دانشکده مهندسی برق

امنیت در سیستم ها و شبکه های کامپیوتری

گزارش پروژه

عنوان:

Zk-SNARK

نگارنده:

علیرضا شیرزاد

شماره دانشجویی:

99201754

خرداد 1400

مقدمه

در سالیان اخیر با گسترش پارادایم های مختلف محاسبات، اعم از پردازش ابری[1]، پردازش مه[2]، پردازش لبه ای[3] و پردازش شبکه ای[4] قدرت محاسباتی به شکل یک سرویس تبدیل شده است و همگان می توانند از آن ها استفاده کنند. به صورت موازی، در دنیای اینترنت اشیا ابزاره هایی تعبیه شده اند که قدرت محاسباتی بسیار اندکی دارند و توانایی انجام پردازش های سنگین اعم از الگویتم های بهینه سازی، هوش مصنوعی ، مدل سازی و ... را ندارند لذا مجبور می شوند تا این محاسبات را به مراکز محاسباتی فوق الذکر، مثلا به مراکز محاسبات ابری ، برون سپاری کنند. این مدل سیستمی که محاسبات هستار ها به شخص ثالثی سپرده شود و نتایج آن به سمت هستار بازگردد، بسیار مورد توجه قرار گرفته زیرا ابزاره های ما می توانند با توان محاسباتی کم و تنها دسترسی به شبکه، به حیات خود ادامه دهند.

اما این پارادایم به همراه خود چالش های امنیتی بسیاری را به بار آورده است. بسیاری از اطلاعات مورد پردازش، که توسط ابزاره[5] ها تولید می شوند کاملا محرمانه و سری می باشند. به عنوان مثال برخی از این ابزاره ها اطلاعات پزشکی یک کاربر را پایش می کند، یک ابزاره ممکن است دما و شرایط محیطی یک رااکتور یا پهباد را ثبت و ضبط کند. در اینجا چالش بسیار مهم محرمانگی پیش می آید و این مسئله مطرح می شود که چگونه می توان اطلاعات را به صورت محرمانه برون سپاری کرد به نحوی که شخص ثالث بتواند روی آن پردازش انجام دهد. این چالش هنوز سئوال بزرگی است که حجم بزرگی از پژوهش روی آن انجام می شود و جواب هایی مانند رمزنگاری های کاملا همریخت از جمله جواب های داده شده به این چالش می باشد.

اما چالش بسیار مهم تر که در این پژوهش به آن می پردازیم، چالش صحت پاسخ شخص ثالث است. نتیجه ی پردازش روی داده های ارسال شده، قاعدتا به صورت یک پاسخ به سمت ابزاره های ما ارسال می شوند. این جواب ها ممکن است محرک ابزاره های مهم دیگری باشد، مثلا قدرت یک موتور را زیاد یا کم کند، لذا صحت و سقم این جواب ها بسیار مهم است. به دلایل مختلفی ، پاسخ پردازش می تواند غلط برگردانده شود، این دلایل می تواند از خرابی های بسیار ساده در مرکز محاسبات شروع شود تا دلایل امنیتی و حملات احتمالی به سیستم مشتری. لذا بسیار مهم است که نتیجه ی پردازش به ابزاره های مشتری اثبات شود و سپس باعث تغییرات یا محرک سیستم بشود. مسئله ی صحت را می توان به صورت یک تابع ریاضی مدل کرد که در آن تابع f و ورودی x از سمت ابزاره به سمت شخص ثالث ارسال می شود و شخص ثالث پس از محاسبه ی مقدار $y = f(x)$ ، y را بر می گرداند.

