ای را روی درختان نشان می دهد. یک نود v

یک "Rumor center" از یک زیر گراف آلوده Gt با ساختار درختی است اگر و تنها اگر هر یک از d زیردرخت مجاور بیشتر از $N(t)/2$ نود در آن نیست.

$$\max_{u \in \mathcal{N}(v)} N_{T_u^v}(t) \leq \frac{N(t)}{2}. \qquad (10)$$

علاوه بر این ، حداقل یک و حداکثر دو "Rumor center" در یک درخت وجود دارد .و منبع واقعی احتمال اکیدا مثبتی برای "Rumor center" بودن روی درختان منظم و درختان هندسی تصادفی دارد. در مورد ما ، ما نمی توانیم به طور مستقیم از "rumor centrality" استفاده کنیم زیرا گره های آلوده واقعی در هر برهه از زمان t مشاهده نمی شوند. ما برآوردگر مرکزیت گزارشگری را به جای گره های آلوده با استفاده از شرایطی مانند (10) روی گره های گزارشگر می سازیم.

تحلیل : ما نشان میدهیم برای درختان با درجه ی d بالا مرکزیت گزارشگری احتمال تشخیص اکیدا بالاتری از برآوردگر "first-timestamp" دارد. احتمال تشخیص آن وقتی $d \longrightarrow \infty$ اکیدا مثبت است.

قضیه 4.2) یک فرایند انتشار "diffusion" را با نرخ $\lambda = 1$ روی یک درخت d–منظم در نظر بگیرید. فرض کنید این فرایند توسط یک دشمن شنودگر مشاهده شده ، که مهرزمانی هر نود را با یک نرخ تاخیر نمایی مستقل $\theta = \lambda 2$ میبیند ، $\theta \geq 1$ . در این صورت برآوردگر مرکزیت گزارشگری احتمال تشخیص ( وابسته به زمان) به صورت زیر دارد:

$$\liminf_{t\to\infty} \mathbb{P}(M_{RC}(\tau, G) = v^*) \geq C_d > 0. \qquad (11)$$

*where*

$$C_d = 1 - d\left(1 - I_{1/2}\left(\frac{1}{d-2}, 1 + \frac{1}{d-2}\right)\right)$$

نتایج شبیه سازی. ما مرکزیت گزارش را در فرآیندهای انتشار روی درختان منظم شبیه سازی می کنیم. شکل 6 عملکرد تجربی گزارش مرکزیت را در مقایسه با حد پایین نظری به طور متوسط روی بیش از 4000 آزمایش نشان می دهد .

برآوردگر در زمان $t = d + 2$ اجرا می شود. شبیه سازی های ما به دلیل محدودیت های محاسباتی تا درجه $d = 5$ اجرا می شوند ، زیرا زیرگراف آلوده به طور تصاعدی با درجه درخت رشد می کند. مشاهده می کنیم که در درجه $d = 5$ ، مرکزیت گزارش به حد پایین نظری در احتمال تشخیص محدود می رسد.

هیچ یک از مرزهای پایین برآورد کننده مرکزیت گزارشگری یا برآورد کننده "first-timestamp" به شدت برتری با دیگری ندارد. برآوردگر "first-timestamp" در گراف هایی با درجه پایین d عملکرد بهتری دارد ، در حالی که مرکزیت گزارشگر در گراف های با درجه بالای d عملکرد بهتری دارد. با در نظر گرفتن حداکثر این دو برآوردگر ، ما حد کمتری را در احتمال ML تشخیص فرآیندهای انتشار در کل دامنه درجه به دست می آوریم. شکل 6 دو برآوردگر را از نظر شبیه سازی و از لحاظ تئوری به عنوان تابعی از درجه d مقایسه می کند. مشاهده می کنیم که گزارش مرکزیت از برآوردگر "first-timestamp" برای درختان درجه 9 و بالاتر پیشی می گیرد.

از آنجا که نتیجه نظری ما تنها حد پایینی در عملکرد مرکز گزارشگری است ، انتقال ممکن است در واقع حتی درمقادیر d کوچکتر اتفاق بیافتد. از نظر تجربی نمودار واقعی بیت کوین تقریباً 8–منظم است ، ناحیه ای که ما انتظار داریم مرکزیت گزارشگری عملکردی مشابه برآوردگر "first-timestamp" داشته باشد.

