دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر

آزمایشگاه امنیت داده و شبکه
http://dnsl.ce.sharif.edu

رمزنگاری نامتقارن (کلید عمومی)

محمد صادق دوستی

فهرست مطالب

مبانی رمزنگاری کلید عمومی

- دیفی و هلمن اولین راه حل را در ۱۹۷۶ ارائه دادند.
- کاربردهای مهم:
  - حل مسأله توزیع کلید در روشهای رمزنگاری متقارن
  - امضای دیجیتال

Martin Edward Hellman
(1945 –)

Bailey Whitfield Diffie
(1944 –)

رمزنگاری کلید عمومی

- کلیدهای رمزگذاری و رمزگشایی متفاوت اما مرتبط هستند.
- رسیدن به کلید رمزگشایی از کلید رمزگذاری از لحاظ محاسباتی ناممکن است.
- (در حفظ محرمانگی) رمزگذاری امری همگانی است و اساساً نیازی به اشتراک گذاشتن اطلاعات محرمانه ندارد.
- (در حفظ محرمانگی) رمزگشایی از طرف دیگر امری اختصاصی بوده و محرمانگی پیام‌ها محفوظ می‌ماند.

نمادها و قراردادها

- **کلید عمومی:** کلید رمزگذاری (در حفظ محرمانگی)

☜ این کلید را برای شخص A با PU_a نشان می‌دهیم.

- **کلید خصوصی:** کلید رمزگشایی (در حفظ محرمانگی)

☜ این کلید را برای شخص A با PR_a نشان می‌دهیم.

نیازمندیهای «دستوری» رمزنگاری کلید عمومی

- از نظر محاسباتی برای طرف A، تولید یک زوج کلید (کلید عمومی PU_a و کلید خصوصی PR_a) آسان باشد.
- برای فرستنده، تولید متن رمز آسان باشد:

$$C = E_{PU_a}(M)$$

- برای گیرنده، رمزگشایی متن با استفاده از کلید متناظر آن آسان باشد:

$$M = D_{PR_a}(C) = D_{PR_a}[E_{PU_a}(M)]$$

نیازمندیهای «امنیتی» رمزنگاری کلید عمومی

- از نظر محاسباتی، تولید کلید خصوصی (PR_a) با دانستن کلید عمومی (PU_a) غیر ممکن باشد.

$$PU_a \nrightarrow PR_a$$

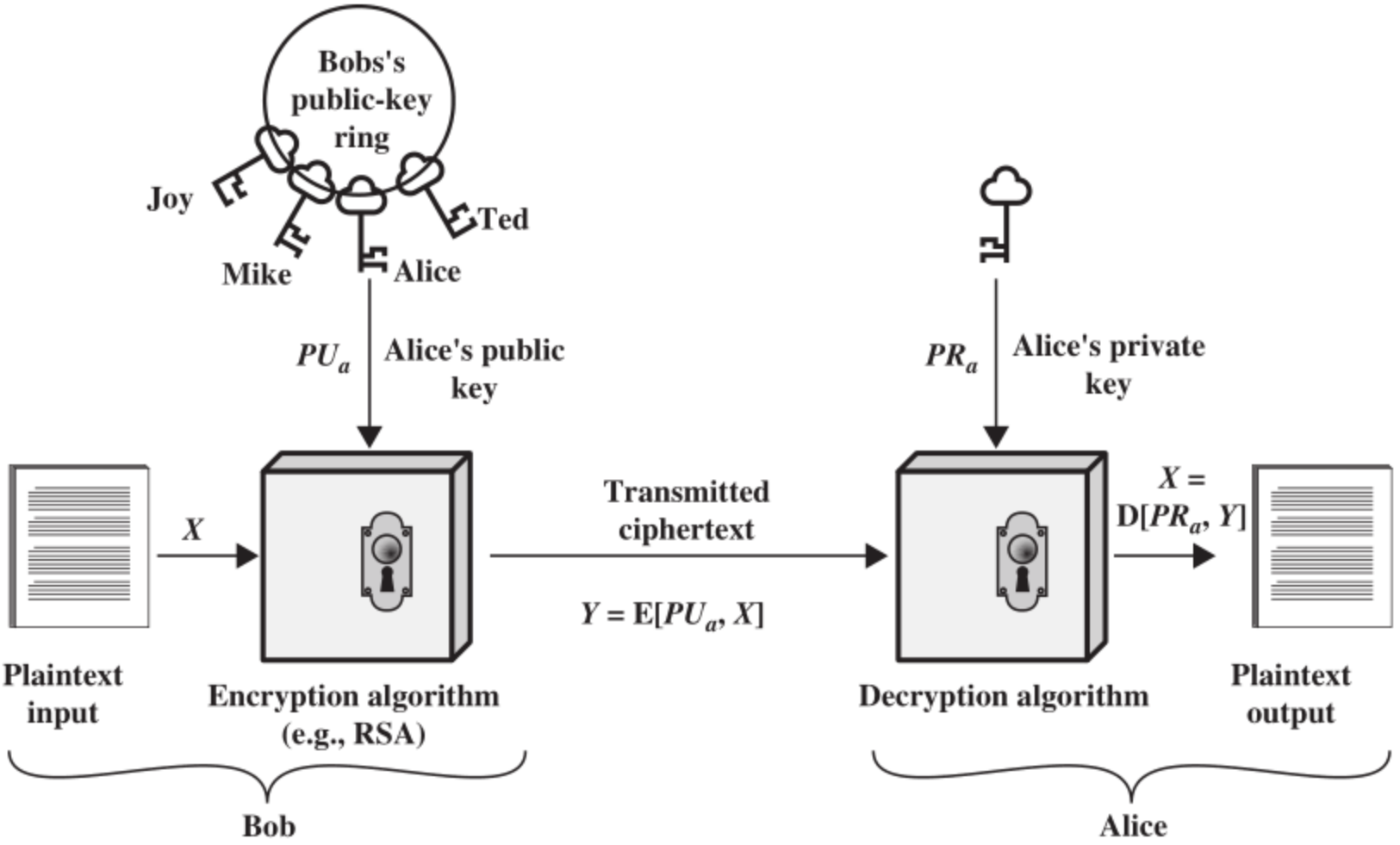
- بازیابی پیام M، با دانستن PU_a و C غیرممکن باشد.