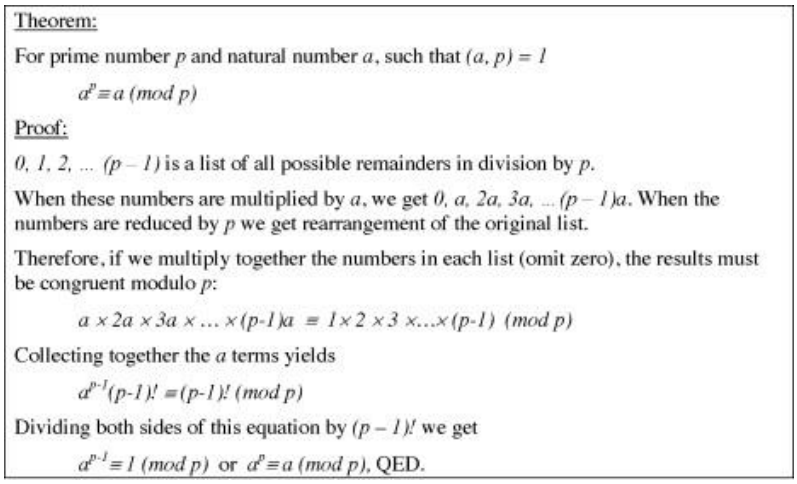
در این پژوهش به توصیف راهی می پردازیم که می توان به وسیله ی آن صحت نتایج یک محاسبه را اثبات کرد. این راه "zk-SNARK" نام دارد و این قدرت را در اختیار ما قرار می دهد که نتیجه یک محاسبه را برای شخص دیگری اثبات کنیم. همچنین این پروتکل، این امکان را برای ما فراهم می سازد که بخشی از جواب را که آن را گواه می نامیم، بتوانیم به صورت دانش صفر منتقل

[1] Cloud Computing
[2] Fog Computing
[3] Edge Computing
[4] Grid Computing
[5] Device

کنیم. پروتکل zk-SNARK نتیجه ی پژوهش های دانشمندان علوم کامپیوتر در زمینه ی رمزنگاری و نظریه ی پیچیدگی محاسبه می باشد که با مفاهیم پایه ای این علم یعنی NP پایه گذاری شده است تا بدینجا رسیده. بخش های بعدی این گزارش تشکیل شده است از بخش های مروری بر مفهوم اثبات، zk-SNARK و بخش یک نمونه کاربرد عملی در صنعت.

مروری بر مفهوم اثبات

اثبات به معنای کلاسیک خود، شامل عباراتی پی در پی می شود که هر گزاره در ادامه ی گزاره ای دیگر آمده است و با قبول هر گزاره تا انتهاب اثبات، گزاره ی مورد نظر اثبات می شود. به این اثبات ها، که بیشتر در دنیای ریاضیات و منطق با آن ها آشنا شده ایم، اثبات های نحوی[6] می گویند. در این اثبات ها ذاتا کرانی برای طول اثبات و زمان قانع شدن یا زمان تولید اثبات وجود ندارد و صرفا خود اثبات مهم است. به عنوان مثال اثبات اینکه تعداد اعداد اول نامحدود است یا اثبات قضیه ی کوچک فرما، از نوع اثبات های نحوی می باشند.



Theorem:

For prime number p and natural number a, such that $(a, p) = 1$

$$a^p \equiv a \ (mod\ p)$$

Proof:

$0, 1, 2, \ldots (p - 1)$ is a list of all possible remainders in division by p.

When these numbers are multiplied by a, we get $0, a, 2a, 3a, \ldots (p - 1)a$. When the numbers are reduced by p we get rearrangement of the original list.

Therefore, if we multiply together the numbers in each list (omit zero), the results must be congruent modulo p:

$$a \times 2a \times 3a \times \ldots \times (p-1)a \equiv 1 \times 2 \times 3 \times \ldots \times (p-1) \ (mod\ p)$$

Collecting together the a terms yields

$$a^{p-1}(p-1)! \equiv (p-1)! \ (mod\ p)$$

Dividing both sides of this equation by $(p - 1)!$ we get

$$a^{p-1} \equiv 1 \ (mod\ p) \text{ or } a^p \equiv a \ (mod\ p), \text{QED.}$$

شکل **1**: مثالی از یک اثبات نحوی، قضیه ی کوچک فرما

با توصیف زبان NP در مقاله های [C71] و [L73] ، دنیای علوم کامپیوتر به نحوی درگیر تعریف اثبات شد. در این مقالات زبان NP به صورت زیر تعریف شد:

تعریف. زبان NP زبانی است که برای آن ماشین تورینگ چند جمله ای R_L وجود دارد به صورتی که

$$\boldsymbol{\mathcal{L} = \{x \mid \exists w,\ |w| = poly(|x|),\ R_L(x, w) = 1\}}$$

این تعریف بیان می کند که برای اثبات عضویت یک رشته در زبان L که $L \subset NP$ تنها کافی است که اثبات (گواه یا شاهد[7]) را توسط یک ماشین چندجمله ای چک کنیم.

