



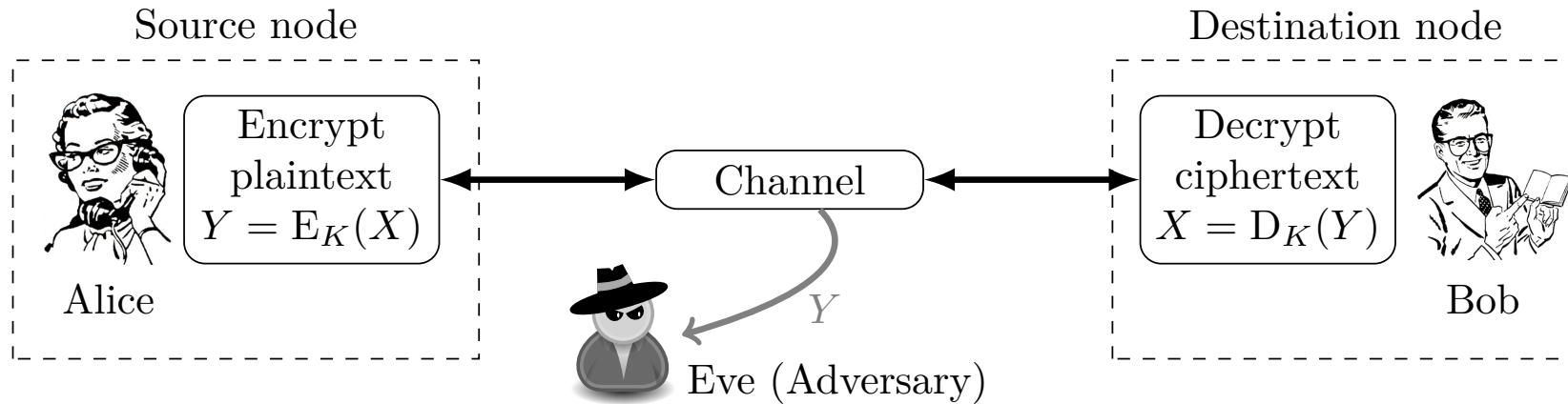
فصل سوم: گلپر عموهی

امنیت سیستم‌های کامپیوتری

ابوالفضل دیانت

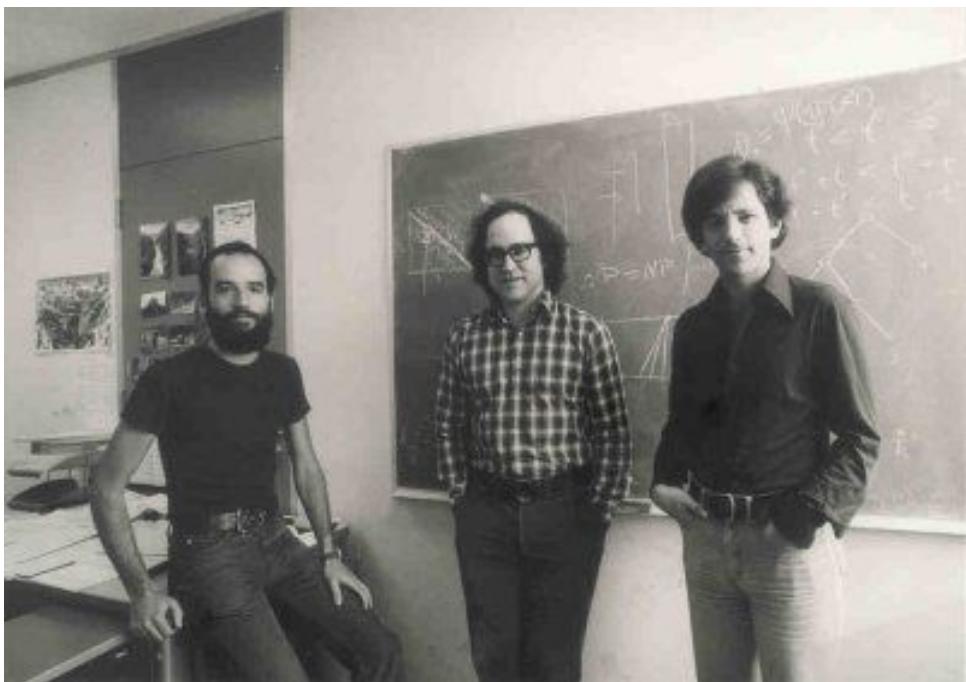
آخرین ویرایش: ۲۵ آذر ۱۴۰۱ در ساعت ۱۳ و ۵۳ دقیقه - نسخه ۳.۱.۳

چالش ما در الگوریتم‌های کلید متقارن ...



- برای محترمانه ماندن پیام می‌بایست از رمزگذاری (Encryption) استفاده کنیم، و برای آن نیاز به کلید داریم. ☞
- استفاده از سازوکارهای برقراری کلید (Key Establishment) [۱]، فصل 12: ☞
- تبادل کلید (Key Transport): یک سمت کلید را تولید کرده و در اختیار طرف مقابل نیز قرار می‌دهد.
- توافق کلید (Key Agreement): هر دو سمت، در فرایند تولید کلید مشارکت می‌کنند.

پیشگامان

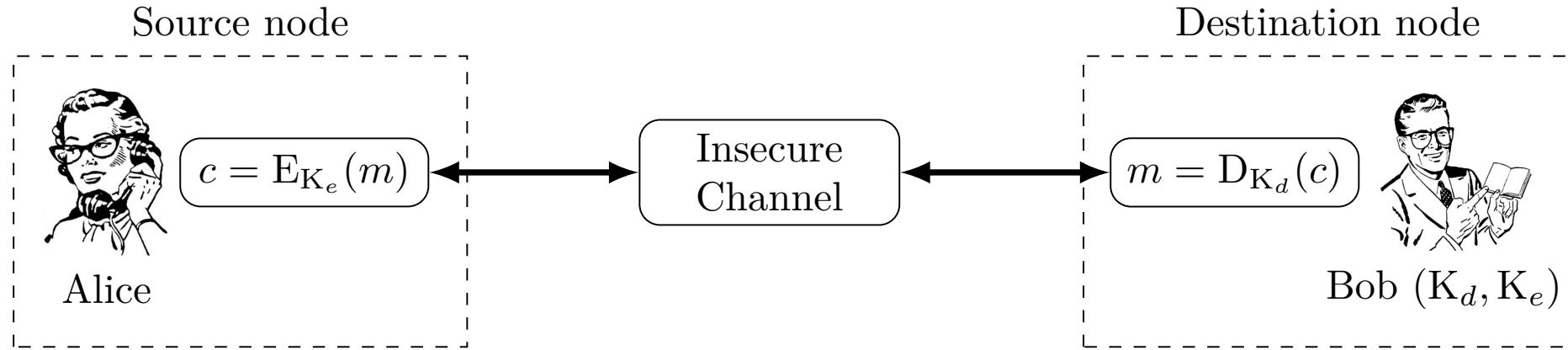


Ron Rivest, Adi Shamir, Leonard Adleman



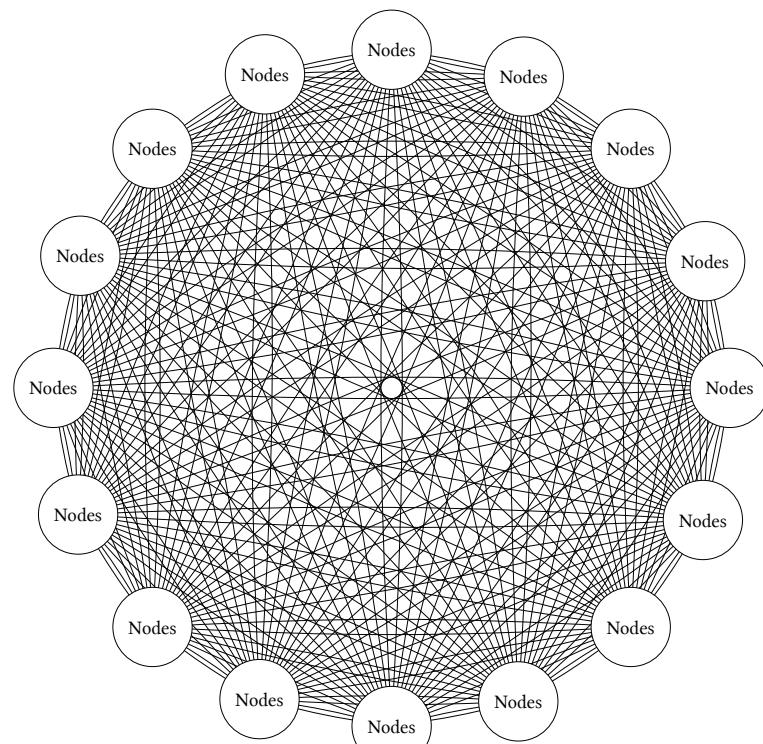
Ralph Merkle, Whitfield Diffie, Martin Hellman

کاربرد الگوریتم کلید نامتقارن - رمزنگاری



- ۱ یک جفت کلید یک کلید عمومی (Public Key) و کلید محرمانه (Private Key) برای حفظ محرمانگی، کلید عمومی را در اختیار همه قرار می‌دهیم (اعلان عمومی) و کلید خصوصی در اختیار فرد گیرنده. مثل Email.
- ۲ همگان می‌توانند رمزگذاری انجام دهند، اما فقط گیرنده می‌تواند رمزگشایی را انجام دهد.
- ۳ آغاز این دوره از سال ۱۹۷۷ و با الگوریتم RSA

الگوریتم کلید نامتقارن - مقایسه



✓ تعداد کلید کمتر برای برقراری ارتباط، $\binom{n}{2}$ در برابر ...

✓ نیاز به کانال امن نداریم، اما ...

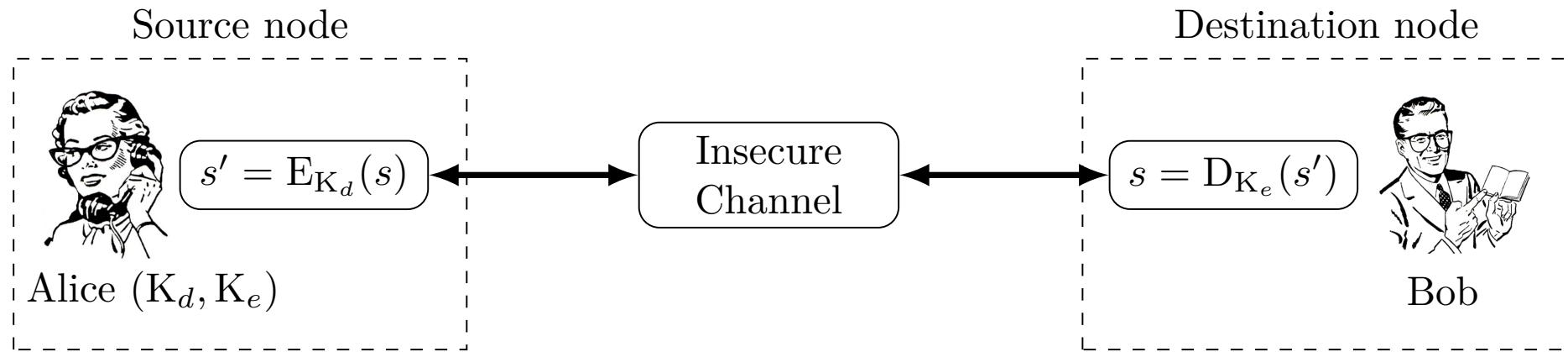
✗ سرعت عمل پایین‌تر نسبت به الگوریتم‌های کلید متقارن. نقش

مکمل برای الگوریتم‌های کلید متقارن در توزیع کلید را دارند!!!

✗ امنیت آن‌ها مبتنی بر مسایل نظریه اعداد است، با طول کلید

بزرگ‌تر (مثلاً یک RSA 3072 بیتی امنیتی معادل AES 128 بیتی دارد).

کاربرد الگوریتم کلید نامتقاضان - امضای دیجیتال (Digital Signature)

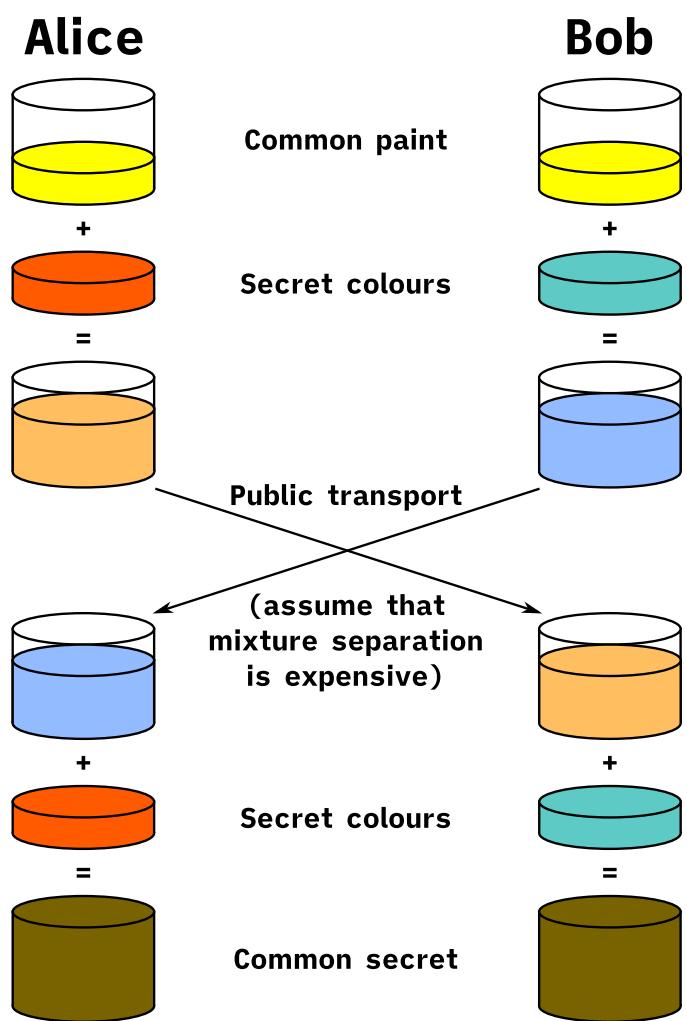


برای امضا، پیام s را با کلید خصوصی خودش، رمزگذاری و برای Bob ارسال می‌کند.

Bob نیز با کلید عمومی Alice می‌تواند امضا را تایید کند.

هر کس که کلید عمومی را دارد، می‌تواند امضا را تایید کند.

کاربرد الگوریتم کلید نامتقارن - توافق کلید



در سال ۱۹۷۶ با الگوریتم تبادل کلید Diffie–Hellman پا به عرصه

وجود گذاشت، و بعدها با کارهای Ralph Merkle تکمیل شد.

فرضیات و روش کار:

- یک رنگ عمومی

و Box و Alice •

انتخاب می‌کنند.

• ترکیب رنگ عمومی با رنگ‌های محرمانه و ارسال به طرف مقابل.

• فرض می‌شود که ترکیب دو رنگ کار ساده‌ای است ولی تجزیه آن کار

بسیار دشواری است.

