بسم الله الرحمن الرحیم

ارائه‌دهنده: درةالسادات دستغیب ۹۳۲۰۱۸۳۵

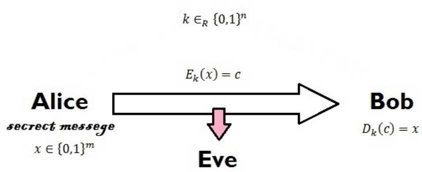
رمزنگاری

مقدمه:

طول عمر رمزنگاری بسیار زیاد است و ایجاد آن به سال‌ها قبل از اختراع کامپیوتر یا نظریه کامپیوتر و محاسبه برمی‌گردد. بهتر است بگوییم از زمانی که انسان نوشتن را ابداع نمود برای تبادل اطلاعات سرّی و محرمانه‌ی خویش، رمزنگاری را به کار گرفت. در این نوشتار قصد بر این داریم تا کمی پیرامون رمزنگاری و پیچیدگی محاسبات صحبت کنیم. در ابتدا تشریح خواهیم کرد که یک طرح رمزنگاری چیست و کمی به سیر تاریخی رمزها می‌پردازیم. بعد از آن بیان خواهیم کرد که یک طرح رمزنگاری با امنیت کامل چه ویژگی دارد و سپس یک طرح رمزگذاری با امنیت کامل را معرفی خواهیم نمود. در ادامه دو نمونه از مهم‌ترین مفهوم‌های مورد استفاده در رمزگذاری، یعنی توابع یک‌طرف و مولدهای شبه‌تصادفی را بیان می‌کنیم و چند قضیه پیرامون آن را ذکر می‌نمائیم. با توضیح مفهوم اثبات ناتروای دانش و ذکر یک مثال این نوشته را خاتمه می‌دهیم.

طرح رمزنگاری چیست؟

ساده‌ترین شکل ارتباط محرمانه‌ای که در رمزنگاری وجود دارد به این شکل است که فردی می‌خواهد پیامی را برای فرد دیگر بفرستند به طوری که دیگران از آن مطلع نشوند. وی برای این منظور پیام خودش را با استفاده از یک کلید که از قبل با نفر دیگر بر سر آنکه چه باشد، به توافق رسیده‌اند، به صورت رمز درآورده و متن رمز را برای آن‌سوی ارتباط ارسال می‌کند. فرد گیرنده، با استفاده از همان کلید، متن رمزشده‌ی دریافتی را رمزگشایی کرده و از محتوای پیام مطلع می‌گردد. در شکل زیر Alice می‌خواهد پیام x را که از طول m است، برای Bob بفرستد.



وی متن x را با استفاده از الگوریتم رمزگذاری E و کلید k را به رمز درآورده و برای Bob ارسال می‌کند. Bob با کلید k و الگوریتم رمزگشایی D ، رمز دریافتی را رمزگشایی کرده و از x مطلع می‌گردد. *یک طرح رمزگذاری (E, D) زوج مرتبی از الگوریتم رمزگذاری E و الگوریتم رمزگشایی D است، به طوریکه داشته باشیم:* $D_k(E_k(x)) = x$

رمزنگاری در طول تاریخ:

یکی از ساده‌ترین نوع رمزنگاری‌هایی که در گذشته مورد استفاده قرار می‌گرفت، رمزنگاری جانشینی[1] بود. در کلید این نوع رمزنگاری به جای هر حرف الفبا یک نماد دیگر به کار برده می‌شد و متن رمزشده چیزی نبود جز همان متن اصلی که با نمادهای آمده در کلید جانشین گشته بود. هرکسی که از کلید نمادها مطلع بود می‌توانست

[1] Substitution Cipher

از متن رمز مطلع باشد. در دوره‌ی سزار برای رفع مشکل نگهداری کلید از رمزگذاری جایگشتی[۲] استفاده می‌شد. کلید این نوع رمزگذاری، همان حروف الفبا بود که توسط یک عدد همگی به جلو هل[۳] داده شده بود. برای آگاه گشتن از متن اصلی، کافی بود تا گیرنده پیام از عدد مربوط به هل داده شدن حروف الفبا مطلع باشد تا یک کلید جایگذاری برای خودش ترتیب داده و رمز را بگشاید. واضح است که این نوع رمزگذاری حالت خاصی از رمزگذاری جانشینی است.