پس از تعریف NP ، یکی از مقالات مهم در زمینه ی اثبات، معرفی اثبات های تعاملی و دانش-صفر بود. در واقع تا بدین جای کار اثبات های ما بدین صورت بود که یک اثبات کننده، اثبات را به صورت کامل می نوشت و یک تایید کننده با خواندن آن اثبات باید

---

[6] Syntactic

[7] Witness

قانع می شد. در مقاله ی [GMR85] مفهوم تازه ای از اثبات توسط نگارندگان معرفی شد. این اثبات ها به اثبات های تعاملی معروف شدند. ذات این اثبات ها به اثبات های استفاده شده در دنیای روزمره ی ما شباهت بیشتری دارد، چرا که در دنیای واقعی، مثلا در اثبات هایی که یک استاد به دانشجویان خود ارائه می دهند، گاها پرسش هایی توسط افراد تایید کننده، در گلوگاه های روند اثبات مطرح می شود و در پاسخ، فرد اثبات کننده ابهامات را برطرف می کند. همچنین برای افزایش قدرت اینگونه اثبات ها، تایید کننده اجازه دارد که سکه پرتاب کند و به صورت تصادفی سئوالاتی را از اثبات کننده بپرسد که این اتفاق، قدرت این سیستم اثبات را به شدت افزایش می دهد. به طور خلاصه، با ورود مفاهیمی من جمله تعامل و تصادف در سیستم اثبات، سعی شد که از سیستم های اثبات کلاسیک نحوی به سمت اثبات های مدرن تر، کوتاه تر و از همه مهم تر، دانش–صفر مهاجرت شود.

منظور از دانش–صفر بودن یک اثبات این است که یک اثبات، یا به طور کلی پیام های ارسالی از سمت اثبات کننده به سمت تایید کننده، چه مقدار دانش را منتقل می کنندو توجه داشته باشید که مفهوم دانش[8] با مفهوم اطلاعات[9] تفاوت دارد. هر بیت ارسالی فارغ از محتوای آن حاوی اطلاعات است اما ممکن است حاوی دانش نباشد، در واقع زمانی یک رشته حاوی دانش است که خود گیرنده نتواند در زمان چندجمله ای محتوای رشته را تولید کند. یعنی با فرض $P \not\subset NP$ ، هر رشته ی گواه $L \subset NP$ حاوی دانش است زیرا بدست آوردن آن قطعا در زمانی بیشتر از چندجمله ای صورت گرفته. حال سئوال مهمی که پیش می آید این است که در اثبات ها، چه مقدار دانش منتقل می شود؟ چه مقدار از این دانش برای اثبات است؟ چه مقدار از این دانش ماهیت حشو دارد و اگر نبود هم اثبات پابرجا باقی می ماند؟ مثلا به مثال زیر توجه کنید.

زبان L را بگیرید مجموعه تمام عبارت های باینری ای که در یک نقطه 1 بشوند (مسئله ی معروف 3-SAT). برای اثبات اینکه $x \in L$، بدیهی ترین کار این است که یک نقطه ی معتبر از متغیرهای عبارت x پیدا کنیم و این عدد گذاری را به عنوان شاهد تحویل دهیم. اما سئوال مهم این است که این شاهد چقدر دانش مازاد برا اثبات جمله ی "x عضو L است" منتقل می کند؟ بدیهتا مقدار دانش منتق شده بسیار بیشتر از صرفا اثبات این جمله است که چرا عدد گذاری دقیق متغیر ها نیز ارسال شده. سئوال این است که آیا می توان اثبات ها را به گونه ای طراحی کرد هیچ دانشی به جز متن اثبات، برای تایید کننده ارسال نشود؟ در مقاله ی [GMA91] به این موضوع پرداخته شده است و نتیجه این است که بله، برای هر زبانی عضو NP اثبات هایی به شکل دانش–صفر وجود دارد.

