# مروری بر کاربرد اثبات ناتراوا در محاسبات اثبات‌پذیر

نویسنده: علیرضا شیرزاد

استاد راهنما: خانم دکتر اقلیدس

دانشگاه صنعتی شریف، دانشکده مهندسی برق

سمینار مخابرات امن و رمزنگاری

ایمیل: alireza.shirzad@ee.sharif.edu

**چکیده**- با پدید آمدن روش‌های برون‌سپاری محاسبات مانند محاسبات ابری، روش‌هایی برای اعتبار سنجی نتیجه‌ی محاسبات پدید آمده‌اند. این روش‌ها عموما بر پایه‌ی تکنیک‌های رمزنگاری، تئوری پیچیدگی محاسباتی و اثبات‌های ناتراوا۱ بنا شده اند که در آن هستار تایید کننده با اجرای پروتکلی از نتیجه‌ی محاسبات هستار اثبات کننده مطمئن می‌شود. یکی از روش‌های محبوب محاسبات اثبات‌پذیر، zkSNARKها هستند که به دلیل پیچیدگی بسیار پایین اثبات و زمان اثبات کننده مورد توجه قرار گرفته اند. یکی از نقاط ضعف این روش، پیچیدگی بالای اثبات کننده است که باعث می‌شود در کاربردهای پیچیدتر مورد استفاده قرار نگیرند. برای رفع این مشکل پروتکل GKR و انواع مختلف آن پیشنهاد داده شده است که اثبات کننده‌ی بسیار کاراتری دارد. در این گزارش در ابتدا اثبات‌های ناتراوا را به شکل دقیق و ریاضی معرفی وسپس به بررسی تکامل روش‌های محاسبات اثبات پذیر و کاربردهای آن می‌پردازیم.

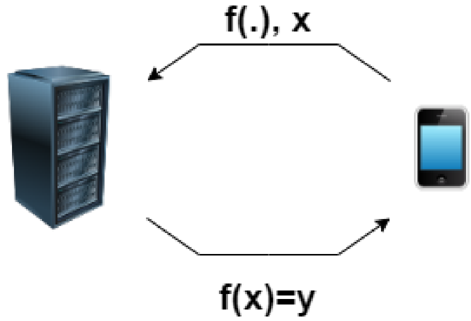
**کلمات کلیدی.** اثبات ناتراوا، محاسبات اثبات پذیر، رایانش امن

## ۱- مقدمه

امروزه با پدیدآمدن مراکز داده‌ی عظیم و روش‌های محاسبات ابری [1]، عدم تقارن در منابع محاسباتی هستارها شدت گرفته است و هستارهای ضعیف‌تر محاسبات سنگین و زمانبر خود را به هستارهای قوی‌تر برون‌سپاری می‌کنند. این چهارچوب محاسبات علی رغم ویژگی‌های مثبت اقتصادی خود، چالش‌های امنیتی بسیار جدی‌ای را بوجود آورده است که محققان در صدد رفع آن برآمده اند. تقریبا تمام این چالش‌ها ناشی از عدم اعتماد به هستار قوی‌تر است که نمونه‌ی آن را می توان در عدم اعتماد به نتیجه‌ی محاسبات دید. در واقع پس از اتمام محاسبات توسط هستار قوی‌تر، نتیجه‌ی محاسبات در اختیار هستار ضعیف تر قرار می‌گیرد که این نتیجه به تنهایی بی اعتبار است و نیاز است تا هستار قوی‌تر با روشی، اعتبار این نتیجه را برای هستار ضیعیف تر اثبات کند. البته این روش تا جایی با معنا است که پیچیدگی

۱ Zero-Knowledge Proof

زمانی هستار ضعیف تر که نقش تاییدکننده را دارد بسیار پایین باشد، در غیر این صورت مسئله زیر سئوال می‌رود.




شکل ۱: مدل سامانه‌ی محاسبات ابری

همانطور که در شکل 1 مشاهده می‌شود، مدل سامانه‌ی محاسبات ابری بدین صورت است که تابع $f$ و ورودی $x$ به هستار قوی‌تر تحویل داده می‌شود و در پاسخ، مقدار محاسبه‌ی این تابع بر روی ورودی به شکل $f(x) = y$ برگردانده می‌شود. در بعضی از مدل‌ها تابع با دو ورودی $f(.,.)$ و $x$ تحویل داده می‌شود و در پاسخ $f(x,w) = y$ برگردانده می‌شود که در این مدل، هستار قوی‌تر باید اثبات کند که $w$ای وجود دارد که مقدار تابع $f$ بر روی $w$ و $x$ برابر با $y$ می‌شود. معمولا مقدار $w$ نیز بر هستار ضعیف‌تر پوشیده است.

برای حل این چالش معمولا از تکنیک اثبات‌های ناتراوا استفاده می‌شود که پایه‌های تئوری و اثبات‌شده ی بسیار قوی‌ای دارد که در ادامه‌ی گزارش به آن پرداخته می‌شود.

ساختار این گزارش بدین ترتیب است که در ابتدا به پایه‌های ریاضی لازم در رمزنگاری و اثبات‌های ناتراوا می‌پردازیم و انواع آن را بررسی می‌کنیم، سپس به توصیف دسته‌ی مهمی از پروتکل‌های محاسبات اثبات‌پذیر،به نام SNARKها می‌پردازیم، در آخر نیز پروتکل GKR را شرح می‌دهیم و نگاه مختصری به پروژه‌ی Muggle می‌اندازیم و بخش جلوگاه و پشت‌گاه آن را بررسی می کنیم.

## ۲- مروری بر مفاهیم اثبات‌های ناتراوا

اثبات‌های ناتراوا که به اثبات‌های دانش-صفر نیز معروف هستند، به صورت غیررسمی، روشی برای اثبات گزاره‌ای هستند که چیزی بیشتر از صحت آن گزارش مشخص نمی‌کند. به طور شهودی، صحت یا عدم صحت یک گزاره باید تنها یک بیت اطلاعات داشته باشد منتها در روش‌های اثباتی که ما در روزمره در علوم ریاضی یا حتی در زندگی روزمره‌ی خود استفاده می‌کنیم، معمولا برای اثبات صحت یک گزاره معمولا بسیار بسیار بیشتر از یک بیت اطلاعات منتقل می‌شود تا هستار تاییدکننده از صحت یک گزاره مطمئن شود.به عنوان مثال فرض کنید می‌خواهیم به کسی اثبات کنیم که راه حل یک مسئله‌ی سخت ریاضی را می‌دانیم. دانستن یا ندانستن این راه حل یک بیت اطلاعات دارد اما در زندگی روزمره گاها مجبوریم تا راه حل را به هستار تاییدکننده بدهیم تا وی را قانع کنیم که این مقدار اطلاعات بسیار بیشتر از یک بیت است.