Whitfield Diffie یکی از مشتاق‌ترین رمزنگاران عصر خویش بود. صرف دیدن او باعث ایجاد یک تصویر چشمگیر و تا حدودی متناقض از او می‌شود. کت و شلوار بی عیب و نقش، منعکس کننده این واقعیت است که در بیشتر دهه ۱۹۹۰، به عنوان مهندسی ممتاز در استخدام یکی از شرکت‌های غول پیکر کامپیوتری آمریکا یعنی Sun Microsystems، بوده. با این حال موها و ریش سفید بلندش، گواهی می‌دهد که قلب او هنوز در دهه ۱۹۶۰ گیر کرده است. Diffie در سال ۱۹۴۴ به دنیا آمد و بیشتر سال‌های اولیه زندگی خود را در نیویورک گذراند. در دوران نوجوانی شیفتۀ ریاضیات شد. در سال ۱۹۶۵ در ماساچوست، به تحصیل ریاضیات ادامه داد، و در دهه ۱۹۷۰ به یکی از معدود متخصصین حوزه امنیت در جهان و رمزنگاران آزاد غیردولتی مبدل شد. در همان زمان جهان در حال آماده شدن برای یک تحول بزرگ بود. ARPANET به عنوان پیشگامی برای شبکه جهانی اینترنت، در حال گسترش بود. گرچه ARPANET هنوز در مراحل اولیه توسعه خود به سر می‌برد، اما برای Diffie پیش‌بینی ظهور یک شبکه جهانی با ارتباطات فراوان، کار چندان دشواری نبود. Diffie با خود می‌اندیشید که اگر در این شبکه، مردم بخواهند از ایمیل برای تبادل اطلاعات استفاده کنند، این حق آنان است که برای حفظ

حریم خصوصی خود، به دنبال رمز کردن پیام‌های خود باشند. اما در این میان یک مشکل وجود داشت، کلید چگونه باید منتقل می‌شد؟! او با خود می‌گفت دو فرد غریبه در یک شبکه، چگونه می‌توانند پیام‌های مبادله شده بین هم‌دیگر را رمز کنند. تعداد زیاد ارتباطات و نرخ بالای تغییر کلید، تبادل کلید را عملًا غیرممکن می‌ساخت.

در سال ۱۹۷۴، Diffie به عنوان یک رمزنگار مستقل، به منظور ارایه راه کارهای خود برای حل مشکل تبادل کلید، به آزمایشگاه توماس واتسون در IBM دعوت شد. بسیاری از شنونده‌های ارایه‌اش، در مورد آینده ایده‌های او دچار تردید بودند. تنها پاسخ مثبت از سوی Alan Konheim (آلن کانهایم)، یکی از متخصصین ارشد رمزنگاری IBM بود. نکته جالبی که او بدان اشاره کرد، این بود که شخص دیگری به نام Hellman که استاد دانشگاه استنفورد بود، در مورد همین چالش، ارایه‌ی در همین آزمایشگاه داشته. Diffie هزاران کیلومتر را پیمود تا به کسی برسد که می‌توانست ایده‌های خود را با او در میان بگذارد.

Martin Hellman در سال ۱۹۴۵ در یک محله یهودی‌نشین در نیویورک متولد شد. او دوست داشت از بقیه متفاوت باشد، به همین سبب علاقه خود به رمزنگاری را در این تمایل به متفاوت بودن می‌دید. همکارانش به او

گفته بودند که دیوانه است که در زمینه رمزگاری تحقیق می‌کند؛ چراکه عملاً دارد با NSA و بودجه چند میلیارد دلاری آن رقابت می‌کند. چگونه می‌توانست به کشف چیزی امیدوار باشد که آن‌ها قبلانمی‌دانستند؟ و اگر چیزی کشف می‌کرد، NSA آن را طبقه بندی می‌کرد.

Hellman در سپتامبر ۱۹۷۴، یک تماس غیرمنتظره داشت، و او کسی جز Diffie نبود. Hellman را نشنیده بود، اما با اکراه با یک قرار ملاقات نیم ساعته برای بعد از ظهر همان روز موافقت کرد. Diffie می‌گوید: «به همسرم قول داده بودم که برای نگهداری از بچه‌ها در خانه باشم، بنابراین او به خانه من آمد و با هم شام خوردیم، و حوالی نیمه شب رفت. ما از لحاظ شخصیتی خیلی با هم متفاوت بودیم. این مثل تنفس یک هوای تازه برای من بود». این دو با هم شروع به مطالعه مشکل توزیع کلید کردند و نامیدانه در تلاش برای یافتن جایگزینی برای انتقال فیزیکی کلیدها در فواصل بسیار دور بودند. در ادامه نیز Ralph Merkle از تیم قبلی تحقیقاتی خود جدا شد و بدان‌ها پیوست.

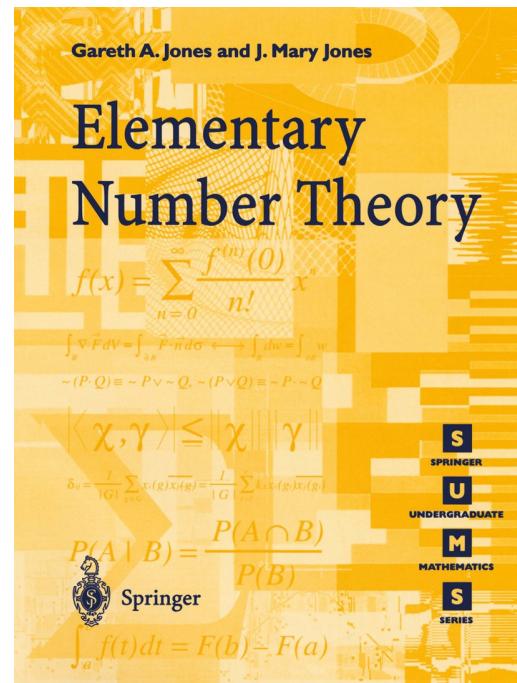
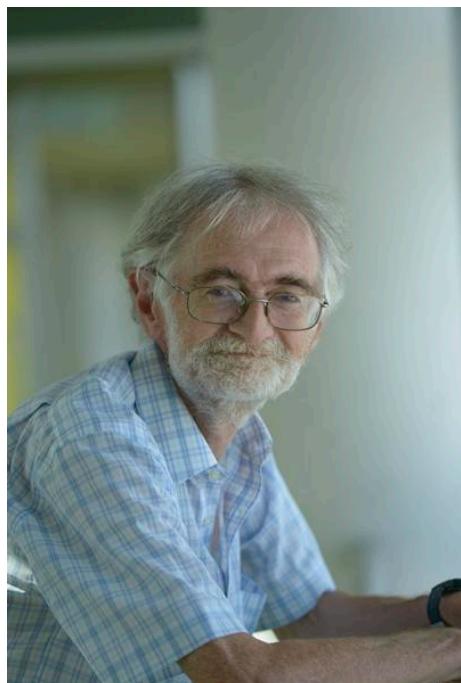
مقاله‌ای با عنوان New Direction in Cryptography و Hellman در آن از ایده اعجاب‌آور

خود سخن گفتند. به جرات می‌توان گفت که این مقاله یکی از مهم‌ترین مقالات در تاریخ علم رمزنگاری است. جالب این است که Malcom Williamson و James H. Ellis, Clifford Cocks، روشی مشابه با کار-Diffie Hellman را پیشتر کشف کرده بودند، اما به دلیل این‌که آن‌ها از کارمندان Government Communications Headquarters (GCHQ) که از سازمان‌های اطلاعاتی انگلیس محسوب می‌شد، بودند، ایده آن‌ها به صورت محروم‌انه باقی ماند. گرچه سرانجام در سال ۱۹۹۷ این ایده اجازه انتشار یافت.

الگوریتم کلید نامتقارن (Asymmetric Key Algorithm) نامی است که ما برای ایده جدید Diffie و Hellman برگزیدیم. در این نوع از سامانه‌ها، همان‌طور که از نام آن پیداست، کلید رمزگذاری و رمزگشایی یکسان نیستند. امروزه الگوریتم‌های کلید نامتقارن (Asymmetric Key Algorithm)، کاربردهای فراوانی دارد، از حوزه رمزگذاری گرفته تا حل چالش تبادل کلید (Key Exchange) و امضای دیجیتال (Digital Signature). اکثریت مسایل موجود در حوزه الگوریتم‌های کلید نامتقارن، مبتنی بر حل مسایل نظریه اعداد (Number Theory) است. بدین‌سان ما در ادامه مروری بر مفاهیم مقدماتی نظریه اعداد خواهیم داشت.

گنروی پر نظر په اعڑا

Jones, Gareth A., Jones, Josephine M.. Elementary Number Theory. United Kingdom: Springer London, 1998.



برای هر دو عدد صحیح a و b و عدد $n \neq 0$ ، a را همنهشت b به پیمانه یا هنگ n گوییم اگر و فقط

تعريف ۱

اگر برای یک عدد صحیح k داشته باشیم:

$$a - b = k \times n \iff a \equiv b \pmod{n}$$

۲۱ همنهشت ۵ در هنگ ۴ است، زیرا: $21 - 5 = 4 \times 4 \implies 21 \equiv 5 \pmod{4}$

مثال ۱

آخرین رقم 7^{100} چند است؟

مثال ۲

بزرگ‌ترین مقسوم‌علیه مشترک

قضیه ۱

اگر a و b دو عدد طبیعی باشند که در آن $0 < b$ باشد، آن‌گاه یک جفت عدد طبیعی q و r وجود دارد به طوری که:

$$a = qb + r \quad 0 \leq r < b$$

به b مقسوم‌علیه، q خارج قسمت و به r باقیمانده گفته می‌شود.

- $\text{gcd}(a, b)$ یا همان بزرگ‌ترین مقسوم‌علیه مشترک.
- اگر باقیمانده صفر باشد آن‌گاه گوییم، a بر b بخش‌پذیر است، یا b یک فاکتور a است ($b|a$).
- اگر $a|c$ و $a|b$ آن‌گاه $a|c|b$.
- اگر $a|b$ و $c|d$ آن‌گاه $ac|bd$.
- اگر $d|a$ و $d \neq 0$ ، آن‌گاه $|d| \leq |a|$.

تعريف ۲ مجموعه کامل ماندها $\mathbb{Z}_n = \{r_1, r_2, \dots, r_n\}$ عدد صحیح مجموعه n را مجموعه کامل

ماندها (Complete Set Of Residues) در هنگ n گوییم، هرگاه برای هر عدد صحیح a دقیقاً یک عدد r_i در

مجموعه وجود داشته باشد، به طوری که

$$a \equiv r_i \pmod{n}$$

مثال ۳ به عنوان مثال مجموعه کامل ماندها در هنگ $n = 4$ است. همچنین:

$$\{1, 2, 3, 4\}, \{-1, 0, 1, 2\}$$

- همواره دارای n عضو است.
- مجموعه $\{0, 1, 2, \dots, n - 1\}$ تنها مجموعه کامل ماندها با اعضای مثبت و کوچکتر از n .

تعريف ۳

مجموعه کاهش یافته مانده‌ها زیرمجموعه‌ای از مجموعه کامل مانده‌ها یعنی $\{1, 2, \dots, n-1\}$

را که کلیه عناصر آن نسبت به n اول باشند را، مجموعه کاهش یافته مانده‌ها (Reduced Set of Residues) در

هنگ n می‌گوییم.

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n | (a, n) = 1\} \quad \mathbb{Z}_n^* \subset \mathbb{Z}_n.$$

مثال ۴

به عنوان مثال مجموعه کامل مانده‌ها در هنگ $n = 8$ است،

و مجموع کاهش یافته مانده‌ها به صورت $\mathbb{Z}_n^* = \{1, 3, 5, 7\}$ خواهد بود.

- تعداد عناصر مجموعه کاهش یافته مانده‌ها را اصطلاحاً تابع اویلر می‌نامیم ($|\mathbb{Z}_n^*| = \phi(n)$).
- مجموعه کاهش یافته مانده‌ها برای یک عدد اول (p) چگونه است؟

یک سوال



بلوک‌های $n = 30$ از حروف را در اختیار دارم. می‌خواهم عملیات جایگشتی را بر روی این بلوک اعمال کنم.
یعنی جایگاه‌های حروف را که مجموعه $\{0, 1, 2, \dots, n - 1\}$ نشان می‌دهم را به هم بریزم.

یک سوال (ادامه)

قضیه ۲

اگر $\{r_1, r_2, \dots, r_{\phi(n)}\}$ مجموع کاهش یافته مانده‌ها باشد، آن‌گاه مجموعه حاصل شده از ضرب عدد a در مجموعه کاهش یافته مانده‌ها یعنی $\{ar_1, ar_2, \dots, ar_{\phi(n)}\}$ یک جایگشت کامل از مجموعه اولیه است، اگر $(a, n) = 1$ باشد.

اثبات. باید دو مورد را اثبات کنیم. نخست آن‌که ar_i عضوی از مجموع کاهش یافته مانده‌ها است. می‌دانیم که

است. در ضمن به ازای هر $(ar_i, n) = 1$

$$\forall i, j \in \{1, 2, \dots, \phi(n)\}, \quad i \neq j, \quad ar_i \neq ar_j$$

یعنی دو عضو به یک عضو نگاشت نشود. اما این مورد را چگونه اثبات کنیم؟!! فرض خلف ...



محاسبه تابع اویلر

Leonhard Euler ریاضیدان و فیزیکدان برجسته سوئیسی قرن ۱۸ که نقش محوری در بسیاری از علوم ریاضی نظیر گراف، آنالیز اعداد، محاسبات نامتناهی، هندسه و ... نقش محوری و پیشگام را ایفا می‌کرد. تقریباً نیمی از آثار اویلر پس از نابینا شدن او تالیف شد.



تابع اویلر ($\phi(n)$) یا همان تعداد عناصر مجموعه کاهش‌یافته مانده‌ها را چگونه می‌توان محاسبه کرد؟

лем ۱. اگر p یک عدد اول باشد، براحتی می‌دانیم که $\phi(p) = p - 1$.

лем ۲. اگر $n = pq$ باشد که p و q دو عدد اول باشند، در این صورت داریم:

$$\phi(n) = (p - 1)(q - 1) \quad (1)$$

лем ۳. اگر $n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t}$ باشد که p_i فاکتورهای n است، در این صورت داریم:

$$\phi(n) = \prod_{i=1}^t p_i^{e_i-1} (p_i - 1) \quad (2)$$

محاسبه تابع اویلر (ادامه)

اگر $n = 72$ باشد، آن‌گاه چه تعداد عدد مثبت کوچکتر از n وجود دارد که نسبت به آن اول باشد؟

مثال ۵

$$n = 72 = 2^3 \times 3^2 \implies \phi(72) = 4 \times (2 - 1) \times 3 \times (3 - 1) = 24$$

اگر $n = p^k$ باشد، آن‌گاه چه تعداد عدد مثبت کوچکتر از n وجود دارد که نسبت به آن اول باشد؟

مثال ۶

$$\phi(p^k) = p^{k-1} \times (p - 1)$$

برای عدد $5^3 \times 2^7 = 16000$ ، تعداد اعدادی که نسبت به این عدد اول هستند برابر است با

مثال ۷

$$\phi(16000) = (1 \times 2^6) \times (4 \times 5^2) = 6800$$

زنگ تفريح



بِرُوزگار سلامت سُكّنگان دیاب
چو سائل از توبه زاری طلب کند چیزی
که جبر خاطر مسکین بلا بکرداند
بده و کرنہ سُمگر به زور بستاد

کاروانی در زمین یونان بِرَدَند و نعمتِ بی قیاس بِپُرَدَند. بازرگانان گریه و زاری کردند و خدا و پیمبر شفیع آوردند
و فایده نبود:

په غم داره از گریه کاروان
په پیروز شد دزد تیره روان

لقمان حکیم اnder آن کاروان بود. یکی گفتش از کاروانیان: مگر اینان را نصیحتی کنی و موعظه‌ای گویی تا
ظرفی از مال ما دست بدارند، که دریغ باشد چندین نعمت که ضایع شود. گفت: دریغ کلمه حکمت با ایشان
گفتند:

آهنی را که هوریانه بفورد
نتوان برد از او به صیقل زنگ
نرود میخ آهنین در سنگ
با سیه دل په سود گفتند وعظ

همانا که جرم از طرف ماست:
به روزگار سلامت شکستگان دریاب
که جبر خاطر مسکین بلا بگرداند
په سائل از تو به زاری طلب کند چیزی

معکوس عدد

معکوس در هنگ n عدد x را معکوس عدد a در هنگ n گوییم، اگر داشته باشیم:

تعريف ۴

$$ax \equiv 1 \pmod{n}$$

قضیه ۳

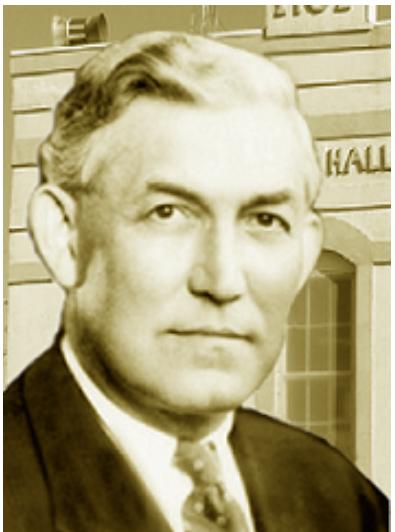
معادله $ax \equiv 1 \pmod{n}$ تنها یک جواب دارد، اگر و فقط اگر $(a, n) = 1$. یعنی تنها در صورتی که n نسبت به a اول باشد، عدد a دارای معکوس یکتا در هنگ n خواهد بود.