پس از به اوج رسیدن اثبات های تعاملی با پروتکل های معروفی من جمله پروتکل *Schnorr* یا پروتکل های سیگما، کماکان سئوال مهمی در ذهن دانشمندان علوم کامپیوتر بی جواب باقی مانده بود و آن هم این بود که آیا می توان قدرت اثبات های دانش–صفر تعاملی را به شکل غیر تعاملی استفاده کرد؟ آیا می توان تعداد راند های تعامل را در سیستم ها اثبات به یک رساند؟ تعدادی راه پیشنهاد شد که مهم ترین آن ها دو تئوری به نام PCP و QAP بود. در PCP ، اثبات کننده یک رشته ی طولانی از اثبات را برای تایید کننده ارسال می کند که تایید کننده پس از نمونه برداری از چند بیت و در زمان بسیار کوتاه با احتمال بسیار زیادی قانع می شود. هرچند قضیه ی *PCP* بسیار قدرتمند است اما ارزشی در دنیای عملی ندارد چرا که طول اثبات ارسالی بسیار بزرگ است و

[8] Knowledge
[9] Information

زمان تولید اثبات نیز هیچ محدودیتی ندارد. در این پژوهش به سراغ QAP می رویم و پس از آن وارد دنیای اثبات های غیر تعاملی که به اختصار $NIZK$ می شویم. در واقع $SNARK$ ها نمونه ی بسیار عالی $NIZK$ها می باشند.

پیش از ورود به مبحث zk- $SNARK$ لازم است ویژگی های مهم یک سیستم اثبات را با هم مرور کنیم:

1. کامل بودن[10]: اثبات های متعارف و درست بتوانند به درستی تایید کننده را قانع کنند.
2. منطقی بودن[11]: اثبات های نادرست به احتمال بسیار کم ، بتوانند به غلط تایید کننده را قانع کنند.
3. دانش-صفر بودن: هیچ دانشی مازاد بر جمله ی مورد اثبات، به تاییدکننده منتقل نشود.

[10] Completeness
[11] Soundness

Zk-SNARK

این پروتکل یکی از پروتکل های معروفی است که به اثبات کننده این اجازه را می دهد که محاسبه ی یک تابع، برنامه ، مدار یا هر مدل محاسباتی دیگری را اثبات کند. در واقع این پروتکل نتیجه ی تلاش دانشمندان علوم کامپیوتر در زمینه ی اثبات های غیر تعاملی می باشد و تقریبا به دنیای عملی نزدیک شده است و در رمز ارز Z–Cash هم به صورت واقعی پیاده شده است. در این پژوهش به نحوه ی کارکرد این پروتکل می پردازیم.

در ابتدا لازم است $zk - SNARK$ را تعریف کنیم:

- ✓ *Zk= Zero–Knowledge*: دانش–صفر است
- ✓ *S= Succinct*: اثبات بسیار کوتاه است.
- ✓ *N= Non–Interactive*: اثبات غیر تعاملی است و تنها یک راند دارد.
- ✓ *AR= Argument*: در این پروتکل فرض میکنیم که قدرت محاسباتی اثبات کننده محدود است.
- ✓ *K= Knowledge*: در این اثبات دانستن یک چیز، من جمله گواه حل یک مسئله، مورد بحث است.

به طور کلی این پروتکل، در ابتدا یک برنامه (محاسبه) را به مدار های محاسباتی و منطقی تبدیل می کند و اثبات می کند که حاصل خروجی این مدار درست است. برای اثبات صحبت خروجی یک مدار، آن را به تعدادی چند جمله ای تبدیل می کند و گزاره ی به خصوصی را درباره ی این چند جمله ای ها اثبات می کند. در ابتدا ابزارهای مورد نیاز خود را مطرح می کنیم و در نهایت پروتکل *zk– SNARK* را با هم می سازیم.