برای درک درستی از اثبات‌های دانش صفر لازم است تا درابتدا به درک درستی از اثبات برسیم و سپس به بررسی مفهوم اثبات ناتراوا بپردازیم.

### ۲-۱ مفهوم اثبات

اثبات در هر قالبی از علم، به معنای عبارتی است که انسان را قانع سازد وصحت یا عدم صحت گزاره‌ای را برای انسان متقن سازد. حال در علوم ریاضی و تا انتهای صده‌ی گذشته، اثبات به همان شکل سنتی و ثابت خود در نظر گرفته می‌شد، به صورتی که یک متن ثابت که شامل

گزاره‌هایی است در جایی نوشته شده. این گزاره‌ها یکی پس از دیگری از یکدیگر نتیجه گرفته می‌شوند و و دنباله‌ی استنتاجی را تشکیل می‌دهند. بهترین نقطه برای شروع بررسی مفهوم اثبات در علوم کامپیوتر تعریف زبان $\mathcal{NP}$ است که در آن داریم، زبان $\mathcal{L} \in \mathcal{NP}$ اگر رابطه‌ی $R_L$ که در زمان چندجمله‌ای تصمیم‌پذیر است وجود داشته باشد به طوری‌که

$$\mathcal{L} = \{x: \exists w \; s.t. (x, w) \in R_L\}$$

و

$$(x, w) \in R_L \; only \; if \; |w| \leq poly(|x|)$$

این تعریف بدین معنی است که یک رشته عضو $\mathcal{NP}$ است اگر برای این عضویت یک شاهد یا اثبات به نام $w$ وجود داشته باشد. این تعریف دقیقا هم ارز تعریف کلاسیک و ریاضی اثبات است.

## ۲-۲ اثبات تعاملی

یک دیدگاه انقلابی نسبت به اثبات که در [2] مطرح شد، دیدگاه سامانه‌ی اثبات تعاملی بود که اثبات را از حالت ثابت و سنتی خارج می‌کرد و دو عنصر مهم

- **تعامل**: به معنای گفت‌وگوی اثبات کننده و تاییدکننده
- **تصادفی بودن**: ورود احتمال به سامانه‌ی اثبات

را به آن اضافه کرد. در این سامانه، هستارها می‌توانند با یکدیگر گفت و گو کنند و از متغیرهای تصادفی استفاده کنند تا در نهایت به اثبات یا عدم اثبات گذاره‌ای برسند.این درست شبیه سناریوی آموزش مطالب در کلاس درس است که فرایند اثبات به صورت سئوال و جواب توسط دانشجو و استاد صورت می‌گیرد. در واقع در این سامانه‌ها هدف این است که تایید کننده با استفاده از چالش‌های تصادفی، اثبات کننده را گیر بیندازد. دو ویژگی اساسی این سامانه عبارتند از:

۱. **کامل بودن**: بدین معنی که در اکثر اوقات، یک اثبات‌کننده‌ی صادق موفق به اثبات ادعای خود بشود یا به زبان ریاضی

$$x \in \mathcal{L} \rightarrow \Pr[< P, V > (x) = 1] \geq \frac{2}{3}$$

۲. **صحیح بودن**: یعنی هیچ ماشین اثبات‌کننده‌ای نتواند گزاره‌ای غلط را به تاییدکننده اثبات کند یا به زبان ریاضی

$$if \; x \notin \mathcal{L} \rightarrow \nexists B^* s.t. \Pr[< B^*, V > (x) = 1] \leq \frac{1}{3}$$

توجه کنید که اعداد $\frac{1}{3}$ و $\frac{2}{3}$ کاملا قرادادی هستن و اثبات می‌شود که این اعداد می‌توانند به اندازه‌ی کافی کوچک یا بزرگ شوند.

**تعریف**. می‌گوییم $\mathcal{L} \in \mathcal{IP}$ است اگر سامانه‌ی اثبات تعاملی‌ای برای $\mathcal{L}$ وجود داشته باشد.

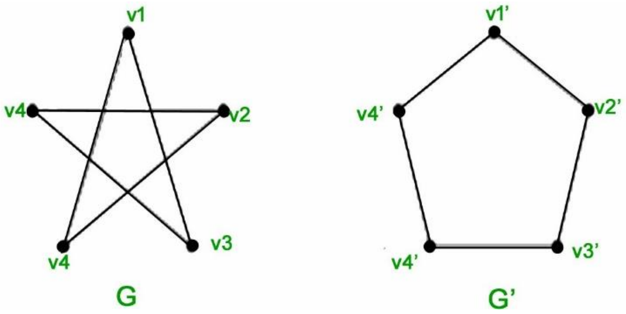
طبق تعریف بالا، سئوال اساسی‌ای که پیش می‌آید این است که قدرت توصیف کلاس $\mathcal{IP}$ چقدر است؟ در واقع قدرت توصیف کلاس $\mathcal{IP}$ نمایانگر قدرتی است که دو خاصیت تعامل وتصادفی بودن به اثبات‌های سنتی می‌دهند. در [3] به این سئوال پاسخ داده شده است که داریم:

**قضیه**: $\mathcal{IP} = \mathcal{PSPACE}$

قضیه‌ی فوق‌الذکر بیانگر این مسئله است که با ورود تعامل و متغیرهای تصادفی در فرآیند اثبات، قدرت توصیف

اثبات‌ها از کلاس $\mathcal{NP}$ بسیار فراتر رفته و معادل کلاس $\mathcal{IP}$ شده است. حال که مفهوم اثبات و اثبات‌های تعاملی توضیح داده شده است، به توصیف دانِش[۲] و تراوش دانِش می‌پردازیم.

برای درک بهتر سامانه‌های اثبات تعاملی، مثال زیر می‌تواند مفید باشد. مسئله‌ی $GNI(Graph\ Non-Isomorphism)$ مسئله‌ای است که در آن به عنوان ورودی دو گراف گرفته می‌شود و در صورت یکریخت نبود گراف‌ها، پذیرفته می‌شود. معادلا GNI زبان تمام جفت گراف‌هایی است که با یکدیگیر یکریخت نیستند.