معکوس عدد ۵ در هنگ ۷، برابر ۳ خواهد بود.

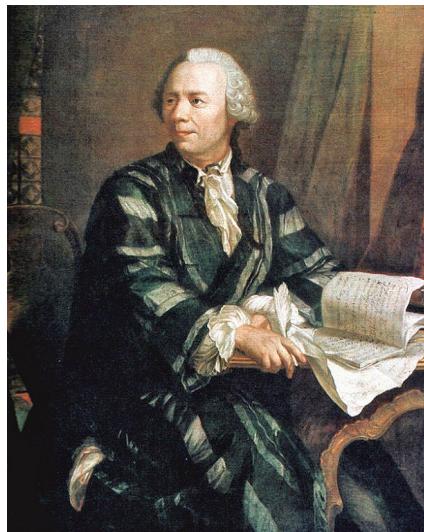
مثال ۸

$$ax \equiv 1 \pmod{n} \implies 5 \times 3 \equiv 1 \pmod{7}$$

قضیه اویلر-فرما



Robert Daniel Carmichael



Leonhard Euler



Pierre de Fermat

قضیه ۴ قضیه اویلر-فرما

اگر دو عدد a و n نسبت به هم دیگر اول باشند، آن‌گاه خواهیم داشت:

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

نتیجه ۱. اگر p یک عدد اول باشد و a نسبت به p اول باشد، آن‌گاه:

اویلر در سال ۱۷۳۶، اثباتی برای قضیه کوچک فرما ارایه داده. در واقع قضیه کوچک فرما بدین گونه است که اگر عدد a نسبت به عدد اول p ، اول باشد، آن‌گاه عبارت زیر همواره برقرار است: $a^{p-1} \equiv 1 \pmod{p}$.

با کمی دقیق توان دریافت که عبارت فوق، به نوعی یک حالت خاص از قضیه اویلر-فرما است. در واقع اویلر قضیه‌ای را که فرما بدون اثبات بیان کرده بود را اثبات کرد، و علاوه بر آن، قضیه را در یک حالت کلی، بدون این قید که n باید عدد اولی باشد، را نیز بیان کرد.

اثبات. اگر $\mathbb{Z}_n^* = \{r_1, r_2, \dots, r_{\phi(n)}\}$ مجموع کاهش‌یافته مانده‌ها باشد، آن‌گاه مجموعه حاصل شده از ضرب عدد a در مجموعه کاهش‌یافته مانده‌ها یعنی $\{ar_1, ar_2, \dots, ar_{\phi(n)}\}$ یک جایگشت کامل از مجموعه اولیه است. پس داریم:

$$\prod_{i=1}^{\phi(n)} (ar_i \pmod n) = \prod_{i=1}^{\phi(n)} r_i \implies \left(a^{\phi(n)}\right) \left(\prod_{i=1}^{\phi(n)} r_i\right) = \left(\prod_{i=1}^{\phi(n)} r_i\right) \implies a^{\phi(n)} \equiv 1 \pmod n$$



مثال ۹

رقم آخر عدد 3^{87} چند است؟

- دقت کنید که در واقع ما به دنبال پاسخ $3^{87} \pmod{10}$ هستم. می‌دانیم که: 
- عدد سه و ده نسبت به هم اول هستند، یعنی $(3, 10) = 1$ ●

- برطبق قضیه اویلر-فرما داریم: $3^4 \equiv 1 \pmod{10}$ ●

 آن‌گاه براحتی می‌توانیم بنویسیم که:

$$3^{87} = 3^{4 \times 21 + 3} = (3^4)^{21} \times (3^3) = 1^{21} \times 27 = 7 \pmod{10}.$$

قضیه ۵

اگر دو عدد a و n نسبت به هم اول باشند، آن‌گاه پاسخ یکتای معادله $ax \equiv b \pmod{n}$ برابر است با:

$$x = ba^{\phi(n)-1} \pmod{n}$$

نتیجه ۲. اگر دو عدد a و n نسبت به هم اول باشند، آن‌گاه پاسخ یکتای معادله $ax \equiv 1 \pmod{n}$ برابر است

$$x = a^{\phi(n)-1} \pmod{n}$$

پاسخ معادله همنهشتی $5x \equiv 3 \pmod{24}$ برابر است با:

$$\phi(24) = 2^2 \times (2 - 1) \times (3 - 1) = 8$$

$$x = 3 \times 5^{\phi(24)-1} = 3 \times 5^7 = 15 \pmod{24}$$

مثال ۱۰

محاسبه معکوس



✓ مامی توانیم معکوس یک عدد را در زمان چندجمله‌ای پیدا کنیم، بدون آن که لزوماً $\phi(n)$ را داشته باشیم.

$$ax \equiv 1 \pmod{n} \implies x = a^{\phi(n)-1} \pmod{n}$$

✓ یافتن $\phi(n)$ خود مستلزم حل شدن مساله تجزیه عدد است.

$$n = p_1^{e_1} p_2^{e_2} \cdots p_t^{e_t} \implies \phi(n) = \prod_{i=1}^t p_i^{e_i-1} (p_i - 1)$$

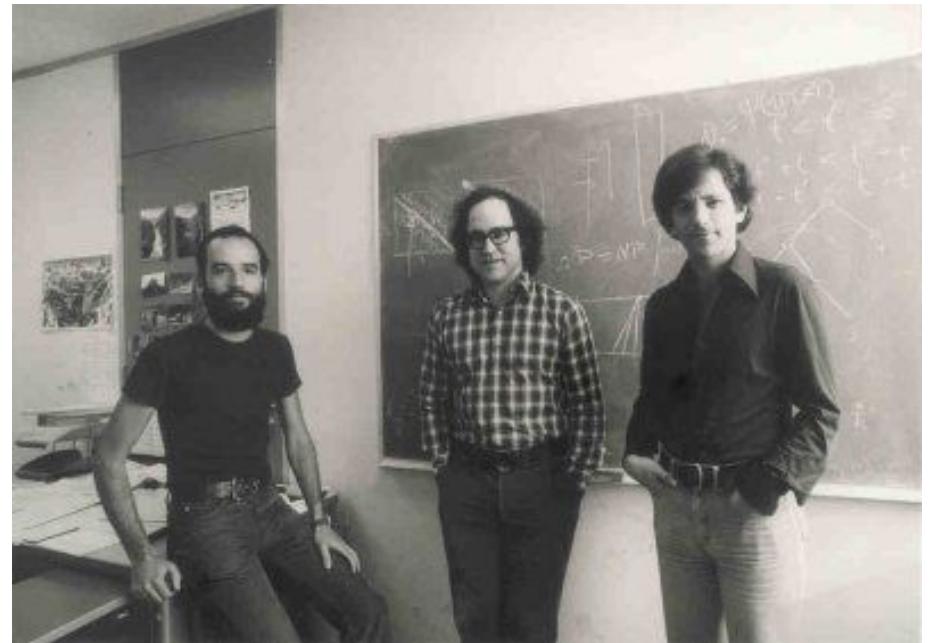
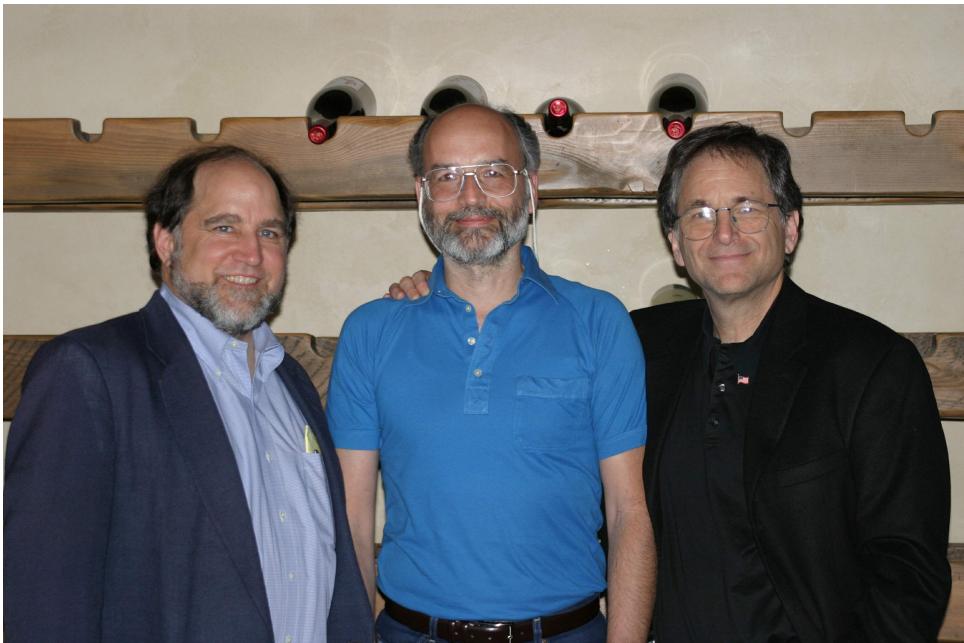
✓ یافتن تجزیه عدد کار سختی است، پس یافتن $\phi(n)$ سخت است.

تمرین

- پیچیدگی زمانی سریع‌ترین الگوریتم برای تجزیه اعداد اول چگونه است؟
- اعداد Semiprime چیست؟ تجزیه این اعداد نسبت به سایر اعداد پیچیده‌تر است یا ساده‌تر؟

RSA **الجُنُوبِيّ**

الگوریتم RSA



Ron Rivest, Adi Shamir, Leonard Adleman

RSA یک الگوریتم کلید نامتقارن است که در سال ۱۹۷۷ منتشر شد. گرچه ادعا می‌شود که در سال ۱۹۷۳ فردی به نام Clifford Cocks در GCHQ، الگوریتم مشابهی را پیشنهاد داده بود.

الگوریتم RSA (ادامه)

۱) عدد n از حاصل ضرب دو عدد اول خیلی بزرگ به مانند p و q حاصل می‌شود ($p \neq q$ و $n = pq$).

۲) پارامتر e را به عنوان کلید عمومی (Public Key) در نظر می‌گیریم، به گونه‌ای که

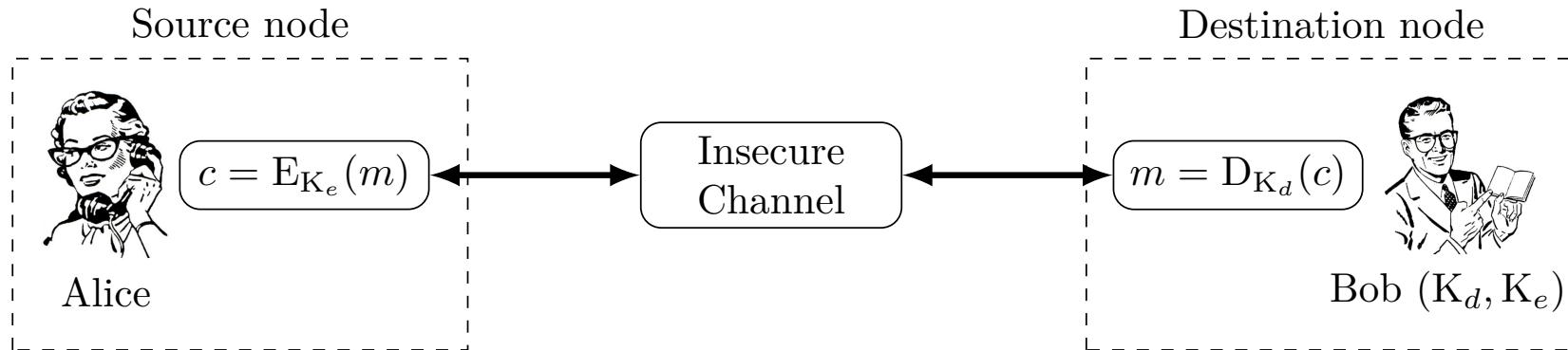
$$1 < e < \phi(n), \quad (e, \phi(n)) = 1.$$

۳) پارامتر d را به عنوان کلید محرمانه (Private Key) در نظر می‌گیریم، به گونه‌ای که:

$$ed \equiv 1 \pmod{\phi(n)},$$

حمله‌گر (Attacker)  حمله‌گر (Attacker) p و q را دارد. اما چون مساله تجزیه عدد NP است، او نمی‌تواند e و n را داشته باشد. همین طور او از روی $\phi(n)$ دست پیدا کند.

الگوریتم RSA (ادامه)



کاربر با کلید عمومی خود پیام را رمز می‌کند.

$$c = m^e \pmod{n}, \quad 0 \leq m < n.$$

Bob نیز به عنوان گیرنده پیام، کلید محربانه d را دارد و می‌تواند این گونه پیام را بازگشایی کند.

$$(c)^d = (m^e)^d = m^{k\phi(n)+1} = (m^{\phi(n)})^k \times m = m \pmod{n}$$



اگر m نسبت به n اول نباشد چه اتفاقی می‌افتد؟ این پیوند را نگاه کنید.

فرض کنید که

مثال ۱۱

$$p = 3, \quad q = 11, \quad n = pq = 3 \times 11 = 33, \quad \phi(33) = (p-1)(q-1) = 20$$

- در گام نخست، e پارامتری است که باید در بازه $(1, \phi(n))$ انتخاب شود، مثلا $7 = e$.
- پارامتر d نیز باید در رابطه $ed \equiv 1 \pmod{\phi(n)}$ صدق کند.

$$7d = 1 \pmod{20} \implies d = 7^{\phi(20)-1} = 7^7 \pmod{20} = 3$$

پس کلید محرمانه برابر با $(e, n) = (7, 33)$ و کلید عمومی نیز $d = 3$ است.

- در ادامه متن رمز $m = 2$ به صورت زیر حاصل می‌گردد.

$$c = m^e \pmod{n} \implies c = 2^7 \pmod{33} = 29$$

- در نهایت نیز در گیرنده، برای رمزگشایی خواهیم داشت:

$$m = c^d \mod n \implies m = 29^3 \mod 33 = 2$$

الگوریتم RSA نخستین بار توسط Leonard Adleman و Ron Rivest, Adi Shamir که هر سه در MIT تدریس می‌کردند، در سال ۱۹۷۷ ارایه شد. ظاهرا یک سامانه معادل در سال ۱۹۷۳ در GCHQ (Government Communications Headquarters) توسط یک ریاضیدان انگلیسی به نام Clifford Cocks، نیز ارایه شده بود. اما به دلیل این که این سامانه طبقه‌بندی شده بود، تا سال ۱۹۹۷ اطلاعاتی از آن به بیرون درز نکرد.