بحث را با این سئوال آغاز می کنیم که آیا می شود اثبات کرد که "چند جمله ای $p(x)$ای را می شناسیم که ریشه های $a_1, a_2, \dots a_n$ را داشته باشد"؟ معادلا آیا می توان اثبات کرد که $p(x)$ ای را می شناسیم که $p(x) = t(x)h(x)$ به صورتی که $t(x) = (x - a_1)(x - a_2) \dots (x - a_n)$ ؟ برای اثبات در ابتدا به یک خاصیت مهم چندجمله ای ها توجه می کنیم. برای اثبات دانش یک چند جمله ای به خصوص همیشه می توان از نقاط آن چند جمله ای استفاده کرد به عنوان مثال برای اثبات دانش $p(x)$ ، تایید کننده می تواند یک s تصادفی مشخص کند و اثبات کننده در پاسخ به آن $p(s)$ را ارسال کند. حال از آنجاییکه تقلب در این طرح بدین معنی است که به جای $p(s)$ ما یک $p'(s)$ را ارسال کنیم، احتمال تقلب بسیار پایین است زیرا معادله $p(x) = p'(x)$ تنها به اندازه درجه ی این چند جمله ای ها جواب دارد و مجموعه ی نمونه برداری تصادفی ما بی نهایت است لذا احتمال تقلب عملا صفر است. لذا این روش به ما یک ایده می دهد که نیازی نیست برای کل یک چندجمله ای یک گزاره اثبات شود، بلکه اثبات گزاره ها در نقاط محدود نیز کافی است.

حال برمی گردیم به سئوال اول: می خواهیم اثبات کنیم که $p(x) = t(x)h(x)$ ، یا به عبارت بهتر می خواهیم اثبات کنیم که چنین $p(x)$ ای می شناسیم. این اثبات بر این پایه بنا نهاده شده است که ما دانش را درباره ی یک نقطه بدست آوریم و با احتمال بسیار زیادی قانع شویم. اما این نقطه چگونه باید تولید شود تا این اثبات برای همگان معتبر باشد. به عبارتی اثبات تولید شده می

بایست برای هر تایید کننده ای قابل وارسی باشد، نه صرفا برای یک تایید کننده ی به خصوص ، درست مثل یک امضای دیجیتال وارس عمومی. به همین دلیل مرحله ی اولیه ای قبل از اثبات مطرح می شود به نام مرحله ی راه اندازی[12] که در آن نقاط مورد نظر تولید می شود. پروتکل نهایی به صورت زیر است:

مرحله راه اندازی:

- Setup
  - sample random values s, α
  - calculate encryptions g^{α} and $\left\{g^{s^i}\right\}_{i\in[d]}, \left\{g^{\alpha s^i}\right\}_{i\in\{0,\ldots,d\}}$
  - proving key: $\left(\left\{g^{s^i}\right\}_{i\in[d]}, \left\{g^{\alpha s^i}\right\}_{i\in\{0,\ldots,d\}}\right)$
  - verification key: $\left(g^{\alpha}, g^{t(s)}\right)$

مرحله تولید اثبات:

- Proving
  - assign coefficients $\{c_i\}_{i\in\{0,\ldots,d\}}$ (i.e., knowledge), $p(x) = c_d x^d + \cdots + c_1 x^1 + c_0 x^0$
  - calculate polynomial $h(x) = \frac{p(x)}{t(x)}$
  - evaluate encrypted polynomials $g^{p(s)}$ and $g^{h(s)}$ using $\left\{g^{s^i}\right\}_{i\in[d]}$
  - evaluate encrypted shifted polynomial $g^{\alpha p(s)}$ using $\left\{g^{\alpha s^i}\right\}_{i\in\{0,\ldots,d\}}$
  - sample random δ
  - set the randomized proof $\pi = \left(g^{\delta p(s)}, g^{\delta h(s)}, g^{\delta\alpha p(s)}\right)$

مرحله تایید اثبات:

---

[12] Setup

- Verification
  - parse proof π as $\left(g^{p}, g^{h}, g^{p'}\right)$
  - check polynomial restriction $e\left(g^{p'}, g\right) = e\left(g^{p}, g^{\alpha}\right)$
  - check polynomial cofactors $e\left(g^{p}, g\right) = e\left(g^{t(s)}, g^{h}\right)$

حال توضیحاتی را درباره پروتکل بالا می دهیم. در مرحله ی راه اندازی، که یک بار برای همیشه انجام می شود،دو پارامتر تصادفی α و s را انتخاب می شود و بر اساس آن، توان های چند جمله ای به صورت رمز شده ساخته می شود، همچنین این توان ها به صورت α-شیفت یافته نیز تولید می شوند تا اثبات کننده بتواند نشان دهد که تنها از چند جمله ای مورد نظر در مراحل اثبات استفاده شده است. این ها کلید اثبات کننده را تشکیل می دهند، سپس لازم است که چندجمله ای هدف (همان $t(s)$) و مقدار شیفت (α) به صورت رمزشده، برای تایید کننده ارسال شود. در مرحله ی اثبات ، اثبات کننده در فضای همریختی ، اثبات می کنید که $p(x)$ و $h(x)$ را در نقطه ی مشخص شده در مرحله ی راه اندازی دارد. در مرحله ی تایید نیز چک می شود که $p(s) = h(s)t(s)$ و اثبات به پایان می رسد. چند نکته ی مهم درباره ی اثبات بالا لازم به ذکر است:

1. مقدار δ باعث می شود که دانش اثبات کننده به صورت دانش-صفر اثبات شود چون این عدد تصادفی است و توسط اثبات کننده تولید می شود.
2. تمامی عملیات های zk-SNARK در فضای همریختی و گاها دوخطی (جاهایی که نیاز به ضرب در فضای همریختی داریم) انجام می شود.
3. مرحله ی راه اندازی اولیه توسط یک شخص ثالث مورد اعتماد انجام می شود که معرف شرایط CRS[13] می باشد.

حال به مسئله ی اصلی zk-SNARK می پردازیم، آیا می توان نتیجه ی یک محاسبه را اثبات کرد؟ هدف نهایی zk-SNARK این است که نتیجه ی یک قطعه کد یا برنامه یا یک سری دستور العمل[14] اثبات شود.

Algorithm 1 Operation depends on an input

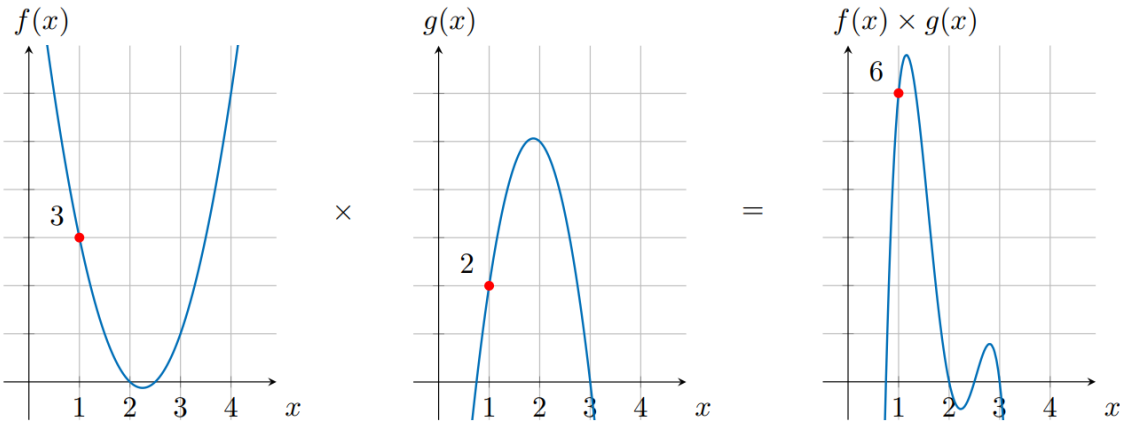
```
function calc(w, a, b)
    if w then
        return $a \times b$
    else
        return $a + b$
    end if
end function
```

شکل 2نمونه ای از یک برنامه که خروجی آن باید توسط zk-SNARK اثبات شود

---

[13] Common Reference String

[14] instruction



در اینجا با استفاده از مفهوم QAP محاسبات خود را تبدیل به یک مدار می کنیم و در نهایت اثبات کارکرد درست این مدار را با چندجمله ای ها اثبات می کنیم.

شکل3 : نحوه عملکرد zk-SNARK

تبدیل یک برنامه به مدار محاسباتی کاری تقریبا آسان به شما می آید و در زمان چندجمله ای نسبت به طول برنامه قابل انجام است. حال سئوال مهم این است که یک مدار محاسباتی را چگونه باید به چند جمله ای تبدیل کرد. با یک مثال مفهوم را توضیح می دهیم و سپس پروتکل نهایی zk-SNARK را می نویسیم.

