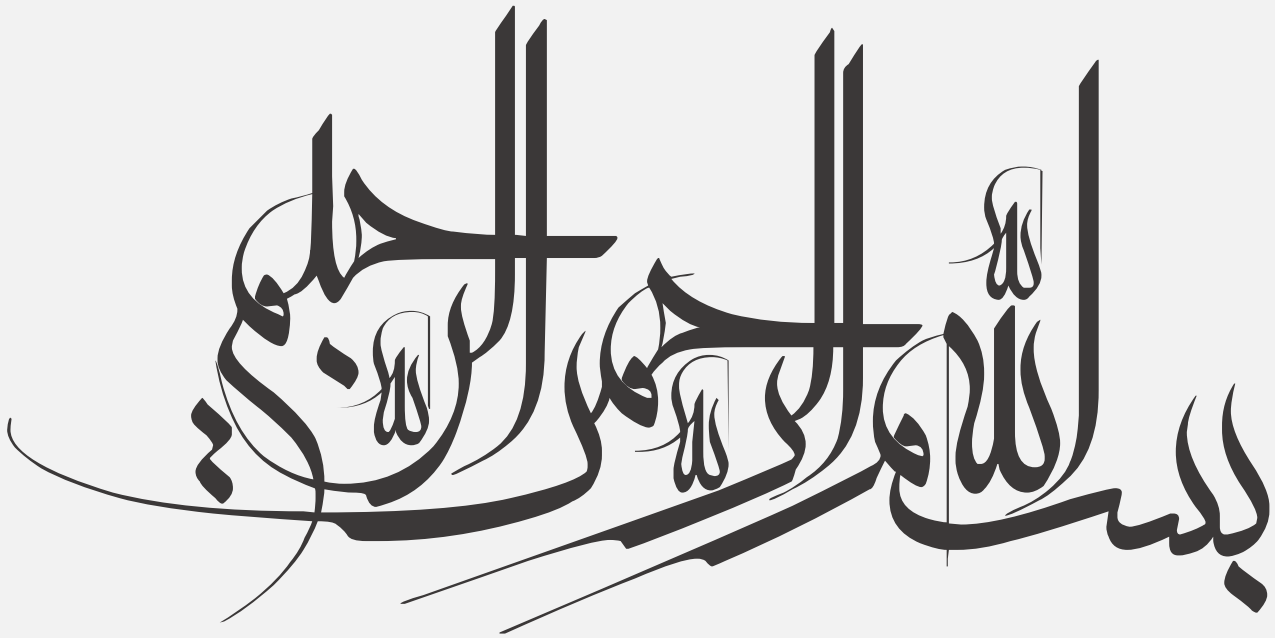



# درس رمز

بخش دوم فصل 9

**استاد**

دکتر لاجوردی

پاییز 1404

# فهرست مطالب پایان‌نامه


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# آزمون میلر–رابین – هدف این بخش

در این قسمت می‌خواهیم آزمون اول‌بودن Miller–Rabin را معرفی کنیم و بعداً «قضیه ۹٫۳۳» را درباره‌ی درستی آن ثابت کنیم.

این آزمون روی مطالب بخش ۹.۱.۵ (گروه‌ها و قضیه‌ی فرما) تکیه دارد، اما نتیجه‌ی این بخش دیگر در ادامه‌ی کتاب مستقیماً استفاده نمی‌شود.

ایده‌ی اصلی این است که یک خاصیت عددی پیدا کنیم که برای همه‌ی اعداد اول درست و برای بیشتر اعداد مرکب غلط باشد.

فرض کن N عدد ورودی تست باشد. اگر N اول باشد، همه‌ی اعداد ۱ تا N−۱ با آن هم‌اول‌اند و اندازه‌ی گروه Z*_N برابر N−۱ است.

در این حالت برای هر عدد a در بازه‌ی ۱ ... N−۱ داریم:

$$a^{N-1} \stackrel{?}{=} 1 \bmod N$$

a^(N−۱) ≡ ۱ (mod N)

این همان شکل گروهیِ قضیه‌ی کوچک فرماست و نقطه‌ی شروع آزمون ماست.



فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# هدف ۹.۲.۲ در یک نگاه

- $Z^*_N$ یعنی: عددهای ۱ تا N−۱ که با N نسبت به هم اول‌اند ($\gcd(a,N)=1$) و ضربشان "مد N " یک گروه می‌سازد.
- "پاس کردن تست" یعنی: الگوریتم برای N می‌گوید prime (ولی ممکن است N واقعاً مرکب باشد.)
- witness (شاهد) برای مرکب‌بودن: عدد a که نشان می‌دهد N قطعاً مرکب است (مثلاً $a^{(N-1)} \bmod N \neq 1$ ).
- strong witness(شاهد قوی): نسخه‌ی قوی‌تر که زنجیره‌ی توان‌ها را چک می‌کند.
- strong liar: عدد a که N مرکب است ولی الگوریتم با آن a اشتباه prime می‌گوید.
- هدف احتمالاتی: کاری کنیم"liar ها" کم باشند ⇒ با چند بار تکرار خطا نمایی کم شود.

`x0=a^u` → `x1=x0^2` → … → `x_{r}=a^{N-1}` → `N-1 = 2^r * u` (فرد u)

# الگوریتم ۹٫۳۴ – تولید یک عدد اول تصادفی nبیتی

- ورودی: n
- خروجی: یک عدد اول nبیتی (تقریباً یکنواخت)
- برای i از ۱ تا ۳ n^۲ :
- ۱۰) یک رشته p' به‌صورت یکنواخت از {۰,۱} ^ {n−۱} بردار
- ۲۰) p = ۱ || p’ (بیت اول را ۱ می‌گذاریم تا عدد n بیتی شود)
- ۳۰) تست میلر–رابین را روی p با پارامتر n^۱ اجرا کن
- ۴۰) اگر خروجی “prime” بود، p را برگردان
- اگر تا آخر پیدا نشد: fail
- نکتهٔ ارائه‌ای:
- - شرطی بر این‌که خروجی واقعاً prime باشد، p یک “عدد اول n بیتی تقریباً یکنواخت” است.
- - بهینه‌سازی رایج در عمل: بیت آخر را ۱ می‌گذارند تا p فرد باشد (سرعت بیشتر).

# ساختن یک تست ساده‌ی اول‌بودن (الگوریتم ۹٫۳۵)

ایده این است:

یک عدد a را به‌طور یکنواخت از مجموعه‌ی ۱ ... N−۱ انتخاب می‌کنیم و مقدار a^(N−۱) mod N را حساب می‌کنیم.

اگر نتیجه مساوی ۱ نباشد، یعنی:

a^(N−۱) ≠ ۱ (mod N)

آن‌وقت N نمی‌تواند اول باشد و حتماً مرکب است.

اگر نتیجه مساوی ۱ شود، این تست را برای این a پاس کرده‌ایم، ولی هنوز ۱۰۰٪ مطمئن نیستیم که N اول است.

کتاب این ایده را به‌صورت رسمی در الگوریتم ۹٫۳۵ – Primality testing: first attempt می‌نویسد و تست را t بار برای مقادیر تصادفی مختلف a تکرار می‌کند.

# مثال عددی برای فهم تست

عدد اول: $N = ۷$ و $a = ۳$

$۳^۶ = ۷۲۹$

و اگر آن را بر ۷ تقسیم کنیم، باقیمانده‌ی آن ۱ است؛ یعنی:

$$۳^۶ \equiv ۱ \pmod ۷$$

پس $N = ۷$ برای این a تست را پاس می‌کند.

عدد مرکب: $N = ۱۵$ و $a = ۲$

اگر $۲^{۱۴}$ را حساب کنیم و بر ۱۵ تقسیم کنیم، باقیمانده‌ی آن ۴ می‌شود؛ یعنی:

$$۲^{۱۴} \equiv ۴ \pmod ۱۵$$

که با ۱ فرق دارد، پس همین یک محاسبه کافی است تا بفهمیم ۱۵ قطعاً مرکب است.

# چرا تست را چند بار تکرار می‌کنیم؟

اگر N اول باشد، برای **همه‌ی** aهای ۱ ... N−۱ داریم:

$$a^{(N-1)} \equiv 1 \pmod N$$

پس این تست هیچ‌وقت اشتباه نمی‌گوید «مرکب».

اما اگر N مرکب باشد، معمولاً برای **بخش بزرگی** از انتخاب‌های a رابطه‌ی بالا برقرار نیست، یعنی:

$$a^{(N-1)} \neq 1 \pmod N$$

چنین aهایی را «شاهد» مرکب‌بودن N می‌نامیم.

با تکرار تست برای t عدد تصادفی، شانس این‌که هیچ شاهدی انتخاب نشود و به‌اشتباه بگوییم prime ، خیلی کم می‌شود و هرچه tبیشتر باشد، این احتمال خطا کوچک‌تر می‌شود.


آزمون میلر–رابین ۱
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# پیچیدگی زمانی و قابل‌اجرا بودن تست

دو کار اصلی در این تست داریم:

محاسبه‌ی a^(N−۱) mod N که با الگوریتم‌های استاندارد مثل روش square-and-multiply در زمان چندجمله‌ای نسبت به طول بیت‌های N انجام می‌شود.

انتخاب یک a یکنواخت از بازه‌ی ۱ ... N−۱ و محاسبه‌ی gcd(a, N) (اگر لازم شود) که آن هم در زمان چندجمله‌ای قابل انجام است.

بنابراین این تست از نظر تئوری **سریع** است و با انتخاب مناسب t می‌توانیم احتمال خطای آن را تا هر حد دلخواه کوچک کنیم.


۱ آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# رفتار کلی الگوریتم نسبت به N

***ALGORITHM 9.35***
**Primality testing – first attempt**

**Input:** Integer $N$ and parameter $1^t$
**Output:** A decision as to whether $N$ is prime or composite

**for** $i = 1$ to $t$:
  $a \leftarrow \{1, \ldots, N-1\}$
  **if** $a^{N-1} \neq 1 \bmod N$ **return** "composite"
**return** "prime"

الگوریتمی که تا الان ساختیم این‌طوری عمل می‌کند:

اگر N واقعاً **اول** باشد، برای همه‌ی انتخاب‌های a در بازه‌ی ۱ ... N−۱ داریم: a^(N−۱) ≡ ۱ (mod N)

پس هیچ‌وقت شرط "a^(N−۱) ≠ ۱ (mod N) برقرار نمی‌شود و الگوریتم **همیشه** خروجی prime می‌دهد.