شکل ۲: نمونه‌ای از دو گراف یکریخت

می‌دانیم که $GNI \in coNP$ است و اثبات‌کننده‌ی $NP$ کارایی برای آن وجود ندارد، منتها در این بخش برای آن یک سامانه‌ی اثبات تعاملی پیشنهاد می‌دهیم:

**سامانه.** فرض کنید ورودی‌های سامانه $G_1 = (V_1, E_1)$ و $G_2 = (V_2, E_2)$ و بدون از دست دادن کلیات فرض کنید:

$$V_1 = V_2 = \{1,2,3, \dots, |V_1|\}$$

تاییدکننده در ابتدا به صورت تصادفی $\sigma \in \{1,2\}$ را انتخاب می‌کند و یک گراف یکریخت با جایگشت تصادفی از $V_\sigma$ را برای اثبات‌کننده ارسال می‌کند. در پاسخ، ارسال‌کننده باید مقدار $\sigma$ را برای تاییدکننده ارسال کند. اگر گراف‌های ورودی یکریخت نباشند، پاسخ اثبات‌کننده در بهترین حالت با احتمال $\frac{1}{2}$ با $\sigma$ برابر است، اما اگر گراف‌ها یکریخت باشند، حتما پاسخ با $\sigma$ برابر است. احتمال تقلب توسط اثبات کننده در هر راند از این سامانه نصف می شود لذا با تکرار این پرسش و پاسخ می‌توان این مقدار را به حد کوچک دلخواهی رساند. همچنین احتمال موفقیت اثبات‌کننده در صورت عدم تقلب نیز 1 می‌باشد. بدیهی است که تاییدکننده محاسباتی چندجمله‌ای را انجام می‌دهد لذا بسیار کارا است.

## 3-2 دانش و تراوش

مفهوم دانِش، از مفاهیم انتزاعی و عجیبی در علوم کامپیوتر است که تعریف صریحی ندارد.در واقع برای دانِش تعریف خاصی مشخص نشده منتها برای تراوشِ دانش و بالتبع عدم تراوشِ دانش، تعریفی آورده شده است که در ادامه به آن می‌پردازیم.

تراوش یا انتقال دانش با انتقال اطلاعات بسیار متفاوت است و گاها اشتباه گرفته می‌شود. رشته‌ای از عبارات می‌تواند حاوی اطلاعات باشد اما حاوی دانش نباشد اما برعکس آن امکان‌پذیر نیست. به عنوان مثال جمله‌ی "عدد 20 بر 2 بخش‌پذیر است" حاوی اطلاعات است اما حاوی دانش نیست، زیر گیرنده‌ی این جمله با دریافت آن دانشی کسب نکرده چر که خود نیز به تنهایی می‌توانست به این حقیقت دست پیدا کند.منتها جمله‌ی "تجزیه‌ی عدد 8051 برابر است با $83 \times 97$" حاوی اطلاعات و دانش است چرا که بدست آوردن تجزیه‌ی یک عدد، راه

[۲] Knowledge

حلی چندجمله‌ای ندارد و گیرنده نمی‌توانست به تنهایی به این حقیقت دست پیدا کند. در ادامه به تعریف ریاضی و دقیق عدم تراوش دانش در سامانه‌های اثبات تعاملی می‌پردازیم:

**تعریف.** می‌گوییم $< P, V >$ سامانه‌ی اثبات تعاملی ناتراوای کامل برای زبان $L$ است اگر برای هر ماشین $V^* \in PPT$ ، الگوریتم $M^* \in PPT$ داشته باشیم به شکلی که برای هر $x \in \mathcal{L}$، دو متغیر زیر توزیع کاملا یکسانی داشته باشند:

- $< P, V^* > (x)$
- $M^*(x)$

در تعریف فوق، به الگوریتم $M^*$، الگوریتم شبیه‌ساز[۳] گفته می‌شود. طبق این تعریف، اگر ماشین تاییدکننده بتواند مکالمه‌ی خود با ماشین $P$ را به تنهایی شبیه سازی کند، در واقع انگار وجود یا عدم وجود ماشین $P$ هیچ تاثیری در پروتکل ندارد و هیچ دانشی از $P$ به سمت تاییدکننده تراوش نکرده است. البته تعریف فوق برای هر ماشین تاییدکننده‌ای است، که این بدین معنی است که ناتراوا بودن یک خاصیت برای ماشین اثبات‌کننده است و ربطی به ماشین تاییدکننده ندارد.

همچنین شایان ذکر است که پروتکل‌های ناتراوا به 3 دسته‌ی کلی

- سامانه‌ی اثبات ناتراوای کامل
- سامانه‌ی اثبات ناتراوای آماری
- سامانه‌ی اثبات ناتراوای محاسباتی

تقسیم می‌شوند. تساوی کامل توزیع‌ها، غیرقابل تمایز بودن توزیع‌ها از لحاظ آماری و غیرقابل تمایز بودن توزیع‌ها از لحاظ محاسباتی، تعریف 3 دسته‌ی بالاست.به عنوان مثال زبان GI(Graph Isomorphism) را درنظر بگیرید. این زبان در $NP$ قرار دارد و بدیهی است چون $\mathcal{NP} \subseteq \mathcal{IP}$ پس اثبات تعاملی برای این زبان وجود دارد. حال سئوال اینجاست که آیا اثبات تعاملی غیرتراوا برای آن وجود دارد یا خیر.

سامانه. در ورودی دو گراف $G_1 = (V_1, E_1)$ و $G_2 = (V_2, E_2)$ گرفته می‌شود. سپس اثبات‌کننده یک جایگشت تصادفی از گراف دوم را به شکل $G' = (V', E')$ برای تاییدکننده ارسال می‌کند. تاییدکننده به تصادف $\sigma \in \{1,2\}$ را انتخاب می‌کند و برای اثبات‌کننده ارسال می‌کند. در نهایت در پاسخ، اثبات کننده می‌بایست مقدار $\sigma$ را به درستی پیدا کند.

در پروتکل بالا باز هم بدیهی است که تاییدکننده پیچیدگی زمانی چندجمله‌ای دارد. همچنین ویژگی‌های کامل بودن و صحیح بودن برای این پروتکل نیز برقرار است. در نهایت کافی است یک ماشین چندجمله‌ای شبیه‌ساز معرفی کنیم تا ناتراوا بودن این پروتکل را اثبات کرده باشیم. به صورت شهودی نیز توجه کنید که تاییدکننده هیچگاه از رابطه‌ی یکریختی بین دو گراف اطلاعاتی کسب نمی‌کند ولی با اطمینان بالایی قانع می‌شود که دو گراف یکریخت هستند. در صورتی که در ماشین تاییدکننده‌ی این زبان در سبک $\mathcal{NP}$ ، ماشین

---

۳ Simulator

تاییدکننده به صورت کامل از این تابع یکریختی آگاه می‌شد.