$$C, PU_a \nrightarrow M$$

مراحل رمزنگاری کلید عمومی

- هر کاربر یک زوج کلید عمومی و خصوصی تولید می‌کند.
- کاربران کلید عمومی خود را به صورت عمومی اعلان می‌کنند درحالی که کلید خصوصی مخفی می‌باشد.
- همگان قادر به ارسال پیام رمز شده برای هر کاربر دلخواه با استفاده از کلید عمومی او هستند.
- هر کاربر می‌تواند با کمک کلید خصوصی پیام‌هایی که با کلید عمومی او رمز شده رمزگشایی کند.

رمزگذاری و رمزگشایی با کلید عمومی



فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- **مقایسه با رمزنگاری سنتی و متقارن**
- کاربردهای رمزنگاری کلید عمومی
- الگوریتم رمز RSA
- الگوریتم دیفی-هلمن
- الگوریتم رمز الجمل (ElGamal)

مقایسه رمزنگاری متقارن و رمزنگاری کلید عمومی

- مشکل مدیریت کلیدها در رمزنگاری متقارن
  - نیاز به توافق بر روی کلید پیش از برقراری ارتباط
  - برای ارتباط n نفر باهم به $\binom{n}{2}$ کلید احتیاج داریم.
- عدم پشتیبانی رمزنگاری متقارن از امضای رقمی (دیجیتال)
- با این وجود الگوریتم‌های رمزنگاری متقارن **بسیار** سریع‌تر هستند.

جایگزینی یا تکمیل؟

- رمزنگاری کلید عمومی ← مکمل رمزنگاری متقارن
- تولید کلید نشست برای رمزنگاری متقارن
- اشتراک کلید نشست توسط رمزنگاری نامتقارن
- رمزگذاری و رمزگشایی پیامهای نشست توسط کلید نشست

سوء برداشت!

- دو تصور اشتباه درباره الگوریتم‌های کلید عمومی

☜ رمزنگاری با کلید عمومی امن‌تر است!

☜ مسئله توزیع کلید در رمزنگاری با کلید عمومی برطرف شده است!

- چگونه مطمئن شویم کلید عمومی لزوماً متعلق به شخص ادعا کننده است؟!
- توزیع کلید عمومی آسانتر است، ولی بدیهی و بدون مشکل نیست.

فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- مقایسه با رمزنگاری سنتی و متقارن
- **کاربردهای رمزنگاری کلید عمومی**
- الگوریتم رمز RSA
- الگوریتم دیفی-هلمن
- الگوریتم رمز الجمل (ElGamal)

کاربردهای رمزنگاری کلید عمومی

- **رمزگذاری/ رمزگشایی:** برای حفظ محرمانگی
- **امضای رقمی:** برای حفظ صحت پیام و معین نمودن فرستنده پیام (پیوند دادن پیام با امضا کننده) یا همان **عدم انکار**
- **توزیع کلید:** برای توافق طرفین روی کلید مخفی نشست، قبل از برقراری ارتباط

جایگاه عملی رمزنگاری کلید عمومی

- کلیدهای این نوع از الگوریتم‌ها، بسیار طولانی‌تر از الگوریتم‌های رمز متقارن هستند.

☜ الگوریتم RSA با پیمانه ۱۰۲۴ بیتی، امنیتی در حد الگوریتم‌های متقارن با کلیدهای ۸۷ بیتی دارد.

- سرعت الگوریتم‌های کلید عمومی از الگوریتم‌های رمزگذاری متقارن پایین‌تر است.
- RSA تقریباً چندین هزار بار کندتر از رمزهای متقارن (با امنیت یکسان) است.

حملات به رمزنگاری کلید عمومی

- جستجوی جامع (Brute force)
- محاسبه کلید خصوصی از کلید عمومی (یا: محاسبه پیام از روی متن رمز + کلید عمومی)
  - ☜ اثبات نشده که غیر ممکن است.
  - ☜ بر مبنای یک یا چند فرض (معقول) استوار است.

فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- مقایسه با رمزنگاری سنتی و متقارن
- کاربردهای رمزنگاری کلید عمومی
- **الگوریتم رمز RSA**
- الگوریتم دیفی-هلمن
- الگوریتم رمز الجمل (ElGamal)

کلیات الگوریتم رمزنگاری RSA

- ارائه توسط رایوست (R)، شامیر (S) و ادلمن (A) در سال ۱۹۷۸
- مشهورترین و پرکاربردترین الگوریتم رمزگذاری کلیدعمومی
- مبتنی بر توان رسانی پیمانه‌ای با پیمانه خیلی بزرگ
- امنیت مبتنی بر دشواری تجزیه اعداد بزرگ

Leonard Adleman
(1945 –)

Adi Shamir
(1952 –)

Ronald Linn Rivest
(1947 –)

- $\mathbb{Z}_n$: مجموعه اعداد نامنفی کمتر از n.
- $\mathbb{Z}_n^*$: مجموعه اعداد طبیعی کمتر از n و اول نسبت به آن.
- مثال:

$$\mathbb{Z}_{۱۲} = \{۰, ۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹, ۱۰, ۱۱\}$$
$$\mathbb{Z}_{۱۲}^* = \{۱, ۵, ۷, ۱۱\}$$

- **تابع $\varphi(n)$ اویلر:** تعداد اعضای $\mathbb{Z}_n^*$
- **مثال:** $\varphi(۱۲) = ۴$
- اگر n عددی اول باشد: $\varphi(n) = n - ۱$
- اگر n حاصل ضرب دو عدد اول p و q باشد:

$$\varphi(n) = \varphi(pq) = (p-۱)(q-۱)$$

- قضیه کوچک فرما (Fermat):

☜ p عددی اول و a عددی صحیحی که مضرب p نیست:

$$a^{p-۱} \equiv ۱ \pmod{p}$$

- مثال:

$$p = ۱۱, \qquad a = ۷$$

$$۷^{۱۱-۱} = ۲۸۲۴۷۵۲۴۹$$

$$= ۲۵۶۷۹۵۶۸ \times ۱۱ + ۱$$

$$\equiv ۱ \pmod{۱۱}$$

❑ قضیه اویلر (تعمیم قضیه فرما):

☜ اگر a و n اعداد طبیعی و نسبت به هم اول باشند:

$$a^{\varphi(n)} \equiv ۱ \pmod{n}$$

❑ مثال:

$$n = ۱۲, \qquad a = ۷$$

$$۷^{\varphi(۱۲)} = ۷^{۴}$$

$$= ۲۴۰۱ = ۲۰۰ \times ۱۲ + ۱$$

$$\equiv ۱ \pmod{۱۲}$$

نمادگذاری RSA

- n: پیمانه عمومی (Public Modulus)
  - ☜ حاصل ضرب دو عدد اول p و q بسیار بزرگ
- e: نمای عمومی (Public Exponent)
- d: نمای خصوصی (Private Exponent)
  - ☜ داریم: $d \leftarrow e^{-1} \pmod{\varphi(n)}$
- m: پیام، عددی متعلق به $\mathbb{Z}_n$
- **تابع RSA:** تابع یکطرفه $c \leftarrow m^e \pmod{n}$
- **تابع معکوس:** $m \leftarrow c^d \pmod{n}$

عدد صحیح k وجود دارد که:

$$ed = k\varphi(n) + 1$$

مثال از RSA

$$p = ۶۱, \quad q = ۵۳$$

$$n = pq = ۳۲۳۳$$

$$\varphi(n) = (۶۱-۱)(۵۳-۱) = ۳۱۲۰$$

$$\left.\begin{array}{l} e = ۱۷ \\ d = ۲۷۵۳ \end{array}\right\} \rightarrow ed \equiv ۱ \pmod{\varphi(n)}$$

$$m = ۶۵$$

$$c = E(m) = ۶۵^{۱۷} \pmod{۳۲۳۳} = ۲۷۹۰$$

$$c = ۲۷۹۰$$

$$m = D(c) = ۲۷۹۰^{۲۷۵۳} \pmod{۳۲۳۳} = ۶۵$$

چرا RSA درست کار می‌کند؟

- اگر $m \in \mathbb{Z}_n^*$ باشد، قضیه اویلر کار می‌کند:

$$\begin{aligned} c^d \pmod n &= \left(\left(m^e \bmod n \right)^d \bmod n \right) \\ &= m^{ed} \pmod n \\ &= m^{k\varphi(n)+1} \pmod n \\ &= m^1 \pmod n \\ &= m \end{aligned}$$

- حالت کلی: $m \in \mathbb{Z}_n$

☜ استفاده از قضیه فرما (تمرین)

روند تولید کلید در RSA

- ابتدا دو عدد اول بزرگ و تقریباً هم اندازه p و q را به طور تصادفی انتخاب کن به گونه‌ای که $p \neq q$.
- عدد n و $\varphi(n)$ را محاسبه کن.
- عدد صحیح e کوچکتر از $\varphi(n)$ را به گونه‌ای انتخاب کن که نسبت به $\varphi(n)$ اول باشد.
- d را محاسبه کن (معکوس e به پیمانه $\varphi(n)$).
- $\mathrm{PU} = (e, n)$ و $\mathrm{PR} = (d, n)$

مربع‌سازی و ضرب (Square & Multiply)

- چگونگی **توان رسانی** سریع (محاسبه a^b) با استفاده از دو عمل پایه:

  ☜ مربع کردن و ضرب در a

- **مثال:** کد به زبان پایتون

```
1 def pow(base, power):
2     accum = 1
3     for bit in bin(power)[2:]:
4         accum *= accum    # square
5         if bit == '1':
6             accum *= base  # multiply
7     return accum
```