فرض کنید که Alice می‌خواهد برای Bob، یک پیام رمز شده را ارسال کند. برطبق الگوریتم‌های کلید نامتقارن، Alice می‌بایست متن اصلی را با کلید عمومی Bob، رمز کرده و برای او ارسال کند. در سوی دیگر، Bob بعد از دریافت متن رمز، می‌تواند با بهره‌گیری از کلید محرمانه خود، پیام را رمزگشایی کند. دقت کنید که تنها Bob می‌تواند این کار را انجام دهد، چراکه تنها او به کلید محرمانه دسترسی دارد.

وجود تابع یک‌طرفه (One-way Function) را می‌توان به جرات هسته اصلی یک الگوریتم نامتقارن دانست، تابعی که محاسبه آن از یک سمت آسان و در سمت معکوس بسیار سخت باشد. Adleman و Rivest, Shamir ماهها تلاش می‌کنند تا به یک تابع یک‌طرفه دست پیدا کنند. جالب است بدانید که Adleman نیز که یک

ریاضیدان بود، وظیفه داشت تا توابع پیشنهادی را بررسی و نقاط ضعف شان را در یک طرفه بودن پیدا کند. سرانجام در آوریل ۱۹۷۷، Rivest به ایده‌ای برای یک تابع یک‌طرفه مناسب دست یافت. این سه نفر به دلیل تلاش‌های بی‌وقفه خود برای ابداع الگوریتم RSA، در سال ۲۰۰۲ جایزه تورینگ (یک جوایز نوبل ریاضیات) را تصاحب کردند. این [فیلم](#) و این [رشته تؤییت‌ها](#) را نیز حتما مشاهده کنید.

مراحل الگوریتم RSA را می‌توان به شرح زیر بیان کرد:

- ابتدا یک عدد طبیعی به مانند n را که حاصل ضرب دو عدد اول بسیار بزرگ به مانند p و q است را در نظر می‌گیریم. پس داریم: $pq = n$. باید دقت کنیم که p نباید مساوی با q باشد. معمول اعدادی که امروزه برای n در نظر گرفته می‌شود، حداقل طولی برابر با ۱۰۲۴ را دارد. این بدان معنا است که p و q ، دو عدد حدود ۳۰۰ رقمی است.

- در گام بعدی، کلید عمومی Bob، که ما آن را با e نمایش می‌دهیم، باید تولید شود. یک عدد دلخواه است،

اما به شرطی که دو قید زیر را برابر ده سازد.

$$1 < e < \phi(n), \quad (e, \phi(n)) = 1.$$

و e و n به عنوان کلید عمومی، در اختیار هر کسی که بخواهد با Bob ارتباط امن داشته باشد، قرار می‌گیرد.

بدیهی است که حمله‌گر نیز به این دو پارامتر دسترسی دارد.

- برای تولید کلید محرمانه d می‌بایست معادله زیر را حل کرد:

$$ed \equiv 1 \pmod{\phi(n)}. \quad (3)$$

در حقیقت d عکس e در هنگ $\phi(n)$ است. این مقدار می‌تواند به صورت بهینه‌ای توسط الگوریتم تعمیم‌یافته

اقلیدس محاسبه شود. علاوه بر d ، سه پارامتر $(p - 1)(q - 1) = \phi(n)$ ، p و q نیز محرمانه است،

که فقط در اختیار Bob است، Alice و حمله‌گر نمی‌توانند با داشتن n در زمان چندجمله‌ای $\phi(n)$ را پیدا کنند،

چراکه p و q را ندارند. در ضمن رسیدن به p و q ، مستلزم حل مساله تجزیه عدد است، که می‌دانیم این مساله

جزو مسایل NP Hard است.

نکته دیگری که باید به آن توجه داشت این است که یافتن معکوس e و رسیدن به d ، یا به عبارت بهتر حل (۳)، می‌تواند توسط Bob در زمان چند جمله‌ای انجام شود. اما حمله‌گر نمی‌تواند این کار را انجام دهد، چراکه اصلاً $\phi(n)$ را ندارد.

• Alice متن اصلی m را به صورت زیر رمز می‌کند:

$$c = m^e \pmod{n}, \quad 0 \leq m < n.$$

سپس پیام c را برای Bob ارسال می‌کند. Bob نیز کافی است عملیات زیر را برای رمزگشایی انجام دهد.

$$(c)^d = (m^e)^d = m^{k\phi(n)+1} = (m^{\phi(n)})^k \times m = m \pmod{n} \quad (4)$$

نخست آن که می‌دانیم فقط Bob به دلیل این که d را دارد، می‌تواند عملیات صورت پذیرفته در (۴) را انجام دهد. در ضمن رابطه $m^{\phi(n)} = 1 \pmod{n}$ زمانی معتبر است که m برابر با

ضریبی از p و/یا q نباشد (یعنی $m \neq kpq$ یا $m \neq kq$ یا $m \neq kp$). با کمی دقت می‌توان دریافت که به احتمال بسیار کمی، پارامتر m نسبت به n اول نیست. گرچه Adleman, Rivest, Shamir در مقاله خود ثابت کردند که در صورتی که حتی m نسبت به n اول نیز نباشد، باز رابطه (۴) برقرار خواهد بود. [این پیوند](#) را نگاه کنید.

نکته دیگر این است که m باید از n کوچکتر باشد (یعنی $n < m$). به دلیل این که محاسبات در پیمانه n است، m های بزرگتر از n نمی‌تواند به مجموعه کامل مانده‌ها به درستی نگاشت شود ([این پیوند](#)).

از دیدگاه کاربرد اگر بخواهیم به RSA نگاهی بیاندازیم، باید به مواردی که در ادامه می‌آید اشاره کرد. فقط آن که بدانیم RSA به عنوان یک الگوریتم‌های کلید نامتقارن، به مراتب نسبت به الگوریتم‌های کلید متقارن نظیر AES، کارایی پایینی دارند. به همین علت به طور مشخص در سامانه‌های رمزگذاری مورد استفاده قرار نمی‌گیرد.

در حقیقت جایگاه RSA را باید در مواردی نظیر حل چالش تبادل کلید و امضای دیجیتال جستجو کرد.

IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-22, NO. 6

New Directions in Cryptography

Invited Paper

WHITFIELD DIFFIE AND MARTIN E. HELLMAN, MEMBER, IEEE

I. INTRODUCTION

WE STAND TODAY on the brink of a revolution in cryptography. The development of cheap digital hardware has freed it from the design limitations of mechanical computing and brought the cost of high grade cryptographic devices down to where they can be used in such commercial applications as remote cash dispensers and computer terminals. In turn, such applications create a need for new types of cryptographic systems which minimize the necessity of secure key distribution channels and supply the equivalent of a written signature. At the same time, theoretical developments in information theory and computer science show promise of providing provably secure cryptosystems, changing this ancient art into a science.



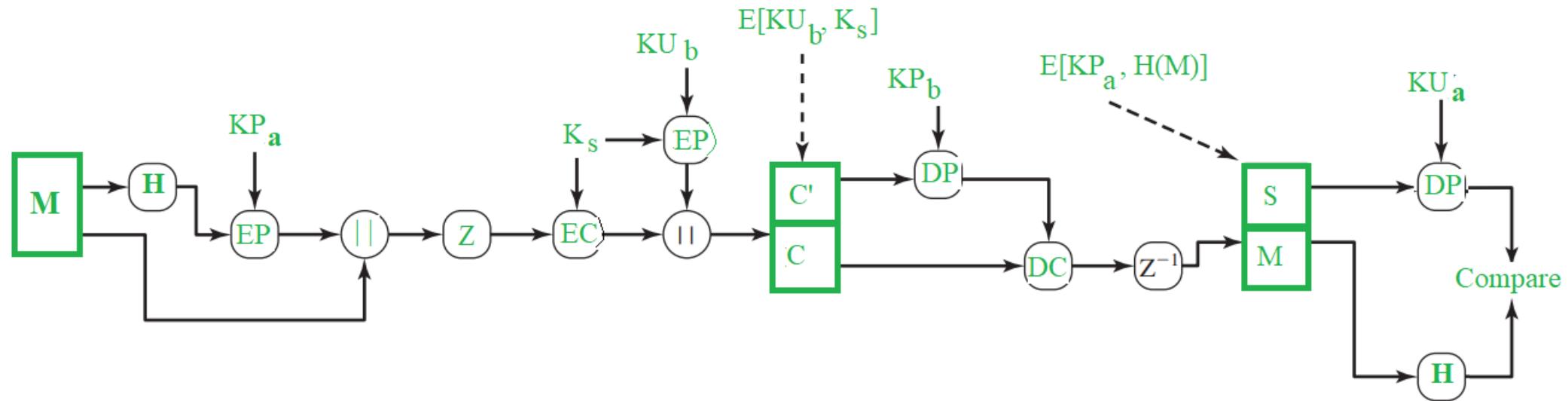
یک راه کار برای رمزگذاری و امضای دیجیتال ایمیل‌ها، فایل‌ها و ... به صورت انتها به انتهای

در سال ۱۹۹۱ توسط Phil Zimmermann ابداع شد، و بعدها توسط IETF در استاندارد RFC 4880 تحت

نام OpenPGP استاندارد شد.

یک پیاده‌سازی متن‌باز از OpenPGP gpg است.

در عمل - PGP - (ادامه) RSA



- ❷ چکیده (Hash) پیام با کلید محروم‌نامه Alice رمز شده و در کنار پیام، به عنوان امضا دیجیتال قرار می‌گیرد.
- ❸ عملیات فشرده‌سازی بر روی پیام و چکیده آن صورت می‌پذیرد.
- ❹ با استفاده از یک کلید تصادفی به عنوان کلید جلسه (K_s) و با بهره‌گیری از یک الگوریتم رمزنگاری متقارن، پیام رمز شده و حاصل کار به همراه رمزشده K_s با کلید عمومی Bob، برای ارسال می‌شود.

من متن زیر را از این رشته توثیق با دخل و تصرف برداشتیم.

باورش ممکنه سخت باشه، اما تا ۱۹۹۲، رمزنگاری به عنوان تجهیزات نظامی کمکی در لیست تسليحات ایالات متحده شناخته می‌شد، و صادرکردن هر نوع سیستم رمزنگاری از روش‌ها گرفته تا حتی شرح اون‌ها، به شدت سخت‌گیرانه و نیازمند مجوز بود. با انتشار عمومی الگوریتم‌های رمزنگاری مثل DES و روش‌های رمزنگاری نامتقارن، ظهور اینترنت، و البته تلاش‌ها و فدایکاری عده‌ای در راستای حفظ حریم خصوصی (با وجود ریسک محاکمه شدن)، راه برای برچیده شدن چنین سیاست‌هایی هموارتر شد.

Phil Zimmermann یکی از اون افراد بود. تا اون موقع، رمزنگاری قوی تنها در سلطه دولتها بود. اما چی می‌شد اگه نرم افزاری داشتیم که الگوریتم‌های رمزنگاری RSA رو به کامپیوترهای شخصی می‌آورد، و افراد عادی می‌توانستن مکالمه‌های روزمره و حتی فایل‌هاشون رو خودشون رمزنگاری کنن؟ این سؤالی بود که در ۱۹۷۷ به ذهن او خطور کرد، اما کار جدی برای پاسخ به اون رو تا ۱۹۸۴ شروع نکرد. هرچی بیشتر به مشکلات پیرامون حریم خصوصی فکر می‌کرد، بیشتر به اهمیت این پروژه پی می‌برد.

که تخصص ویژه‌ای در رمزنگاری نداشت، کند پیش می‌رفت. در حالی که شغل اصلی خودش زیمرمن (Zimmermann) رو داشت، دارای همسر و دو فرزند هم بود، و همین موضوع باعث می‌شد تا نتونه با اون سرعتی که می‌خواهد پروژه را پیش ببره. با این حال، ازش دست نکشید. در ۱۹۸۶ موفق شد RSA را پیاده‌سازی کنه. تا سال ۱۹۹۱، او همچنان نرم‌افزار کاملی برای عرضه نداشت، تا اینکه جو بایدن (Joe Biden) (که در اون زمان سناتور بود) قدمی برداشت که باعث شد زیمرمن (Zimmermann) چندین ماه بی‌وقفه روی پروژه کار کنه تا اون رو به پایان برسونه. در ژانویه ۱۹۹۱، بایدن (Biden) لایحه‌ای را پیشنهاد داد. جایی در این متن او مده که شرکت‌ها موظف هستند در صورت ارایه درخواست قانونی محتوای متنی/صوتی و داده افراد را در اختیار دولت قرار بدن. این همون آینده‌ای بود گورکن (George Orwell) در رمان ۱۹۸۴ ترسیم کرده بود، و زیمرمن (Zimmermann) تلاش می‌کرد ازش جلوگیری کنه. همین موضوع به او هدف تازه‌ای داد. حالا مسیرش مشخص بود. باید قبل از اینکه کنگره راهی پیدا می‌کرد تا جلوی ارتباط امن و خصوصی افراد را بگیره، نرم‌افزارش را آماده و عرضه می‌کرد.

پنجم ژوئن ۱۹۹۱ روزی بود که پس از گذر از مسیری پر پیچ و خم که چندین سال از عمرش را صرفش کرده بود،

بالاخره نرم افزارش رو عمومی کرد و اسم او را PGP (Pretty Good Privacy) گذاشت یا «حریم خصوصی بسیار خوب». فکر گرفتن کارمزد جهت استفاده از PGP از ذهنش عبور کرده بود، اما از اونجایی که می‌ترسید روزی دولت استفاده از رمزنگاری رو ممنوع کنه، می‌خواست قبل از رسیدن چنین روزی همه تا حدامکان از ابزارهای حریم خصوصی بهره‌مند بشن. درنتیجه، تصمیم گرفت ثمره سال‌ها زحمتش رو مجانی منتشر کنه. Zimmermann حتی تا مرز از دستدادن خونه خودش هم رفت چون از ابتدای ۱۹۹۱ تا زمان انتشار از پرداخت پنج قسط وام خونه‌ش عقب مونده بود و مجبور شد بانک رو قانع کنه تا خونه‌ش رو نگیره. این، از دید من، نشون از فدایکاری بزرگش داره.

او که در اون زمان اطلاعات چندانی راجع به اینترنت نداشت، اولین نسخه PGP رو به دو نفر از دوستانش داد تا اون رو بارگذاری کنن. نرم افزار خیلی زود دست به دست شد و سر از اروپا و کشورهای دیگه درآورد. «مثل هزاران دونه قاصدک در دست باد»، Zimmermann توصیف می‌کنه PGP در اینترنت پخش می‌شد. درحالی که استفاده از PGP در ایالات متحده آزاد بود، وقتی در کشورهای دیگه پدیدار شد، در درسرهای قانونی‌ای برای

Zimmermann در پی داشت. همون طور که می‌دونیم، صادرکردن چنین رمزگاری قدرتمندی در اون زمان غیرقانونی محسوب می‌شد. سال ۱۹۹۳ و به مدت سه سال، زیمرمن درگیر یه پرونده قضایی با دولت آمریکا شد. جرم؟ زیرپاگذاشتن قانون کنترل صادرات اسلحه.

Zimmermann در پاسخی زیرکانه، با همکاری انتشارات MIT، یکی از برجسته‌ترین ناشرها در سطح ملی و جهانی، سورس کد PGP را در قالب کتاب منتشر کرد. براساس متمم اول قانون اساسی ایالات متحده و زیرمجموعه قانون آزادی بیان، نشر و صادرات کتاب هیچ‌گونه محدودیتی نداره. با انتشار کد PGP در قالب کتاب و فروش در سطح جهانی، Zimmermann سعی داشت نشون بده اتهامش در صادرکردن «نرم‌افزار» بی‌معنیه. در سال ۱۹۹۷، او پیشنهادی به IETF برای ایجاد یک استاندارد PGP متن‌باز (Open Source) ارایه کرد. این پیشنهاد پذیرفته شد و منجر به ایجاد پروتکل OpenPGP شد که فرمتهای استاندارد را برای کلیدهای رمزگذاری و پیام‌ها تعریف می‌کند. اگرچه در ابتدا فقط برای ایمن کردن پیام‌های ایمیل و پیوست‌ها استفاده می‌شد، اما امروزه برای طیف گسترده‌ای از موارد استفاده از جمله امضای دیجیتال، رمزگذاری کامل دیسک و محافظت

از شبکه استفاده می‌شود. PGP خدمات مختلفی نظیر محترمانگی، احراز اصالت، یکپارچگی، فشرده‌سازی و بخش‌بندی (Segmentation) را برای ما به ارمغان می‌آورد.