**قدرت توصیف اثبات ناتراوا**

همانطور که در بخش قبل دیدیم $\mathcal{NP} = \mathcal{IP}$ که قدرت توصیف زبان $\mathcal{NP}$ را مشخص نمود. حال سئوالی که پیش می‌آید این است که قدرت توصیف اثبات‌هایی که شاخصه‌ی ناتراوایی نیز دارند به چه صورت است. در ابتدا لازم است که کلاس‌های پیچیدگی زیر را تعریف کنیم:

- $\mathcal{PZK}$: کلاس زبان‌هایی که دارای اثبات ناتراوای کامل هستند
- $\mathcal{SZK}$: کلاس زبان‌هایی که دارای اثبات ناتراوای آماری هستند
- $\mathcal{CZK}$: کلاس زبان‌هایی که دارای اثبات ناتراوای محاسباتی هستند

حال سئوال مهم این است که این کلاس‌ها در چه نقطه‌ای از کلاس‌های پیچیدگی قرار دارند و قدرت توصیف آن‌ها به چه میزان است. برای این کار می‌توان از قضایای زیر استفاده کرد:

**قضیه**: با فرض وجود طرح تعهد[۴]، مسئله‌ی رنگ آمیزی گراف با 3 رنگ دارای اثبات ناتراوای محاسباتی است.

**اثبات**: رجوع کنید به [4]

**قضیه**: $\mathcal{NP} = \mathcal{CZK}$ [5]

**اثبات**: مسئله‌ی رنگ آمیزی گراف یک مسئله‌ی NP-Complete است، حال طبق قضیه‌ی قبل، چون این مسئله دارای اثبات ناتراوا است و تمام مسائل $\mathcal{NP}$ به مسئله‌ی رنگ آمیزی کاهش پیدا می‌کنند، لذا تمام مسائل $\mathcal{NP}$ دارای اثبات ناتراوا هستند. رابطه‌ی برگشت تساوی نیز بدیهی است. ■

حال طبق بحث‌هایی که تا به حال انجام شده است، رابطه‌ی کلاس‌های مطرح شده به صورت زیر است:

$$\mathcal{BPP} \subseteq \mathcal{PZK} \subseteq \mathcal{SZK} \subseteq \mathcal{CZK} \subseteq \mathcal{IP}$$

که اثبات می‌شود رابطه‌ی $\mathcal{CZK} \subseteq \mathcal{IP}$ با فرض وجود توابع یک‌طرفه، می تواند به $\mathcal{CZK} = \mathcal{IP}$ تبدیل شود.

## ۲-۴ اثبات دانش[۵]

تا بدین جای کار، مقوله‌ی اثبات درباره‌ی یک گزاره بحث شده بود که با زبان $\mathcal{NP}$ می توانستیم با استخراج یک شاهد ($w$) و اثبات وجود چنین شاهدی، گزاره را اثبات کنیم. به عنوان مثال در اثبات ناتراوای مسئله‌ی $GI$، دیدیم که می‌توان به صورت ناتراوا اثبات کرد که دو گراف یکریخت هستند. حال سئوال اینجاست که آیا می‌توان اثبات کرد دو گراف یکریخت هستند، همچنین اثبات کننده از این تابع یکریختی آگاه است؟ به عبارتی به جای اثبات وجود چنین تابع یکریختی، اثبات دانشِ چنین تابعی مطرح شود. به این دسته از اثبات‌های ناتراوا، اثبات دانش می‌گویند. تعریف ریاضی این سامانه‌ها در حوصله‌ی این گزارش نمی‌گنجد اما به طور خلاصه، برای این سامانه‌ها ماشین استخراجی[۶] معرفی می‌شود که می‌تواند

[۴] Commitment Scheme

[۵] Proof of Knowledge

[۶] Extractor Machine

با تعامل با ماشین اثبات کننده، این دانش را استخراج کند.

### ۲-۵ سامانه‌های استدلال

در تعریف سامانه‌های اثبات تعاملی، شرطی به نام صحت وجود دارد که نمایانگر قدرت این سامانه در برابر اثبات‌کننده‌های متقلب است که قصد اثبات گزاره‌های نادرست را دارند. در این تعریف بیان شد که سامانه‌ی ما باید توان مقابله با هر ماشین متقلب را داشته باشد یا به عبارتی

$$if\ x \notin \mathcal{L} \to \nexists B^* s.t. \Pr[< B^*, V > (x) = 1] \leq \frac{1}{3}$$

برقرار باشد. در سامانه‌های استدلال این شرط کمی ریلکس‌تر می‌شود و نیازی نیست که سامانه‌ی ما در مقابل تمام ماشین‌های اثبات‌کننده مقاوم باشد بلکه کافی است در مقابل تمام ماشین‌های اثبات‌کننده‌ی $PPT$ مقاوم باشد یا به عبارتی

$$if\ x \notin \mathcal{L} \to \nexists B^* \in PPT\ s.t. \Pr[< B^*, V > (x) = 1] \leq \frac{1}{3}$$

## ۳- اثبات غیرتعاملی

تا بدین‌جای کار سامانه‌های گفته شده همگی تعاملی بودند و اثبات‌کننده و تاییدکننده باید با گفتگو و ارسال و دریافت چالش‌های متفاوت، به یک نتیجه‌ی واحد قبول یا رد برسند. با این حال که این روش قدرت زیادی را وارد سامانه‌ی اثبات کرد، منتها در برخی از سناریوها تعامل مقدور نیست و هستار اثبات‌کننده نمی‌تواند همیشه برخط باشد تا به چالش‌ها پاسخ بدهد. از طرفی غیرتعاملی کردن این سامانه در نگاه اول، برگشت به مسئله‌ی $\mathcal{NP}$ است چرا که در آنجا نیز ما هیچونه تعاملی نداشتیم و عملا قدرت سامانه‌ی ما دوباره به $\mathcal{NP}$ نزول می‌کند. اما در [6] نشان داده شد که با ورود به فرض $CRS$[7] یا رشته‌ی مشترک مرجع، می‌توان دوباره به همین اهداف رسید.