چنین طرح‌های رمزگذاری با وجود فضای حالت بزرگی که برای کلید آن موجود است، با استفاده از بررسی تناوب تکرار حروف یک زبان به سادگی شکسته می‌شود. یعنی یک متخاصم[۴]، نیازی به هیچ چیز اضافه‌ای جز همان متن رمز شده ندارد تا بتواند از متن اصلی سر در بیاورد. به چنین حمله‌هایی، حمله فقط متن رمز[۵] گفته می‌شود و این از بدترین نوع حمله‌هایی است که می‌تواند به یک طرح رمزگذاری شود.

نوع رمزگذاری دیگری که بعد از این مدل رمزگذاری‌هایی به کار برده می‌شد، رمزگذاری Vigener است، که در اوایل قرن ۱۹ کسی به نام Vigener آن را معرفی نمود. ایده این طرح به این صورت بود که یک کلمه را به عنوان کلید در نظر می‌گرفتند؛ سپس کلید را پشت سر هم زیر متن اصلی می‌نوشتند. در نهایت باقیمانده‌ی به پیمانه‌ی ۲۶ جمع شماره‌های حروف الفبای زبان را محاسبه می‌کردند و حروفی که معادل عددی آن‌ها بود، همان متن رمز بود.

```
  COMPUTERSC I ENCE
  CA KECAKECAKECAK
 ------------------ mod 26
  F P YUXUPwVDTJQDP
```

با وجودی که چنین طرح رمزگذاری مشکل رمزگذاری‌های قبلی را ندارد، برای مثال حرف p هم از جمع e و k و هم از جمع o و a حاصل گشته است، اما باز با حمله‌ی فقط رمز می‌توان چنین طرح رمزگذاری را مورد حمله قرار داد.

در جنگ جهانی دوم، از ماشین‌های چرخان[۶] برای رمزنگاری استفاده می‌شد. ایده این بود که کلید ماشین در هر باری که نویسنده متن رمز می‌خواهد حرفی را تایپ کند، توسط چرخ‌دنده‌های ماشین تغییر کند.متفقین با کمک جاسوسان خود در آلمان و بهره بردن از ریاضی‌دانان به نامی مانند آلن تورینگ توانستند، سر از کار ماشین‌های چرخانی که توسط آلمان‌ها استفاده می‌شد، در بیاورند و رمزهای آنان را بشکنند.

«نبوغ بشر نمی‌تواند رمزی بسازد که نبوغ بشر نتواند آن را بشکند.» این جمله‌ایست که E.A.Poe در سال ۱۸۴۱ آن را بیان کرد. بعد از ماجرای جنگ جهانی و اختراع کامپیوتر، دولت‌ها به فکر این افتادند که شیوه‌ای از رمزنگاری را به کار بگیرند که از امنیت کامل برخوردارباشد. برای این منظور دستور دادند تا رمزنگاری به صورت آکادمیک و علمی مورد بررسی واقع شود و یک استاندارد برای آن تولید گردد. در اواسط دهه‌ی ۱۹۷۰ بود که رمزنگاری مدرن به دنیا آمد. اینجا نقطه‌ای بود که برای یک طرح رمزنگاری، مخفی ماندن الگوریتم رمزگذاری و رمزگشایی یک مزیت محسوب نمی‌شد. الگوریتم‌های مختلف در جوامع علمی مطرح شده و شیوه‌های گوناگون حمله به آن‌ها مورد تحلیل و بررسی قرار می‌گرفت. پیچیدگی محاسباتی مورد توجه دانشمندان قرار گرفت و در رمزنگاری مدرن به کار گرفته شد. همان طور که بیان شد سختی رمزنگاری مدرن بر پایه مخفی نگه داشتن طرح رمزنگاری نیست، بلکه سختی از نظر محاسبه و پیچیدگی محاسبه‌ی طرح‌های رمزگذاری آشکاری مثل RSA است که امنیت آن‌ها را تضمین می‌کند.

