رمزنگاری نامتقارن(کلید عمومی)

Asymmetric (Public Key) Encryption

اصغر اصل اصغریان – دانشگاه ارومیه

فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- کاربردها و مقایسه با رمزنگاری متقارن
- الگوریتم رمز RSA
- پروتکل دیفی هلمن
- الگوریتم رمزالجمل(ElGamal)

نیازمندی‌های منجر به رمزنگاری کلید عمومی

- کلیدهای رمزنگاری و رمزگشایی متفاوت اما مرتبط باشند
- رسیدن به کلید رمزگشایی از کلید رمزگذاری از لحاظ محاسباتی ناممکن باشد
  - در عمل ممکن ولی سخت است
- رمزگذاری امری همگانی است و باید بدون نیاز به اطلاعات محرمانه مشترک ممکن باشد(در کاربردهای حفظ محرمانگی)
- رمزگشایی امری اختصاصی است و باید فقط توسط صاحب پیام ممکن باشد (در کاربردهای حفظ محرمانگی)
- دیفی و هلمن اولین بار در سال ۱۹۷۶ یک راه راه حل ارائه کردند

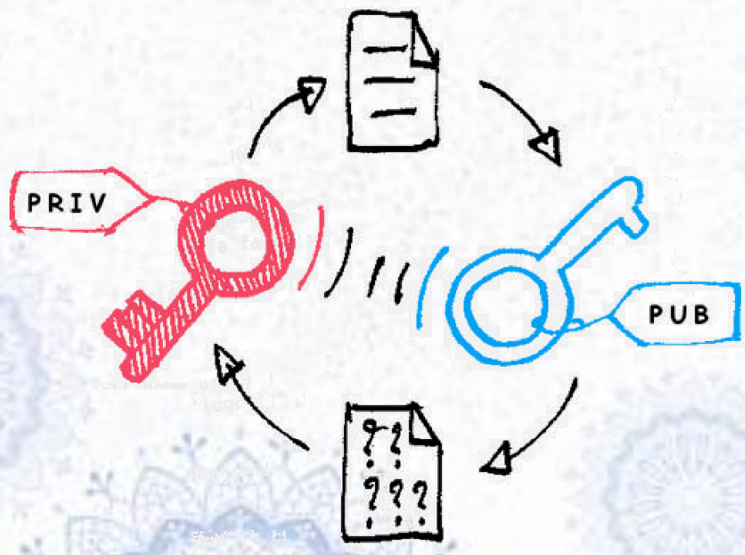
Bailey Whitfield Diffie
(1944 –)

Martin Edward Hellman
(1945 –)

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



نیازمندی های رمزنگاری کلید عمومی

- **نیازمندی‌های عملیاتی:**
  - از نظر محاسباتی برای طرف A، تولید یک زوج کلید آسان باشد:
    - کلید عمومی PU_a و کلید خصوصی PR_a
  - برای فرستنده تولید متن رمز آسان باشد:

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

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

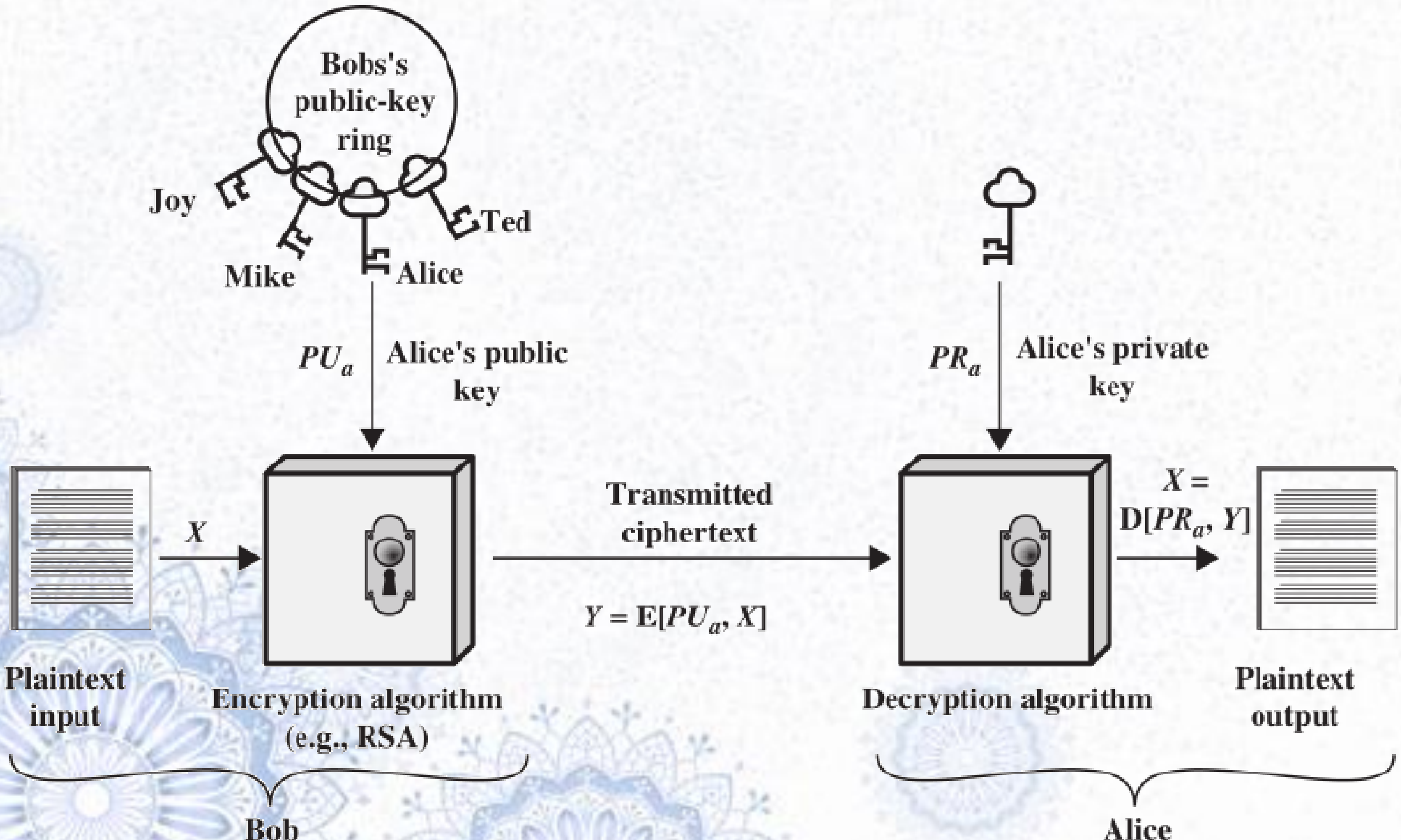
$$M=D_{PR_a}(C)= D_{PR_a}(E_{PU_a}(M))$$

- **نیازمندی‌های امنیتی:**
  - از نظر محاسباتی به دست آوردن کلید خصوصی PR_a، با دانستن کلید عمومی Pu_a غیر ممکن باشد
  - بازیابی پیام M با دانستن، متن رمز C و کلید عمومی PU_a غیرممکن باشد

مراحل رمزنگاری کلید عمومی

- هر کاربر یک زوج کلید عمومی و خصوصی تولید می کند
- کاربران کلید عمومی خود را به صورت عمومی اعلان می کنند در حالی که کلید خصوصی مخفی نگه می دارند
- 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 رمز گشایی با کلید عمومی



فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- کاربردها و مقایسه با رمزنگاری متقارن
- الگوریتم رمز RSA
- پروتکل دیفی هلمن
- الگوریتم رمزالجمل(ElGamal)

کاربردهای رمز نگاری کلید عمومی

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

چند نکته درباره رمز نگاری کلید عمومی

- کلیدهای این نوع از الگوریتم‌ها, اغلب بسیار طولانی‌تر از الگوریتم‌های رمز متقارن هستند
- الگوریتم RSA با پیمانه ۱۰۲۴بیتی, امنیتی در حد الگوریتم‌های متقارن با کلید ۸۷ بیتی دارد
- سرعت الگوریتم های کلید عمومی از الگوریتم های رمزگذاری متقارن پایین‌تر است
- الگوریتم RSA چندین هزار بار کندتراز رمزهای متقارن (با امنیت یکسان) است

رمزنگاری متقارن در مقایسه با رمزنگاری کلید عمومی

- **مشکلات رمزنگاری متقارن**
  - نیاز به توافق بر روی کلید قبل از برقرار ارتباط
  - نیاز به n^2-n کلید برای ارتباط محرمانه n نفر باهم
  - عدم پشتیبانی از امضای دیجیتال
- **مزایای رمزنگاری متقارن**
  - سرعت بسیار بالا
  - پیاده‌سازی کاراتر
  - طول کلید کوتاه‌تر
- **بنابراین**
  - رمزنگاری کلید عمومی مکمل رمزنگاری متقارن است (و ~~نه~~ جایگزین آن)
    - استفاده از رمزنگاری کلید عمومی برای توزیع کلید نشست
    - استفاده از رمزنگاری متقارن برای رمز کردن پیام‌های نشست

سوء برداشت!