اگر N **مرکب** باشد، هر وقت در یکی از تکرارها عددی a پیدا کنیم که برایش: a^(N−۱) ≠ ۱ (mod N)

فوری نتیجه می‌گیریم N مرکب است و الگوریتم خروجی composite می‌دهد.

یعنی:

عدد مرکب را فقط وقتی از دست می‌دهیم که در تمام تکرارها، هیچ a ای با a^(N−۱) ≠ ۱ (mod N) انتخاب نشود.


۱ آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# چرا فقط a های داخل Z*_N برای ما مهم‌اند؟

کتاب یک نکته‌ی ظریف می‌گوید اگر a در Z*_N نباشد یعنی gcd(a, N) ≠ ۱، آن‌وقت:

gcd(a, N) = d > ۱

a^(N−۱) هم حتماً با N همین مقسوم‌علیه مشترک d را دارد،پس gcd(a^(N−۱), N) ≠ ۱ می‌شود.

اما عددی که با N مقسوم‌علیه مشترک بزرگ‌تر از ۱ دارد، نمی‌تواند هم‌مانده‌ی ۱ باشد،چون ۱ همیشه با N هم‌اول است (مقسوم‌علیه مشترک ۱ دارد).

پس اگر gcd(a, N) ≠ ۱، محال است a^(N−۱) ≡ ۱ (mod N) برقرار باشد.

در نتیجه چنین aهایی خودبه‌خود شاهد مرکب بودن N هستند و حتی لازم نیست زیاد به آن‌ها فکر کنیم.برای همین کتاب می‌گوید:

«از این به بعد توجه‌مان را محدود می‌کنیم به a هایی که در Z*_N هستند»؛ چون اگر a در Z*_N نباشد، همین که gcd(a, N) ≠ ۱ را ببینیم، می‌فهمیم N مرکب است.

مثال خیلی ساده : بگذار N = ۱۵ و a = ۵.

gcd(۵, ۱۵) = ۵ ≠ ۱، پس a در Z*_۱۵ نیست.

هر توانی از ۵ (مثلاً ۵^۲ = ۲۵، ۵^۳ = ۱۲۵، ...) هم هنوز بر ۱۵ بخش‌پذیر است یا دست‌کم مقسوم‌علیه مشترک ۵ با ۱۵ دارد. پس ۵^(۱۴) mod ۱۵ نمی‌تواند برابر ۱ شود، چون ۱ با ۱۵ هیچ مقسوم‌علیه مشترکی جز ۱ ندارد.

یعنی همین a = ۵ خودش نشان می‌دهد ۱۵ مرکب است؛ اصلاً نیازی به قضیه‌ی فرما هم نیست.

# تعریف witness (شاهد مرکب‌بودن)

کتاب حالا برای اعداد داخل $Z^*_N$ یک اسم می‌گذارد:

فرض کن $a$ در $Z^*_N$ باشد

(یعنی $1 \le a \le N-1$ و $\gcd(a, N) = 1$).

اگر برای این $a$ داشته باشیم:

$$a^{(N-1)} \neq 1 \pmod N$$

آن‌وقت می‌گوییم $a$ یک **شاهد مرکب‌بودن** $N$ است، یا به‌اختصار یک **witness** است.

معنی‌اش ساده است:

این $a$ دارد «شهادت می‌دهد» که $N$ نمی‌تواند عدد اول باشد، چون برای عدد اول باید

$$a^{(N-1)} \equiv 1 \pmod N$$

باشد و اینجا این اتفاق نمی‌افتد.

# شهود احتمالی: چرا امیدواریم witness زیاد باشد؟

حالا که witnessرا تعریف کردیم، امید منطقی ما این است:

وقتی N مرکب باشد، تعداد زیادی witness وجود داشته باشد.

اگر بخش بزرگی از Z*_N شاهد باشد، آن‌وقت وقتی الگوریتم چند بار به‌طور تصادفی a را انتخاب می‌کند،

احتمال این‌که هیچ witness ی انتخاب نشود، خیلی کم می‌شود.

نتیجه: برای اعداد مرکب، الگوریتم با احتمال بالا در یکی از تکرارها witness را می‌بیند و می‌گوید composite

کتاب در این نقطه می‌گوید:

این شهود درست است، به شرطی که حداقل یک witness وجود داشته باشد.

برای این‌که نشان بدهیم «اگر یک witness وجود داشته باشد، در واقع witnessهای زیادی وجود دارند»، کتاب دو تا لم گروهی (group-theoretic lemmas) معرفی می‌کند و بعد با آن‌ها این ادعا را اثبات می‌کند.

اون دو لم و ادامه‌ی استدلال هنوز تو متنی که دادی نیامده و بعداً می‌آید.


آزمون میلر–رابین ۱
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# نتیجه‌ی اول و دوم – Proposition ۹.۳۶ (زیرگروه بودن H)

در این بخش دو نتیجه از نظریهٔ گروه‌ها لازم داریم.

قضیهٔ کمکی اول می‌گوید: اگر Gیک گروه متناهی باشد و Hیک زیرمجموعهٔ غیرخالی از Gکه تحت عمل گروه بسته است (یعنی هر وقت a و b در H باشند، a·b هم در H باشد)، آن‌وقت H درواقع یک زیرگروه از G است.

به‌خاطر متناهی بودن G، برای هر a در H توان‌های a داخل H می‌مانند و در نهایت به ۱ و$a^{-1}$ می‌رسیم؛ پس H هم عنصر واحد را دارد و هم معکوس هر عضو را، بنابراین شرایط زیرگروه بودن کامل می‌شود.

قضیهٔ کمکی دوم: اگر H یک زیرگروهِ واقعیِ G باشد (یعنی H ≠ G)، آن‌وقت اندازهٔ H حداکثر نصف اندازهٔ G است؛ یعنی |H| ≤ |G|/۲.

ایده این است که عنصری h از G می‌گیریم که در H نیست و مجموعهٔ جدید hH = {h·x | x∈H} را می‌سازیم. ضرب‌کردن در h یک تناظرساز یک‌به‌یک بین H و hH می‌دهد، پس این دو مجموعه هم‌اندازه‌اند. از طرفی نشان می‌دهیم هیچ عضوی مشترک ندارند؛ بنابراین G حداقل به اندازهٔ H به‌اضافهٔ hH عضو دارد، یعنی |G| ≥ ۲|H|، و از این نامساوی نتیجه می‌گیریم |H| ≤ |G|/۲.

# ربط این دو قضیه به آزمون میلر–رابین

این دو قضیه برای تحلیل رفتار آزمون اول‌بودن استفاده می‌شوند.

هدف این است که نشان بدهیم اگر عدد مرکب N حتی یک شاهد داشته باشد؛ یعنی عددی a در $Z^*_N$ وجود داشته باشد که شرط

$$a^{(N-1)} \neq 1 \pmod N$$

را نقض کند، آن‌وقت این فقط یک مورد خاص نیست.

در واقع ثابت می‌شود مجموعهٔ عناصری که شاهد نیستند یک زیرگروهِ واقعی از $Z_N$ است و بنابراین نمی‌تواند بیشتر از نصف کل عناصر باشد.

پس نتیجه می‌گیریم **حداقل نیمی از عناصر $Z_N$ شاهد هستند**.

این نکته برای الگوریتم حیاتی است.

چون در هر مرحله یک a تصادفی انتخاب می‌شود و اگر نصف فضا شاهد باشد، احتمال پیدا کردن شاهد در هر مرحله حداقل ۱/۲ است.

بنابراین احتمال این‌که الگوریتم در t مرحله هیچ شاهدی نبیند و اشتباهی N را prime اعلام کند، حداکثر:

$$2^{(-t)}$$

به این ترتیب، اگر N مرکب باشد و حداقل یک شاهد وجود داشته باشد، با چند تکرار ساده احتمال خطا بسیار کوچک می‌شود.


آزمون میلر–رابین ۱
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# قضیه ۹٫۳۸ – چند تا شاهد داریم؟

اگر N مرکب باشد و حداقل یک شاهد برای مرکب بودنش وجود داشته باشد یعنی عددی a در Z*_N که

a^(N-۱) != ۱ (mod N)

آن‌وقت حداقل نصف عناصر Z*_N شاهد هستند.

ایدهٔ اثبات این است که مجموعهٔ عددهایی که شاهد نیستند را تعریف می‌کنیم:

{ Z*_N | a^(N-۱) = ۱ (mod N) } a در = Bad

این مجموعه تحت ضرب بسته است و ۱ را هم دارد، پس با قضیهٔ کمکی قبلی یک زیرگروه از Z*_N می‌شود.

چون فرض کردیم حداقل یک شاهد هست، Bad نمی‌تواند کل Z*_N باشد؛ زیرگروهِ واقعی است و اندازه‌اش حداکثر نصف است.

پس بقیهٔ عناصر (حداقل نصف Z*_N) شاهد هستند.

# نتیجهٔ احتمالاتی تست – کران خطا

- فرض کن N مرکب است و حداقل یک witness وجود دارد.
- طبق قضیهٔ ۹.۳۸، حداقل نصف عناصر $Z^*_N$ witness هستند.
- اگر $\gcd(a,N)\neq 1$ باشد، همان لحظه مرکب بودن معلوم می‌شود.
- پس در هر تکرار، احتمال گرفتن witness یا خروج از $Z^*_N$ ≥ ۱/۲ است.
- اگر t بار تکرار کنیم، احتمال اینکه هیچ‌کدام رخ ندهد) ≤ $(۱/۲)^t$ است.
- بنابراین احتمال خطای اعلامِ «prime» برای عدد مرکب ≥ $۲^{(-t)}$ است.
- نمونه ⇒ t=۲۰ :خطا ≤ ۱/۱,۰۴۸,۵۷۶ (حدود یک در یک میلیون).
- هرچه t بزرگ‌تر، اعتماد بیشتر (ولی زمان هم بیشتر).
- این تحلیل برای Fermat-test است؛ برای Carmichael کافی نیست.
- برای همین می‌رویم سراغ Miller–Rabin.