امنیت کامل[۷]

اولین کسی که به صورت رسمی و علمی یک تعریف برای امنیت طرح رمزگذاری مطرح کرد شانون[۸] در

[۲]Caesar Cipher
[۳]Shift
[۴]Adversary
[۵]Cipher Only Attack
[۶]Rotor Machines
[۷]Perfect Secrecy
[۸]Shanon

سال ۱۹۴۹ بود. وی طرحی را از امنیت کامل برخوردار دانست که هیچ گونه اطلاعاتی را در مورد متن اصلی پیام لو ندهد.

تعریف امنیت کامل: *فرض کنید (E, D) یک طرح رمزگذاری برای پیام‌هایی به طول m با کلیدهایی به طول n باشد. گوییم (E, D) دارای امنیت کامل است، اگر برای هر زوج پیام $x, x' \in \{۰, ۱\}^m$ توزیع $E_{U_n}(x)$ و $E_{U_n}(x')$ یکسان باشد.*

طبق این تعریف اگر یک متن رمز به ما بدهند، در صورتی که نتوانیم که بین دو پیام x و x' تمایز قائل شده و بگوییم که متن رمزشده احتمالا مربوط به کدامیک بود، طرح رمزگذاری امنیت دارد. به عبارت دیگر احتمال اینکه متن رمزشده‌ی c را با اجرای الگوریتم E روی کلید k و پیام x حاصل شود، برابر باشد با احتمال اینکه c متن رمز حاصل از اجرای الگوریتم E روی کلید k و پیام x' باشد.

$$Pr[E_k(x) = c] = Pr[E_k(x') = c] \quad \forall x, x' \in \{۰, ۱\}^m$$

طرح رمزگذاری یک بار مصرف (!)[۹]

یکی از طرح‌های رمزگذاری که در تعریف امنیت کامل صدق می‌کند، طرح رمزگذاری یک بار مصرف است. ایده‌ی این طرح به این صورت است که یک کلید به اندازه‌ی طول پیام تصادفی انتخاب کرده و سپس محتوای کلید و پیام را باهم بیت به بیت XOR می‌کنیم.

چون کلید این طرح به صورت تصادفی در فضای یکنواخت انتخاب شده است. متن رمز حاصل یک عدد تصادفی یکنواخت خواهد بود و در تعریف امنیت کامل صدق می‌کند. از آنجایی که استفاده‌ی یک کلید برای دو پیام اطلاعاتی غیربدهی را در اختیار متخاصم قرار می‌دهد، از هر کلید دقیقا باید یک بار استفاده شود.

متاسفانه طرح رمزگذاری یک بار مصرف ایجاب می‌کند تا ما حتما کلیدهایی با طول هم‌اندازه‌ی پیام داشته باشیم و برای دنیال وسیع و پر اطلاعات امروزه این اصلا به صرفه نیست. اولا اینکه باید به ازای هر پیام طولانی یک کلید طولانی تولید کنیم. ثانیا باید چنین کلیدهایی را از دسترس متخاصمین حفظ کنیم! آیا به اندازه‌ی مناسبی کوچک بودن کلید در طرح‌های رمزگذاری مشکلی ایجاد می‌کند؟ لم زیر بیان می‌کند که تا زمانی که $P \neq NP$ پاسخ منفی است.