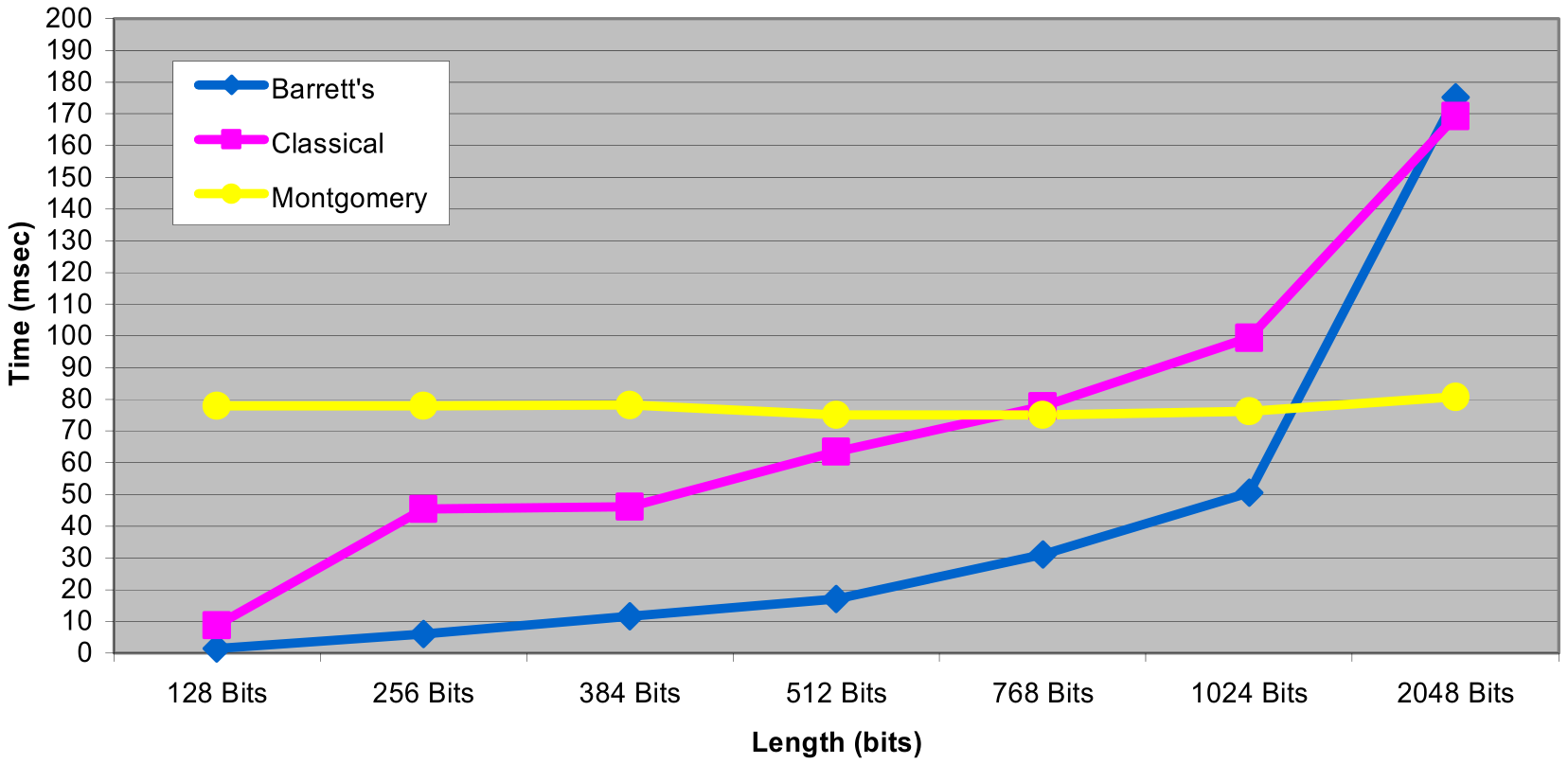
توان‌رسانی پیمانه‌ای سریع

- **هدف:** محاسبه سریع $a^b \mod n$.
- برخی از الگوریتم‌ها:
  - روش کلاسیک: «مربع‌سازی و ضرب» با mod گرفتن در هر گام
  - روش مونتگومری (Montgomery)
  - روش بارِت (Barrett)

https://en.wikipedia.org/wiki/Montgomery_reduction

https://en.wikipedia.org/wiki/Barrett_reduction

سرعت سه الگوریتم توان‌رسانی پیمانه‌ای (Pentium III)



M. Johnson, B. Phung, T. Shackelford, S. Rueangvivatanakij. **Modular Reduction of Large Integers Using Classical, Barrett, Montgomery Algorithms**, Tech. Report, 2002.

انتخاب نمای عمومی (e) در RSA

- با توجه به الگوریتم توان‌رسانی سریع، بهترین «نما» عددی است که کمترین ۱ را در نمایش بیتی خود داشته باشد. (چرا؟)
- در RSA، عدد e حتماً مقداری فرد است (چرا؟)
- بنابراین در نمایش بیتی e حداقل دو بیت ۱ است (چرا؟)
- معمولاً e را برابر عدد kاُم فرما قرار می‌دهند ($F_k = 2^{2^k} + 1$).
- اعداد صفرم تا سوم فرما (۳، ۵، ۱۷، ۲۵۷) کوچک هستند و به آنها حمله وارد است ← استفاده بسیار متداول از عدد چهارم فرما (۶۵۵۳۷).