# وقتی Miller–Rabin گفت prime، چقدر می‌توانیم اعتماد کنیم؟

$$\Pr(\neg P \mid MR_k) = \frac{\Pr(\neg P \wedge MR_k)}{\Pr(\neg P \wedge MR_k) + \Pr(P \wedge MR_k)}$$
$$= \frac{1}{1 + \frac{\Pr(MR_k|P)}{\Pr(MR_k|\neg P)} \frac{\Pr(P)}{\Pr(\neg P)}}$$
$$= \frac{1}{1 + \frac{1}{\Pr(MR_k|\neg P)} \frac{\Pr(P)}{1-\Pr(P)}}$$

- دو رویداد تعریف کن:
- P : "عدد واقعاً اول است"
- MR_k: "k بار Miller–Rabin را پاس کرده"
- چیزی که ما می‌خواهیم : Pr(¬P | MR_k) (یعنی احتمال اینکه با وجود پاس کردن، مرکب باشد)
- این مقدار با قضیه بیز نوشته می‌شود
- چون برای عدد اول Miller–Rabin همیشه پاس می‌شود Pr(MR_k | P) = ۱ :
- اگر برای عدد مرکب، احتمال پاس شدن هر بار ≥ ۱/۲ باشد Pr(MR_k | ¬P) ≤ ۲^(−k)
- نتیجه شهودی: با k تکرار، خطا "نمایی" کوچک می‌شود.

"P یعنی اول بودن" و "MR_k یعنی k بار پاس شدن". بدون این توضیح، برای مخاطب تبدیل می‌شود به «فرمول ترسناکِ بی‌ربط».


آزمون میلر–رابین ۱
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# مشکل اعداد کارمایکل

این تحلیل یک «اما» دارد:

بی‌نهایت عدد مرکب N وجود دارند که هیچwitness ی ندارند؛ یعنی برای تمامa های مناسب، همچنان:

$$a^{(N-1)} = 1 \pmod N$$

صادق است.

این اعداد را «اعداد کارمایکل (Carmichael numbers) می‌نامند.

برای این Nها، تست قبلی همیشه مثل عدد اول رفتار می‌کند و هرچقدر t را زیاد کنیم باز هم اشتباه می‌گوید prime

به‌همین دلیل لازم است تست را قوی‌تر کنیم.

# مثال کارمایکل: ۵۶۱ چرا خطرناک است؟

۵۶۱ عددی مرکب است چون ۳×۱۱×۱۷.

اما یک عدد «کارمایکل» است و برای هر a که با ۵۶۱ نسبت به هم اول باشد، داریم: a^۵۶۰ ≡ ۱ (mod ۵۶۱).

پس تست ساده‌ی فرما ممکن است همیشه گول بخورد و ۵۶۱ را “prime ” اعلام کند.

دلیلش این است که بعضی عددهای مرکب طوری ساخته شده‌اند که رفتار عدد اول را تقلید کنند.

Miller–Rabin فقط نتیجه‌ی نهایی را چک نمی‌کند، مسیرِ توان‌ها را هم بررسی می‌کند.

در این تست n−۱ را به شکل ۲ s × d می‌نویسیم که d فرد است.

بعد به جای فقط a^(n−۱)، این زنجیره را نگاه می‌کنیم: a^d، a^(۲d)، a^(۴d)، ...همه mod n). اگر n اول باشد، این زنجیره باید خیلی محدود رفتار کند و در نهایت فقط به ۱ یا −۱ برسد.

ولی در عددهای مرکب (مثل ۵۶۱) معمولاً وسط راه یک مقدار “غیرعادی” می‌بینیم که نشان می‌دهد عدد مرکب است.

برای همین Miller–Rabin خطاهای مدل فرما را خیلی کمتر می‌کند و در عمل بسیار قابل‌اعتمادتر است.


آزمون میلر–رابین ۱
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# ایدهٔ تقویت تست – شکستن N−۱ و دنبالهٔ توان‌ها

برای ساخت نسخهٔ قوی‌تر، N−۱ را به شکل زیر می‌نویسیم:

N - ۱ = ۲^r * u که u فرد است و r >= ۱

حساب کردن r و u از روی N آسان است (N را آن‌قدر بر ۲ تقسیم می‌کنیم تا فرد شود).

این فرض یعنی N عددی فرد است؛ برای N زوج که تشخیص مرکب بودن ساده است.

در نسخهٔ قبلی فقط a^(N-۱) = a^(۲^r * u) را بررسی می‌کردیم.

حالا به **دنبالهٔ کامل توان‌ها** نگاه می‌کنیم:

a^u , a^(۲u) , a^(۴u) , ... , a^(۲^r * u) (همه پیمانهٔ N)

هر جمله مربع جملهٔ قبلی است؛

اگر یکی از این‌ها ۱ شود، تمام جملات بعدی هم برابر ۱ خواهند بود.

**مثال کوتاه:**

برای N = ۲۱ داریم:

r = ۲ , u = ۵ ⇒ ۵ * ۲^۲ = ۲۰ = ۲۱ - ۱

اگر a = ۲ بگیریم، دنبالهٔ ۵^۲ , ۱۰^۲ , ۲۰^۲ (mod ۲۱) را حساب می‌کنیم و می‌بینیم که الگویش مثل اعداد اول نیست؛ پس ۲ برای ۲۱ یک شاهد قوی می‌شود (این را می‌توانی روی کاغذ حساب کنی).

# تعریف «شاهد قوی» و رابطه با شاهد معمولی

کتاب حالا a را در Z*_N یک **شاهد قوی مرکب بودن N** می‌نامد (strong witness) اگر:

مقدار اول دنباله، یعنی a^u mod N برابر ۱ نشود؛

در هیچ‌کدام از مراحل میانی، دنباله به «الگوی مجاز» اعداد اول نرسد (جایی که قبل از رسیدن به ۱، مقدار −۱ mod N ظاهر شود و بعد مربعش ۱ شود).

به زبان ساده: اگر رفتار دنباله‌ی

a^u , a^(۲u) , ... , a^(۲^r * u)

مثل چیزی که از یک عدد اول انتظار داریم نباشد، a یک شاهد قوی برای مرکب بودن N است.

کتاب نشان می‌دهد:

اگر a شاهد قوی **نباشد**، دنباله در نهایت به الگوی مجاز می‌رسد و نتیجه می‌دهد a^(N-۱) = ۱ (mod N)؛ یعنی a اصلاً witness معمولی هم نیست.

بنابراین هر witness معمولی حتماً witness قوی هم هست؛ در نتیجه تعداد شاهدهای قوی حداقل به اندازهٔ شاهدهای معمولی است.

این نکته مهم است چون وقتی می‌رویم سراغ نسخهٔ نهایی الگوریتم Miller–Rabin، همیشه با «شاهدهای قوی» کار می‌کنیم و همان تحلیل احتمال خطا (از نوع $۲^{-t}$)برای آن‌ها هم برقرار می‌ماند.


آزمون میلر–رابین ۱
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# ریشهٔ دومِ ۱ پیمانهٔ N و نتیجهٔ ۹٫۳۹

می‌گوییم عدد x ریشهٔ دومِ ۱ پیمانهٔ N است اگر:

$$x^2 \equiv 1 \pmod N$$

نتیجهٔ ۹٫۳۹ می‌گوید: اگر N یک عدد اولِ فرد باشد، تنها ریشه‌های دومِ ۱ پیمانهٔ N همین دو تا هستند:

$$x \equiv 1 \pmod N$$

$$x \equiv -1 \pmod N$$

ایدهٔ اثبات: اگر $x^2 \equiv 1 \pmod N$، آن‌وقت:

$$x^2 - 1 \equiv 0 \pmod N$$

$$(x - 1)(x + 1) \equiv 0 \pmod N$$

چون N اول است، باید N یکی از این دو ضرب‌اندر را بخش‌پذیر کند؛ یعنی یا $x \equiv 1 \bmod(N)$ یا $x \equiv -1 \pmod N$ و حالت سومی وجود ندارد.

# چرا برای N اول، strong witness وجود ندارد؟

فرض کن N یک عدد اولِ فرد است و $N-1 = 2^r * u$ با u فرد.

یک a دلخواه در $Z^*_N$ بگیر و دنبالهٔ زیر را نگاه کن:

$$a^u , a^{(2u)} , \dots , a^{(2^r * u)} \pmod N$$

یک i مینیمم وجود دارد که :

$$a^{(2^i * u)} \equiv 1 \pmod N$$

چون در آخر، $a^{(2^r * u)} = a^{(N-1)} \equiv 1 \pmod N$ است، حتماً چنین iای بین ۰ و r پیدا می‌شود.

دو حالت داریم:

اگر $i = 0$، یعنی $a^u \equiv 1 \pmod N$؛ در این صورت a اصلاً strong witness محسوب نمی‌شود.

اگر $i > 0$، بگذار$x = a^{(2^{(i-1)} * u)}$ ؛ آن‌وقت :

$$x^2 = a^{(2^i * u)} \equiv 1 \pmod N$$

پس x یک ریشهٔ دومِ ۱ پیمانهٔ N است.با نتیجهٔ ۹٫۳۹، تنها گزینه‌ها $x \equiv 1$ یا $x \equiv -1 \pmod N$ است. چون i کمترین مقدار بود، x نمی‌تواند ۱ باشد، پس $x \equiv -1 \pmod N$ و دنباله جایی شکل «... , -۱ , ۱» می‌گیرد؛ این همان الگویی است که strong witness را رد می‌کند. در هر دو حالت، a strong witness نیست.پس وقتی N یک عدد اولِ فرد است، **هیچ strong witness ی وجود ندارد.**

# مثال

مثال ( N = ۷، عدد اول)

$r = ۱ \Rightarrow N-۱ = ۶ = ۲\times۳$، $u = ۳$

یک پایه انتخاب می‌کنیم: $a = ۳$

مرحله ۱) $x_۰ = a^u \bmod N$

$x_۰ = ۳^۳ \bmod ۷ = ۲۷ \bmod ۷ = ۶ \Rightarrow ۶ \equiv -۱ \pmod ۷$

پس دنباله:

$(x_۰ , x_۱) = ( -۱ , ۱ )$

چون برای عدد اول دیدن ۱- در دنباله مجاز است،

a=۳ برای “strong witness” N=۷ نیست (یعنی الگوریتم N را composite اعلام نمی‌کند).

# prime power و هدف جدید

یک عدد مرکب N را «توانِ اول» می‌نامیم اگر:

N = p^r

برای یک عدد اول p و عدد صحیح r >= ۲؛ مثلا ۲۷ = ۳^۳ یا ۴۹ = ۷^۲.