لم: *فرض کنید $P = NP$ و یک طرح رمزگذاری (E, D) محاسبه‌پذیر با زمان اجرای چندجمله‌ای باشد. همچنین فرض کنید که طول کلید کوچک‌تر از طول متن پیام باشد. آنگاه یک الگوریتم $\mathscr{A}$ وجود دارد که به ازای هر ورودی به طول m یک زوج پیام $x_۰, x_۱ \in \{۰, ۱\}^m$ هست که*

$$Pr[\mathscr{A}(E_k(x_b))] \geq \frac{۳}{۴}, \quad b \in_R \{۰, ۱\}, k \in_R \{۰, ۱\}^n$$

در واقع لم فوق می‌گوید اگر $P = NP$ آنگاه می‌توانیم یک الگوریتم داشته باشیم که دو پیام x و x' وجود دارند که اگر رمزشده‌ی آن‌ها را به این الگوریتم بدهیم، می‌توانیم اطلاعات نابدیهی در مورد پیام‌ها به دست آوریم و این امنیت کامل را نقض می‌کند.

در ادامه تابعی را معرفی می‌کنیم که در قضایای رمزنگاری بسیاری مورد، از جمله برخی که بیان خواهیم کرد، استفاده قرار می‌گیرد و آن هم یک تابع ناچیز است!

تعریف تابع ناچیز[۱۰]: تابع $\varepsilon : \mathbb{N} \to [۰, ۱]$ یک تابع ناچیز گفته می‌شود اگر $\varepsilon(n) = n^{-\omega(۱)}$ یعنی برای هر عدد ثابت c و داشتن n به اندازه‌ی کافی بزرگ داریم $\varepsilon(n) < n^{-c}$

تعریف فوق می‌گوید که ε از هر تابع چندجمله‌ای که در نظر گرفته شود، کوچک‌تر است. به عبارتی با بزرگ شدن ورودی، مقدار تابع‌های ناچیز به سرعت به صفر میل می‌کند. پیشامدهایی که احتمال ناچیزی داشته باشند در کاربردهای عملی و تئوری نادیده گرفته می‌شوند.

یکی از حدس‌های مهم اهل فن در پیچیدگی محاسباتی وجود توابع یک‌طرفه است. چنین توابعی تاثیر مهم و به‌سزایی در اثبات امنیت طرح‌های رمزنگاری دارد.

[۹] One Time Pad
[۱۰] Negligible Function

تعریف تابع یک طرفه [۱۱] : تابع محاسبه‌پذیر $f:\{0,1\}^* \to \{0,1\}^*$ که زمان اجرای چندجمله‌ای دارد را یک‌طرفه می‌گوییم، اگر برای هر الگوریتم احتمالاتی با زمان اجرای چندجمله‌ای مانند $\mathscr{A}$ یک تابع ناچیز ε وجود داشته باشد که

$$\forall n \in \mathbb{N} \quad Pr[\mathscr{A}(y) = x \text{ s.t } f(x) = y] < \varepsilon(n), \quad x \in_R \{0,1\}^n, f(x) = y$$

طبق این تعریف یک تابع را یک طرفه گوییم اگر مقدار خود تابع را بتوانیم در زمان خوبی، مثلا چندجمله‌ای، محاسبه کنیم ولی محاسبه معکوس تابع بسیار دشوار است. در واقع هر الگوریتم احتمالاتی که تلاش کند معکوس تابع را به دست بیاورد، شکست بخورد و به عبارت دیگر احتمال موفق شدنش بسیار ناچیز باشد.

می‌توان نشان داد که اگر تابع یک‌طرفه وجود داشته باشد، آنگاه $P \neq NP$! وجود نامزدهای خوبی برای تابع‌های یک‌طرفه، دلیل این است که اهل فن در پیچیدگی محاسبه گمان ببرند که تابع‌های یک طرفه وجود داشته باشند. سه مورد از نامزدهای تابع یک‌طرفه را ذکر می‌کنیم:

- تابع ضرب! بیش‌تر دانشمندان باور دارند که تابعی که یک وروی n بیتی شامل دو عدد $\frac{n}{2}$ بیتی به عنوان ورودی بگیرد و ضرب این دو عدد را به خروجی بدهد یک تابع یک‌طرفه هست. معکوس کردن این تابع به عنوان مسئله‌ی تجزیه‌ی عوامل صحیح[۱۲] شناخته شده است. البته که پیدا کردن عوامل صحیح عدد مانند N می‌تواند با حداکثر $\sqrt{N}$ تقسیم، صورت بپذیرد؛ اما چون ما عدد N را در این مسئله با lgN بیت نمایش می‌دهیم، این الگوریتم به یک الگوریتم با زمان اجرای نمائی تبدیل می‌شود.