فرض کنید که Alice می‌خواهد پیامی را برای Bob ارسال کند و برای تامین امنیت آن به صورت انتهای، از PGP استفاده کند. او می‌بایست گام‌های زیر را بردارد:

- ابتدا یک امضا بر روی پیام به منظور احراز اصالت و حفظ یکپارچگی، می‌زند. برای امضا دیجیتال، نخست از پیام با استفاده از الگوریتم‌هایی نظیر SHA-1, MD5 و یا SHA-XXXX یک چکیده را بدست آورده، سپس آن را با استفاده از کلید محترمانه RSA خودش رمز می‌کند. هر کس دیگری می‌تواند این پیام را با استفاده از کلید عمومی Alice باز نماید و صحت امضا را تایید کند.
- در ادامه عملیات فشرده‌سازی با بهره‌گیری از الگوریتم‌هایی نظیر ZIP, ZLIB و یا BZip2، بر روی خروجی مرحله قبلی صورت می‌پذیرد. این عملیات حتماً باید بعد از امضا و قبل از رمزگذاری باشد.
- برای رمزگذاری، نیز Alice یک کلید تصادفی به عنوان کلید جلسه (K_s) را تولید می‌کند. سپس با استفاده

از این کلید، متن اصلی را با بکارگیری یک الگوریتم رمزنگاری متقارن نظیر 3DES, AES, CAST5, IDEA و Blowfish رمز می‌کند. این کلید را نیز با استفاده از کلید عمومی Bob رمز شده و در کنار متن رمز قرار می‌گیرد. در نهایت نیز برای Bob ارسال می‌شود.

پر واضح است که در سمت گیرنده Bob، عکس تمامی مراحل قبل را برای رسیدن به متن اصلی، تایید هویت فرستنده و صحت امضا طی می‌کند.

 چرا اول امضا کردیم و سپس رمز کردیم؟ در پاسخ باید گفت که فرض کنید که گیرنده‌ای بخواهد از محتوای نامناسب ایمیلی که از سمت شما دریافت کرده و با PGP تامین امنیت شده، پیش مراجع قضایی شکایت ببرد. در حالت فعلی او کافی است که اصل متن اصلی و امضا را پیش خود نگه دارد. اما در صورتی که اول رمز می‌کردیم و سپس امضا، او می‌بایست متن متن رمز به همراه امضا، و همچنین کلید و نوع الگوریتم رمزنگاری را ذخیره و به مراجع قضایی تحويل دهد.

 چرا فشرده‌سازی بعد از امضا و قبل از رمزنگاری قرار گرفته است؟ باز به همان علت قبل برای جلوگیری از

ذخیره نوع الگوریتم فشرده‌سازی بعد از امضا قرار گرفته. همچنین باید بدانیم که در فشرده‌سازی، افزونگی‌ها (Redundancy) از بین می‌رود و این خود ما را در برابر حملات موجود به الگوریتم‌های رمزگاری بیشتر مصون می‌کند. به این دلایل، دلیل کاهش سربار بخش رمزگذاری را نیز باید اضافه کرد.

GPG یک پیاده‌سازی متن‌باز از پروتکل OpenPGP است، که بر روی بسیاری از سیستم‌عامل نصب و اجرا می‌شود. ما در اینجا، PGP را بر روی Linux تست خواهیم کرد. برای شروع کار بسته‌های زیر را نصب کنید.

```
¹ sudo apt update  
² sudo apt install gnupg2 gpa
```

در مرحله نخست باید یک جفت کلید عمومی و کلید محرمانه بسازیم. بدین منظور دستور زیر را تایپ کنید.

```
¹ gpg --full-gen-key
```

گزینه RSA and RSA (default) (1)، را انتخاب کنید و سپس عدد 4096 را به عنوان طول کلید RSA وارد کنید.

با انتخاب گزینه (Expiration Time) = 0 اجازه دهید تا یک جفت کلید بدون زمان انقضا (key does not expire) ایجاد کنیم. در ادامه باید نام و ایمیل خود را وارد کنید. در نهایت نیز کلید ساخته می‌شود و به مخزن کلیدهای شما اضافه می‌شود. کلیدها با پست الکترونیکی که در زمان تولید کلید وارد کردید، تطابق پیدا می‌کند. برای گرفتن خروجی کافی است که دستور زیر را بزنیم:

```
gpg --export --armor adiyanat@iust.ac.ir > publickey.asc
```

این دستور یک نسخه از کلید عمومی را در اختیار شما قرار می‌دهد و شما می‌توانید آن را به هر فردی که می‌خواهد با شما ارتباط امن داشته باشد، بدهید. فرض کنید Alice می‌خواهد پیامی را برای من ارسال کند. کافی است که او ابتدا کلید عمومی را بر روی سیستم عامل خودش وارد کند:

```
gpg --import publickey.asc
```

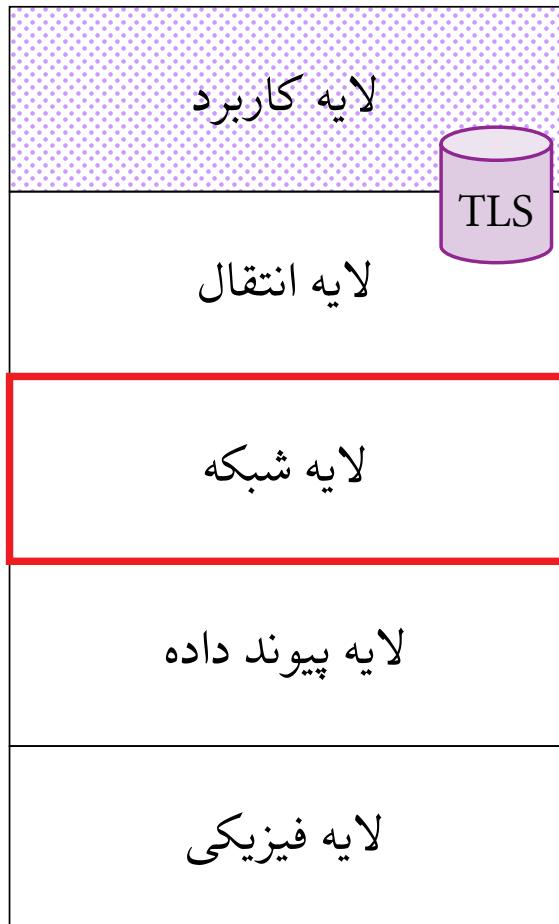
و سپس با استفاده از دستور زیر متن دلخواه خود را که در اینجا محتوای فایل filename.txt را رمز نماید.

```
gpg --encrypt --recipient 'adiyanat@iust.ac.ir' --recipient 'ADiyanat'  
filename.txt
```

خروجی یک فایل با پسوند gpg. خواهد بود. در حقیقت این یک فایل رمز شده است که شما با اطمینان می توانید آن را برای فرد مورد نظر ارسال کنید. برای فرایند رمزگشایی نیز براحتی خواهیم داشت:

```
gpg -d filename.txt.gpg > filename-copy.txt
```

ابزاری به نام gpa و یا kleopatra نیز وجود دارد که این موارد را به صورت گرافیکی می توانید مشاهده و تست کنید.



ما باید بر طبق استاندارد عمل کنیم:

SSL 2.0 (1995), SSL 3.0 (1996), TLS 1.0 (1999), TLS 1.1 (2006),

TLS 1.2 (2008), TLS 1.3 (2018)

در استاندارد TLS 1.3، تصمیم گرفت که پروتکل تبادل کلید مبتنی بر RSA را، با یک پروتکل دیگر جایگزین کند، البته برای امضای دیجیتال هنوز از RSA استفاده می‌شود.

همواره ما با این سوال مواجه هستیم که امنیت را باید در کدام یک از لایه‌های شبکه برآورده سازیم. اگر امنیت را به لایه کاربرد بسپاریم، اتفاقی که رخ می‌دهد این است که به مرور ما هزاران برنامه کاربردی خواهیم داشت با روش‌های مختلف تامین امنیت. این موضوع علاوه بر این‌که کار پیاده‌ساز را دشوار می‌سازد، ریسک وجود خلاهای امنیتی را در برنامه‌های کاربردی بالا می‌برد. یک راه کار مناسب این است که امنیت را به درگاه متصل کننده بین لایه کاربرد و لایه انتقال ببریم. استاندارد TLS که پیشتر ما آن را با عنوان SSL می‌شناختیم، به همین منظور توسط IETF (Internet Engineering Task Force) ارایه شده است.

IETF در استاندارد TLS برای تبادل کلید از RSA استفاده می‌کرد. اما بعدها در TLS 1.3 این الگوریتم را با الگوریتم‌های دیگری جایگزین کرد. گرچه باید توجه داشته باشیم که امضای پیام و تایید آن، هنوز با بهره‌گیری از الگوریتم RSA صورت می‌پذیرد.

یک چالش بزرگ



تا مدت‌ها ستاره بازی TurboTime RoadBlasters بود. همیشه عاشق دیده شدن بود. با ورود Turbo محبویت TurboTime کم شد. حسادتش باعث شد، برای خراب کردن RoadBlasters بازیش را ترک کند. او موفق شد، اما باعث شد هر دو بازی از مدار خارج شود. مدت‌ها مخفی زندگی کرد، تا این‌که بازی SugarRush وارد مدار شد. او بازی را هک کرد و خودش را در نقش شاه نبات وارد بازی کرد. تلاش کرد که کدهای پرنس را پاک کند، اما ناموفق بود و پرنس به یک glitch تبدیل شد. به همین خاطر تمام تلاش خود را بکار بست تا پرنس به عنوان یک مسابقه دهنده از خط پایان عبور نکند، چراکه با این کار بازی دوباره ریست می‌شد.

- ☞ آیا لازم است $m, n = 1$ باشد؟
- ☞ پارامتر e بهتر است فرد باشد یا زوج؟
- ☞ اعداد فرما چیست و نقش آن در تولید پارامترهای RSA را ذکر کنید؟
- ☞ چند نمونه الگوریتم بهینه برای عملیات به توان رسانی در محاسبات پیمانه‌ای ذکر کنید؟
- ☞ یک RSA 1024 و 2048 بیتی معادل یک الگوریتم رمزنگاری بلوکی چند بیتی است؟
- ☞ در مورد نحوه تولید اعداد اول مناسب در RSA به طور مختصر توضیح دهید؟
- ☞ یک لینک مناسب: people.csail.mit.edu/rivest/Rsapaper.pdf ✓

الگوریتم (یعنی - همان)

پروتکل توافق کلید دیفی-هلمن



- ۱) هدف ارایه یک پروتکل برای توافق کلید توسط دیفی-هلمن در سال ۱۹۷۶.
- ۲) تحولی بزرگ و به نوعی آغازگر دوره رمزنگاری نامتقارن
- ۳) بر مبنای مساله لگاریتم گستته (Discrete Logarithm Problem) که یک مساله NP-Hard است:

$$a^x = b \pmod{p}$$

تعریف ۵. عدد g را یک ریشه اولیه عدد n می‌گوییم، اگر به ازای هر i عضو مجموعه کاهش‌یافته مانده‌ها (Reduced Set of Residues) $\{g^{r_1}, g^{r_2}, \dots, g^{r_{\phi(n)}}\}$ ، یک جایگشت کامل از مجموعه کاهش‌یافته مانده‌ها (\mathbb{Z}_n^*) باشد.

قضیه ۶. اثبات می‌شود که فقط اعداد این مجموعه ریشه اولیه دارند. $\{1, 2, 4, p^k, 2 \times p^k\}$

قضیه ۷. تعداد ریشه‌های اولیه عدد n برابر با $\phi(\phi(n))$.

عدد g را یک ریشه اولیه p می‌گوییم، اگر x یک مقدار $\{1, 2, 3, \dots, p - 1\}$ باشد، یک جایگشت از مجموعه $\mathbb{Z}_p^* = \{1, 2, 3, \dots, p - 1\}$ باشد.

مثال ۱۲

عدد ۱۳ یک عدد اول است. پس حتماً ریشه اولیه (Primitive Root) دارد.

$$\{1, 2, 4, p^k, 2 \times p^k\}$$

تعداد چهار ریشه اولیه برای عدد ۱۳ وجود دارد: $\{2, 6, 7, 11\}$. اما چگونه این تعداد را بدست آور迪م:

$$\phi(\phi(n)) \implies \phi(\phi(13)) = \phi(12) = 4$$

مثلاً برای $g = 2$ داریم، که مجموعه زیر

$$\{2^1, 2^2, 2^3, 2^4, 2^5, 2^6, 2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}\} \pmod{13}$$

یک جایگشت از مجموعه کاهاش یافته مانده‌ها است.

$$\mathbb{Z}_{13}^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$$

مساله مساله لگاریتم گسته (Discrete Logarithm Problem)

فرض کنید که پارامترهای h و g را می‌دانیم. x را به گونه‌ای پیدا کنید که رابطه زیر برقرار باشد:

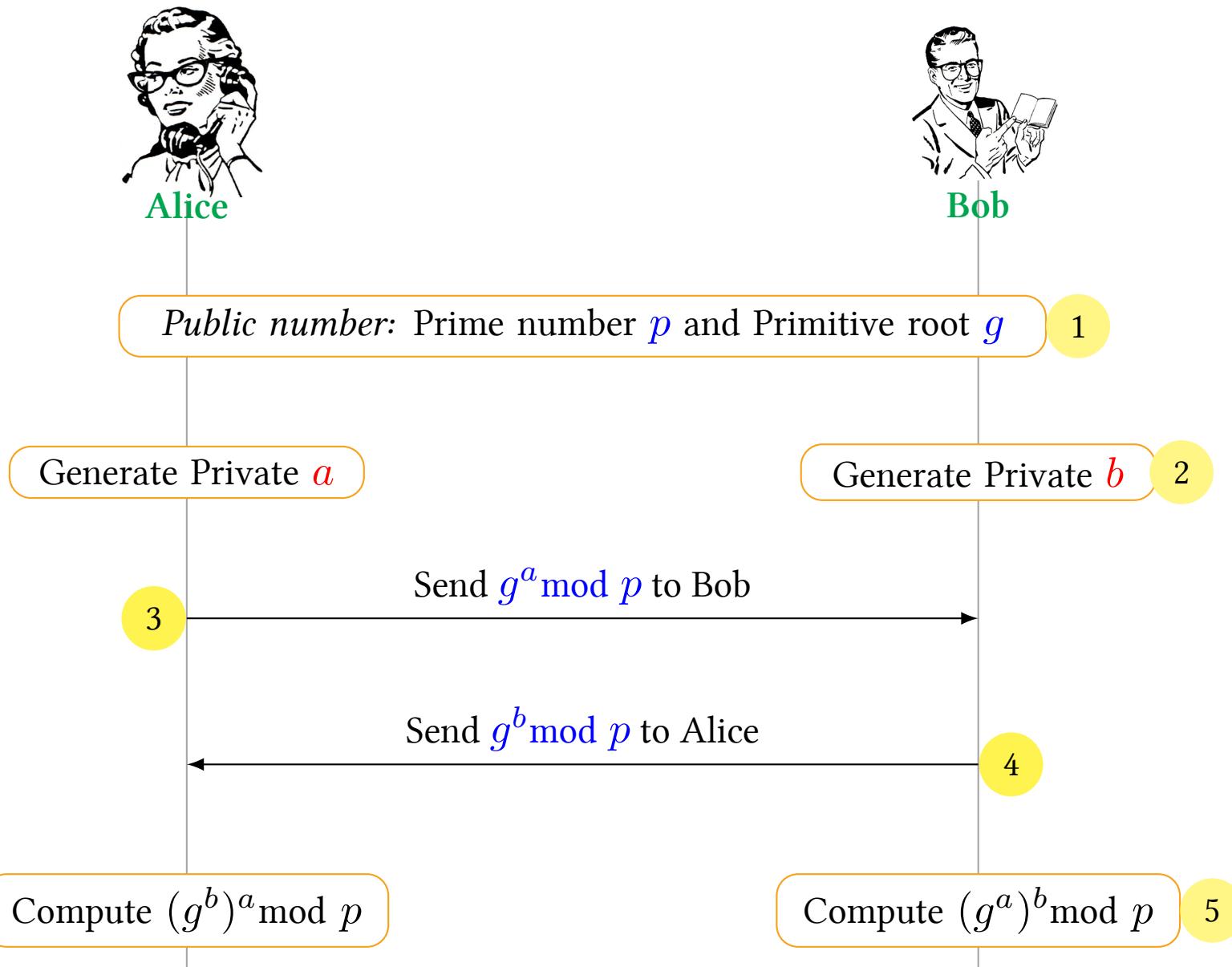
$$h = g^x \pmod{n}$$

به مساله فوق، مساله لگاریتم گسته، گفته می‌شود که هنوز الگوریتم کارایی برای حل آن نداریم.