در این بخش ، ما مقایسه کاملی از "trickle and diffusion" ، هم از لحاظ نظری و هم از نظر شبیه سازی ارائه می دهیم.

برای مقایسه مستقیم ، جدول 2 شامل نتایج نظری ما در کنار هم ، برای مورد خاص $\theta = 1$ و برای $\theta$ عمومی است. همانطور که قبلاً بیان کردیم ، عملکرد هردو شبیه هستند ، به ویژه هنگامی که $\theta = 1$ . اگرچه نتایج حداکثر احتمال در نگاه اول دشوار به نظر می رسد ، نکته اینجاست که با $d, t \to \infty$ هر دوی آن ها به یک ثابت مثبتی نزدیک می شوند. برای انتشار "trickle" آن ثابت برابر ½ است در حالی که برای انتشار "diffusion" تقریبا برابر 0.307 می باشد.

تا کنون ، ما در درجه اول عملکرد "trickle and diffusion" را تابعی از درجه d در نظر گرفته ایم. با این حال ، در عمل ، نمودار زیرساخت بیت کوین ثابت است. تنها مقدار متغیر منابع دشمن است که توسط $\theta$ ضبط می شود ، تعداد اتصالات خراب در هر گره صادق. با این حال ، مطالعه احتمال مجانبی تشخیص هنگامی که $\theta \to 0$ منطقی نیست ، از آنجایی که $\theta \geq 1$ . از طرف دیگر هنگامی که $\theta \to \infty$ احتمال تشخیص بدون توجه به پروتکل انتشار به سمت 1 میل میکند . بنابراین ما امیدواریم که به صورت عددی بفهمیم که برای یک d ثابت ، پروتکل های "trickle and diffusion" به عنوان تابعی از $\theta$ چطور مقایسه می شوند.

شکل 7 عبارات تحلیلی و نتایج شبیه سازی را برای برآوردگر "first-timestamp" در درختان 4 منظم مقایسه می کند.

توجه داشته باشید که نتایج نظری ما احتمال تشخیص بیشتری برای انتشار "diffusion" نسبت به "trickle" را نشان می دهد .این محصولی از حد پایینی ما در احتمال تشخیص قطره است.

در شبیه سازی ، متوجه می شویم که قطره "trickle" و انتشار "diffusion" در واقع عملکرد تقریباً یکسانی از خود نشان می دهند ، که با احتمال نظری تشخیص برای انتشار "diffusion" موافق است. علاوه بر این ، ما با کنار گذاشتن تحقق هایی که چندین گره همزمان در اولین زمان گزارش میدهند ، حد پایین قطره را در شبیه سازی تأیید می کنیم. این نتایج شبیه سازی (خط شرابی در شکل 7) با پیش بینی نظری ما هماهنگ است.

ما همچنین دریافتیم که با افزایش d ، شکاف بین حد پایین قطره ”trickle“ و احتمال شبیه سازی شده شناسایی برای قطره کاهش می یابد. در همین حال ، شباهت بین ”trickle and diffusion“ (در شبیه سازی) حتی برای درختان با درجه زیاد همچنان وجود دارد – حداقل در درختان منظم.

گراف حقیقی بیتکوین. شبکه حقیقی بیتکوین یک درخت منظم نیست. برای تأیید تصمیم خود در مورد تحلیل درختان منظم ، ما انتشار ”trickle and diffusion“ را روی یک ”snapshot“ شبکه حقیقی بیتکوین از 2015 شبیه سازی میکنیم. شکل 8 این نتایج را برای برآوردگر ”first-timestamp“ به عنوان تابعی از θ مقایسه میکند. از آن جایی که گراف بیتکوین ساختار درختی ندارد ، ما برای هر دو پروتکل ”trickle and diffusion“ کمبود برآوردگر ML داریم.