- تابع RSA در این تابع e و N ثابت در نظر گرفته شده و خروجی تابع، برای x ای که به عنوان ورودی به آن داده می‌شود، برابر است با $RSA_{N,e}(x) = x^e(modN)$

- تابع Rabin این تابع مشابه RSA است با این تفاوت که در میدان QR_N عمل می‌کند و برابر است با $x \in QR_N, \quad Rabin(x) = x^2(modN)$

یکی از کاربردهای توابع یک‌طرفه در طرح‌های رمزنگاری است کلیدهایی کوتاه‌تر از طول پیام دارند.
قضیه رمزگذاری از یک تابع یک‌طرفه [۱۳]: *فرض کنید که تابع یک‌طرفه وجود داشته باشد، آنگاه به ازای هر $c \in \mathbb{N}$ یک طرح رمزگذاری از نظر محاسباتی امن مثل (E, D) وجود دارد که برای پیام‌های به طول n^c از کلیدهایی به طول n استفاده می‌کند.*

در قضیه فوق عبارت از نظر محاسباتی امن به کار برده شد. چیزی را از نظر محاسباتی امن گوییم که متخاصم نتواند هیچ بیت تصادفی انتخاب شده‌ای را با احتمال غیر ناچیز بیش از $\frac{1}{2}$ حدس بزند. به عبارت دیگر طرح رمزگذاری (E, D) را از نظر محاسباتی امن گوییم اگر برای هر الگوریتم احتمالاتی چندجمله‌ای مثل $\mathscr{A}$ یک تابع ناچیز مثل ε وجود داشته باشد که

$$Pr[\mathscr{A}(E_k(x)) = (i,b) \text{ s.t. } x_i = b] \leq \frac{1}{2} + \varepsilon(n), \quad k \in_R \{0,1\}^n, x \in_R \{0,1\}^m$$

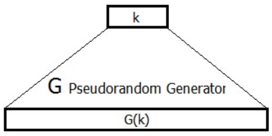
ایده‌ی اثبات قضیه‌ای که در فوق بیان شد، از این پیروی می‌کند که ما یک کلید تصادفی کوچک به طول n برداریم و آن‌را به طریقی تا طول پیام بزرگ‌نمائی کنیم طوری که همچنان تصادفی به نظر برسد! مولدهای شبه تصادفی این امر را میسر می‌کنند.
تعریف مولد شبه تصادفی امن [۱۴]: فرض کنید که یک تابع محاسبه‌پذیر $G : \{0,1\}^* \to \{0,1\}^*$ با زمان اجرای چندجمله‌ای داشته باشیم و نیز فرض کنید که $l : \mathbb{N} \to \mathbb{N}$ هم یک تابع محاسبه‌پذیر با زمان اجرای چندجمله‌ای باشد، به طوریکه داشته باشیم $l(n) > n$. آنگاه Gرا یک مولد شبه‌تصادفی امن با امتداد $l(n)$

[۱۱] One Way Function
[۱۲] integer factorization
[۱۳] Encryption from One Way Function
[۱۴] Secure Pseudorandom Generator

گوییم اگر $|G(x)| = l(|x|) \quad \forall x \in \{۰,۱\}^*$ و برای هر الگوریتم احتمالاتی با زمان اجرای چندجمله‌ای مثل $\mathscr{A}$ یک تابع ناچیز ε وجود داشته باشد به طوریکه

$$\forall n \in \mathbb{N} \quad |Pr[\mathscr{A}(U_n)) = ۱] - Pr[\mathscr{A}(U_n) = ۱]| < \varepsilon(n)$$

این تعریف بیان می‌کند که G یک مولد شبه تصادفی است اگر کلید تصادفی x را بگیرد و کلید شبه‌تصادفی $G(x)$ را به ما تحویل بدهد(شکل زیر).