حالا هدف کتاب این است که نشان دهد:

اگر N عدد مرکبِ فردی باشد که توانِ اول نیست، آن‌وقت این عدد N خیلی strong witness دارد.

قضیهٔ ۹٫۴۰ می‌گوید:

اگر N عددی فرد و مرکب باشد و نتوان آن را به شکل p^r نوشت (یعنی prime power نباشد)،

آن‌وقت حداقل نصف عناصر Z*_N strong witness هستند و مرکب بودن N را لو می‌دهند.

از نظر شهودی، این قضیه می‌گوید: برای این نوع N،

فضای عناصری که الگوریتم Miller–Rabin را قانع می‌کنند که « N مرکب است» خیلی بزرگ است

و انتخاب تصادفی a تقریباً همیشه به یک strong witness می‌خورد.


۱ آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# ایدهٔ اثبات ۹٫۴۰ با مجموعه‌های Bad

برای اثبات، کتاب دوباره از ایدهٔ «زیرگروهِ کوچک» استفاده می‌کند.

Bad = مجموعهٔ عناصری از Z*_N که **strong witness نیستند**.

سپس یک مجموعهٔ کوچک‌تر به نام Bad' تعریف می‌شود از عناصری که در گامِ مشخصی از دنباله، a^(۲^i * u) ≡ ۱ (mod N) دارند.

کتاب نشان می‌دهد:

Bad' زیرمجموعه‌ای از Bad است؛

Bad' یک زیرگروهِ واقعی از Z*_N است؛ بنابراین با نتیجهٔ قبلی:

|Bad'| ≤ |Z*_N| / ۲

از ۱ نتیجه می‌گیریم |Bad| هم حداکثر همین‌قدر است، چون Bad' داخل Bad است.

پس **حداقل نصف عناصر** Z*_N strong witness هستند.

(خود Bad لزوماً زیرگروه نیست؛ برای همین با Bad' کار می‌کنیم.)

# ایدهٔ اثبات ۹٫۴۰ با مجموعه‌های Bad و Bad۲

N عددی فرد و مرکب است و prime power نیست. می‌نویسیم:

$N-1 = 2^r \cdot u$ (u فرد)

Bad = همهٔ aهای داخل $Z^*_N$ که **strong witness نیستند**

iرا این‌طور در نظر می‌گیریم:

i بزرگ‌ترین عدد در $\{0,\dots,r-1\}$ که برای یک a در Bad داشته باشیم:

$a^{(2^i \cdot u)} \equiv -1 \pmod N$

Bad۲ را تعریف می‌کنیم:

$\text{Bad۲} = \{ a \text{ در } Z^*_N \mid a^{(2^i \cdot u)} \equiv 1 \text{ یا } -1 \pmod N \}$

سه ادعا:

ادعا ۹٫۴۱: $\text{Bad} \subseteq \text{Bad۲}$

ادعا ۹٫۴۲: Bad۲ یک زیرگروه از $Z^*_N$ است

ادعا ۹٫۴۳: Bad۲ زیرگروهِ واقعی است $\Rightarrow |\text{Bad۲}| \le |Z^*_N|/2$

نتیجه: چون $|Z_N|//2 \le |\text{Bad۲}| \le |\text{Bad}|$ و $\text{Bad} \subseteq \text{Bad۲} \Rightarrow$

*پس حداقل نصف عناصر* $Z_N$ strong witness هستند.

# ادعا ۹٫۴۱ – Bad۲ زیرمجموعهٔ Bad است

- Badیعنی پایه‌هایی که Miller–Rabin را گول می‌زنند(خروجی"prime").
- Bad۲ یعنی پایه‌هایی که برای یک i داریم.$a^{(2^i \cdot u)} = \pm 1 \pmod N$
- اگر a در Bad باشد، دنباله‌اش یکی از الگوهای "غیرشاهد" را دارد.
- در هر دو الگو، حتماً یک مرحله داریم که مقدار دنباله ۱± می‌شود.
- یعنی برای همان i: $a^{(2^i \cdot u)} \equiv \pm 1 \pmod N$.
- پس a عضو Bad۲ است.
- نتیجه : Bad۲ ⊆ Bad (نه برعکس)
- همه‌ی این‌ها داخل $Z^*_N$ تعریف می‌شود، نه کل $Z_N$.
- اگر Bad۲ زیرگروهِ واقعی باشد، اندازه‌اش ≥ نصف $Z^*_N$ است.
- پس خطای t تکرار ≥ ۲ به توان منفی t

# ادعا ۹٫۴۲ – Bad۲ یک زیرگروه از Z*_N است

- عضو همانی:

۱ در Bad' است چون $1^{(2^{i \cdot u})} \equiv 1 \pmod N$

- بسته بودن تحت ضرب: اگر a و b در Bad' باشند، داریم:

$a^{(2^{i \cdot u})} \equiv \pm 1 \pmod N$

$b^{(2^{i \cdot u})} \equiv \pm 1 \pmod N$

پس:

$(ab)^{(2^{i \cdot u})} \equiv a^{(2^{i \cdot u})} \cdot b^{(2^{i \cdot u})} \equiv \pm 1 \pmod N$

ab ⇒در Bad' است

- نتیجه : Bad' ناتهی و تحت ضرب بسته است. چون Z*_N متناهی است،

این یعنی Bad' تحت معکوس‌ها هم بسته می‌شود Bad' ⇒ زیرگروه است.

# ادعا ۹٫۴۳ – Bad۲ زیرگروهِ واقعی است (کل Z*_N نیست)

- چون N prime power نیست، می‌توان نوشت:

N = N۱ · N۲ و gcd(N۱,N۲)=۱

- با CRT هر a در Z_N متناظر یک زوج (a۱,a۲) در Z*{N۲} × Z{N۱} است.
- طبق تعریف i ، یک a در Bad۲ داریم که:

a^(۲^i·u) ≡ −۱ (mod N)

پس a۱^(۲^i·u) ≡ −۱ (mod N۱) و: a۲^(۲^i·u) ≡ −۱ (mod N۲)

- حالا با CRT عدد b را طوری می‌سازیم که:

b ↔ (a۱ , ۱)

آنگاه:

b^(۲^i·u) ↔ (−۱ , ۱)

که نه +۱ است و نه −۱ در پیمانه N

⇒ b در Bad۲ نیست

- پس Bad۲ ≠ Z_N ⇒ Bad۲ زیرگروهِ واقعی ⇒ |Bad۲| ≤ |Z_N|/۲
- و چون Bad ⊆ Bad۲ ⇒ |Bad| ≤ |Z*_N|/۲

# مثال

مثلاً N = ۲۱ = ۳ * ۷ یک عدد فرد است و prime power نیست.

طبق قضیهٔ ۹٫۴۰، حداقل نصف عناصر Z*_۲۱ باید strong witness باشند.

اگر چند عدد مثل a = ۲ , ۵ , ۱۱ را امتحان کنی و دنبالهٔ

a^u , a^(۲u) , a^(۴u) (mod ۲۱)

را حساب کنی ( با u = ۵ , r = ۲) ⇒ ۵ * ۲^۲ = ۱-۲۱)، می‌بینی بسیاری از این a ها دنباله‌ای می‌سازند که شبیه رفتار اعداد اول نیست و در نتیجه strong witness می‌شوند.

این مثال عددی نشان می‌دهد چرا «فضای strong witnessها» برای چنین N هایی واقعاً بزرگ است.

# ادامه مباحث

تا این‌جا این زنجیره‌ی فکری رو ساختیم:

اول ایده‌ی ساده‌ی فرما رو داشتیم: اگر N اول باشد، برای هر a در $Z^*_N$ داریم

$a^{(N-1)} \equiv 1 \pmod N$؛

اگر برای یک a این برقرار نباشد، به آن a می‌گوییم شاهد مرکب بودن N.

با نتایج گروهی نشان دادیم: اگر حتی یک شاهد وجود داشته باشد، حداقل نصف عناصر $Z^*_N$ شاهد هستند (قضیهٔ ۹٫۳۸). بنابراین در تست تصادفی ساده، احتمال خطای بعد از t تکرار حداکثر $2^{-t}$ است.

مشکل: اعداد کارمایکل؛ بی‌نهایت عدد مرکبی که برایشان هیچ witnessی نیست و تست فرما را کاملاً گول می‌زنند.

برای حل این مشکل، $N-1 = 2^r * u$ را با u فرد نوشتیم و به دنبالهٔ

$a^u , a^{(2u)} , \dots , a^{(2^r * u)}$ (پیمانهٔ N) نگاه کردیم.

با توجه به شکل مجاز این دنباله برای اعداد اول، تعریف کردیم که یک a در $Z^*_N$ اگر این دنباله «رفتار غیرمجاز» داشته باشد، شاهد قوی (strong witness) است.

ثابت شد:

اگر N اولِ فرد باشد، اصلاً strong witness وجود ندارد؛

اگر N مرکبِ فرد و prime power نباشد، حداقل نصف عناصر $Z^*_N$ strong witness هستند (قضیهٔ ۹٫۴۰).

# الگوریتم میلر–رابین (۹٫۴۴)


۱ آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش


***ALGORITHM 9.44***
**The Miller–Rabin primality test**

**Input:** Integer $N > 2$ and parameter $1^t$
**Output:** A decision as to whether $N$ is prime or composite

**if** $N$ is even, **return** "composite"
**if** $N$ is a perfect power, **return** "composite"
**compute** $r \geq 1$ and $u$ odd such that $N - 1 = 2^r u$
**for** $j = 1$ to $t$:
  $a \leftarrow \{1, \ldots, N-1\}$
  **if** $a^u \neq \pm 1 \bmod N$ and $a^{2^i u} \neq -1 \bmod N$ for $i \in \{1, \ldots, r-1\}$
    **return** "composite"
**return** "prime"

N>۲، پارامتر t (تعداد تکرار)

اگر N زوج است ← composite

اگر N یک perfect power است ← composite

بنویس N−۱ = ۲^r · u (u فرد)

برای j=۱..t : عدد تصادفی a از {۲..N−۲}

اگر gcd(a,N)≠۱ ← composite

x = a^u mod N

اگر x=۱ یا x=N−۱ ← این دور "قبول"

وگرنه r−۱ بار: x = x^۲ mod N، اگر N−۱ شد "قبول"

اگر هیچ‌وقت N−۱ نشد ← composite؛ اگر همه قبول شدند ← prime با خطای ≤ ۲^(−t)

# مثال عددی کامل از آزمون میلر–رابین

- عدد مرکب نمونه:

$N = ۲۲۱ = ۱۳ \times ۱۷$

- مرحلهٔ تجزیهٔ : $N - ۱$

$N - ۱ = ۲۲۰ = ۲^۲ \times ۵۵$

پس $r = ۲$ و $u = ۵۵$ ($u$ فرد است)

- یک پایهٔ تصادفی انتخاب می‌کنیم، مثلاً:

$a = ۲$

- دنبالهٔ توان‌ها (پیمانهٔ ۲۲۱):

- $x_۰ = a^u = ۲^{۵۵} \equiv ۱۲۸ \pmod{۲۲۱}$

۱۲۸نه ۱ است و نه ۲۲۰ (= −۱ پیمانهٔ ۲۲۱).