تولید کلید RSA (توسط نرم‌افزار OpenSSL)

```
> openssl genrsa -out rsa1024.pem 1024

Loading 'screen' into random state - done
Generating RSA private key, 1024 bit long modulus
....++++++
.....................................++++++
e is 65537 (0x10001)
```

خواندن کلید RSA (توسط نرم‌افزار OpenSSL)

```
> openssl rsa -text -in rsa1024.pem
Private-Key: (1024 bit)
modulus:
    00:bf:6b:7a:8b:2d:a4:19:33:1d:72:81:1d:26:4e:
    73:cb:95:17:db:11:d1:d2:46:ad:6e:ac:6f:26:52:
    c8:fa:0a:07:a7:7f:86:fd:22:e8:0b:d1:b4:fc:32:
    7f:33:6e:de:5f:c3:6a:11:2e:bb:ef:cf:83:e7:83:
    0b:95:02:3b:72:15:03:18:38:24:e8:31:7d:b4:5c:
    a8:f5:2d:c6:84:e2:18:f8:a6:3b:65:96:eb:e4:07:
    71:5e:3f:79:18:52:8d:a6:ec:10:d7:b0:61:fc:6f:
    7d:90:c6:04:73:d9:f7:e6:1f:c9:61:c1:8e:48:76:
    95:97:ac:b7:92:60:cc:ca:9f
publicExponent: 65537 (0x10001)
```

خروجی ادامه دارد ...

نسبت اندازه اعداد در مثال قبلی

p = 0xE670E1AF4273ED120BAEBE947D5015F264889145B4237DFEA173
F49348E542115B4A19E643C8BE65F950C7B0607696E0AEDFECD2
FC4FB79F97D04C230D55A61D (512 bits)

q = 0xD4A6A2B20ABB7D00115D1D9CA4C32D1B36A6BCC06F74265B810B
C66AA414F31E235FD4DC9FFA368EE43458779C73CD96D00F108E
668BC923EEE97ED3F38926EB (512 bits)

e = 0x010001 (24 bits)

d = 0x196333D989B01DF77D8C563B7B7D2436780BB5EE6319B46E0423
B28A2EA8A120FB6AE7AB0B9FB98EF7BD3D45A541390F1D3C59B0
F5B5CF5482760E175727F8A22A0AE88CB207BBCB35426E260237
401FE29EF5A7FA9CD4EC21053B55D2339C4984A560C7C96BBE1E
3163DA17A75E96FB313245E5CB5CA42DBC39BBCFCFA54CE1

1024 bits

n = 0xBF6B7A8B2DA419331D72811D264E73CB9517DB11D1D246AD6EAC
6F2652C8FA0A07A77F86FD22E80BD1B4FC327F336EDE5FC36A11
2EBBEFCF83E7830B95023B721503183824E8317DB45CA8F52DC6
84E218F8A63B6596EBE407715E3F7918528DA6EC10D7B061FC6F
7D90C60473D9F7E61FC961C18E48769597ACB79260CCCA9F

1024 bits

حملات علیه RSA

- در صورتی که پارامترهای RSA به درستی انتخاب شوند...

  ☜ بهترین حمله علیه RSA، تجزیه n است.

- بهترین الگوریتم تجزیه برای عدد **دلخواه**، GNFS نام دارد.

- General Number Field Sieve

- پیچیدگی تجزیه توسط GNFS برای عدد n:

$$\exp\left(\left(\sqrt[3]{\frac{64}{9}}+o(1)\right)(\ln n)^{\frac{1}{3}}(\ln\ln n)^{\frac{2}{3}}\right)$$

$$\exp(x)=e^{x}$$

پیچیدگی تجزیه (امنیت) و سرعت RSA (کد: PyCrypto)

- صرف نظر از مقدار $o(1)$ در فرمول GNFS:

حدود عدد n	حدود پیچیدگی تجزیه	زمان هزار رمزگذاری (ثانیه)	زمان هزار رمزگشایی (ثانیه)	زمان هزار رمزنگاری Blowfish-160
$۲^{۱۰۲۴}$	$۲^{۸۷}$	۰٫۰۷	۳	۰٫۰۰۲
$۲^{۲۰۴۸}$	$۲^{۱۱۷}$	۰٫۲۵	۱۲	۰٫۰۰۴
$۲^{۴۰۹۶}$	$۲^{۱۵۶}$	۰٫۸۵	۷۶	۰٫۰۰۶

- **به همین دلیل می‌گوییم:** الگوریتم RSA با پیمانه ۱۰۲۴ بیتی، امنیتی در حد الگوریتم‌های متقارن با کلیدهای ۸۷ بیتی دارد.

حملات کانال جانبی (Side-Channel Attacks)

- حملاتی با دسترسی به اطلاعات جانبی (توان محاسباتی / زمان محاسبه) از CPUای که رمزنگاری روی آن انجام می‌شود.
- هنگام توان‌رسانی سریع در RSA، بیتهایی از d که ۱ هستند زمان / توان بیشتری مصرف می‌کنند.
- راه‌های مقابله
  - استفاده از توان رساندن با زمان ثابت محاسباتی
  - اضافه کردن تأخیرهای تصادفی
  - قرار دادن اعمال اضافی و گمراه کننده در بین محاسبات

قطعی بودن RSA

- RSA سنتی (معروف به Textbook RSA) قطعی است.

  ☜ رمز پیام m، همواره یک c ثابت است.

- **ایراد:** مهاجم تکرار شدن پیام را می‌فهمد.