یک مولد شبه‌تصادفی در صورتی امن محسوب می‌شود که اگر متخاصمی یک نمونه از آنچه که مولد تولید کرده و یک نمونه تصادفی از توزیع یکنواخت را ببیند، نتواند بین آن‌ها تمایزی قائل شود و تشخیص دهد که کدامیک واقعا تصادفی است!

قضیه مولدهای شبه‌تصادفی با استفاده از توابع یک‌طرفه [۱۵]*: اگر تابع یک‌طرفه موجود باشد، آنگاه برای هر $c \in \mathbb{N}$ یک مولد شبه‌تصادفی امن با امتداد $l(n) = n^c$ وجود دارد.*

طبق این قضیه، با شرط وجود توابع یک‌طرفه، می‌توانیم با در دست داشتن یک کلید کوچک تصادفی، یک کلید بزرگ که تصادفی به نظر می‌رسد را تولید کنیم. با اجرای الگوریتم طرح رمزگذاری یک‌بار مصرف، یک طرح رمزگذاری امن خواهیم داشت.

اثبات‌های ناتراوای دانش [۱۶]:

واضح است که وقتی اثبات یک مطلب را می‌خوانیم، از طرح، ایده و راه حل اثبات آگاه می‌شویم اما چه طور یک نفر می‌تواند مطلبی را برای فرد دیگر اثبات کند به طوری هیچ گونه اطلاعاتی درمورد ایده و راه‌حلش در اختیار او نگذارد؟! پروتکل اثبات نارتراوای دانش این امکان را برای یک اثبات‌کننده فراهم می‌کند تا بدون اینکه اطلاعتی از ایده و راه‌حل اثبات تراوش کند، مطلبش را به یک تصدیق‌گر اثبات نماید.

تعریف اثبات ناتراوای دانش: فرض کنید که $L \in NP$ و نیز فرض کنید که M یک ماشین تورینگ با زمان اجرای چندجمله‌ای باشد به طوریکه

$$x \in L \iff \exists u \in \{۰,۱\}^{p(|x|)} \text{ s.t. } M(x,u) = ۱$$

که در آن $p()$ یک تابع چندجمله‌ای است. به زوج الگوریتم احتمالاتی P و V که با هم در تعامل هستند، یک اثبات ناتراوای دانش برای L گوییم اگر سه شرط برقرار باشد:

- (تمامیت)

$$\forall x \in L, \forall u \text{ s.t. } M(x,u) = ۱ \quad Pr[out_v < P(x,u), V(x) >] \geq \frac{۲}{۳}$$

که $I =< P(x,u), V(x) >$ نمایانگر تعامل بین P و V است. ورودی P برابر x و u است و ورودی V برابر x است. $out_V I$ نشان‌دهنده‌ی خروجی Vدر انتهای تعامل I است.

- (سازگاری) اگر $x \in L$، آنگاه برای هر استراتژی P^* و ورودی u داریم:

$$Pr[out_V < P^*(x,u), V(x) >] \leq \frac{۱}{۳}$$

[۱۵] Pseudorandom Generators from One Way Functions
[۱۶] Zero Knowledge Proof

- (ناتراوایی دانش کامل)[۱۷] برای هر استراتژی تعاملی احتمالاتی مانند V^* که در زمان چندجمله‌ای اجرا می‌شود، یک الگوریتم S^* با زمان اجرای چندجمله‌ای وجود دارد به طوریکه

$$\forall x \in L, \forall u \quad out_{V^*} < P(x,u), V^*(x) > \equiv S^*(x)$$

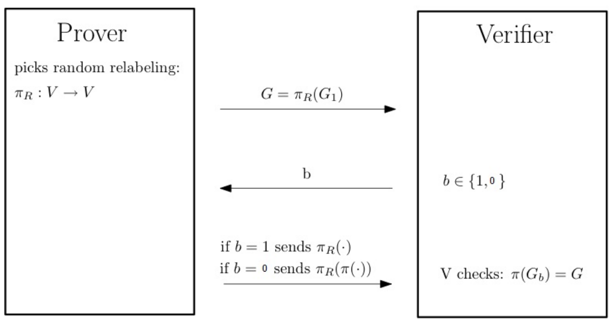
از آنجایی که الگوریتم S^* خروجی تعامل V^* را با اثبات‌کننده شبیه‌سازی می‌کند، الگوریتم شبیه‌ساز برای V^* نامیده می‌شود.