**تعریف**. جفت ماشین‌های $(\mathcal{P}, \mathcal{V})$ یک سامانه‌ی اثبات غیرتعاملی را برای زبان $\mathcal{L}$ تشکیل می‌دهند اگر $\mathcal{V}$ چندجمله‌ای باشد و دو شرط زیر برقرار باشد:

- کامل‌بودن: برای هر $x \in \mathcal{L}$ داریم

$$\Pr\left[\mathcal{V}\big(x, R, P(x, R)\big) = 1\right] \geq \frac{2}{3}$$

که $R$ یک متغیر تصادفی با توزیع یکنواخت در $\{0,1\}^{poly(|x|)}$ است.

- برای هر $x \notin L$ و هر الگوریتم $B$ داشته باشیم

$$\Pr\left[\mathcal{V}\big(x, R, B(x, R)\big) = 1\right] \leq \frac{1}{3}$$

که مجددا $R$ یک متغیر تصادفی با توزیع یکنواخت در $\{0,1\}^{poly(|x|)}$ است.

در تعریف فوق $R$ همان $CRS$ است. در عمل رسیدن به این $CRS$ کار آسانی نیست و گاها با تکنیک‌هایی از قبیل MPC تولید می‌شود[7], [8].

---

۷ Common Reference String

## zkSNARK 1-3

می‌توان گفت در حال حاضر مهم‌ترین خانواده‌ی تکنیک‌های اثبات ناتراوای محاسبات ، خانواده‌ی zkSNARKها است. در ابتدا باید گفت zkSNARK مخفف عبارت " سامانه‌ی غیرتعاملی استدلال اثباتِ مختصر و ناتراوای دانش[۸]" است. همانطور که در نام این خانواده مشخص است، این اثبات باید ویژگی‌های زیر را داشته باشد:

- **ناتراوا** باشد
- **مختصر** باشد، بدین معنی که طول اثبات به اندازه‌ی ثابت و کوتاه باشد و به محاسبات بستگی نداشته باشد.
- **غیرتعاملی** باشد
- **سامانه‌ی استدلال** باشد
- **اثبات دانش** باشد

تقریبا می‌توان گفت که تمام ویژگی‌های خوب یک سامانه‌ی اثبات ناتراوا در این سامانه‌ها وجود دارد، به جز یک ویژگی که در بخش‌های آینده راجع به آن صحبت می‌شود.


شکل ۳: سامانه‌ی اثبات zk-SNARK

در شکل بالا مدل ساده‌ای از سامانه قابل مشاهده است.

## ۲-۳ ساخت zk-SNARK

برای ساخت سامانه‌های اثبات خانواده SNARKها روش‌های متفاوتی وجود دارد که عمدتا می‌توان آن‌ها را به دو دسته‌ی زیر تقسیم کرد:

- PCP –مبنا
- QAP-مبنا یا QSP-مبنا

در این گزارش بیشتر روی اثبات‌های QAP-مبنا تمرکز می‌کنیم چرا که کاربرد بیشتری نیز در جامعه‌رمزنگاری دارند. از آنجایی که توصیف کامل پروتکل [9] در حوصله‌ی این گزارش نمی‌گنجد به مفاهیم و نکات مهم آن می‌پردازیم و کلیات آن را توضیح می‌دهیم و از ورود به جزئیات می‌پرهیزیم.

## 3-3 SP[9] و QSP[10]

استراتژی تولید اثبات ناتراوا برای یک گزاره‌ی $\mathcal{NP}$ ، جستجو برای یک راه جدید جهت توصیف این زبان است. به احتمال زیاد تا به حال با دو روش توصیف این زبان آشنا شده‌اید، این دو روش عبارتند از:

- روش استفاده از رابطه‌ی شاهد و وجود تاییدکننده‌ی کارا که پیشتر توضیح داده شد
- روش وجود ماشین تورینگ غیرقطعیِ کارا برای حل مسئله

واضحا نتیجه‌ی این دو توصیف به یک مجموعه منجر می‌شود منتها نحوه‌ی مشخصه یابی[۱۱] برایشان فرق

[۸] Zero-Knowledge succinct non-interactive argument of knowledge
[۹] Span Program
[۱۰] Quadratic Span Program
[۱۱] Characterization

می‌کند. یکی از توصیف‌های بسیار مهم در تولید اثبات‌های خانواده‌ی SNARK ، توصیف QSP است که به آن می‌پردازیم. برای درک بهتر توصیف QSP بهتر است از توصیف SP برای زبان‌های سطح پایین تر شروع کنیم.

**تعریف.** یک SP [10]روی یک میدان $\mathcal{F}$ شامل یک بردار هدف $t$ روی $\mathcal{F}$، یک دسته بردار $\mathcal{V} = \{v_1, \dots, v_m\}$، یک افراز از اندیس‌های $\mathcal{I} = \{1, \dots, m\}$ به دو دسته‌ی $\mathcal{I}_{labeled}$ و $\mathcal{I}_{free}$ و یک افراز دیگر از $\mathcal{I}_{labeled}$ به صورت $\bigcup_{i\in[n],j\in\{0,1\}} \mathcal{I}_{ij}$ است. سایز یک SP ، $m$ است.

**تعریف.** می‌گوییم یک SP تابع $f$ را محاسبه می‌کند اگر برای همه‌ی $u \in \{0,1\}^n$ داشته باشیم، بردار هدف در span بردارهای متعلق به $u$ (در اندیس‌ها) قرار داشته باشد اگر و تنها اگر $f(u) = 1$.

نکته‌ی قابل توجه این است که دسته‌ی زبان‌هایی که SP با سایز چندجمله‌ای دارند همان کلاس NC است و حتی کل کلاس P را نیز شامل نمی‌شود. برای افزایش قدرت این توصیف جبری، مدل دیگری به نام QSP پیشنهاد شد که تعمیمی از مدل SP است.

تعریف. یک QSP [11]به نام Q روی میدان $\mathcal{F}$ شامل دو دسته چندجمله‌ای $\mathcal{V} = \{v_k(x): k \in \{0, \dots, m\}$ و $\mathcal{W} = \{\omega_k(x): k \in \{0, \dots, m\}$ و یک چندجمله‌ای هدف به نام $t(x)$ بر روی $\mathcal{F}[x]$ است. همچنین Q شامل یک افراز روی اندیس‌های $\mathcal{I} = \{1, \dots, m\}$ به دو دسته‌ی $\mathcal{I}_{labeled}$ و $\mathcal{I}_{free}$ و یک افراز دیگر از $\mathcal{I}_{labeled}$ به صورت $\bigcup_{i\in[n],j\in\{0,1\}} \mathcal{I}_{ij}$ است.