- **راهکار:** استفاده از یک Padding تصادفی برای پیامها

☜ **استاندارد PKCS #1 v2**: روش Padding به نام OAEP

PKCS: Public-Key Cryptography Standards
OAEP: Optimal Asymmetric Encryption Padding

خاصیت همریختی Textbook RSA

- RSA سنتی، همریخت (Homomorphic) است:

$$m \equiv m_1 \times m_2 \pmod{n}$$

$$\mathrm{RSA}(m) \equiv \mathrm{RSA}(m_1) \times \mathrm{RSA}(m_2) \pmod{n}$$

- می‌خواهیم $c = c_1 \times c_2$ را رمزگشایی کنیم.
- استفاده از حمله **متن رمز منتخب**:
  - c_1 و c_2 را به رمزگشا می‌دهیم و m_1 و m_2 را می‌گیریم.
  - متن آشکار معادل c برابر است با $m_1 \times m_2$.
- RSA با OAEP این مشکل را ندارد.

فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- مقایسه با رمزنگاری سنتی و متقارن
- کاربردهای رمزنگاری کلید عمومی
- الگوریتم رمز RSA
- **الگوریتم دیفی-هلمن**
- الگوریتم رمز الجمل (ElGamal)

- فرض کنید p عددی اول باشد.
- مجموعه توانهای مختلف عدد a به پیمانه p را با $<a>_p$ نمایش می‌دهیم.
- مثال: $p = ۷$.

$$\langle ۳ \rangle_۷ = \{۱, ۳, ۲, ۶, ۴, ۵\} \qquad \langle ۲ \rangle_۷ = \{۱, ۲, ۴\}$$

- g را یک مولد (Generator) برای $\mathbb{Z}_p^*$ می‌خوانیم اگر:

$$\langle g \rangle_p = \mathbb{Z}_p^*$$

- **قضیه ۱:** $\mathbb{Z}_p^*$ حتماً مولد دارد.
- **قضیه ۲:** مولد $\mathbb{Z}_p^*$ الزاماً یکتا نیست.
- عدد اول p و مولد دلخواه g از $\mathbb{Z}_p^*$ را در نظر بگیرید.
- عدد α را به تصادف از $\mathbb{Z}_p$ انتخاب کنید.
- **مسئله لگاریتم گسسته (DL):** پیدا کردن α با داشتن مقادیر:

$$p, \quad g, \quad g^{\alpha} \pmod{p}$$

- اعداد α، β و γ به تصادف از $\mathbb{Z}_p$ انتخاب می‌شوند.
- همه محاسبات به پیمانه p انجام می‌شود.
- **مسئله دیفی-هلمن محاسباتی (CDH):**

☜ محاسبه $g^{\alpha\beta}$ با داشتن: $p,\quad g,\quad g^{\alpha},\quad g^{\beta}$

- **مسئله دیفی-هلمن تصمیمی (DDH):**

☜ تمیز دادن دو زوج:

$$\left(p,\quad g,\quad g^{\alpha},\quad g^{\beta},\quad g^{\alpha\beta}\right)$$

$$\left(p,\quad g,\quad g^{\alpha},\quad g^{\beta},\quad g^{\gamma}\right)$$

- مسأله DDH به گونه‌ای که بیان شد به سادگی قابل حل است.
  - تمیز دادن دو زوج با استفاده از مفهومی به نام «نماد لژاندر»
- لازم است مسأله DDH را با فرضهای زیر اصلاح کنیم:
  - فرض کنیم $g \in \mathbb{Z}_p^*$ به گونه‌ای باشد که $|\langle g \rangle_p| = q$.
  - می‌توان ثابت کرد که $q|p-1$.
  - فرض کنیم q اول بوده و همه محاسبات به پیمانه p انجام شود.
  - اعداد α، β و γ به تصادف از $\mathbb{Z}_q$ انتخاب می‌شوند.

مبانی ریاضی – ۵

- **فرض:** برای p و q خیلی بزرگ، مسائل DL، CDH و DDH دشوار هستند.
- پیچیدگی بهترین الگوریتمی که DL به پیمانه p را می‌شکند برابر $\sqrt{p}$ است.
  - **مثال:** برای امنیت معادل ۱۲۸ بیت ($۲^{۱۲۸}$)، اندازه p باید ۲۵۶ بیت باشد ($۲^{۲۵۶}$).
- شکستن CDH ساده‌تر از DL است (اگر DL بشکند، CDH هم می‌شکند).
- شکستن DDH ساده‌تر از CDH است (اگر CDH بشکند، DDH هم می‌شکند).