بنابراین برآورگر ”first-timestamp“ یک انتخاب منطقی است. منحنی نظری برای درخت منظم با d = 8 محاسبه می شود مگر اینکه خلاف آن مشخص شده باشد، از آن جایی که این میانگین درجه ی مجموعه داده ما است . ما اول مشاهده می کنیم که عملکرد شبیه سازی شده انتشار ”diffusion“ به پیش بینی نظری ما نزدیک است. این اتفاق می افتد چرا که به احتمال زیاد برآوردگر ”first-timestamp“ برای تخمین *V از همسایگی محلی استفاده میکند.

به نظر می رسد که قطره ”trickle“ با شدت بیشتری به بی نظمی های گراف نسبت به ”diffusion“ پاسخ می دهد ، احتمالاً به دلیل ماهیت ساختار یافته تر آن است ، که ما در تجزیه و تحلیل خود بسیار مورد استفاده قرار می دهیم. مشاهده می کنیم که از نظر تجربی ، احتمال تشخیص برای ”diffusion“ نسبت به انتشار قطره کمتر است.

این نشان می دهد که شهود توسعه دهندگان بیت کوین صحیح بوده است – انتشار ”diffusion“ دارای ویژگی های گمنامی کمی بهتر از ”trickle“ است. هنوز هم تفاوت کم است. توجه داشته باشید که در شکل 8 ، با افزایش تعداد اتصالات خراب ، انتشار ”diffusion“ و قطره به طور فزاینده ای شبیه به هم می شوند. در حملات عملی [8] ، دشمن شنودگر قادر بود تا 50 اتصال به برخی از گره ها برقرار کند.

در این نواحی ، شکل 8 نشان می دهد که قطره و انتشار احتمال تشخیص بسیار مشابه (بالا) دارد ، حتی برای برآوردگر غیربهینه ای مثل ”first-timestamp“ . بنابراین ، حتی از نظرعددی ، ”diffusion“ و قطره در شرایط عملیاتی عملی خصمانه تفاوت اساسی ندارند.

به طور خلاصه ، در می یابیم که قطره و انتشار دارای احتمال تشخیص مشابهی هستند ، هم از نظر مجانب و هم عددی ، همانطور که در شبیه سازی و تجزیه و تحلیل نظری دیده شد .

ما این را در کلاس متعارف درختان d–منظم و از طریق شبیه سازی در یک نمودار واقعی گراف بیتکوین ارزیابی کرده ایم.

**نتیجه**

در این مقاله ، ما ویژگی های ناشناس بودن شبکه نظیر به نظیر بیت کوین ، با توجه ویژه به تغییر از ”trickle“ به انتشار ”diffusion“ در سال 2015 را تجزیه و تحلیل کردیم . دریافتیم که ”trickle“ و ”diffusion“ به صورت مشابه خصوصیات گمنامی ضعیفی دارند. در درختان منظم ، آن ها خواص مقیاس پذیری مشابهی را به طور مجانبی در d و احتمال تشخیص مشابهی را از نظر عددی برای یک d ثابت نشان می دهند. در شبیه سازی روی یک توپولوژی گراف واقعی بیتکوین ، ”diffusion“ و ”trickle“ همچنین از نظر عددی عملکرد مشابهی را نشان میدهند که با پیش بینی های تئوری ما همسو است. این ما را به این نتیجه می رساند که پروتکل های جاری ”flooding“ که در شبکه بیت کوین استفاده می شود به اندازه کافی از بی هویتی کاربر محافظت نمی کند.

یک سوال جالب این است که چگونه می توان پشته شبکه را تغییر داد تا بتواند در برابر حملات حذف گمنامی منبع ، قدرت ایجاد کند. یک دلیل کلیدی این است که حذف گمنامی در حال حاضر ممکن است به دلیل تقارن پروتکل های انتشار فعلی باشد. به این معنی که ، ”diffusion“ و ”trickle“ هر دو محتوا را روی گراف زیرساختی به همه جهت ها با نرخی تقریبا مشابه منتشر میکنند. این تقارن حملات مبتنی بر مرکزیت قدرتمندی را ممکن میکند. بنابراین ، یک راه حل طبیعی شکستن تقارن ”diffusion“ و ”trickle“ است.

درک چگونگی شکست تقارن بدون آسیب رساندن به عملکرد هم از نظر تئوری و هم از نظر عملی مورد توجه است.