- $x_۱ = a^{(۲u)} = ۲^{۱۱۰} \equiv ۳۰ \pmod{۲۲۱}$

باز هم $۳۰ \neq ۲۲۰$.

- $x_۲ = a^{(۴u)} = ۲^{۲۲۰} \equiv ۱ \pmod{۲۲۱}$.

- برای عدد اول، باید در این دنباله از جایی مقدار ۲۲۰ (یعنی −۱ پیمانهٔ ۲۲۱) دیده شود یا از همان ابتدا به ۱ برسیم.
- این‌جا دنباله با ۱ تمام می‌شود ولی هرگز ۲۲۰ ظاهر نمی‌شود؛

بنابراین $a = ۲$ یک strong witness است و آزمون میلر–رابین عدد $N = ۲۲۱$ را «مرکب» اعلام می‌کند.

# چرا این الگوریتم درست کار می‌کند

کتاب سه حالت را جدا می‌کند:

**N اولِ فرد:**

قبلاً نشان دادیم در این حالت هیچstrong witness ی وجود ندارد.

رفتار دنبالهٔ توان‌ها همیشه مطابق الگوی مجاز است و الگوریتم در هیچ تکراری به شرط composite نمی‌رسد.

پس خروجی همیشه prime است و خطایی در جهت «مرکب گفتنِ عدد اول» نداریم.

**N زوج یا perfect power :**

خط‌های اول الگوریتم این‌ها را مستقیم می‌گیرند و همیشه composite می‌گویند؛

برای این موارد اصلاً وارد بخش تصادفی نمی‌شویم.

**N مرکبِ فرد و prime power نیست:**

این‌جا از قضیهٔ ۹٫۴۰ استفاده می‌کنیم: حداقل نصف عناصر Z*_N strong witness هستند.

علاوه بر این، اگر a اصلاً در Z*_N نباشد (یعنی gcd(a, N) ≠ ۱)، همان اول کار هم می‌توانیم مرکب بودن را بفهمیم.

پس در هر تکرار، احتمال این‌که یا strong witness انتخاب شود یا aای بیرون از Z*_N پیدا کنیم، **حداقل ۱/۲** است.

بنابراین احتمال این‌که در هیچ‌یک از t تکرارها به شرط composite نرسیم، حداکثر:

$(t-)^{۲}$

این یعنی اگر N از این نوع مرکب باشد، الگوریتم فقط با احتمال حداکثر (t-) ۲ ^ ۲ اشتباه می‌کند و می‌گوید prime

# یک مثال عددی ساده برای میلر–رابین

N = ۲۱ (مرکب: ۳ * ۷).

N−۱ = ۲۰ = ۲^۲ * ۵ ⇒ r = ۲ و u = ۵.

یک a تصادفی مثل a = ۲ انتخاب می‌کنیم.

گام‌ها:

x = ۲^۵ mod ۲۱ ⇒ ۲^۵ = ۳۲ و mod ۲۱ = ۱۱ ⇒ x = ۱۱. ۳۲

نه ۱ است، نه N−۱ = ۲۰ ⇒ وارد حلقهٔ داخلی می‌شویم.

یک‌بار حلقه (چون r−۱ = ۱):

x = x^۲ mod ۲۱ = ۱۱^۲ mod ۲۱ = ۱۲۱ mod ۲۱ = ۱۲۱−۱۰۵ = ۱۶.

x نه ۱ است و نه N−۱ ⇒ حلقه تمام می‌شود بدون دیدن ۲۰

پس الگوریتم فوراً N = ۲۱ را **composite** اعلام می‌کند.

این مثال برای کلاس قابل‌حساب است و دقیقاً نشان می‌دهد چطور الگوریتم از دنبالهٔ توان‌ها استفاده می‌کند تا خیلی سریع به مرکب بودن برسد.

# تعریف – GenModulus تولیدکنندهٔ مدول

کتاب اول یک الگوریتم به‌نام GenModulus تعریف می‌کند.

این الگوریتم ورودی‌اش 1^n است (یعنی عدد n به‌صورت یک رشته از n تا ۱؛ فقط برای این‌که طول ورودی برابر n شود) و خروجی‌اش سه‌تا عدد است:

N , p , q

طوری که N = p * q

و هر کدام از p و q با احتمال خیلی زیاد، **یک عدد اول n بیتی** هستند.

راه طبیعی ساختن چنین GenModulus این است که:

دو عدد اول n بیتی تقریباً یکنواخت انتخاب کنیم، p و q،

بعد N = p * q را حساب کنیم و سه‌تایی (N , p , q) را برگردانیم.

این همون کاریه که تو RSA هم می‌کنیم: دو اول بزرگ می‌گیریم و ضرب‌شان می‌شود مدول.


آزمون میلر–رابین

۲ فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# آزمایش فاکتورگیری – Factor(A, GenModulus, n)

برای سنجیدن توانایی یک الگوریتم A در فاکتورگیری، کتاب این «آزمایش» را تعریف می‌کند Factor(A, GenModulus), (n)

GenModulus(۱^n) را اجرا می‌کنیم و خروجی (N , p , q) را می‌گیریم.

فقط عدد N را به الگوریتم A می‌دهیم.

الگوریتم A باید دو عدد p’ و q' بدهد که هر دو بزرگ‌تر از ۱ باشند.

خروجی آزمایش ۱ است اگر p' * q' = N باشد، و در غیر این صورت ۰ است.

اگر خروجی ۱ شود، در حالت معمول یعنی p’ و q' همان p و q اصلی‌اند (مگر این‌که خود p یا q مرکب بوده باشند، که احتمال این اتفاق طبق تعریف GenModulus ناچیز است).


آزمون میلر–رابین
۲ فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# تعریف رسمی «سخت بودن فاکتورگیری» نسبی به GenModulus

حالا می‌توانیم سختی فاکتورگیری را دقیق تعریف کنیم.

می‌گوییم:

**فاکتورگیری نسبت به GenModulus سخت است**

اگر برای هر الگوریتم تصادفی با زمان چندجمله‌ای A

یک تابع ناچیز negl(n) وجود داشته باشد طوری که

$$\Pr[\mathsf{Factor}_{\mathcal{A},\mathsf{GenModulus}}(n) = 1] \leq \mathsf{negl}(n).$$

یعنی هر الگوریتم معقول و سریع A، اگر روی N هایی که GenModulus می‌سازد امتحانش کنیم، فقط با احتمال **ناچیز** (خیلی کوچک، مثلاً کوچک‌تر از ۱ روی هر توان چندجمله‌ای از n) موفق می‌شود که N را درست فاکتور کند.

یک‌جمله‌ای ساده:

هیچ الگوریتم سریع عمومی‌ای وجود ندارد که بتواند مدول‌های ساخته‌شده توسط GenModulus را با احتمال قابل توجه فاکتور کند.


آزمون میلر–رابین
۲ فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# خود «فرض فاکتورگیری» چیه؟

آزمایش فاکتورگیری Factor(A, GenModulus, n)

مرحله ۱) اجرای تولیدکننده

GenModulus(۱^n) را اجرا می‌کنیم و می‌گیریم:

(N, p, q) که N = p×q و p,q اول‌های n بیتی‌اند.

مرحله ۲) حمله/الگوریتم A

به A فقط N را می‌دهیم (نه p نه q).

A باید دو عدد p’ و q' > ۱ خروجی بدهد.

مرحله ۳) معیار موفقیت

خروجی آزمایش = ۱ اگر N = p'×q'

در غیر این صورت خروجی = ۰

تعریف “سخت بودن فاکتورگیری نسبت به GenModulus” :

برای هر الگوریتم چندجمله‌ای A،

احتمال موفقیت Pr[Factor(A,GenModulus,n)=۱] باید ناچیز (negligible) باشد.


آزمون میلر–رابین

۲ فرض فاکتورگیری

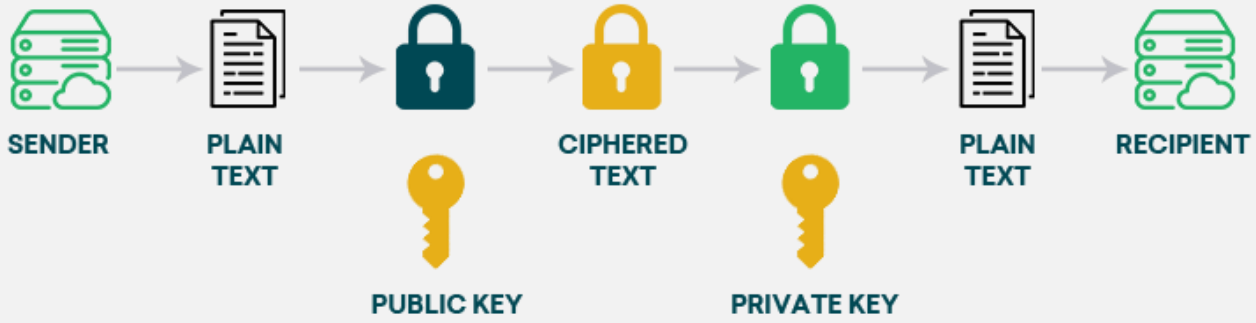
RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# ۹٫۲٫۴ – فرض RSA (RSA Assumption)




مسئلهٔ فاکتورگیری قرن‌هاست بررسی شده و هنوز هیچ الگوریتم واقعاً سریع و عمومی برایش پیدا نشده.