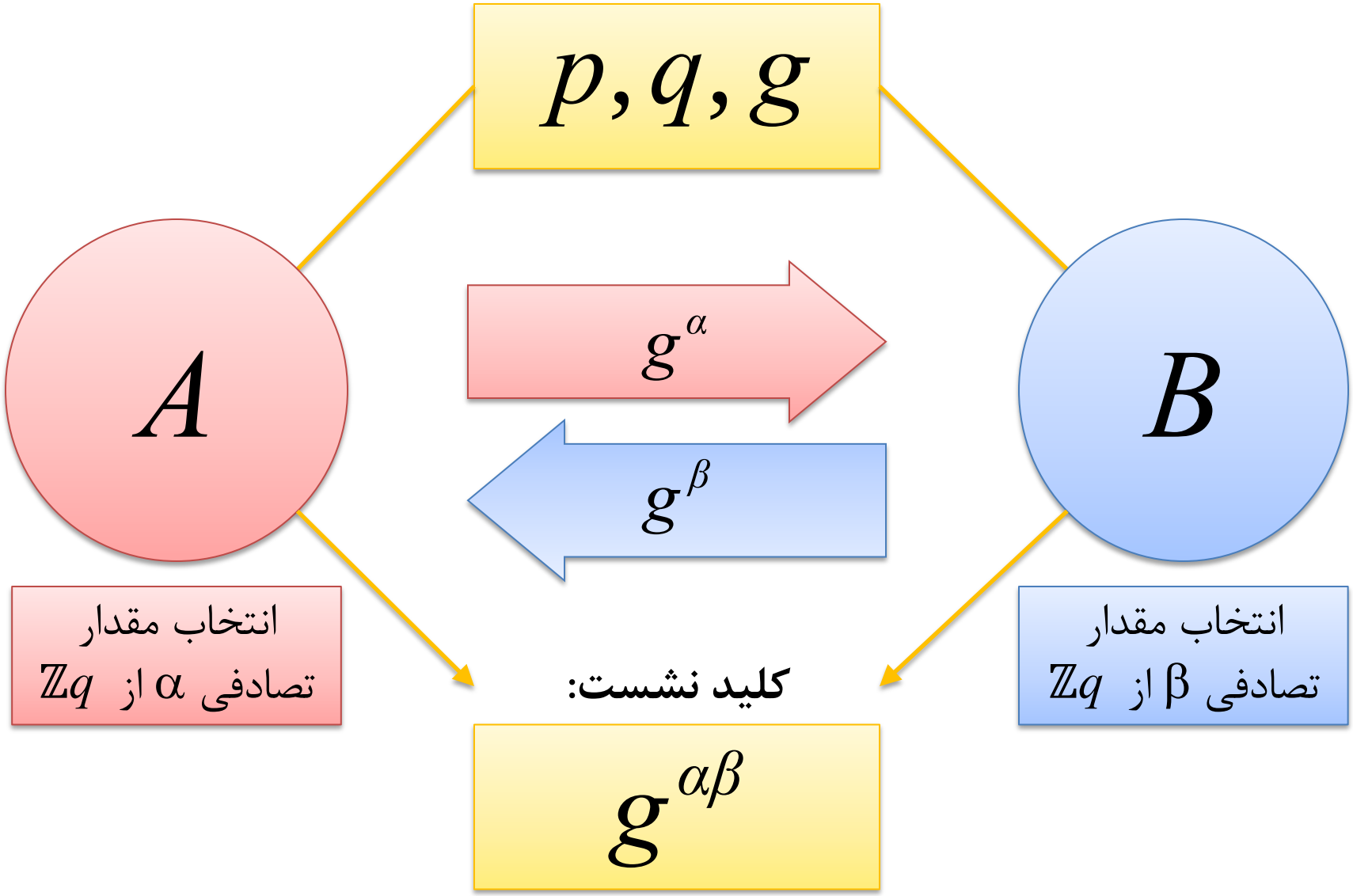
الگوریتم دیفی–هلمن

- توسط Diffie و Hellman در سال ۱۹۷۶ ارائه شد.
- برای تبادل کلید مورد استفاده قرار می‌گیرد.
- کلید نشست باید غیر قابل تمایز از یک مقدار تصادفی باشد.
- امنیت روش مبتنی بر دشواری شکستن DDH است.

- طرفین از قبل روی مقادیر p، q و g توافق می‌کنند.