- دو تصور اشتباه درباره الگوریتم های کلید عمومی:
  - رمزنگاری کلید عمومی امنیت بیشتری دارد
  - رمزنگاری کلید عمومی مسئله توزیع کلید را حل می کند
    - چگونه مطمئن شویم کلید عمومی لزوما متعلق به شخص ادعا کننده است؟!
    - حمله مرد میانی
    - نیاز به زیر ساخت کلید عمومی

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 غیرممکن است
  - معمولا امنیت هر روش برمبنای یک یا چند فرض معقول استوار است
    - مثلا در rsa به سختی تجزیه اعداد بزرگ به عوامل اول

فهرست

- مبانی رمزنگاری کلید عمومی
- کاربردها و مقایسه با رمزنگاری متقارن
- الگوریتم رمز RSA
- پروتکل دیفی هلمن
- الگوریتم رمزالجمل(ElGamal)

کلیات الگوریتم رمزنگاری RSA

- ارائه کنندگان:
  - رایوست R، شامیر S، و ادلمن A (۱۹۷۸)
- مشهورترین و پرکاربردترین الگوریتم رمزنگاری نامتقارن
- مبتنی بر توان رسانی پیمانه‌ای با اعداد خیلی بزرگ
- مبتنی بر دشواری تجزیه اعداد خیلی بزرگ به عوامل اول

مبانی ریاضی (۱)

- $\mathbb{Z}_n$: مجموعه اعداد نامنفی کوچکتر از n
- $\mathbb{Z}^*_n$: مجموعه اعداد طبیعی کوچکتر از n که نسبت به n اول هستند
- مثال:

$$\mathbb{Z}_{12}=\{0,1,2,3,4,5,6,7,8,9,10,11\}$$

$$\mathbb{Z}^*_{12}=\{1,5,7,11\}$$

به صور کلی

مبانی ریاضی (۲)

- **تابع اویلر** $\varphi(n)$: تعداد اعضای $\mathbb{Z}^*_n$
  - مثال:

$$\varphi(12)=4$$

- **اگر n عددی اول باشد:**

$$\varphi(n)=n-1$$

- **اگر n حاصلضرب دو عدد اول p و q باشد:**

$$\varphi(n)= \varphi(p)\times\varphi(q) =(p-1) \times(q-1)$$

مبانی ریاضی (۳)

- **قضیه کوچک فرما (Fermat):**
  - اگر p عددی اول و a عدد صحیحی که مضرب p نیست باشد:

$$a^{p-1} \equiv 1 \pmod{p}$$

- **مثال:**

$p=11, \quad a=7$

$$\begin{aligned} 7^{11-1} &= 282475249 \\ &= 25679568 \times 11 + 1 \\ &\equiv 1 \pmod{11} \end{aligned}$$

- **قضیه اویلر (تعمیم قضیه فرما):**
  - اگر a و n اعدادی طبیعی و نسبت به هم اول باشند:

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

- **مثال:**

$n=12, \quad a=7$

$$7^{\varphi(12)} = 7^4$$
$$= 2401 = 200\times 12 + 1$$
$$\equiv 1 \pmod{12}$$

ویژگی‌های RSA

- **متن آشکار و متن رمز اعداد صحیح بزرگ در نظر گرفته می‌شوند**
  - اعداد بین ۰ و n-1
- **هر بلاک از متن آشکار یک عدد صحیح خیلی بزرگ فرض می‌شود**
  - تعداد بیت‌های هر بلاک باید بین ۰ و $\log(n)$ باشد
- **عملیات رمزگذاری و رمزنگاری مبتنی بر عمل به توان رسانی**
- **همه اعمال در پیمانه n انجام می شود**