**تعریف.** می‌گوییم یک SP تابع $f$ را محاسبه می کند اگر $(a_1, \dots, a_m)$ و $(b_1, \dots, b_m)$ وجود داشته باشند به طوریکه داشته باشیم $f(u) = 1$ اگر و تنها اگر $a_k = 0 = b_k$ برای $k \notin \mathcal{I}_u$ و همچنین

$$t(x)|(v_0(x) + \sum_{k=1}^{m} a_k v_k(x)).(\omega_0(x) + \sum_{k=1}^{m} a_k \omega_k(x)).$$

## ۳-۴ نحوه استفاده از QSP در SNARKها

برای تولید SNARKهای مبتنی بر QSP در ابتدا می‌بایست مسئله‌خود را به یک نمونه از مسئله‌ی QSP تبدیل کنیم. پس از ای تبدیل، ایده‌ی کلی ماجرا این است که عبارت تقسیم‌پذیری QSP را که در بخش قبلی معرفی شد، در یک نقطه‌ی کاملا تصادفی چک کنیم[11], [12]. در واقع اثبات‌کننده باید به نحوی دانش این ضرایب $a_k$ها و $b_k$ها را به تاییدکننده اثبات کند. نقطه‌ی تصادفی پروتکل نیز درCRS قرار می‌گیرد.

## ۳-۵ مشکل خانواده‌ی SNARK

این سبک از سامانه‌های اثبات همانطور که گفته شد دارای ویژگی‌های بسیار خوبی در سمت تاییدکننده هستن و همچنین طول اثبات بسیار مناسبی دارند، به عنوان مثال در روش groth16 [12] پیچیدگی محاسباتی تاییدکننده حدودا به اندازه‌ی دو توان رسانی و یک ضرب در گروه‌های دوخطی است که این ویژگی فارغ از تابع مورد محاسبه و ورودی این تابع می‌باشد. اما در سمت اثبات‌کننده هیچگونه تضمینی مبنی بر مناسب بودن پیچیدگی وجود ندارد. این مسئله شاید تا به حال چندان مهم نبوده است چرا که معمولا کاربردهای

سامانه‌های اثبات در مسائل بسیار پیچیده نبوده است و در مسائلی مانند اثبات هویت[13] و زنجیره‌ی قالبی[14] که اندازه‌ی مسئله بزرگ نیست استفاده می‌شده. منتها با بوجود آمدن نیازهای روزافزون امنیت در مسائل مختلف و با پی‌بردن به پتانسیل بسیار بالای اثبات‌های ناتراوا در حل چالش‌های امنیتی مختلف، نیاز به اثبات‌کننده‌های به مراتب کاراتر حس شده است. به عنوان مثال در کاربردهای هوش مصنوعی[15] و امنیت شبکه[16] ابعاد QSP بسیار بزرگ می‌شود و پیچیدگی اثبات‌کننده به حدی زیاد می‌شود که آن را غیرقابل پیاده سازی می‌کند.

## 4- پروتکل GKR

راه حل جامعی در سال 2015 برای اثبات صحت محاسبه‌ی مدارهای محاسباتی ارائه شد[17] که درآن پیچیدگی محاسبه‌ی اثبات‌کننده بسیار قابل قبول بود. این پروتکل به نام GKR معروف شد که ابتدای اسامی نویسندگان مقاله‌ی فوق‌الذکر هستند. در این روش هم تاییدکننده و اثبات‌کننده کارا هستند اما تاییدکننده به اندازه‌ی روش‌های SNARK کارا نیست و خصوصیات جذاب آن مانند مختصربودن، غیر تعاملی بودن و حتی ناتراوا بودن را ندارد. البته با تکنیک‌هایی می‌توان به سادگی آن را تبدیل به یک پروتکل غیرتعاملی [13] و ناتراوا [18] کرد. برای بررسی این پروتکل [19] در ابتدا می‌بایست یکی از بلوک‌های اساسی این پروتکل به نام پروتکل Sumcheck را بررسی کرد.

### ۴-۱ پروتکل Sumcheck

در این پروتکل یک هستار به دیگری ثابت می کند که جمع یک چندجمله‌ای روی تمام ورودی‌های ممکن، یک عدد خاص است. به صورت رسمی و ریاضی داریم:

فرض کنید $g(x_1, \dots x_v): \{0,1\}^v \to \mathcal{F}$ یک چندجمله‌ای چندمتغیره است و داشته باشیم

$$\sum_{b_1 \in \{0,1\}} \sum_{b_2 \in \{0,1\}} \dots \sum_{b_v \in \{0,1\}} g(b_1, \dots, b_v) = H$$

هستار اثبات‌کننده ($\mathcal{P}$) ادعا می‌کند که حاصل این جمع برابر است با $C_1$. سپس طبق پروتکل زیر این ادعا را اثبات می‌کند:

**پروتکل.**

- در ابتدای پروتکل $\mathcal{P}$ مقدار $C_1$ را به $\mathcal{V}$ ارسال می‌کند و ادعا می‌کند که همان $H$ است.
- در اولین مرحله هستار $\mathcal{P}$ چندجمله‌ای تک متغیره‌ی $g_1(X_1)$ را برای $\mathcal{V}$ ارسال می‌کند و ادعا می‌کند که برابر با چندجمله‌ای زیر است

$$\sum_{(x_2, \dots, x_v) \in \{0,1\}^{v-1}} g(X_1, x_2, \dots, x_v)$$

- در مرحله‌ی بعد $\mathcal{V}$ پس از دریافت $g_1(X_1)$ چک می‌کند که آیا

$$g_1(0) + g_1(1) = C_1$$

و $g_1$ تک متغیره باشد و درجه‌ی آن حداکثر برابر با درجه‌ی متغیر اول در چندجمله‌ای $g$ باشد.

سپس $\mathcal{V}$ عدد تصادفی $r_1$ را انتخاب می‌کند و برای $\mathcal{P}$ ارسال می‌کند.

- در دور $j$ام از پروتکل که $1 < j < v$ هستار $\mathcal{P}$ چندجمله‌ای $g_j(X_j)$ را برای $\mathcal{V}$ ارسال می‌کند و ادعا می‌کند که این چندجمله‌ای تک متغیره برابر است با

$$\sum_{(x_{j+1},\ldots,x_v)\in\{0,1\}^{v-j}} g(r_1,\ldots,r_{j-1},X_j,x_{j+1},\ldots,x_v)$$

مجددا $\mathcal{V}$ چک می‌کند که آیا $g_j$ چندجمله‌ای تک متغیره است، آیا درجه‌ی آن برابر با درجه‌ی $j$امین متغیر در $g$ است و آیا تساوی زیر برقرار است یا خیر

$$g_{j-1}(r_1) = g_j(0) + g_j(1)$$

سپس مقدار تصادفی $r_j$ را برای $\mathcal{P}$ ارسال می‌کند.