فرض فاکتورگیری به ما یک تابع یک‌طرفه می‌دهد، ولی به شکل مستقیم یک رمزنگاری عملی و تمیز نمی‌سازد (در فصل‌های بعدی کتاب یک راهِ معادل با فاکتورگیری برای ساخت سیستم عملی می‌دهند).

برای همین، سراغ مسائل دیگری رفته‌اند که سختی‌شان با فاکتورگیری «مرتبط» باشد؛ مهم‌ترین و معروف‌ترین‌شان مسئله‌ای است که ۱۹۷۸ توسط Rivest–Shamir–Adleman معرفی شد و امروز به‌اسم مسئلهٔ RSA می‌شناسیم.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
۳
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# تعریف غیررسمی مسئلهٔ RSA

فرض کن یک مدول N و یک عدد e > ۲ داریم که نسبت به ϕ(N) هم‌اول است.

طبق نتیجهٔ قبلی کتاب (Corollary ۹.۲۲)، نگاشت

$$x \mapsto x^e \bmod N$$

روی $Z^*_N$ یک پرموتیشن است؛ یعنی برای هر y دقیقاً یک x وجود دارد که:

$$x^e \equiv y \pmod N$$

کتاب این x را به‌صورت $y^{(1/e)} \bmod N$ یاد می‌کند؛ یعنی ریشهٔ eامِ y پیمانهٔ N

مسئلهٔ RSA غیررسمی این است که:

وقتی فقط N و e و y را می‌دانیم (و فاکتورهای N را نمی‌دانیم)، این $x = y^{(1/e)} \bmod N$ را حساب کنیم.


آزمون میلر–رابین

فرض فاکتورگیری

۳ RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# الگوریتم GenRSA و آزمایش RSA-inv

برای تعریف دقیق، یک الگوریتم تصادفی GenRSA در نظر می‌گیریم که روی ورودی ۱^n سه خروجی می‌دهد:

N = p * q که حاصل ضرب دو عدد اول n بیتی است، یک توان عمومی e و یک توان محرمانه d با شرط‌های:

gcd(e , ϕ(N)) = ۱

e * d ≡ ۱ (mod ϕ(N))

(وجود d به این خاطر است که e پیمانهٔ ϕ(N) وارون ضربی دارد؛ بعداً در رمزگشایی از آن استفاده می‌شود.)

ممکن است GenRSA با احتمال ناچیز خراب شود، ولی این احتمال از نظر رمزنگاری قابل چشم‌پوشی است. حالا آزمایش زیر را برای یک الگوریتم A و پارامتر امنیتی n تعریف می‌کنیم:

RSA-inv_A,GenRSA(n):

اجرای GenRSA(۱^n) و به‌دست آوردن سه‌تایی (N , e , d)

انتخاب یک y یکنواخت از Z*_N دادن N , e , y به الگوریتم A؛ خروجی A عددی x در Z*_N است. خروجی آزمایش برابر ۱ است اگر: x^e ≡ y (mod N)

و در غیر این صورت ۰ است. به‌زبان ساده: این آزمایش می‌سنجد آیا A می‌تواند ریشهٔ eام پیمانهٔ N را برای یک yتصادفی پیدا کند یا نه.

# تعریف سخت بودن مسئلهٔ RSA نسبت به GenRSA

می‌گوییم مسئلهٔ RSA نسبت به الگوریتم تولید GenRSA سخت است اگر:

برای هر الگوریتم تصادفی با زمان چندجمله‌ای A،

یک تابع ناچیز negl(n) وجود داشته باشد به‌طوری‌که:

$$\Pr[\text{RSA-inv}_{\mathcal{A},\text{GenRSA}}(n) = 1] \leq \text{negl}(n)$$

یعنی هیچ الگوریتم سریع A وجود ندارد که بتواند،

با احتمال غیرناچیز، از روی N , e , y عدد x را پیدا کند که:

x^e ≡ y (mod N)

فرض RSA می‌گوید:

یک الگوریتم تولید کلید GenRSA وجود دارد

که نسبت به آن، مسئلهٔ RSA سخت است.

یعنی اگر GenRSA را طبق یک روش استاندارد اجرا کنیم

(مثل تولید N = p * q با p , q اولِ بزرگ)،

هیچ الگوریتم چندجمله‌ای شناخته‌شده‌ای نمی‌تواند به‌طور مؤثر

ریشهٔ eام پیمانهٔ N را برای y های تصادفی به دست بیاورد.

# طرح کلی GenRSA (الگوریتم ۹٫۴۷)

**ALGORITHM 9.47**
GenRSA – **high-level outline**

**Input:** Security parameter $1^n$
**Output:** $N$, $e$, $d$ as described in the text

$(N, p, q) \leftarrow \mathsf{GenModulus}(1^n)$
$\phi(N) := (p-1)(q-1)$
**choose** $e > 1$ such that $\gcd(e, \phi(N)) = 1$
**compute** $d := [e^{-1} \bmod \phi(N)]$
**return** $N, e, d$

دو عدد اول بزرگ p و q انتخاب کن

N = p·q (مدول عمومی)

ϕ(N) = (p−۱)(q−۱)

یک e انتخاب کن که gcd(e, ϕ(N))=۱

d = e^(−۱) mod ϕ(N) (معکوس پیمانه‌ای)

کلید عمومی: (N, e)

کلید خصوصی: d (و معمولاً N)

رمز/امضا: y = x^e mod N

بازگشایی/برگرداندن: x = y^d mod N

امنیت: بدون دانستن p q, یا d، پیدا کردن ریشهٔ eام سخت فرض می‌شود

# مثال ۹٫۴۸ – ساختن GenRSA با عددهای مشخص

فرض کن GenModulus این سه‌تا رو خروجی داده:

$N = 143$

$p = 11$

$q = 13$

اول $\varphi(N)$ رو حساب می‌کنیم:

$$\varphi(N) = (p-1)(q-1) = 10 \times 12 = 120$$

حالا باید یک $e$ انتخاب کنیم که با $\varphi(N) = 12$ هم‌اول باشد.

در مثال کتاب، $e = 7$ انتخاب شده (چون ۷ با ۱۲۰ مقسوم‌علیه مشترک بزرگ‌تر از ۱ ندارد).

بعد، باید $d$ را طوری پیدا کنیم که:

$$e \cdot d \equiv 1 \pmod{\varphi(N)}$$

یعنی معکوس ضربی ۷ پیمانهٔ ۱۲۰ را حساب کنیم. جواب کتاب: $d = 103$ چون:

$$7 \cdot 103 = 721 \equiv 1 \pmod{120}$$

پس در این مثال:

$N = 143$

$e = 7$

$d = 103$

و الگوریتم GenRSA این سه‌تایی $(N, e, d) = (143, 7, 103)$ را برمی‌گرداند.

# Example ۹.۴۸ – محاسبه قدم‌به‌قدم GenRSA

- دو عدد اول انتخاب می‌کنیم: $p=۱۱$ و $q=۱۳$
- حاصل‌ضرب‌شان می‌شود $N = p\times q = ۱۴۳$ این همان “مدول” و بخش اصلی کلید است.
- سپس تابع اویلر را حساب می‌کنیم : $\varphi(N) = (p-۱)(q-۱) = ۱۰\times۱۲ = ۱۲۰$
- یک عدد e انتخاب می‌کنیم که با $\varphi(N)$ نسبت به هم اول باشد: $e=۷$ چون $\gcd(۷,۱۲۰)=۱$
- بعد d را پیدا می‌کنیم که معکوس پیمانه‌ای e نسبت به $\varphi(N)$ باشد:

$d = e^{-۱} \bmod ۱۲۰ = ۱۰۳$ چون $۷\times۱۰۳ = ۷۲۱ \equiv ۱ \pmod{۱۲۰}$

- خروجی : $(N,e,d) = (۱۴۳, ۷, ۱۰۳)$
- کلید عمومی : $(N, e) = (۱۴۳, ۷)$
- کلید خصوصی : $d = ۱۰۳$ (به‌همراه N)

# مسئلهٔ RSA در این مثال (ریشهٔ هفتم ۶۴)

داده‌های مثال (خروجی GenModulus)

N = ۱۴۳  p = ۱۱  q = ۱۳

گام ۱) محاسبه φ(N)

φ(N) = (p−۱)(q−۱) = ۱۰×۱۲ = ۱۲۰

گام ۲) انتخاب e

یک e>۱ می‌خواهیم که gcd(e, φ(N)) = ۱

در این مثال e = ۷ (چون gcd(۷,۱۲۰)=۱)

گام ۳) محاسبه d (معکوس پیمانه‌ای e نسبت به φ(N))

d = e^(-۱) mod ۱۲۰ ⇒ d = ۱۰۳

چک: ۷×۱۰۳ = ۷۲۱ و ۷۲۱ = ۱ mod ۱۲۰

پس خروجی GenRSA در این مثال:

(N, e, d) = (۱۴۳, ۷, ۱۰۳)

# RSA-inv در این مثال یعنی چی؟

- داده‌ها:

N = ۱۴۳ e = ۷ y = ۶۴

- مفهوم RSA (در یک خط) : اگر x پیام/عدد اصلی باشد، خروجی RSA می‌شود : y = x^e (mod N)
- مسئله RSA-inv یعنی چی؟ یعنی برعکسِ RSA را انجام بدهیم
- پیدا کردن x طوری که : x^۷ ≡ ۶۴ (mod ۱۴۳) چرا این مسئله "سخت" فرض می‌شود؟ چون اگر p و q (فاکتورهای N) را ندانیم، معمولاً نمی‌توانیم d را بسازیم و بدون d ، پیدا کردن "ریشه-e ام" در mod N قرار است سخت باشد اما اگر d را داشته باشیم (کلید خصوصی):
- x = y^d (mod N)
- در این مثال: x = ۶۴^۱۰۳ (mod ۱۴۳) = ۲۵ :
- چک سریع (برای اطمینان):
- ۲۵^۷ (mod ۱۴۳) = ۶۴
- پس جواب درست است و RSA-inv حل شد

y → (mod N g توان e) → Forward: x
x → (mod N g توان d) → Inverse: y

Public: (N,e)
Private: d

# مثال ۹٫۴۹ – وقتی d را داریم، مسئله راحت می‌شود

اگر d = ۱۰۳ را داشته باشیم، طبق قضیهٔ قبلی می‌دانیم:

x = y^d mod N

در این مثال داریم:

x = ۶۴^۱۰۳ mod ۱۴۳

کتاب نتیجه را می‌دهد:

x = ۲۵

حالا برای اطمینان چک می‌کنیم:

mod ۱۴۳ = ۶۴ ۷^۲۵

پس واقعاً ۲۵ همان ریشهٔ هفتمِ ۶۴ پیمانهٔ ۱۴۳ است.