☜ کلیه محاسبات به پیمانه p

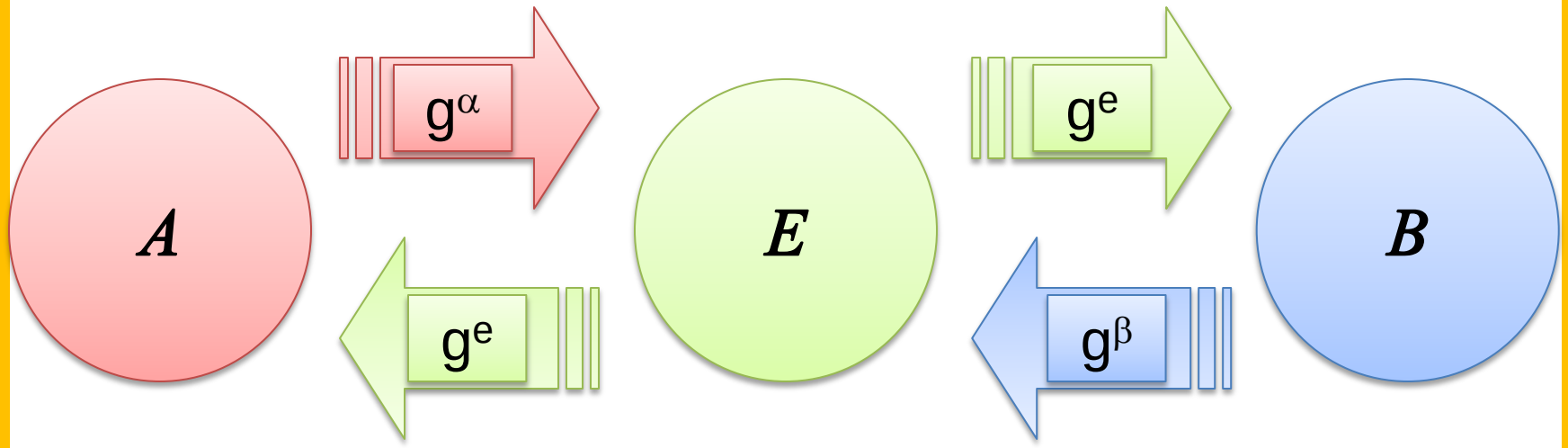
تبادل کلید دیفی–هلمن



حمله مرد میانی

- با فرض دشواری DDH، پروتکل دیفی-هلمن در برابر حملات منفعلانه (passive) امن است.
- اما این پروتکل در برابر حملات فعال (active) امن نیست.
- **حمله مرد میانی (MITM)**
  - مهاجم برای A وانمود می‌کند که B است.
  - مهاجم برای B وانمود می‌کند که A است.

حمله مرد میانی



$$K_1 = g^{\alpha e}$$

$$K_2 = g^{\beta e}$$

A گمان می‌کند کلید K_1 را با B به اشتراک گذاشته است.

- A پیامهای به مقصد B را با K_1 رمز می‌کند.
- B پیامهای به مقصد A را با K_2 رمز می‌کند.
- E می‌تواند همه پیامها را بخواند، و با کلید مناسب برای فرد منظور رمز نماید.

B گمان می‌کند کلید K_2 را با A به اشتراک گذاشته است.

رفع مشکل تبادل کلید دیفی–هلمن

- طرفین باید قبل از شروع پروتکل، یک کلید طولانی مدت (LTK) را به اشتراک گذاشته باشند.
  - LTK می‌تواند متقارن یا نامتقارن باشد.
  - در حالت نامتقارن، طرفین کلید عمومی یکدیگر را دارند.
- از LTK برای حفظ صحت g^{α} و g^{β} استفاده می‌شود.
  - Authenticated Diffie–Hellman (ADH)
  - در صورت حفظ صحت، مهاجم نمی‌تواند MITM را اجرا کند.

خاصیت محرمانگی پیشرو (Forward Secrecy)

- گاه به آن PFS هم گفته می‌شود (Perfect Forward Secrecy).
- **تعریف:** در صورت لو رفتن LTK در زمان T، کلیدهای نشستی که قبل از زمان T تبادل شده‌اند امن بمانند.
- ADH دارای خاصیت PFS است.

☜ از LTK فقط برای حفظ صحت و نه محرمانگی استفاده می‌شود.

☜ محرمانگی کلید نشست وابسته به LTK نیست.

- مبانی رمزنگاری کلید عمومی
- مقایسه با رمزنگاری سنتی و متقارن
- کاربردهای رمزنگاری کلید عمومی
- الگوریتم رمز RSA
- الگوریتم دیفی-هلمن
- **الگوریتم رمز الجمل (ElGamal)**

رمز الجمل (ElGamal)

- ابداع توسط الجمل، رمزنگاری مصری-آمریکایی، در سال ۱۹۸۵
  - ☜ در ایران بیشتر با نام «الجمال» شناخته می‌شود.
  - ☜ الجمال دانشجوی دکترای هلمن در دانشگاه استنفورد بود.
- امنیت رمز الجمل مبتنی بر دشواری DDH

طاهر الجمل
(۱۹۵۵ –)

تولید کلید الجمل

- انتخاب عدد اول بزرگ p
- انتخاب $g \in \mathbb{Z}_p^*$ به گونه‌ای که $|\langle g \rangle_p| = q$.
  - q باید اول و بزرگ باشد.
  - کلیه محاسبات به پیمانه p.
- انتخاب عدد تصادفی α از $\mathbb{Z}_q$ و محاسبه $h = g^{\alpha}$.
- p، q و g پارامترهای عمومی (همه مقادیر آنها را می‌دانند).
- α کلید خصوصی و h کلید عمومی.

رمزگذاری و رمزگشایی الجمل

- رمزگذاری عدد $m \in \langle g \rangle_p$:
  - انتخاب عدد تصادفی r از $\mathbb{Z}_q$.
  - مقدار رمز عبارت است از زوج: $c = (g^r, mh^r)$.
- رمزگشایی از زوج $c = (c_1, c_2)$ با استفاده از کلید خصوصی α:

$$m = \frac{c_2}{(c_1)^{\alpha}}$$

خاصیت همریختی رمز الجمل

- رمز الجمل بر خلاف Textbook RSA، رمزی قطعی نیست.

☜ اگر پیامی چند بار رمز شود، هر بار متن رمز جدیدی تولید می‌شود.

- با این حال، رمز الجمل نیز همریخت است.

$$m \equiv m_1 \times m_2 \pmod{q}$$

$$\mathrm{ElGamal}(m) \equiv \mathrm{ElGamal}(m_1) \times \mathrm{ElGamal}(m_2) \pmod{q}$$

- مشابه Textbook RSA، رمز الجمل نیز در برابر حمله متن رمز انتخابی شکننده است.

پایان

صفحه درس:

http://ce.sharif.edu/courses/94-95/1/ce442-1/

مراجعه حضوری جهت رفع اشکال: شنبه‌ها ۱۵ الی ۱۶

(طبقه پنجم دانشکده، درب شیشه‌ای جنب آسانسور)

یا در زمانهای دیگر با قرار قبلی

یا به وسیله رایانامه: dousti@ce