نمادهای مورد استفاده در RSA

- **n: پیمانه عمومی (Public Modulus)**
  - حاصل‌ضرب دو عدد اول بسیار بزرگ p و q
    - یعنی $n=pq$
    - p و q مخفی نگهداری می‌شوند
- **e: نمای عمومی (Public Exponent)**
  - e باید بین ۱ و $\varphi(n)$ و نسبت به $\varphi(n)$ اول باشد
- **d: نمای خصوصی (Private Exponent)**
  - به طوریکه d معکوس ضربی e در پیمانه $\varphi(n)$ است یعنی:

$$ed \equiv 1 \pmod{\varphi(n)}$$
$$ed = k\varphi(n) + 1$$

- **m: متن آشکار، در قالب یک عدد متعلق به $\mathbb{Z}_n$**
- **تابع رمزگذاری**
- **تابع رمزگشایی**

$$c = E(m) = m^e \pmod{n}$$
$$m = D(c) = c^d \pmod{n}$$

مثالی از RSA

$p = 61, \quad q = 53$

$n = pq = 3233$

$\varphi(n) = (61\text{-}1)(53\text{-}1) = 3120$

$e = 17$

$d = 2753 \quad \rightarrow ed = 15 * 3120 + 1$

$m = 65$

$c = E(m) = 65^{17} \text{ (mod 3233)} = 2790$

$m = D(c) = 2790^{2753} \text{ (mod 3233)} = 65$

محاسبه معکوس ضربی

- **الگوریتم اقلیدس توسعه یافته**
  - معکوس ضربی a به پیمانه n (a و n باید نسبت به هم اول باشند)
- **مثال:**
  - معکوس ضربی ۲۷۵۳ را به پیمانه ۳۱۲۰

	۳۱۲۰	۱۷	۹	۸	۱	۰
q_i		۱۸۳	۱	۱	۸	
p_i	۰	۱	-۱۸۳	۱۸۴	-۳۶۷=۲۷۵۳	

	۳۱۲۰	۲۷۵۳	۳۶۷	۱۸۴	۱۸۳	۱	۰
q_i		۱	۷	۱	۱	۱۸۳	
p_i	۰	۱	-۱	۸	-۹	۱۷	

- $p_0 = 0, p_1 = 1$
- $p_i = p_{i-2} - p_{i-1}\, q_{i-1}$

اثبات درستی RSA

- اگر $m \in \mathbb{Z}_n$ باشد آنگاه با استفاده از قضیه اویلر اثبات ساده است:

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

- درغیر اینصورت اثبات کمی پیچیده‌تر است
  - استفاده از قضیه باقیمانده چینی و قضیه فرما

فرایند تولید کلید RSA

- ابتدا دو عدد اول بزرگ (برای مثال ۵۱۲ بیتی) p و q را به طور تصادفی انتخاب کنید $(p \neq q)$
  - با کمک الگوریتم‌های تصادفی آزمون اول بودن
- اعداد n و $\varphi(n)$ را محاسبه کنید
- عدد صحیح e کوچکتر از $\varphi(n)$ را انتخاب کنید به طوریکه نسبت به $\varphi(n)$ اول باشد
- معکوس ضربی e در پیمانه $\varphi(n)$ را پیدا کنید و آن را d بنامید
- کلید عمومی عباتست از: $\mathbf{PU=(e,n)}$
- کلید خصوصی عباتست از: $\mathbf{PR=(d,n)}$
- اعداد p، q و $\varphi(n)$ باید محرمانه بمانند
  - در فرایند رمزگشایی نیازی به آنها وجود ندارد و می‌توان آنها را امحا کرد

انتخاب نمای عمومی (e)

- بهتر است نماها تعداد کمی ۱ در نمایش بیتی خود داشته باشند
  - برای سرعت بیشتر الگوریتم‌ها به توان رسانی
- در RSA، حتما مقداری فرد است
  - چون p و q اول (و فرد) هستند، $\varphi(n)$ حتما زوج است
  - e باید نسبت به $\varphi(n)$ اول باشد
- بنابراین در نمایش بیتی e حداقل دو بیت ۱ داریم (چرا؟)
- معمولا از عدد kام فرما ($2^{2^k}+1$) به عنوان e استفاده می شود
- به چهار عدد اول فرما (۳، ۵، ۱۷، ۲۵۷) حملاتی وارد است
- استفاده از عدد چهارم (۶۵۵۳۷) ، متداول است
- چون e کوچک است، معمولا عملیات رمزنگاری بسیار سریعتر است
  - در عوض d مقداری بزرگتر و عملیات رمزگشایی کندتر است