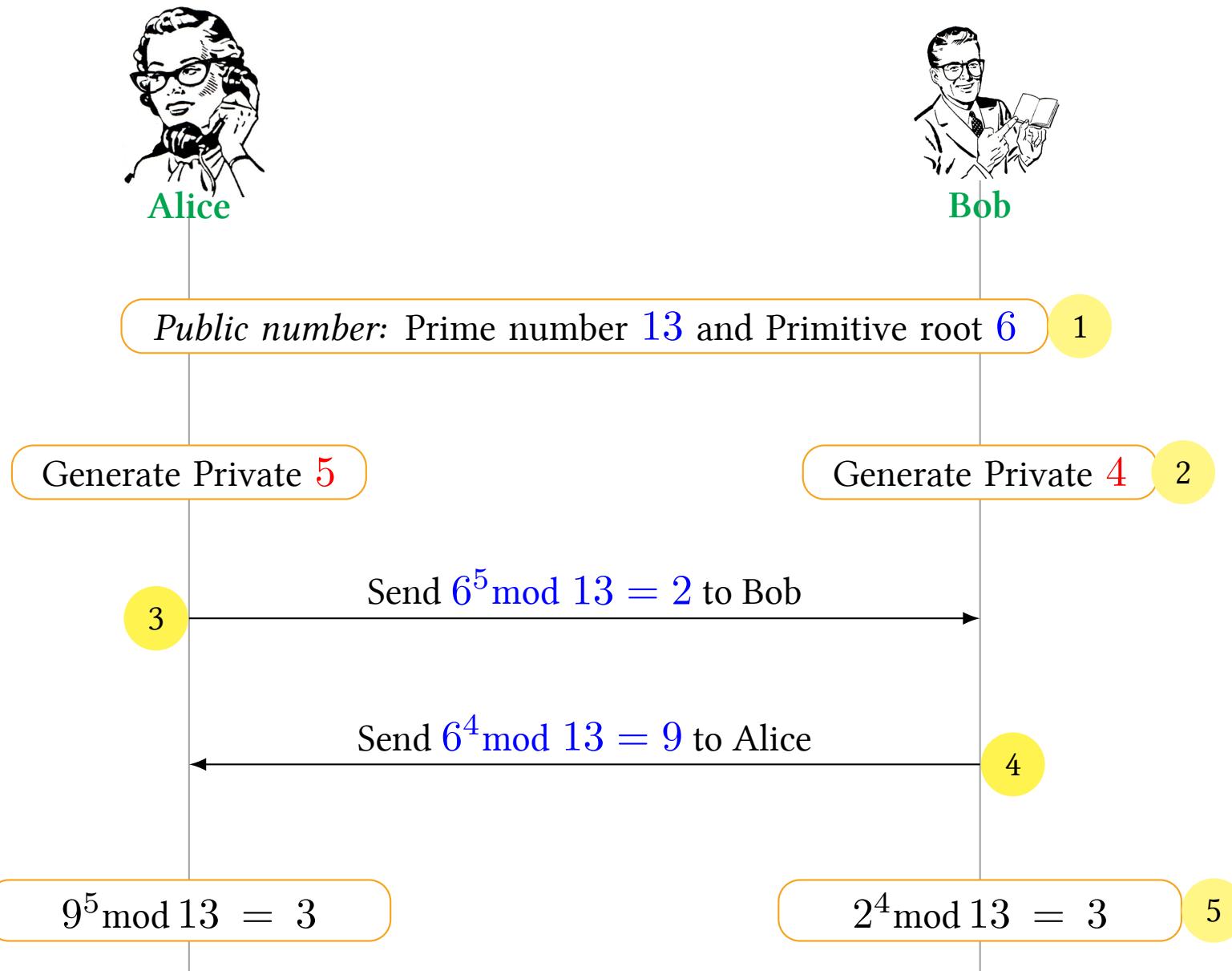
اگر n یک عدد اول به مانند p باشد، آن‌گاه:

- حتماً ریشه اولیه به مانند g دارد.
- اگر g یک ریشه اولیه باشد، آن‌گاه برای هر x عضو \mathbb{Z}_n^* در همان مجموعه \mathbb{Z}_n^* یک به یک خواهد بود.

پروتکل توافق کلید دیفی-هلمن



پروتکل توافق کلید دیفی-هلمن - مثال



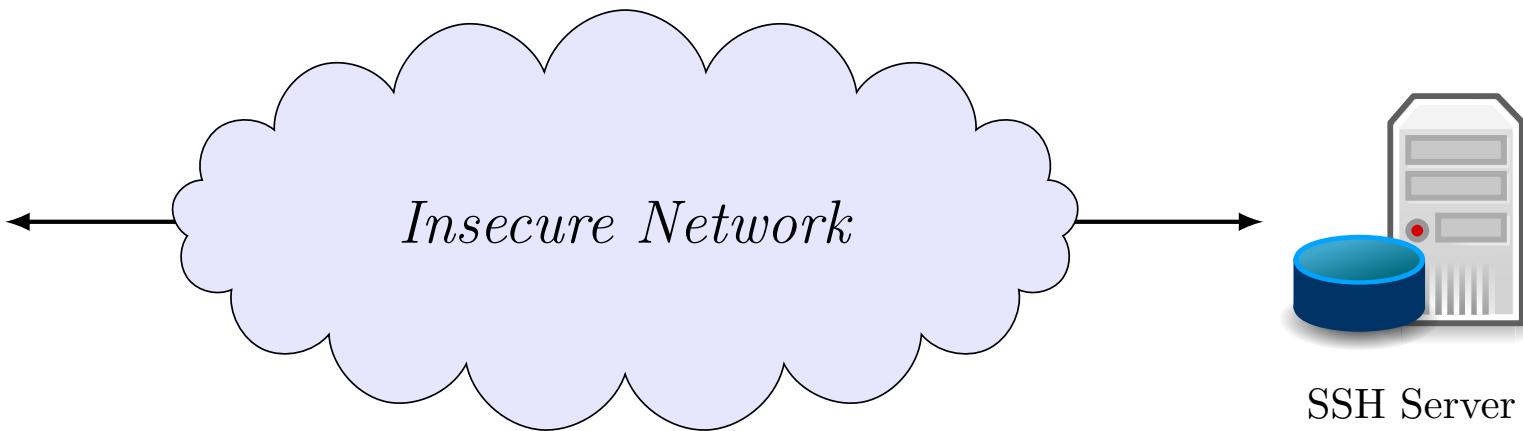
عدد اول p که یک عدد اول بزرگ هست را در نظر بگیرید. در ضمن فرض کنید که g یک ریشه اولیه عدد p باشد. این دو پارامتر عمومی است و همه آن را می‌دانند. هر دو سمت Alice و Bob هر کدام یک پارامتر به صورت تصادفی تولید می‌کنند. ما این پارامترها را با a و b نمایش می‌دهیم. یعنی Alice پارامتر محربانه a را برگزیده است و Bob نیز پارامتر b را. این دو مقدار باید عضو مجموعه $\{1, \dots, p-1\}$ باشند.

در ادامه، Alice پارامتر g^a را حساب می‌کند و برای Bob ارسال می‌کند. از سوی دیگر، Bob نیز پارامتر g^b را با دریافت g و p از سمت Alice محاسبه می‌کند و برای او ارسال می‌کند. در نهایت کلید مشترک به صورت g^{ab} حاصل خواهد شد. با کمی دقت می‌توان دریافت که هر دو طرف این الگوریتم، می‌توانند به این کلید مشترک دست پیدا کنند. اما Eve تنها به پارامتر g^a و g^b دسترسی دارد. او برای این که بتواند به کلید مشترک دست پیدا کند، باید مساله مساله لگاریتم گستته را حل نماید، که می‌دانیم این کار بسیار دشوار است.

پروتکل توافق کلید دیفی-هلمن - کاربرد



SSH Client



SSH Server

☞ در سال ۱۹۹۵ توسط یک فنلاندی به نام **Tatu Ylönen** ارایه شد، برای امن کردن پروتکل **Telnet**

☞ ما در SSH سه خدمت زیر را داریم:

- احراز اصالت (Authentication)

- رمزگذاری (Encryption)

- یکپارچگی (Integrity)

بعد از این که ارتباط TCP بین مشتری (Client) و خدمت‌گزار (Server) برقرار شد، مشتری اولین گام در پروتکل SSH را که به آن اصطلاحاً رویه شناسایی (Identification) گفته می‌شود، را انجام می‌دهد. در این پیام که نمایی از آن در کد زیر قرار داده شده است، نسخه پروتکل SSH و همچنین نسخه نرم‌افزار مورد استفاده، از سوی مشتری ارسال می‌گردد.

```
1 Protocol: SSH-2.0-libssh_0.9.3  
2 [Direction: client-to-server]
```

در پاسخ نیز خدمت‌گزار اطلاعات سمت خودش را برای مشتری ارسال می‌کند.

```
1 Protocol: SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5  
2 [Direction: server-to-client]
```

در گام بعدی مشتری و خدمت‌گزار به سراغ مذاکره بر روی الگوریتم‌های تامین امنیت می‌روند. مشتری در ابتدا لیست الگوریتم‌های پیشنهادی خود را به ترتیب اولویت برای خدمت‌گزار ارسال می‌کند. این الگوریتم‌ها شامل

الگوریتم تبادل کلید، رمزگذاری و فشرده‌سازی برای ارتباط بین مشتری به خدمت‌گزار و بالعکس است. به این مرحله اصطلاحا Key Exchange Init گفته می‌شود.

```
1 SSH Version 2 (encryption:aes256-gcm@openssh.com mac:<implicit> compression
2   :none)
3   Packet Length: 940
4   Padding Length: 4
5   Key Exchange (method:curve25519-sha256)
6     Message Code: Key Exchange Init (20)
7     Algorithms
8       Cookie: f6ef99c585b95624a1a8c6483b554b30
9       kex_algorithms length: 266
10      kex_algorithms string [truncated]: curve25519-sha256,curve25519-
11        -sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-
12        -sha2-nistp521,diffie-hellman-group18-sha512,diffie-hellman-
13        -group16-sha512,diffie-hellman-group-exchange-sha256,di
14      server_host_key_algorithms length: 113
15      server_host_key_algorithms string: ssh-ed25519,ecdsa-sha2-
16        -nistp521,ecdsa-sha2-nistp384,ecdsa-sha2-nistp256,rsa-sha2-
17        -512,rsa-sha2-256,ssh-rsa,ssh-dss
18      encryption_algorithms_client_to_server length: 120
```

```
13    encryption_algorithms_client_to_server string: aes256-
14        gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,
15        aes128-ctr,aes256-cbc,aes192-cbc,aes128-cbc,3des-cbc
16    encryption_algorithms_server_to_client length: 120
17    encryption_algorithms_server_to_client string: aes256-
18        gcm@openssh.com,aes128-gcm@openssh.com,aes256-ctr,aes192-ctr,
19        aes128-ctr,aes256-cbc,aes192-cbc,aes128-cbc,3des-cbc
20    mac_algorithms_client_to_server length: 123
21    mac_algorithms_client_to_server string: hmac-sha2-256-
22        etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha1-
23        etm@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-sha1
24    mac_algorithms_server_to_client length: 123
25    mac_algorithms_server_to_client string: hmac-sha2-256-
26        etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha1-
27        etm@openssh.com,hmac-sha2-256,hmac-sha2-512,hmac-sha1
28    compression_algorithms_client_to_server length: 4
29    compression_algorithms_client_to_server string: none
30    compression_algorithms_server_to_client length: 4
31    compression_algorithms_server_to_client string: none
32    languages_client_to_server length: 0
33    languages_client_to_server string:
34    languages_server_to_client length: 0
```

```
۲۷ languages_server_to_client string:  
۲۸ First KEX Packet Follows: 0  
۲۹ Reserved: 00000000  
۳۰ [hasshAlgorithms [truncated]: curve25519-sha256,curve25519-  
    sha256@libssh.org,ecdh-sha2-nistp256,ecdh-sha2-nistp384,ecdh-  
    sha2-nistp521,diffie-hellman-group18-sha512,diffie-hellman-  
    group16-sha512,diffie-hellman-group-exchange-sha256,diffie-h]  
۳۱ [hassh: ba19a58a192de4fe89e3f7e2f9917fc9]  
۳۲ Padding String: 00000000  
۳۳ [Direction: client-to-server]
```

در گام بعدی نیز خدمت‌گزار الگوریتم‌های خودش را برای مشتری ارسال می‌کند (مرحله Init). Key Exchange Init همان‌طور که ملاحظه می‌شود، این دو بر روی الگوریتم‌های امضای دیجیتال، تابع چکیده‌ساز، رمزگذاری و فشرده‌سازی، می‌بایست توافق کنند. اگر مشتری و خدمت‌گزار به یک الگوریتم مشترک رسیدند، که کار را ادامه می‌دهند. در غیراین صورت خدمت‌گزار اولین الگوریتم پیشنهادی مشتری را برمی‌گزیند. اما اگر خدمت‌گزار این الگوریتم را پشتیبانی نمی‌کرد، به ناچار اتصال باید قطع شود.

در گام بعدی، دونهاد درگیر وارد مرحله تبادل کلید می‌شوند. همان‌طور که می‌دانید، تقریباً در اکثر پروتکل‌های امنیتی، این ایده وجود دارد که برای حل مشکل تبادل کلید از الگوریتم کلید نامتقارن استفاده شود، اما مزگذاری بر عهده الگوریتم کلید متقارن قرار داده شود. در پیام قبل مشاهده کردید، قرار شد دو سوی درگیر به منظور ایجاد کلید جلسه، از پروتکل Elliptic Curve Diffie-Hellman، استفاده کنند. این پروتکل، یک نسخه قدرتمندتر و بهینه از الگوریتم سنتی Diffie-Hellman، با بهره‌گیری از خم‌های بیضوی (Elliptic Curve) است. البته به دلیل این‌که خارج از حوزه این نوشتار است، از بیان توضیح آن خودداری می‌کنیم.