شرط ناتروایی دانش کامل تضمین می‌کند که هر اطلاعاتی که تصدیق‌گر از یک تعامل با اثبات‌کننده به‌دست می‌آورد را نیز خودش هم بتواند با استفاده از یک شبیه‌ساز به دست‌آورد. از آنجایی که برآورده ساختن این شرط بسیار سخت و حتی در مواردی ممکن است امکان‌پذیر نباشد، در پروتکل‌های اثبات ناتراوای دانش، به جای این شرط، شرط ناتروایی دانش آماری [۱۸] یا شرط ناتراوایی دانش محاسباتی [۱۹] به کار می‌رود. شرط ناتراوایی دانش آماری از توزیع‌هایی استفاده می‌کند که فاصله‌ی آماری کمی دارند و در شرط ناتراوایی دانش محاسباتی توزیع‌هایی که از نظر محاسبه‌پذیری تمایزناپذیرند استفاده می‌شود. رده‌ی مسائلی باتوجه به این شرط ناتراوایی دانش کامل، ناتروایی دانش آماری و ناتراوایی دانش محاسباتی ، یک اثبات ناتراوای دانش دارند به ترتیب PZK، SZK و CZK است و رابطه‌ی زیر برای‌شان برقرار است:

$$BPP \subseteq PZK \subseteq SZK \subseteq CZK \subseteq IP$$

اثبات ناتروای دانش برای گراف ایزومورفیسم:

در شکل زیر می‌توانید یک پروتکل اثبات ناتراوای دانش که برای اثبات ایزومورفیسم بودن دو گراف به کار گرفته می‌شود را ملاحظه نمائید. اثبات‌کننده و تصدیق‌گر هر دو از گراف‌های G_0 و G_1 مطلع هستند اما اثبات کننده از جایگشت رئوس π آگاه است که ثابت می‌کند دو گراف G_0 و G_1 با هم ایزومورفیسم هستند. وی بدون اینکه این جایشگت را بخواهد در اختیار تصدیق‌گر قرار دهد، این موضوع را به وی اثبات می‌کند.



وی ابتدا یک جایگشت تصادفی π_R را انتخاب کرده، آن را روی گراف G_1 اعمال نموده و گراف حاصل شده را برای تصدیق‌گر ارسال می‌کند. تصدیق‌گر بیت تصادفی $b \in_R \{0, 1\}$ را برمی‌گزیند و آن را برای اثبات کننده می‌فرستد. حال اثبات‌کننده وظیفه دارد تا جایگشتی را برای تصدیق‌گر بفرستد که گراف متناظر با بیت ارسالی او را به وی تحویل دهد. در واقع اگر $b = 1$ اثبات‌کننده جایگشت $\pi_R(.)$ و اگر $b = 0$ اثبات‌کننده جایگشت $\pi_R(\pi(.))$ را برای تصدیق‌گر ارسال می‌کند. تصدیق‌گر جایگشت دریافتی را روی گرافی که در تعامل اول با اثبات‌کننده به دست آورده بود اعمال می‌کند. اگر اثبات‌کننده واقعا اثباتی درست برای ایزومورفیسم بودن دو گراف در دست داشته باشد، گراف حاصل شده همانی است که مورد انتظار تصدیق‌گر است این تضمین می‌کند که شرط تمامیت ارضا شود. در غیر این صورت به احتمال $\frac{1}{2}$ اثبات کننده در تعامل سوم جایگشت ارسالی‌اش ناصحیح است و تصدیق‌گر نخواهد پذیرفت و این شرط سازگاری را ارضا می‌کند. به سادگی می‌توان

[۱۷] Perfect Zero Knowledge
[۱۸] Statistical Zero Knowledge
[۱۹] Computational Zero Knowledge

بررسی کرد که این پروتکل در شرط ناتراوایی دانش کامل صادق است.