تولید کلید RSA با استفاده از ابزار 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 نرم افزار (Open SSL)

```
> 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
n = 0xBF6B7A8B2DA419331D72811D264E73CB9517DB11D1D246AD6EAC
       6F2652C8FA0A07A77F86FD22E80BD1B4FC327F336EDE5FC36A11
       2EBBEFCF83E7830B95023B72150318382 4E8317DB45CA8F52DC6
       84E218F8A63B6596EBE407715E3F7918528DA6EC10D7B061FC6F
       7D90C60473D9F7E61FC961C18E48769597ACB79260CCCA9F
```

1024 bits

1024 bits

حملات علیه RSA

- در صورتی که پارامترهای RSA به درستی انتخاب شوند بهترین حمله علیه آن تجزیه n است
- بهترین الگوریتم تجزیه برای یک عدد دلخواه:
  - حمله: General Number Field Service
  - پیچیدگی حمله نشان می‌دهد مقدار کار مورد نیاز برای شکستن RSA با کلید ۱۰۲۴ بیت برابر است با
    - مقدار کار مورد نیاز برای جستجوی جامع برای یک الگوریتم متقارن با کلید ۸۷ بیتی

حملات کانال جانبی(Side Channel Attacks)

- حملاتی با دسترسی به اطلاعات جانبی(توان محاسباتی/زمان محاسبه)ازCPUای که رمزنگاری روی آن انجام می شود
- هنگام توان رسانی سریع در RSA, بیتهایی از dکه ۱ هستند زمان/توان بیشتری مصرف می کنند
  - از زمان یا توان مصرفی می توان حدس زد که آیا یک بیت خاص از d یک هست یا نه
- راه های مقابله
  - استفاده از توان رساندن با زمان ثابت محاسباتی
  - اضافه کردن تاخیرهای تصادفی
  - قرار دادن اعمال اضافی و گمراه کننده در بین محاسبات

قطعی بودن RSA

- **الگوریتم RSA در حالت عادی قطعی است**
  - رمز یک پیام ثابت با یک کلید ثابت همواره یک متن رمز می دهد
  - معروف به Textbook RSA
- **ایراد: مهاجم تکرار پیام را می فهمد**
- **راهکار: استفاده از یک padding تصادفی برای پیام‌ها**
  - pad: مقدار تصادفی (اضافی) که به انتهای پیام چسبانده می شود
- **استاندارد مرسوم: PKCS#1**
  - یک روش padding بر مبنای شبکه فایستل به نام OAEP

خاصیت همریختی RSA سنتی

- RSA سنتی هم‌ریخت (Homomorphic) است:

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

$$m^d \equiv m_1{}^d \times m_2{}^d \qquad \pmod{n}$$

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

- برای رمزگشایی $c=c_1\times c_2$ می‌توان از حمله **متن رمز منتخب** استفاده کرد:
  - c_1 و c_2 را به رمزگشا می دهیم و m_1 و m_2 را می‌گیریم
  - $m_2\times m_1$ معادل m همان متن آشکار مربوط به c است
- استفاده از روش‌های padding این مشکل را از بین می برد
- در برخی از کاربردها ویژگی همریختی مطلوب یا مورد نیاز است

فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- کاربردها و مقایسه با رمزنگاری متقارن
- الگوریتم رمز RSA
- پروتکل دیفی هلمن
- الگوریتم رمزالجمل(ElGamal)

مبانی ریاضی

- فرض کنید p عددی اول باشد
- مجموعه توان‌های مختلف عدد a به پیمانه p را با $<a>_p$ نشان می‌دهیم:
  - زیر گروه $\mathbb{Z}^*_7$ تولید شده توسط a
- مثال:

$$p = 7$$
$$<2>_7 = \{2^i \bmod 7 \mid i \in N\} = \{1,2,4\}$$
$$<3>_7 = \{1,3,2,6,4,5\}$$

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

$$<g>_p = \mathbb{Z}^*_p$$

مبانی ریاضی (۲)

مثال: توان‌های اعداد مختلف به پیمانه ۱۹

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	17	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	16	4	1	5	6	11	17	9	7	16	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	9	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1	18	1

مبانی ریاضی (۳)

- قضیه ۱: برای هر $\mathbb{Z}^*_p$ حداقل یک مولد وجود دارد (با فرض: p اول است)
- قضیه ۲: مولد $\mathbb{Z}^*_p$ همیشه یکتا نیست، ولی با داشتن تجزیه p-1 می‌توان به سادگی یک مولد دلخواه برای آن یافت.