می دانیم عملگرهای باینری به صورت زیر نوشته می شوند:

$$\text{left operand} \quad \textbf{operator} \quad \text{right operand} \quad = \quad \text{output}$$

که همگی عملوند ها و خروجی عدد هستند. حال می توان گفت که اگر به جای عملوند ها، چند جمله ی هایی را قرار دهیم که در نقاط مخصوصی دقیقا برابر با همین عملوند ها باشند، محاسبه ی ما با این اعداد، مساوی است با محاسبه روی چندجمله ای در نقاط به خصوصی که حاصل چند جمله ای ها با همین اعداد برابر است. به عنوان مثال محاسبه ی زیر را در نظر بگیرید:

$$2 \times 3 = 6$$

حال چند جمله های زیر را در نظر بگیرید:

مشاهده می کنیم که عملوند ها و خروجی در نقطه ی 1 دقیقا برابر با محاسبات ما می باشند، منتها تمام محاسبات در فضای چندجمله ای ها انجام شده است. به عبارت دیگر می توانیم بنویسیم:

$$l(x)\ \textbf{operator}\ r(x) = o(x)$$

حال اگر تشکیل دهیم $p(x) = l(x)\ \boldsymbol{OPERATOR}\ r(x) - o(x)$، اثبات اینکه محاسبه انجام شده است، تبدیل به مسئله ی ریشه داشتن $p(x)$ در نقطه ی a می شود. بدین شکل توانستیم یک محاسبه را به مسئله ی چندجمله ای ها تبدیل کنیم. حال به سراغ پروتکل اصلی می رویم:

مرحله ی راه اندازی

- Setup
  - select a generator g and a cryptographic pairing e
  - for a function $f(u) = y$ with n total variables of which m are input/output variables, convert into the polynomial form[27] $\left(\{l_i(x), r_i(x), o_i(x)\}_{i\in\{0,\ldots,n\}}, t(x)\right)$ of degree d (equal to the number of operations) and size $n+1$
  - sample random $s, \rho_l, \rho_r, \alpha_l, \alpha_r, \alpha_o, \beta, \gamma$
  - set $\rho_o = \rho_l \cdot \rho_r$ and the operand generators $g_l = g^{\rho_l}, g_r = g^{\rho_r}, g_o = g^{\rho_o}$
  - set the proving key:

$$\Big(\quad \left\{g^{s^k}\right\}_{k\in[d]}, \left\{g_l^{l_i(s)}, g_r^{r_i(s)}, g_o^{o_i(s)}\right\}_{i\in\{0,\ldots,n\}},$$
$$\left\{g_l^{\alpha_l l_i(s)}, g_r^{\alpha_r r_i(s)}, g_o^{\alpha_o o_i(s)}, g_l^{\beta l_i(s)} g_r^{\beta r_i(s)} g_o^{\beta o_i(s)}\right\}_{i\in\{m+1,\ldots,n\}},$$
$$g_l^{t(s)}, g_r^{t(s)}, g_o^{t(s)}, g_l^{\alpha_l t(s)}, g_r^{\alpha_r t(s)}, g_o^{\alpha_o t(s)}, g_l^{\beta t(s)}, g_r^{\beta t(s)}, g_o^{\beta t(s)} \quad \Big)$$

  - set the verification key:

$$\left(g^1, g_o^{t(s)}, \left\{g_l^{l_i(s)}, g_r^{r_i(s)}, g_o^{o_i(s)}\right\}_{i\in\{0,\ldots,m\}}, g^{\alpha_l}, g^{\alpha_r}, g^{\alpha_o}, g^{\gamma}, g^{\beta\gamma}\right)$$

مرحله ی اثبات:

- Proving
  - for the input u, execute the computation of $f(u)$ obtaining values $\{v_i\}_{i\in\{m+1,\dots,n\}}$ for all the itermediary variables
  - assign all values to the unencrypted variable polynomials $L(x) = l_0(x) + \sum_{i=1}^{n} v_i \cdot l_i(x)$ and similarly $R(x), O(x)$
  - sample random δ_l, δ_r and δ_o
  - find $h(x) = \frac{L(x)R(x) - O(x)}{t(x)} + \delta_r L(x) + \delta_l R(x) + \delta_l \delta_r t(x) - \delta_o$
  - assign the prover's variable values to the encrypted variable polynomials and apply *zero-knowledge* δ-shift $g_l^{L_p(s)} = \left(g_l^{t(s)}\right)^{\delta_l} \cdot \prod_{i=m+1}^{n} \left(g_l^{l_i(s)}\right)^{v_i}$ and similarly $g_r^{R_p(s)}$, $g_o^{O_p(s)}$
  - assign its α-shifted pairs $g_l^{L'_p(s)} = \left(g_l^{\alpha_l t(s)}\right)^{\delta_l} \cdot \prod_{i=m+1}^{n} \left(g_l^{\alpha_l l_i(s)}\right)^{v_i}$ and similarly $g_r^{R'_p(s)}$, $g_o^{O'_p(s)}$
  - assign the variable values consistency polynomials
    $g^{Z(s)} = \left(g_l^{\beta t(s)}\right)^{\delta_l} \left(g_r^{\beta t(s)}\right)^{\delta_r} \left(g_o^{\beta t(s)}\right)^{\delta_o} \cdot \prod_{i=m+1}^{n} \left(g_l^{\beta l_i(s)} g_r^{\beta r_i(s)} g_o^{\beta o_i(s)}\right)^{v_i}$
  - compute the proof $\left(g_l^{L_p(s)}, g_r^{R_p(s)}, g_o^{O_p(s)}, g^{h(s)}, g_l^{L'_p(s)}, g_r^{R'_p(s)}, g_o^{O'_p(s)}, g^{Z(s)}\right)$

مرحله ی تایید

- Verification
  - parse a provided proof as $\left(g_l^{L_p}, g_r^{R_p}, g_o^{O_p}, g^h, g_l^{L'_p}, g_r^{R'_p}, g_o^{O'_p}, g^Z\right)$
  - assign input/output values to verifier's encrypted polynomials and add to 1:
    $g_l^{L_v(s)} = g_l^{l_0(s)} \cdot \prod_{i=1}^{m} \left(g_l^{l_i(s)}\right)^{v_i}$ and similarly for $g_r^{R_v(s)}$ and $g_o^{O_v(s)}$
  - variable polynomials restriction check :
    $e\left(g_l^{L_p}, g^{\alpha_l}\right) = e\left(g_l^{L'_p}, g\right)$ and similarly for $g_r^{R_p}$ and $g_o^{O_p}$
  - variable values consistency check:
    $e\left(g_l^{L_p} g_r^{R_p} g_o^{O_p}, g^{\beta\gamma}\right) = e\left(g^Z, g^\gamma\right)$
  - valid operations check:
    $e\left(g_l^{L_p} g_l^{L_v(s)}, g_r^{R_p} g_r^{R_v(s)}\right) = e\left(g_o^{t(s)}, g^h\right) \cdot e\left(g_o^{O_p} g_o^{O_v(s)}, g\right)$

در این پروتکل دقیق مانند پروتکل قبلی، دانش داشتن یک چندجمله ای اثبات می شود منتها این چند جمله ای در نقاط به خصوصی مساوی است با محاسبات انجام شده.

مراجع

- [C71] Cook, Stephen A. "The complexity of theorem-proving procedures." *Proceedings of the third annual ACM symposium on Theory of computing*. 1971.
- [L73] Levin, Leonid Anatolevich. "Universal sequential search problems." *Problemy peredachi informatsii* 9.3 (1973): 115-116.
- [GMR85] S Goldwasser, S Micali, and C Rackoff. 1985. The knowledge complexity of interactive proof-systems. In Proceedings of the seventeenth annual ACM symposium on Theory of computing (STOC '85). Association for Computing Machinery, New York, NY, USA, 291–304.
- [GMA91] Oded Goldreich, Silvio Micali, and Avi Wigderson. 1991. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM 38, 3 (July 1991), 690–728.
- Nitulescu, Anca. A tale of SNARKs: quantum resilience, knowledge extractability and data privacy. Diss. Paris Sciences et Lettres, 2019
- Parno, Bryan, et al. "Pinocchio: Nearly practical verifiable computation." 2013 IEEE Symposium on Security and Privacy. IEEE, 2013.
- Nitulescu, Anca. "zk-SNARKs: A Gentle Introduction."
- Arora, Sanjeev, and Boaz Barak. Computational complexity: a modern approach. Cambridge University Press, 2009.
- Sipser, Michael. "Introduction to the Theory of Computation." ACM Sigact News 27.1 (1996): 27-29.