آنچه که باقی ماند

دنیای رمزنگاری وسیع‌تر است از آنچه ما توصیف کردیم. مطالب بسیاری باقی ماند که تنها به اختصار به برخی از آن‌ها اشاره می‌کنیم.

- پیش‌بینی نشدن شبه‌تصادفی بودن را در بردارد

برای آن‌که مطمئن شویم یک مولد شبه‌تصادفی داریم، کافی است تا مطمئن شویم که بیت‌هایی که مولد تولید می‌کند پیش‌بینی نشدنی هستند. به عبارت دیگر اگر با در دست داشتن بیت‌های یک تا i یک رشته‌ی تولید شده توسط مولد نتوانیم بیت $i+۱$ ام آن را تشخیص دهیم، می‌گوییم مولد پیش‌بینی نشدنی است و می‌توان ثابت کرد که مولدی که پیش‌بینی نشدنی باشد، شبه‌تصادفی امن هم هست.

- تولید یک بیت شبه‌تصادفی اضافه

اگر یک تابع یک‌طرفه داشته باشیم، آنگاه می‌توانیم مولد شبه‌تصادفی داشته باشیم که یک بیت شبه‌تصادفی اضافه تولید می‌کند. با توسعه دادن این ایده، می‌توان نشان داد که می‌توان یک مولد شبه‌تصادفی به دست آورد که به اندازه‌ی امتداد چندجمله‌ای بیت اضافه تولید می‌کند.

- خانواده‌ی توابع شبه‌تصادفی

می‌توان بحث مولد شبه‌تصادفی را خانواده‌ی توابع شبه‌تصادفی تعمیم داد. در یک تابع شبه‌تصادفی جدول ارزش اندازه‌ای نمائی دارد و الگوریتمی که در زمان چندجمله‌ای قرار است تمیز دهد؛ فقط می‌تواند برای تعداد محدودی از ورودی‌ها از مقدار تابع مطلع شود. خانواده‌ی توابع شبه‌تصادفی در بسیاری از موارد از جمله آنچه در زیر آورده شده است، کاربرد دارد:

  - صحت پیام[۲۰]
  - احراز هویت[۲۱]
  - شیر و خط از پشت تلفن(!) و داشتن بیت تعهد[۲۲]
  - محاسبات امن چندنفره[۲۳]
  - کران پایین برای یادگیری ماشین[۲۴]
  - طرح رمزگذاری با کلید عمومی(!) [۲۵]

همان‌طور که در اوایل این نوشته نیز اشاره کردیم، طرح‌های رمزنگاری مدرن و پروتکل‌های امروزی وابسته به مخفی نگه‌داشتن الگوریتم‌های رمزگذاری و رمزگشایی نیست و چیزی که موجب می‌شود شکستن رمزهای مدرن سخت باشد، توان محدود محاسباتی دنیای امروز است. به همین خاطر است که ما به مفاهیم تصادفی در تعاریف مولدها، توابع یک‌طرفه و قضایای متفاوت رو می‌آوریم. تصادفی عمل کردن تنها چاره‌ی کنونی ممکن برای به دست آوردن اطلاعات غیربدیهی از متن رمز است و الگوریتم‌های رمزگذاری کنونی تضمین می‌کند که در زمان چندجمله‌ای که عمر ما می‌تواند کفایت کند تا به محاسبه بپردازیم، شکسته نشوند.

[۲۰] Authentication
[۲۱] Authorization
[۲۲] Tossing coin over phone and bit commitment
[۲۳] Secure MultiParty Computation
[۲۴] Lower bounds for Machine Learning
[۲۵] Public Key Encryption