- عدد اول p و یک مولد دلخواه مانند g برای $\mathbb{Z}^*_p$ را در نظر بگیرید
- عدد α را به تصادف از $\mathbb{Z}_{p-1}$ انتخاب کنید

- مسئله لگاریتم گسسته (DL):
  - پیدا کردن α با داشتن مقادیر زیر:

$$p,\ \ g,\ \ g^{\alpha}\ (\mathrm{mod}\ p)$$

مسئله دیفی هلمن

- اعداد α، β، و γ را به تصادف از $\mathbb{Z}_{p\text{-}1}$ انتخاب می‌کنیم
- همه محاسبات به پیمانه p انجام می‌گیرد

- مسئله دیفی هلمن محاسباتی (CDH):
  - محاسبه $g^{\alpha\beta}$ با داشتن: $p, g, g^{\alpha}, g^{\beta}$
- مسئله دیفی هلمن تصمیمی(DDH):
  - تمیز دادن دو زوج زیر:

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

به عبارت دیگر با داشتن یک مقداری تشخیص بدهیم که آن مقدار $g^{\alpha\beta}$ هست یا نه

مسئله دیفی هلمن (۲)

- راه‌حل‌های کارایی برای مساله DDH وجود دارد
  - تمیز دادن دو زوج با استفاده از مفهومی به نام «نماد لژاندر»

- برای سخت کردن DDH مسئله را اصلاح می‌کنیم
  - به جای اینکه g را مولد $\mathbb{Z}^*_p$ بگیریم،
  - $g \in \mathbb{Z}^*_p$ را به گونه‌ای انتخاب می‌کنیم که $|<g>_p|=q$
    - یعنی توان‌های g فقط تعداد q تا از اعضای $\mathbb{Z}^*_p$ را تولید کند
    - می‌توان ثابت کرد که: $q|p\text{-}1$

  - علاوه‌بر این g طوری انتخاب می‌شود که q اول باشد
  - اعداد α، β، و γ به تصادف از $\mathbb{Z}^*_q$ انتخاب می‌شوند
  - کماکان همه محاسبات به پیمانه p انجام می‌شود که اول است

مسئله دیفی هلمن (۳)

- پیچیدگی بهترین الگوریتمی که DL به پیمانه p را حل می‌کند $p^{1/2}$ است
- شکستن CDH و DDH سخت‌تر از DL نیست
  - اگر DL را بتوانیم حل کنیم هر دو مسئله DH قابل حل خواهد بود