- در نهایت در دور $v$ام، هستار $\mathcal{P}$ چندجمله‌ای $g_v(X_v)$ را برای $\mathcal{V}$ ارسال می‌کند و ادعا می‌کند که برابر با چندجمله‌ای زیر است

$$g(r_1,\ldots,r_{v-1},X_v)$$

سپس $\mathcal{V}$ چک می‌کند که آیا درجه‌ی $g_v$ برابر است با درجه‌ی $v$امین متغیر از $g$ و آیا تساوی زیر برقرار است یا خیر

$$g_{v-1}(r_{v-1}) = g_v(0) + g_v(1)$$

در نهایت $\mathcal{V}$ یک مقدار تصادفی $r_v$ را انتخاب می‌کند و چک می‌کند که آیا $g_v(r_v) = g(r_1,\ldots,r_v)$ برقرار است یا خیر

در هر کدام از مراحل بالا اگر $\mathcal{V}$ چک را تایید نکند، در اثبات قانع نمی‌شود و پروتکل قطع می‌شود.

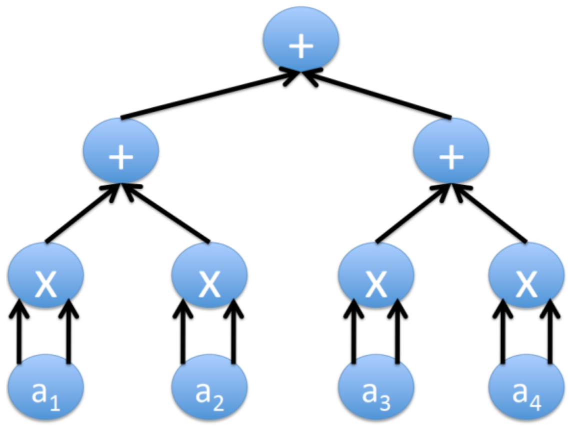
جدول ۱: پیچیدگی پروتکل Sumcheck

| پیچیدگی $\mathcal{P}$ | پیچیدگی $\mathcal{V}$ | تعداد دور | پیچیدگی مخابراتی |
|---|---|---|---|
| $O(2^v T)$ | $O(v + \sum_{i=1}^{v} \deg_i(g)) + T$ | $v$ | $O(\sum_{i=1}^{v} \deg_i(g))$ field elements |

نکته‌ی جالب این پروتکل این است که تاییدکننده دیگر نیازی نیست تا $2^v$ کار انجام دهد بلکه تنها به میزان تعداد متغیرها کار انجام می‌دهد که همین مسئله پروتکل را جذاب می کند. حال با استفاده از همین پروتکل به عنوان بلوک اصلی، پروتکل GKR را شرح می‌دهیم.

## ۲-۴ شرح پروتکل

پروتکل GKR براساس مدار محاسباتی عمل می‌کند. در واقع در ابتدا باید محاسبات خود را به شکل یک مدار محاسباتی بر روی مقادیری از میدان $\mathcal{F}$ بدهیم و سپس پروتکل را روی آن اجرا کنیم.




شکل ۴: نمونه ای از مدار محاسباتی مورد استفاده در پروتکل GKR

در این پروتکل $\mathcal{P}$ ادعایی درباره‌ی خروجی مدار دارد و می‌خواهد آن را اثبات کند. در ابتدا می‌بایست چند علامت‌گذاری را مشخص کنیم:

- $C$ : مدار محاسباتی لایه‌ای در $\mathcal{F}$

- $S$: اندازه مدار
- $d$: عمق مدار
- $S_i = 2^{k_i}$: تعداد گیت‌ها در لایه‌ی $i$ام مدار
- تابع $W_i$: تابعی است که برچسب گیت در لایه‌ی $i$ام را می‌گیرد و خروجی آن گیت را می‌دهد.

$$W_i: \{0,k\}^{k_i} \to \mathcal{F}$$

- توابع $in_{1,i}$ و $in_{2,i}$ که داریم:

$$in_{1,i}, in_{2,i}: \{0,1\}^{k_i} \to \{0,1\}^{k_{i+1}}$$

که در آن برچسب گیت در لایه $i$ به عنوان ورودی و برچسب گیت درلایه $i+1$ به عنوان خروجی است و مشخص می‌کند که ورودی هر گیت از چه گیتی در لایه‌ی قبل وارد می‌شود.

- توابع $add_i, mult_i$ که داریم:

$$add_i, mult_i: \{0,1\}^{k_i+2k_{i+1}} \to \{0,1\}$$

که با هم توابع سیم‌کشی[12] لایه $i$ را تشکیل می‌دهند. این توابع اسامی 3 گیت $(a,b,c)$ را به عنوان ورودی میپذیرند و تنها زمانی خروجی 1 می‌دهند که

$$(b,c) = (in_{1,i}(a), in_{2,i}(a))$$

و گیت a برای تابع add باید جمع و برای تابع mult باید ضرب باشد.

طبق تعاریف بالا خواهیم داشت

$$\widetilde{W_\iota}(z) = \sum_{(b,c)\in\{0,1\}^{k_{i+1}}} \widetilde{add_\iota}(z,b,c)(\widetilde{W_{\iota+1}}(b) + \widetilde{W_{\iota+1}}(c)) + \widetilde{mult_\iota}(z,b,c)(\widetilde{W_{\iota+1}}(b) + \widetilde{W_{\iota+1}}(c))$$

حال $\mathcal{P}$ با ادعایی درباره‌ی مقدار $W_o(z)$ها شروع می‌کند. برای این ادعا، این هستار باید پروتکل sumcheck را روی این چندجمله‌ای اجرا کند. پس از اجرای پروتکل Sumcheck ، هنوز مقادیر $W_{i+1}$ ها مشخص نیستند، لذا $\mathcal{V}$ نمی‌تواند مقدار $g(r_1, \dots, r_v)$ را بدست بیاورد. به همین خاطر این پروتکل روی لایه‌ی بعدی و روی جمع زیر اجرا می‌شود

$$\widetilde{W_{\iota+1}}(z) = \sum_{(b,c)\in\{0,1\}^{k_{i+1}+1}} \widetilde{add_{\iota+1}}(z,b,c)(\widetilde{W_{\iota+2}}(b) + \widetilde{W_{\iota+2}}(c)) + \widetilde{mult_{\iota+1}}(z,b,c)(\widetilde{W_{\iota+2}}(b) + \widetilde{W_{\iota+2}}(c))$$

و به همین ترتیب تا لایه‌ی $d$ام که ورودی است اجرا می‌شود. چون ورودی‌ها مشخص هستند لذا با جایگذاری آن‌ها پروتکل Sumcheck به صورت کامل اجرا می‌شود.