مشتری یک جفت کلید عمومی و کلید محرمانه را تولید می‌کند. سپس کلید عمومی خود را برای خدمت‌گزار در قالب پیامی به صورت زیر برای خدمت‌گزار ارسال می‌کند.

```
1 SSH Version 2 (encryption:aes256-gcm@openssh.com mac:<implicit> compression
2 :none)
3   Packet Length: 44
4   Padding Length: 6
5   Key Exchange (method:curve25519-sha256)
```

```
5   Message Code: Elliptic Curve Diffie-Hellman Key Exchange Init (30)
6   ECDH clients ephemeral public key length: 32
7   ECDH clients ephemeral public key (Q_C): 6387684984
8     b6eb05f043ea92750a35d9b93943154657b464ebf63010f74b087a
9   Padding String: 000000000000
[DIRECTION: client-to-server]
```

با رسیدن این پیام به دست خدمت‌گزار، او نیز جفت کلید عمومی و کلید محرمانه خود را تولید می‌کند. سپس او با استفاده از کلیدهای تولیدی خودش و همچنین کلید عمومی مشتری، کلید جلسه مشترک را تولید می‌کند. او در ادامه می‌بایست یک چکیده را تولید کند. اگر یادتان باشد، مشتری و خدمت‌گزار در ابتدای امر بر روی یک تابع چکیده‌ساز، به تفاهم رسیدند. خدمت‌گزار از همان تابع استفاده می‌کند، و با استفاده از ورودی‌های زیر مقدار چکیده پیام را درست می‌کند:

- Client Identification Id: SSH-2.0-libssh_0.9.3
- Server Identification Id: SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.5

- Client Key Exchange Init
 - Server Key Exchange Init
 - Server Public Key for signature (Host Key)
 - Client Public Key for ECDH
 - Server Public Key for ECDH
 - Shared Session Key

بعد از این که این چکیده تولید شد، خدمت‌گزار آن را با کلید عمومی خودش امضا می‌کند. وقتی که این کلید عمومی را با کلید عمومی تولید شده برای ECDH، اشتباه نگیرید. اجازه بدهید به این کلید جدید بگوییم. در ادامه، خدمت‌گزار پیام پاسخی برای مشتری، به صورت زیر ارسال می‌کند.

```
| SSH Version 2 (encryption: aes256-gcm@openssh.com mac:<implicit> compression  
| :none)  
|  
|     Packet Length: 188
```

```
1 Padding Length: 8
2 Key Exchange (method:curve25519-sha256)
3   Message Code: Elliptic Curve Diffie-Hellman Key Exchange Reply (31)
4   KEX host key (type: ssh-ed25519)
5   ECDH servers ephemeral public key length: 32
6   ECDH servers ephemeral public key (Q_S):
7     bdc5efd64f7a70102cf175c8c57db2c5234724fb97ec02c65a5679acfe15a72f
8   KEX H signature length: 83
9   KEX H signature: 0000000
10  ...b7373682d6564323535313900000040278278764410cdb36b42999c25d69a6fb9
11 Padding String: 0000000000000000
12 SSH Version 2 (encryption:aes256-gcm@openssh.com mac:<implicit> compression
13 :none)
14   Packet Length: 12
15   Padding Length: 10
16   Key Exchange (method:curve25519-sha256)
17     Message Code: New Keys (21)
18   Padding String: 00000000000000000000000000000000
19 SSH Version 2 (encryption:aes256-gcm@openssh.com mac:<implicit> compression
20 :none)
21   Packet Length: 208
22   Encrypted Packet: 7
```

...d73b0988439536c9300ffe6f767f0eed637c54c2691c97e79f4e01535a59289

۲۱ MAC: 9f94c46da4eadaa79a455702ec090e7d

۲۲ [Direction: server-to-client]

همان‌طور که در این پیام مشاهده می‌کنید، خدمت‌گزار علاوه بر چکیده، کلید عمومی خودش برای اجرای الگوریتم ECDH و کلید عمومی برای امضای دیجیتال (Host Key) را نیز قرار داده است. به محض این که این پیام به دست مشتری رسید، او با استفاده از کلید عمومی خدمت‌گزار برای ECDH، سعی می‌کند کلید جلسه را تولید بکند. اکنون زمانی است که مشتری می‌تواند چکیده را تولید کند و مطمئن شود که در مراحل قبلی هیچ‌گونه تداخلی ایجاد نشده و یکپارچگی همه پیام‌ها حفظ شده است. همچنین مشتری باید کلید عمومی برای امضای دیجیتال (Host Key) خدمت‌گزار را نیز تایید کند. ممکن است از قبل پایگاه داده داشته باشد، ممکن است از کanal دیگری تایید کند. در هر صورت، در صورتی که هیچ‌کدام از این موارد وجود نداشت، این مورد به عهده خود مشتری است که اتصال را بپذیرد یا خیر؟! نمونه‌ای از هشدار در سمت مشتری در سیستم عامل

Linux به صورت زیر است:

- ۱ The authenticity of host 195.144.107.198 cant be established.
- ۲ ECDSA key fingerprint is SHA256:0zvpQxRUzSfV9F/ECMXbQ7B7zbK0FCBUno65c.
- ۳ Are you sure you want to **continue** connecting (yes/no)?

با تایید Host Key، مشتری می‌تواند امضا دیجیتال خدمت‌گزار را نیز تایید کند. در ادامه مشتری و خدمت‌گزار با کلید جلسه بدست آمده، شش کلید دیگر (هر کدام سه کلید) تولید می‌کنند. دو کلید برای رمزگذاری، دو بردار اولیه برای الگوریتم رمزگذاری، دو کلید نیز برای یکپارچگی مورد استفاده قرار خواهد گرفت. در نهایت نیز دو سوی درگیر، پیام‌هایی با عنوان New Keys برای یکدیگر ارسال می‌کنند، تا همدیگر را مطمئن سازند که پروتکل تبادل کلید با موفقیت به پایان رسیده است. پس از این مشتری می‌تواند درخواست ارایه خدمت خود را ارسال کند و ادامه روند Login، را انجام دهد (وارد کردن نام کاربری و رمز عبور).

امنیت پروتکل توافق کلید دیفری-هلمن



al-Hafiz



Cohen



Nasser

Generate Private a

Generate a', b'

Generate Private b

Send $g^a \text{ mod } p$ to Bob

Send $g^{a'} \text{ mod } p$ to Bob

Send $g^b \text{ mod } p$ to Alice

Send $g^{b'} \text{ mod } p$ to Alice

$$K_1 = (g^{b'})^a \text{ mod } p$$

$$K_1 = (g^a)^{b'}, K_2 = (g^b)^{a'}$$

$$K_2 = (g^{a'})^b \text{ mod } p$$

الیاهو (الی) بن شائول کوهن، یکی از بزرگترین و معروف‌ترین جاسوس‌های موساد بود که تاکنون فیلم‌ها و نوشته‌های زیادی در مورد او منتشر شده. در یک خانواده ارتدوکس و صهیونیست در ۱۹۲۵ متولد شد. خانواده او به مصر رفتند و در ۱۹۵۱ به دلیل فعالیت‌هایی که برای اسرائیل داشت، بازداشت شد، اما به دلیل عدم وجود مدرک کافی، آزاد شد. در نهایت در ۱۹۵۶ مصر را به قصد اسرائیل ترک کرد. در اسرائیل، موساد تصمیم به استخدام کوهن گرفت. مأموریت او نفوذ به دولت سوریه بود. هویتی جعلی به عنوان یک بازرگان سوری که از آرژانتین به کشور خود بازمی‌گردد، برای او تهییه شد. در ژانویه ۱۹۶۲، با نام مستعار کامل امین ثابت، سوار بر کشتی به بیروت رفت. در کشتی، با مجید شیخ الارض دیدار کرد. او که به عنوان خبرچین برای آمریکایی‌ها کار می‌کرد، در ازای ۴۰۰ لیره با بکارگیری دوستانی که داشت، تجهیزات جاسوسی کوهن را از مرز رد کرد و بدین‌سان کوهن به سلامت وارد دمشق شد.

کوهن موفق شد اعتماد افراد زیادی در ارتش و دولت سوریه را به دست آورد. از بلندی‌های جولان بازدید کرد و موفق شد اطلاعات نظامی سوریه در آن منطقه را به اسرائیل منتقل کند. کوهن وانمود کرد که برای کمک به

سر بازانی که زیر آفتاب کشیک می‌دادند قصد دارد درختانی را در بالای سر آن‌ها بکارد. بعداً این درخت‌ها به عنوان هدف برای ارتش رژیم صهیونیستی مورد استفاده قرار گرفت. بخش جالب ماجرا این بود که کوهن دستور گرفته بود که تا می‌تواند علیه اسرائیل بگوید و بنویسد. مثلاً در مجلس ملی سوریه فریاد می‌زد و اسرائیل را به هزار کار کرده و نکرده متهم می‌کرد.

کار بدان جا پیش‌رفت که بعد از این که امین الحافظ رئیس جمهور شد، کوهن برای پست وزیر دفاع در نظر گرفته شد. او موفق شد از برنامه مخفی ارتش سوریه برای ایجاد پدافند چندلایه پرده بردارد. رژیم صهیونیستی همواره کوهن را یکی از مهم‌ترین جاسوس‌های خود معرفی کرده. ایوی شکول، نخست وزیر پیشین اسرائیل، اعلام کرده بود اطلاعاتی که کوهن فراهم کرده بود منجر شد تا اسرائیل در جنگ شش‌روزه ۱۹۶۷ پیروز شود. روایت‌های زیادی از چگونگی فاش شدن هویت او و بازداشت او منتشر شده است. به گفته رئیس اطلاعات سوریه، سوری‌ها وقتی به کوهن شک کردند که از تعاملات وی با «مردی با روابط مشکوک» مطلع شدند. اسرائیل همواره کوهن را به بی توجهی و بی احتیاطی متهم می‌کرد. اما برخی دیگر دلیل بازداشت کوهن را توطئه علیه

او در داخل خود موساد بوده باشد یا شاید جاسوسی در موساد بوده که به نفع طرف های مشخصی فعالیت کرده و او را به سوری ها لو داده باشد.

اسرائیل یک برنامه جهانی برای حمایت از اوراه اندازی کرد تا جلوی اعدام اورا بگیرد. با اینکه خیلی از شخصیت های بین المللی از جمله پاپ پل ششم و دولت های شوروی، فرانسه، بلژیک و کانادا از دولت سوریه درخواست کردند حکم اعدام را لغو کند، او در ساعت ۳ بامداد ۱۸ مه ۱۹۶۵ در میدان «المرجه» دمشق در ملاء عام اعدام شد. برادر کو亨 که وی نیز یک جاسوس موساد بود، ادعای دارد که در زمان اعدام، کو亨 نفر سوم برای جایگزینی نخست وزیر سوریه بود.



تعريف ۶. محربانگی پس رو (Backward Secrecy)

اگر کاربر به سامانه وارد شد، داده های گذشته لو نزود.

تعريف ۷. محربانگی پیش رو (Forward Secrecy)

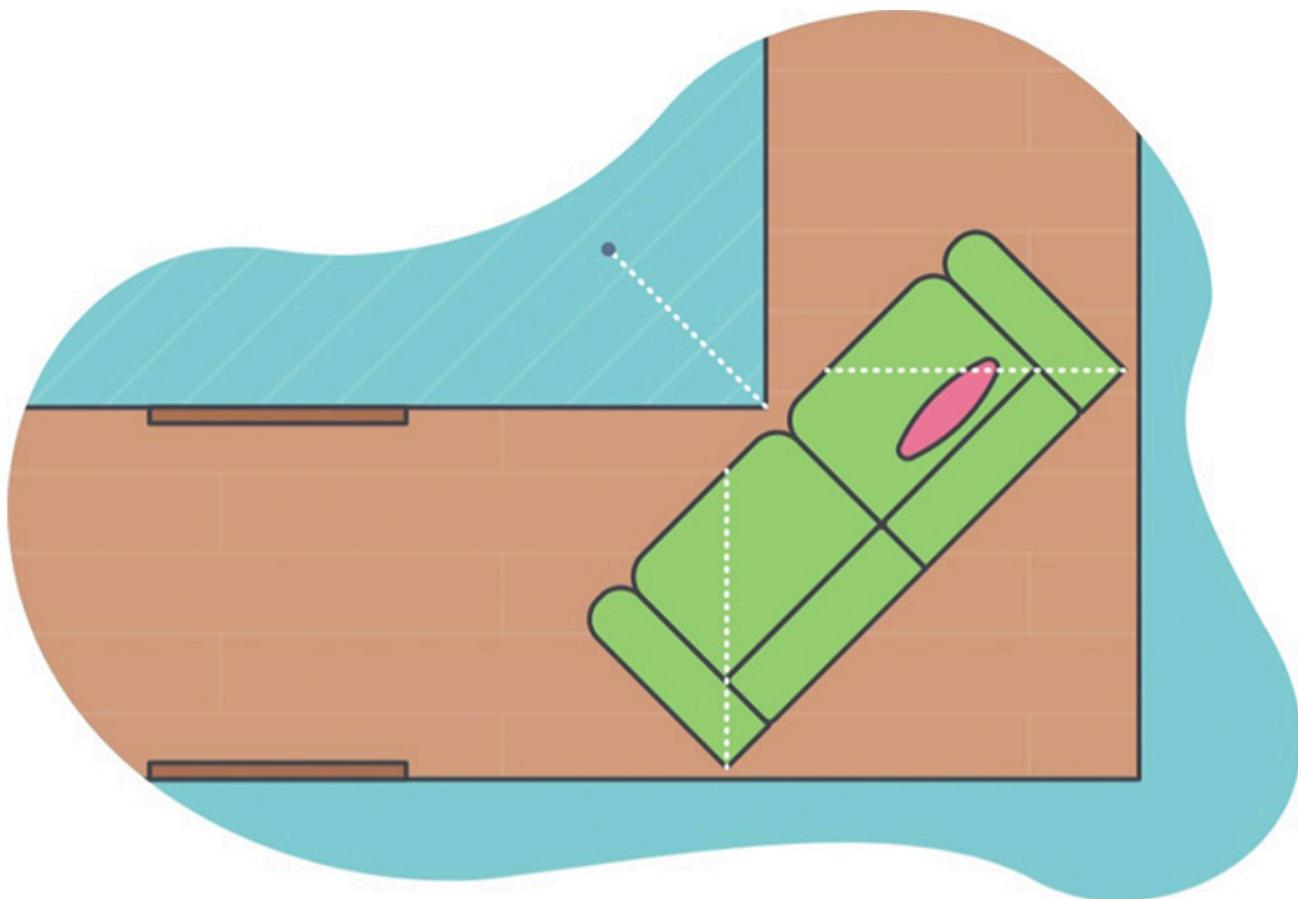
زمانی کاربر سامانه را ترک کرد، داده های در آینده لو نزود.

تعريف ۸. محربانگی پیش رو یا محربانگی پیش رو کامل (Perfect Forward Secrecy) - دیدگاه دیگر

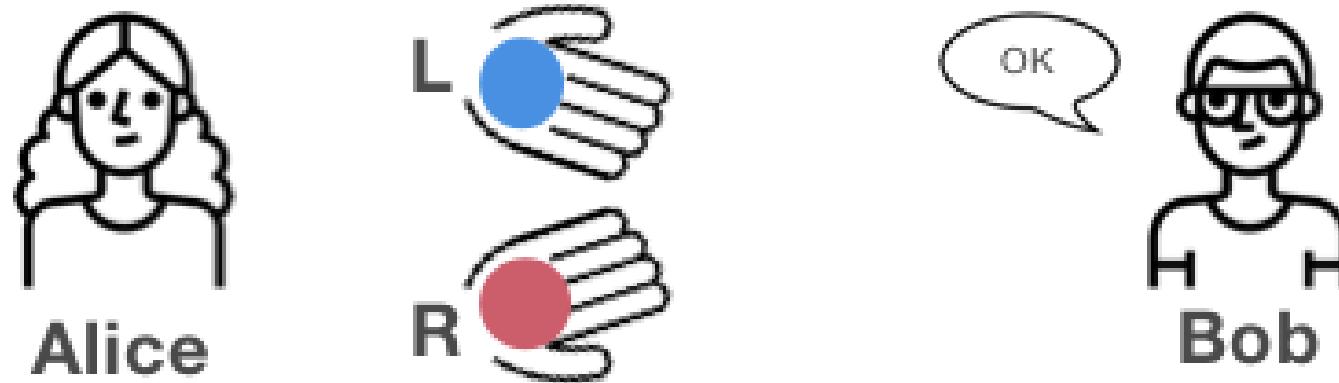
اگر کلید Long Term در زمان T افشا شود، کلیدهای نشستی که قبل از زمان T تبادل شده اند، امن بمانند.