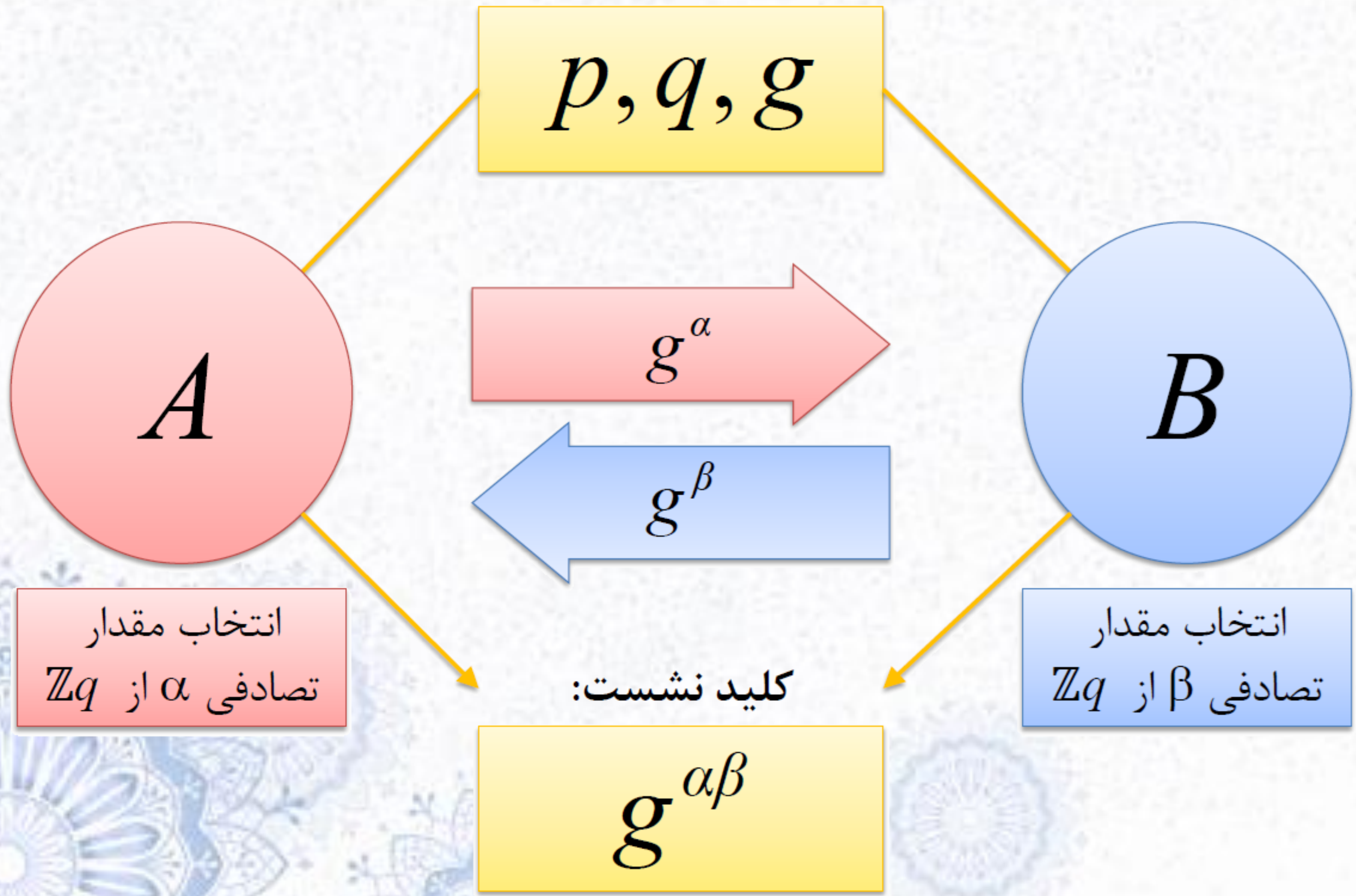
پروتکل دیفی هلمن

- ارائه شده توسط **Diffie** و **Hellman** در سال ۱۹۷۶
- کاربرد: تبادل کلید
- کلید نشست مبادله شده باید غیرقابل تمایز از یک مقدارتصادفی باشد
- امنیت روش مبتنی بر دشواری شکستن **DDH** است

- طرفین از قبل روی مقادیر p، q و g توافق می‌کنند
  - به طوری که p و q اول باشند و ($p=2q+1$)
  - و g مولد $\mathbb{Z}^*_q$ باشد
  - اصطلاحا q را Sophie Germain prime و p را safe prime متناظر با آن می‌نامند

- **کلیه محاسبات به پیمانه p**

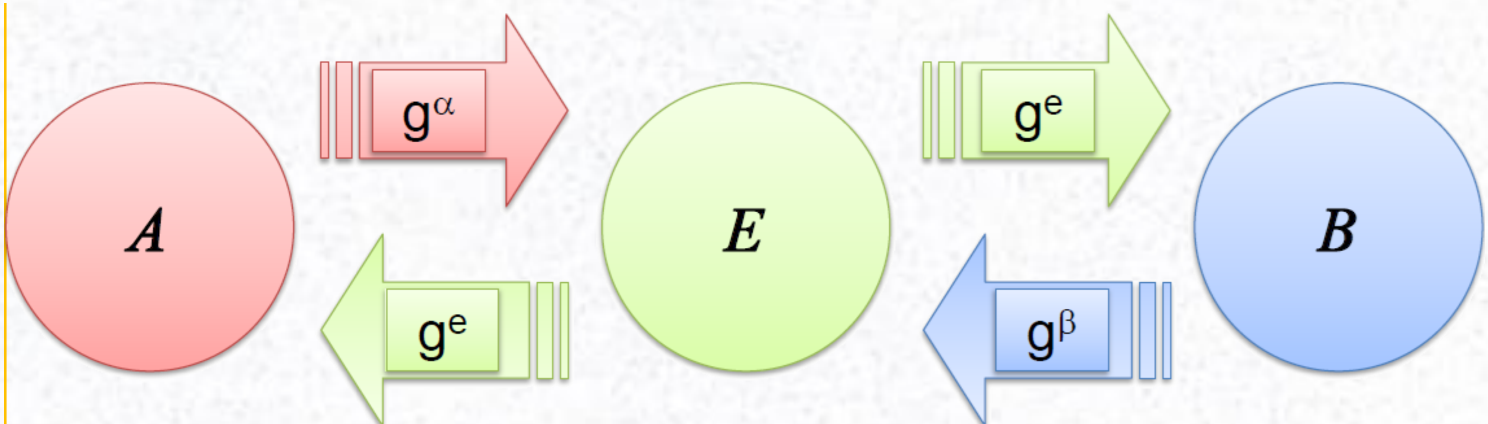
تبادل کلید دیفی هلمن



حمله مرد میانی

- با فرض دشواری DDH, پروتکل دیفی‌هلمن 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 می کند که B است
    - مهاجم برای B وانمود می کند که A است