این دقیقاً نشان می‌دهد:

بدون دانستن d (یا فاکتورهای N)، پیدا کردن x سخت به‌نظر می‌رسد؛ اما وقتی d را داریم، با یک توان‌رسانی ساده جواب به‌دست می‌آید.

این «نامتقارن بودن» هستهٔ ایدهٔ رمزنگاری کلیدعمومی RSA است:

عموم مردم فقط N و e را می‌دانند (حل مسئله سخت است)،صاحب کلید خصوصی، dرا هم دارد (حل مسئله برای او آسان است).

# چرا دانستن d یعنی "کلید خصوصی"؟

- نکته اصلی RSA :کلید عمومی (N, e) :برای "قفل کردن/امضا را بررسی کردن"
- کلید خصوصی d :برای "باز کردن/امضا کردن"
- اگر d را داشته باشی، RSA-invخیلی ساده می‌شود: برای هر y در Z*_N: x = y^d (mod N) یعنی "ریشه-e ام" را مستقیم پیدا می‌کنی
- ارتباط d با phi(N) :
- d معکوسِ e نسبت به phi(N) است:
- یعنی e*d ≡ ۱ (mod phi(N))
- چرا فاکتورگیری مهم است؟
- اگر p و q را داشته باشی،phi(N) را راحت حساب می‌کنی:
- phi(N) = (p−۱)(q−۱)
- بعد d را با معکوس پیمانه‌ای از روی e درمی‌آوری

p, q → phi(N) → d → RSA-inv solved

به همین دلیل d باید محرمانه بماند.


آزمون میلر–رابین
فرض فاکتورگیری
۳ RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# سختی وقتی d یا فاکتورگیری را بدانیم

نکتهٔ مهم:

اگر d را بدانیم، محاسبهٔ ریشه‌های e- ام پیمانهٔ N ساده می‌شود؛ کافی است برای هر y بگوییم:

x = y^d mod N

اگر phi(N) را بدانیم، می‌توانیم d را حساب کنیم چون باید:

e * d ≡ ۱ mod phi(N)

اگر فاکتورهای N یعنی p و q را بدانیم، اول phi(N) = (p-۱)*(q-۱) را حساب می‌کنیم و بعد از رابطهٔ بالا d را به‌دست می‌آوریم.

کتاب در فصل بعد نشان می‌دهد دانستن هرکدام از این سه مورد (d، یا phi(N)، یا p,q) به‌صورت کارآمد به دوتای دیگر منتهی می‌شود.

برای همین:

وقتی فقط N و e را داریم، **مسئلهٔ RSA سخت به نظر می‌رسد**؛

اما وقتی اطلاعات مخفی مثل dرا داشته باشیم، مسئله با یک توان‌رسانی ساده حل می‌شود.


آزمون میلر–رابین
فرض فاکتورگیری
۳ RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# انتخاب e – چرا ۳ و ۶۵۵۳۷ محبوب‌اند؟

سختی مسئله‌ی RSA به‌طور جدی به انتخاب eوابسته نیست (تا وقتی gcd(e, ϕ(N)) = ۱ باشد)، ولی از نظر کارایی و حملات، انتخاب e اهمیت دارد.

دو انتخاب رایج:

e = ۳

توان‌رسانی به توان ۳ پیمانه‌ی N فقط به دو ضرب نیاز دارد، پس خیلی سریع است.

اما برای اینکه کلید معتبر باشد باید gcd(۳, ϕ(N)) = ۱ برقرار باشد. اگر N = pq و ϕ(N) = (p−۱)(q−۱) ، این شرط یعنی:

gcd(۳, (p−۱)(q−۱)) = ۱

پس هیچ‌کدام از p−۱ یا q−۱ نباید مضرب ۳ باشد. برای p و q اول (و بزرگ)، این عملاً یعنی:

p ≡ ۲ (mod ۳) و q ≡ ۲ (mod ۳)

(این شرط به طور خودکار باعث می‌شود pq ≡ ۱ (mod ۳)، اما عکس آن درست نیست؛ ممکن است p*q ≡ ۱ (mod ۳) باشد ولی هر دو p≡۱ و q≡۱ (mod ۳) باشند که در آن صورت gcd(۳, ϕ(N)) = ۳ می‌شود و کلید نامعتبر است.)

همچنین اگر پروتکل یا padding درست طراحی نشود، برخی حملات low exponent برای e=۳ وجود دارند.

e = ۶۵۵۳۷ = ۲^۱۶ + ۱

یک عدد اول با Hamming weight کم است (تعداد بیت‌های ۱ کم)، بنابراین توان‌رسانی هنوز سریع است.

نسبت به e = ۳ کمی کندتر است، ولی از نظر عملی انتخاب رایج‌تری است و در بسیاری از پیاده‌سازی‌های واقعی، e = ۶۵۵۳۷ یکی از استانداردترین گزینه‌هاست؛ ضمن اینکه در برابر بخشی از ریسک‌های رایجِ توان کم (در صورت طراحی بد/padding پروتکل) هم حاشیه‌ی امن بهتری می‌دهد.

# چرا نباید d کوچک یا با وزن هَمینگ کم باشد؟

نکتهٔ امنیتی مهم:

اگر d خیلی کوچک باشد، مهاجم می‌تواند روی d brute force بزند؛

بازهٔ کوچک ⇒ امتحان کردن همهٔ مقادیر ممکن است.

حتی اگر d آن‌قدر بزرگ باشد که brute force نشود، ولی هنوز $d < N^{(1/4)}$ باشد،

حمله‌های معروف (مثل حملهٔ Wiener) می‌توانند از روی N و e، مقدار d را بازیابی کنند.

انتخاب d با Hamming weight کم (تعداد بیت‌های ۱ کم) هم بد است؛

چون ساختار خاص بیت‌ها می‌تواند توسط بعضی الگوریتم‌ها مورد سوءاستفاده قرار بگیرد.

به همین خاطر، در GenRSA معمولاً:

e را کوچک و با Hamming weight کم می‌گیرند (مثل ۶۵۵۳۷)،

ولی d را بزرگ و به‌قدر کافی تصادفی انتخاب می‌کنند تا این‌جور حمله‌ها جواب ندهد.

# جهت اول: اگر فاکتورگیری آسان باشد، RSA سخت نیست

فرض کن GenRSA دقیقاً طبق الگوریتم ۹.۴۷ ساخته شده باشد:

اول GenModulus عددهای p , q و N = p*q را می‌سازد؛

بعد phi(N) = (p-۱)*(q-۱)،

بعد e و d با شرط e*d ≡ ۱ mod phi(N)

اگر ما بتوانیم N را فاکتور کنیم، یعنی p و q را پیدا کنیم، آن‌وقت:

phi(N) = (p-۱)*(q-۱) را حساب می‌کنیم،

بعد هم d = e^(-۱) mod phi(N) را به‌سادگی با الگوریتم استاندارد (مثل Euclid توسعه‌یافته) به‌دست می‌آوریم.

نتیجه:

**اگر فاکتورگیری نسبت به GenModulus آسان باشد، مسئلهٔ RSA نسبت به همین GenRSA نمی‌تواند سخت باشد.**

پس RSA از نظر سختی «از فاکتورگیری جلوتر نمی‌زند»؛

حداکثر می‌تواند **سخت‌تر یا هم‌سطح** باشد، نه سخت‌تر از آن.

# سؤال برعکس: آیا سختی RSA از سختی فاکتورگیری نتیجه می‌شود؟

این‌جا سؤال اصلی این است:

اگر فاکتورگیری سخت باشد،

آیا خودِ مسئلهٔ RSA هم حتماً سخت است؟

جوابِ فعلی علم: **نمی‌دانیم.**

چیزی که می‌توانیم ثابت کنیم این است:

اگر بتوانیم از روی N و e، کلید خصوصی d را حساب کنیم،

آن‌وقت می‌توانیم N را هم فاکتور کنیم.

یعنی **به‌دست آوردن d از (N,e)** از نظر سختی **حداقل به سختی فاکتورگیری است.**


آزمون میلر–رابین

فرض فاکتورگیری

۳ RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# ایدهٔ قضیهٔ ۹٫۵۰ – از «ریشهٔ kام ۱» به فاکتورگیری

قضیهٔ ۹٫۵۰ می‌گوید:

اگر یک زیرروال داشته باشیم که برای هر x در Z*_N

عددی k > ۰ بدهد که : x^k ≡ ۱ mod N

آن‌وقت می‌توانیم N را در زمان چندجمله‌ای فاکتور کنیم.

ایدهٔ کلی (برای N = p*q با p,q اولِ فرد):

برای چنین N، عدد ۱ دقیقاً **چهار ریشهٔ مربعی** پیمانهٔ N دارد:

۱ ، -۱ و دو ریشهٔ غیر بدیهی (در نمایش CRTبرابر (۱,-۱) و (-۱,۱)).

اگر یک «ریشهٔ دوم غیر بدیهی» y از ۱ داشته باشیم،

می‌توانیم با:

gcd(y-۱ , N) یا gcd(y+۱ , N)

یکی از عوامل Nرا پیدا کنیم.

# یک مثال کوتاه

برای N = ۱۵ :

y = ۴ داریم: $۲^{۴} = ۱۶ \equiv ۱ \bmod ۱۵$

y نه ۱ است، نه -۱ ⇒ ریشهٔ غیر بدیهی است.

حالا:

$gcd(۴-۱ , ۱۵) = gcd(۳ , ۱۵) = ۳$

$gcd(۴+۱ , ۱۵) = gcd(۵ , ۱۵) = ۵$

هر دو مقسوم‌علیه‌های غیر trivial هستند ⇒ ۱۵ = ۳ * ۵.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها ۳

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# استراتژی قضیهٔ ۹٫۵۰

برای N بزرگ:

یک x یکنواخت از Z*_N انتخاب می‌کنیم.

زیرروال به ما k می‌دهد با x^k ≡ ۱ mod N.