من مساله ثابت مبل (۱۹۶۶م) را حل کردم !!!

در یک کنج L شکل (زاویه ۹۰ درجه) با عرض یک متر، در حالت دو بعدی، **بزرگترین سطحی** را که می توانیم بدون برخورد به دیوارها منتقل کنیم چقدر است؟ $2.2195 \leq A \leq 2.37$.

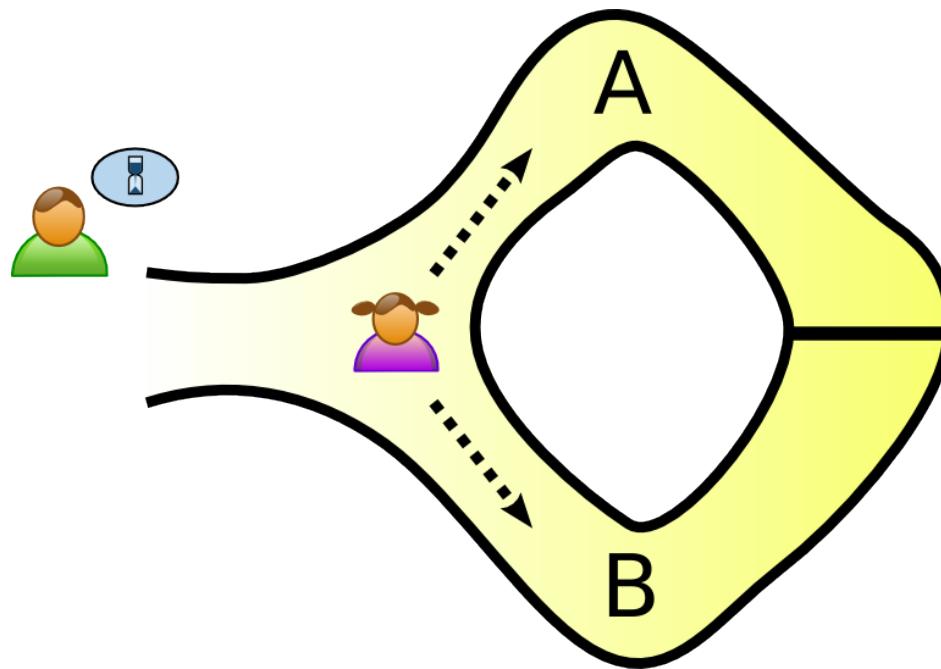


یک بازی



چشمان Alice بسته است و Bob به عنوان فرد اثبات‌کننده می‌خواهد به Alice اثبات کند که رنگ دو گوی با هم متفاوت است بدون این که رنگ آن‌ها را بگوید! یعنی Bob که از رنگ گوی‌ها با خبر است، نباید اطلاعاتی در مورد رنگ گوی‌ها به Alice بدهد. دقیق کنید که Bob ممکن است آدم راستگویی نباشد.

یک بازی دیگر - غار علی بابا



رمز درب قرار گرفته شده در وسط غار را می‌داند، او به عنوان اثبات‌کننده، می‌خواهد به Bob به عنوان Alice فرد تاییدکننده، این اطمینان را بدهد که رمز عبور را دارد. اما نمی‌خواهد رمز عبور را در اختیار Bob قرار دهد. دقت کنید که Alice ممکن است آدم راستگویی نباشد، و به دروغ بگوید که رمز درب را دارد.

اثبات دانایی صفر - ویژگی

هدف



(اثبات‌کننده) می‌خواهد به Bob (وارسی‌کننده) ثابت کند پارامتر M را می‌داند، بدون این‌که این پارامتر را برای Bob افشا کند؟!

ویژگی‌ها:

- ✓ کامل بودن: اگر هر دو کاربر قوانین را رعایت کنند، تاییدکننده بدون هیچ کمکی در اثبات یک عبارت درست، متقادع شود.
- ✓ صحت: در صورت نادرست بودن عبارت، تاییدکننده در هیچ حالت قانع نشود.
- ✓ دانش صفر: در همه موارد تاییدکننده اطلاعات بیشتری دریافت نکند.

هدف



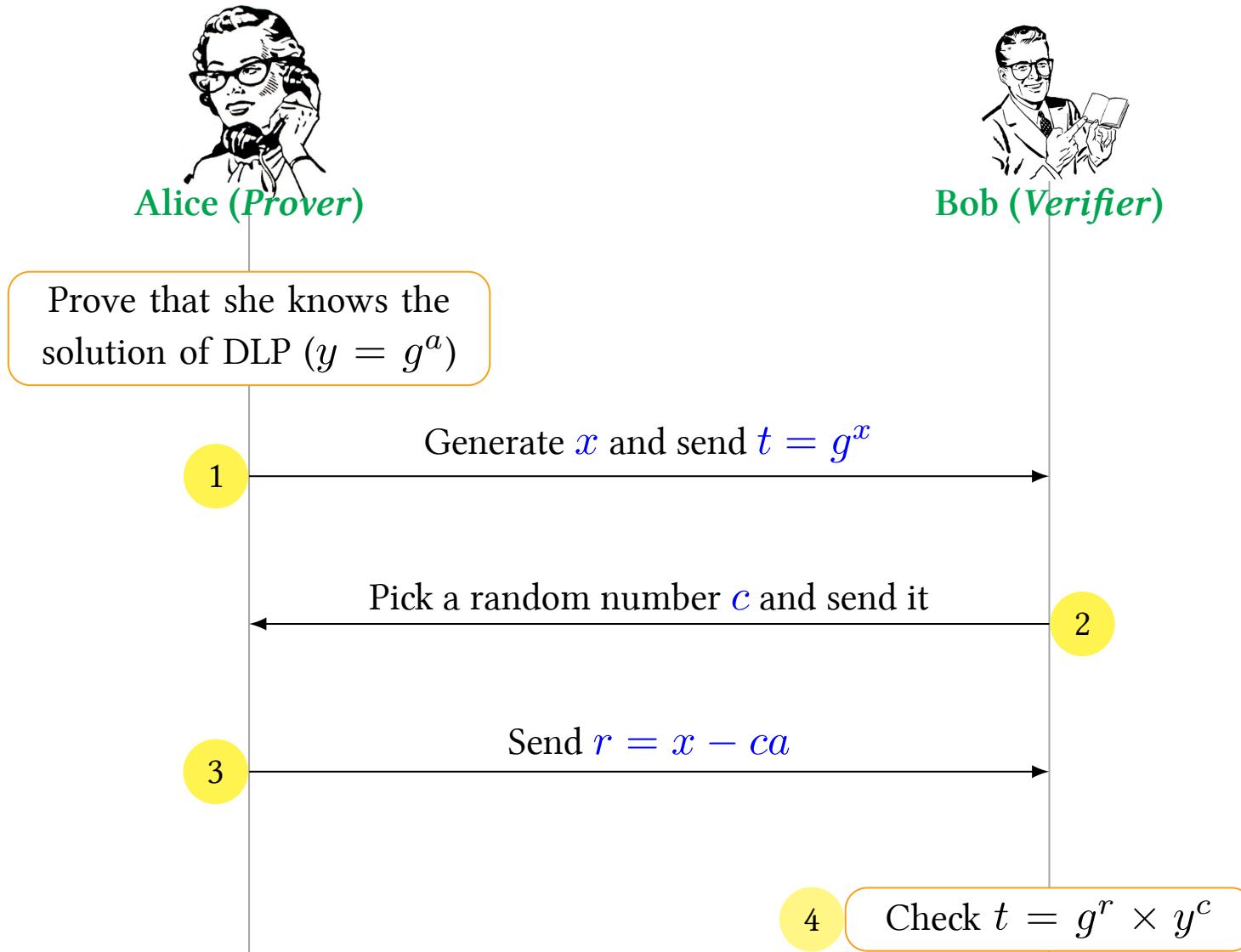
(اثبات‌کننده) می‌خواهد به Bob (وارسی‌کننده) ثابت کند پارامتر M را می‌داند، بدون این‌که این پارامتر را برای Bob افشا کند؟!

☞ در حال کلی دو روش برای این کار وجود دارد:

- روش تعاملی (Interactive)

- روش غیر تعاملی (Non-Interactive)

اثبات دانایی صفر - مثال



Alice می‌خواهد اثبات کند که پارامتری به نام a را دارد. a در حقیقت پاسخ یک مساله لگاریتم گسته به صورت $y = g^a \text{ mod } p$ است. اما در روند این اثبات نمی‌خواهد اطلاعاتی از a را در اختیار Bob به عنوان وارسی‌کننده (Verifier) بگذارد. در گام نخست، Alice یک عدد تصادفی x را تولید کرده و پارامتر

$$t = g^x \text{ mod } p,$$

را برای Bob ارسال می‌کند. بدیهی است که Bob با داشتن پارامترهای عمومی نمی‌تواند این مساله را حل کند، چراکه حل این مساله مستلزم حل مساله لگاریتم گسته است.

در گام بعدی، Bob عدد تصادفی c را تولید کرده و برای Alice ارسال می‌کند. Alice نیز مقدار $r = x - ca$ را محاسبه و مجدداً برای Bob می‌فرستد. اکنون Bob مقدار زیر را محاسبه می‌کند. اگر حاصل عبارت زیر برابر با t شد، Bob به این نتیجه می‌رسد که Alice واقعاً پارامتر a را می‌داند.

$$g^r \times y^c = g^{(x-ca)} \times (g^a)^c = g^x = t \text{ mod } p$$

تمرین



عنکبوت را در این شکل پیدا کنید؟

فرض کنید در مساله صفحه قبل، Alice از Bob می‌خواهد که یک عنکبوت را در شکل یاد شده پیدا کند. Bob این کار را با موفقیت انجام می‌دهد. او می‌خواهد به Alice اثبات کند که توانسته است که عنکبوت را پیدا کند، اما نمی‌خواهد از مکان عنکبوت اطلاعاتی را به Alice بدهد. یک راه کار (Zero-knowledge Proof) برای حل این مساله پیشنهاد دهید؟

یک روش اثبات دانایی صفر به صورت غیرتعاملی (Non-Interactive) که در زنجیره بلوکی (Blockchain) استفاده می‌شود را تشریح کنید؟

- [1] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. Discrete Mathematics and Its Applications, CRC Press, 1996.

فهرست اختصارات

A

AES Advanced Encryption Standard

ARPANET Advanced Research Projects Agency Network

D

DES Data Encryption Standard

E

ECDH Elliptic Curve Diffie Hellman

G

GCHQ Government Communications Headquarters

I

IETF Internet Engineering Task Force

N

NSA Non-Standalone

P

PGP Pretty Good Privacy

R

RSA Rivest Shamir Adleman

S

SSH Secure Shell

SSL Secure Sockets Layer

T

TCP Transmission Control Protocol

TLS Transport Layer Security

Z

ZKP Zero-knowledge Proof

واژه‌نامه انگلیسی به فارسی

B

Backward Secrecy محرمانگی پس‌رو احراز اصالت

Blockchain زنجیره بلوکی برنامه کاربردی

Application Layer لایه کاربرد

Ciphering رمزگذاری الگوریتم کلید نامتقارن

Ciphertext متن رمز Algorithm حمله‌گر

Attacker حمله‌گر

Digital Signature	امضای دیجیتال	مشتری
Discrete Logarithm Problem	مساله لگاریتم گسته	مجموعه کامل مانده‌ها.
E	Compression	فشرده‌سازی
Expiration Time	زمان انقضا	محرمانگی
Elliptic Curve	خم بیضوی	اتصال
Encryption	رمزگذاری	رمزنگاری
End to End	انتها به انتها	D
Entity	نهاد	پایگاه داده
	Database	رمزگشایی
	Decryption	

Integrity یکپارچگی F

محرمانگی پیش رو Forward Secrecy

K

Key Agreement توافق کلید H

Key Exchange تبادل کلید چکیده Hash

Key Establishment برقراری کلید Hash Function تابع چکیده ساز

Key Transport تبادل کلید

I

M

Identification شناسایی

Message پیام

P

Password	رمز عبور	N
Perfect Forward Secrecy	محرمانگی پیش رو کامل .	Number Theory
Performance	کارایی	O
Public Key	کلید عمومی	
Plaintext	متن اصلی	تابع یک طرفه
Privacy	حریم خصوصی	متن باز
Private Key	کلید محرمانه	سیستم عامل
Primitive Root	ریشه اولیه	Operating System
Procedure	رویه	

اثبات‌کننده Service خدمت Prover

کلید جلسه Session Key

الگوریتم کلید متقارن Symmetric Key Algorithm . R

مجموعه کاہش‌یافته مانده‌ها Reduced Set of T Residues

افزونگی Transport Layer لایه انتقال Redundancy

U S

بخش‌بندی Upload بارگذاری Segmentation

خدمت‌گزار Username Server نام کاربری

V

وارسی کننده Verifier

W

هشدار Warning

Z

اثبات دانایی صفر Zero Knowledge Proof

واژه‌نامه فارسی به انگلیسی

۱

Algorithm

اتصال Digital Signature امضای دیجیتال Connection

اثبات دانایی صفر End to End انتها به انتها Zero Knowledge Proof

اثبات‌کننده Prover

احراز اصالت Authentication

ب

افزونگی Redundancy

الگوریتم کلید متقارن Symmetric Key Algorithm

الگوریتم کلید نامتقارن Asymmetric Key

بارگذاری Upload

بخش‌بندی Segmentation

برقراری کلید Key Transport تبادل کلید Key Establishment

برنامه کاربردی Key Agreement توافق کلید Application

ج

پایگاه داده Hash Database چکیده

پیام Message

ح

ت خصوصی Privacy حریم

تابع چکیدهساز Attacker حمله‌گر Hash Function

تابع یک‌طرفه One-way Function

خ

Cryptography رمزنگاری

Procedure رویه Service

Primitive Root ریشه اولیه Server

خدمت گزار خم بیضوی

ز

Expiration Time زمان انقضا

Blockchain زنجیره بلوکی Password

رمزگذاری Ciphering

رمزگذاری Encryption

رمزگشایی Decryption

س

System عامل Performance کارایی Operating System

Session Key کلید جلسه

Public Key کلید عمومی

Private Key کلید محرمانه

ش

Identification شناسایی

ل

Transport Layer لایه انتقال

Compression فشرده سازی

Application Layer لایه کاربرد

ف

Discrete Logarithm مساله لگاریتم گسته

متن اصلی Problem Plaintext

Client مشتری Ciphertext متن رمز

متن باز Open Source

مجموعه کامل ماندها Complete Set Of Residues

ن Reduced Set of مجموعه کاهش یافته ماندها

Username نام کاربری Residues

Number Theory نظریه اعداد Confidentiality محرمانگی

Entity نهاد Backward Secrecy محرمانگی پس رو

محرمانگی پیش رو Forward Secrecy

محرمانگی پیش رو کامل Perfect Forward Secrecy

و

وارسی کننده Verifier

ه

هشدار Warning

ی

یکپارچگی Integrity