حمله مرد میانی



$$K_1 = g^{\alpha e} \qquad K_2 = g^{\beta e}$$

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.

مقابله با حمله مرد میانی در دیفی‌هلمن

- **استفاده از یک کلید طولانی مدت (Long-Term Key)**
  - برای احراز اصالت پیام‌های مبادله شده: g^α و g^β
- **طرفین باید قبل از شروع پروتکل این کلید LTK را به اشتراک بگذارند**
  - از یک LTK می توان در چندین پروتکل استفاده کرد
  - LTK می‌تواند کلید متقارن یا یک زوج کلید عمومی باشد

- **پروتکل حاصل را ADH می‌نامند**
  - Authenticated Diffie-Hellman

خاصیت محرمانگی پیشرو(Forward secrecy)

- **تعریف: در صورت لو رفتن LTKدر زمان *t*,کلیدهای نشستی که که قبل از زمان *t* تبادل شده اند امن بمانند**

- **گاه به آن PFS هم گفته می شود(Perfect forward Secrecy)**

- **ADHدارای خاصیتPFSاست**
  - از LTK فقط برای حفظ صحت و نه محرمانگی استفاده می شود
  - محرمانگی کلید نشست وابسته به LTK نیست
  - اگر LTK لو برود فقط از زمان لو رفتن آن به بعد امکان حمله MITM وجود دارد

فهرست مطالب

- مبانی رمزنگاری کلید عمومی
- کاربردها و مقایسه با رمزنگاری متقارن
- الگوریتم رمز RSA
- پروتکل دیفی هلمن
- الگوریتم رمزالجمل (ElGamal)

رمز الجمل (ElGamal)

- **ابداع توسط طاهر الجمل، رمزنگار مصری آمریکایی (۱۹۸۵)**
  - الجمل دانشجوی دکترای هلمن در دانشگاه استنفور بوده است
- **امنیت مبتنی بر دشواری DDH**

- **الگوریتم امضای دیجیتال (DSA) گونه‌ای از رمز الجمل است**
  - استاندارد مورد استفاده برای امضای دیجیتال

- **الجمل یک رمز غیر قطعی (احتمالاتی) است (بر خلاف RSA)**
  - متن رمز حاصل از یک متن آشکار با یک کلید همواره یکسان نیست
- **الجمل یک رمز همریخت است (مشابه RSA)**
- **طول متن رمز از طول متن آشکار بیشتر است**
  - تقریبا دو برابر

تولید کلید الجمل

- **انتخاب عدد اول بزرگ p**
  - پیمانه‌ی کلیه محاسبات
- **انتخاب $g\in\mathbb{Z}^*_p$ به گونه‌ای که $|<g>_p|=q$**
  - q باید عددی اول و بزرگ باشد
- **p، q و g پارامترهای عمومی خوانده می‌شوند**
  - مقدار آنها را همه می‌دانند

- **انتخاب عدد تصادفی α از $\mathbb{Z}^*_q$ و محاسبه‌ی $h=g^\alpha$**
- **کلید خصوصی**: α
- **کلید عمومی**: h

رمزگذاری و رمزگشایی الجمل

- **برای رمزگذاری پیام M**
  - ابتدا پیام M به یک روش قابل برگشت به $m \in <g>_p$ نگاشته می‌شود
  - یک عدد تصادفی مانند r از $\mathbb{Z}^*_q$ انتخاب می‌شود
  - متن رمز عبارت است از:

$$c = (g^r, mh^r)$$

- **رمز گشایی از متن رمز $c=(c_1,c_2)$ با کلید خصوصی α:**

$$m = c_2 \times c_1^{-\alpha} = mg^{r\alpha}g^{-r\alpha}$$

- **رمزنگاری نیازمند دو عمل توان‌رسانی و رمزگشایی نیازمند یک عمل توان‌رسانی**