جدول ۲: پیچیدگی پروتکل GKR

| پیچیدگی $\mathcal{P}$ | پیچیدگی $\mathcal{V}$ | تعداد دور |
|---|---|---|
| $O(S^3)$ | $O(n + d\log(S))$ | $d\log(S)$ |

## 5- جمع‌بندی و نتیجه گیری

در این گزارش در ابتدا با مفاهیم اثبات و اثبات‌های ناتراوا و انواع آن آشنا شدیم و سپس یکی از مهم‌ترین

---

۱۲ Wiring Predicate

خانواده‌های این اثبات‌ها، یعنی zkSNARK ها را مشاهده کردیم. سپس دیدیم که در کاربردهای سنگین‌تر و با توابع پیچیده‌تر ، پیچیدگی اثبات‌کننده بسیار بالا می‌رود و عملا آن را غیر قابل استفاده می‌سازد، لذا پروتکل GKR را بررسی کردیم.

اثبات‌های ناتراوا برای محاسبات اثبات‌پذیر روز به روز نقش پررنگ‌تری را در تکنولوژی محاسبات ابری و دیگر زمینه‌های علوم کامپیوتر مانند هوش مصنوعی بازی می‌کنند. به همین خاطر می‌بایست کارایی آن‌ها را در کاربردهای صنعتی تضمین کرد. هرچند هنوز چالش‌های بسیار در تولید اینگونه اثبات‌ها مانند مقاومت آن‌ها در برابر حملات کوانتومی، پیش‌پردازش پروتکل، اندازه و تولید CRS ، عدم تبدیل برخی از محاسبات به مدار و ... وجود دارد، اما می‌توان این تکنیک رمزنگاری را یکی از اجزای مهم رمزنگاری مدرن دانست و از کاربردهای آن در زمینه‌های مختلف اعم از رایانش ابری، هوش مصنوعی، امنیت شبکه، امنیت نرم‌افزارهای تحت وب و ... استفاده کرد.

## ۶- مراجع


[1] B. P. Rimal, E. Choi, and I. Lumb, “A taxonomy and survey of cloud computing systems,” NCM 2009 - 5th International Joint Conference on INC, IMS, and IDC, pp. 44–51, 2009, doi: 10.1109/NCM.2009.218.

[2] S. Goldwasser, S. Micali, and C. Rackoff, “The Knowledge Complexity of Interactive Proof Systems,” http://dx.doi.org/10.1137/0218012, vol. 18, no. 1, pp. 186–208, Jul. 2006, doi: 10.1137/0218012.

[3] A. Shamir, “IP = PSPACE,” Journal of the ACM (JACM), vol. 39, no. 4, pp. 869–877, Oct. 1992, doi: 10.1145/146585.146609.

[4] “Foundations of Cryptography: A Primer - Oded Goldreich - Google Books.” https://books.google.com/books?hl=en&lr=&id=1u7j2ljt0bQC&oi=fnd&pg=PA7&dq=oded+goldreich+foundations+of+cryptography&ots=vW9t2ud-aD&sig=iRpiq6FIq6LBg8KuXP6x0j7wVuc#v=onepage&q=oded%20goldreich%20foundations%20of%20cryptography&f=false (accessed Apr. 09, 2022).

[5] O. Goldreich, S. Micali, and A. Wigderson, “Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems,” Journal of the ACM (JACM), vol. 38, no. 3, pp. 690–728, Jul. 1991, doi: 10.1145/116825.116852.

[6] M. Blum, P. Feldman, and S. Micali, “Non-interactive zero-knowledge and its applications,” Proceedings of the Annual ACM Symposium on Theory of Computing, pp. 103–112, 1988, doi: 10.1145/62212.62222.

[7] “Setup Ceremonies - ZKProof Standards.” https://zkproof.org/2021/06/30/setup-ceremonies/ (accessed Jun. 30, 2022).

[14] “Privacy-protecting digital currency | Zcash.” https://z.cash/ (accessed Apr. 09, 2022).

[15] “ZEN: Efficient Zero-Knowledge Proofs for Neural Networks.” https://eprint.iacr.org/archive/2021/87/20210127:132648 (accessed May 29, 2022).

[16] “Cryptology ePrint Archive: Report 2021/1022 - Zero-Knowledge Middleboxes.” https://eprint.iacr.org/2021/1022 (accessed Apr. 09, 2022).

[17] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum, “Delegating Computation,” Journal of the ACM (JACM), vol. 62, no. 4, Sep. 2015, doi: 10.1145/2699436.

[18] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler, and M. Walfish, “Doubly-Efficient zkSNARKs Without Trusted Setup,” Proceedings - IEEE Symposium on Security and Privacy, vol. 2018-May, pp. 926–943, Jul. 2018, doi: 10.1109/SP.2018.00060.

[19] J. Thater, Proofs, Arguments, and Zero-Knowledge. 2022. Accessed: Jul. 01, 2022. [Online]. Available: https://people.cs.georgetown.edu/jthaler/ProofsArgsAndZK.html

[8] “Diving into the zk-SNARKs Setup Phase | by Daniel Benarroch | QEDIT | Medium.” https://medium.com/qed-it/diving-into-the-snarks-setup-phase-b7660242a0d7 (accessed Jun. 30, 2022).

[9] M. Petkus, “Why and How zk-SNARK Works,” Jun. 2019, doi: 10.48550/arxiv.1906.07221.

[10] S. Jukna, “Span Programs,” pp. 205–218, 2001, doi: 10.1007/978-3-662-04650-0_18.

[11] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, “Quadratic Span Programs and Succinct NIZKs without PCPs,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 7881 LNCS, pp. 626–645, 2013, doi: 10.1007/978-3-642-38348-9_37.

[12] J. Groth, “On the size of pairing-based non-interactive arguments,” Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 9666, pp. 305–326, 2016, doi: 10.1007/978-3-662-49896-5_11/TABLES/2.

[13] U. Feige, A. Fiat, and A. Shamir, “Zero-knowledge proofs of identity,” Journal of Cryptology 1988 1:2, vol. 1, no. 2, pp. 77–94, Jun. 1988, doi: 10.1007/BF02351717.