K را به صورت k = ۲^s * v با v فرد می‌نویسیم.

دنبالهٔ زیر را محاسبه می‌کنیم:

x^v , x^(۲v) , x^(۴v) , … , x^(۲^s * v) (mod N)

به‌دنبال بزرگ‌ترین j می‌گردیم که y = x^(۲^j * v) هنوز ۱ نشده است؛

آن‌وقت y^۲ ≡ ۱ mod N می‌شود.

اگر y هم ۱ نباشد و هم ۱- mod N نباشد،

یک ریشهٔ دوم غیر بدیهی پیدا کرده‌ایم ⇒ می‌توانیم N را فاکتور کنیم.

اگر این اتفاق نیفتد ⇒ یک x دیگر امتحان می‌کنیم.

با استفاده از ایدهٔ زیرگروهِ Bad (مثل کاری که برای Miller–Rabin کردیم)

کتاب نشان می‌دهد احتمال موفقیت در هر انتخاب x حداقل ۱/۲ است.

پس با چند تکرار، با احتمال خیلی بالا N فاکتور می‌شود.

# از d تا فاکتورگیری N (راه عملی)

- اگر d را داریم، k = e*d - ۱ یک مضرب از phi(N) است.
- k را بنویس : k = ۲^s * r که r فرد است.
- یک x تصادفی از Z*_N انتخاب کن.
- a = x^r mod N را حساب کن.
- اگر a = ۱ یا a = -۱ شد، یک x دیگر بگیر.
- حالا a را پشت سر هم مربع کن تا به ۱ برسی.
- اولین جایی که از مقدار “نه ±۱” به ۱ می‌رسی، ریشه غیر بدیهی ۱ داری.
- آن‌وقت gcd(a-۱, N) یک عامل غیر بدیهی N را می‌دهد.
- این همان پل قضیه ۹.۵۰ به نتیجه ۹.۵۱ است.
- پس داشتن d یعنی فاکتورگیری N عملاً ممکن است.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها ۳
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# نتیجهٔ ۹٫۵۱ – داشتن (N,e,d) برای فاکتورگیری کافی است

حال اگر یک الگوریتمی N و e و d را داشته باشد با شرط:

$e * d \equiv 1 \bmod \phi(N)$

می‌توانیم قضیهٔ قبلی را روی آن پیاده کنیم:

تعریف می‌کنیم: $k = e*d - 1$

می‌دانیم $k$، $\phi(N)$ را تقسیم می‌کند ⇒ برای همهٔ x در $Z^*_N$ داریم:

$x^k \equiv 1 \bmod N$

پس زیرروال قضیهٔ ۹٫۵۰ را خیلی ساده پیاده می‌کنیم:

«هر وقت ورودی x آمد، جواب همیشه همان k است».

با این کار، خودِ قضیهٔ ۹٫۵۰ یک الگوریتم چندجمله‌ای برای فاکتورگیری N به ما می‌دهد.

نتیجهٔ مهم:

اگر فاکتورگیری واقعاً سخت باشد،

هیچ الگوریتم چندجمله‌ای‌ای نمی‌تواند از روی (N,e)، کلید خصوصی d را به‌طور مطمئن پیدا کند.

# تصویر کلی رابطهٔ فرض فاکتورگیری و فرض RSA

اگر فاکتورگیری عدد N آسان باشد، آنگاه مسئله‌ی RSA نیز قطعاً آسان است؛ زیرا با دانستن عوامل اول p و q می‌توان $\phi(N)$ را محاسبه کرد و سپس مقدار d را به‌دست آورد.

همچنین اگر الگوریتمی وجود داشته باشد که با دانستن (N , e , d) بتواند عدد N را فاکتور کند، در این صورت می‌توان گفت مسئله‌ی پیدا کردن d از روی (N , e) حداقل به سختی مسئله‌ی فاکتورگیری است.

با این حال، هنوز رابطه‌ی دقیق بین سختی مسئله‌ی RSA و سختی مسئله‌ی فاکتورگیری مشخص نیست. به‌طور خاص، معلوم نیست که آیا می‌توان مسئله‌ی RSA را در زمان چندجمله‌ای حل کرد بدون آن‌که عدد N را فاکتور کرد. به بیان دیگر، ممکن است از نظر تئوریک مسئله‌ی RSA در زمان چندجمله‌ای قابل حل باشد، در حالی که فاکتورگیری همچنان سخت باقی بماند.

به همین دلیل، از دیدگاه منطق نظری نمی‌توان گفت که فرض RSA معادل یا قوی‌تر از فرض فاکتورگیری است. تنها رابطه‌ی اثبات‌شده این است که آسان بودن فاکتورگیری، آسان بودن مسئله‌ی RSA را نتیجه می‌دهد، اما عکس این گزاره ثابت نشده است.

با این وجود، هنگامی که الگوریتم GenRSA با استفاده از GenModulus (مانند الگوریتم ۹٫۴۷) ساخته می‌شود، حدس غالب در جامعه‌ی رمزنگاری این است که اگر فاکتورگیری نسبت به GenModulus سخت باشد، آنگاه مسئله‌ی RSA نیز نسبت به GenRSA سخت خواهد بود؛ هرچند این موضوع به‌صورت قضیه‌ی رسمی اثبات نشده است.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها ۳
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# فرض‌های رمزنگاری در گروه‌های دوری

ایده‌ی کلی این بخش:

اول کمی درباره‌ی گروه‌های دوری (cyclic groups) و مولد (generator) حرف می‌زنیم؛

بعد، فرض‌های رمزنگاری‌ای که روی این گروه‌ها تعریف می‌شود (مثل DLP و DH)معرفی می‌شوند؛

و در آخر، دو مثال مهم از گروه‌های واقعی (عددی و بیضوی) را می‌بینیم که این فرض‌ها در آن‌ها «سخت» فرض می‌شوند.

الان فقط روی قسمت اول، یعنی ساختار گروه دوری و مولد تمرکز می‌کنیم.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# زیرگروهی که یک عنصر می‌سازد و «مرتبه» عنصر

فرض کن:

G یک گروه متناهی باشد با اندازه‌ی m

و g یک عنصر دلخواه از G باشد. توان‌های مختلف g را نگاه می‌کنیم:

$g^0 , g^1 , g^2 , \dots$

چون گروه متناهی است، طبق قضیه‌ی ۹٫۱۴ می‌دانیم:

$g^m = 1$

حالا بگذار i کوچک‌ترین عدد طبیعی مثبت باشد که:

$g^i = 1$

در این صورت دنباله‌ی بالا بعد از i قدم تکرار می‌شود، و مجموعه‌ای که g تولید می‌کند این است:

$\langle g \rangle = \{ g^0 , g^1 , \dots , g^{(i-1)} \}$

این مجموعه دقیقاً i عضو دارد و خودش یک زیرگروه از G است. به آن می‌گوییم: زیرگروهِ تولیدشده توسط g، و به عدد i می‌گوییم مرتبه‌ی g

تعریف ۹٫۵۲:

مرتبه‌ی g کوچک‌ترین عدد طبیعی مثبت iاست که:

$g^i = 1$

# گزاره ۹٫۵۳ و ۹٫۵۴ – توان‌ها فقط تا «مرتبه» مهم‌اند

اگر مرتبه‌ی g برابر i باشد، یعنی:

g^i = ۱ و برای هیچ عدد کوچک‌تر از i این اتفاق نمی‌افتد.

گزاره ۹٫۵۳:

برای هر عدد صحیح x داریم:

g^x = g^[x mod i]

یعنی توان گرفتن از g عملاً ماژولوی مرتبه‌ی g معنی دارد.

گزاره ۹٫۵۴ (قوی‌تر):

g^x = g^y اگر و تنها اگر x ≡ y mod i

به زبان ساده:

دو توان از g برابرند دقیقاً وقتی که توان‌هاشان در پیمانه‌ی i برابر باشند.

این دو گزاره در ادامه، برای تعریف مسألهٔ لگاریتم گسسته و DH کلیدی‌اند؛ چون نشان می‌دهند در یک گروه دوری، «نمای‌ها» مثل اعداد مدولوی کار می‌کنند.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
۴
DLP/DH
یک‌طرفه و هش

# گروه دوری و مولد (cyclic group, generator)

در هر گروه:

عنصر همانی (۱) تنها عنصری است که مرتبه‌ی ۱ دارد،

و زیرگروهی که می‌سازد فقط {۱} است.

اگر در یک گروه متناهی G با اندازه‌ی m، عنصری g وجود داشته باشد که مرتبه‌اش m باشد، یعنی:

g> = G>

در این صورت:

می‌گوییم G یک **گروه دوری** (cyclic group) است؛

و g را یک **مولد** (generator) برای G می‌نامیم.

در این حالت، هر عضو h از G را می‌توان این‌طور نوشت:

h = g^x برای یک عدد x در بازه‌ی ۰ ... m-۱.

نکته: یک گروه دوری معمولاً **چندین مولد** دارد، بنابراین نمی‌توان گفت the generator، بلکه فقط می‌گوییم a generator


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

۴ گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# محدودیت روی مرتبه‌ها و گروه با اندازه‌ی اول

**گزاره ۹٫۵۵:**

اگر گروه G اندازه‌ی m داشته باشد و مرتبه‌ی g برابر iباشد،

آن‌وقت:

iحتماً **مقسوم‌علیه** m است، یعنی i، m را تقسیم می‌کند.

ایده: چون g^m = ۱ و مرتبه‌ی g همان کوچک‌ترین i با g^i = ۱ است،

نتیجه می‌گیریم m ≡ ۰ mod l

**نتیجه‌ی مهم – هم‌ارز ۹٫۵۶:**

اگر G گروهی باشد با اندازه‌ی یک عدد اول p، آن‌وقت:

G حتماً **گروه دوری** است؛

و **تمام اعضا به‌جز همانی، مولد هستند.**

چرا؟

چون مقسوم‌علیه‌های p فقط ۱ و p هستند.

پس مرتبه‌ی هر عنصر یا ۱ است (فقط برای همانی) یا p است که یعنی آن عنصر کل گروه را تولید می‌کند.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# مثال‌ها

دو مثال مهم از گروه‌های دوری:

**گروه جمعی Z_N** (با عمل جمع پیمانه‌ای):

اعضا: ۰,۱,...,N-۱

عمل: a + b mod N

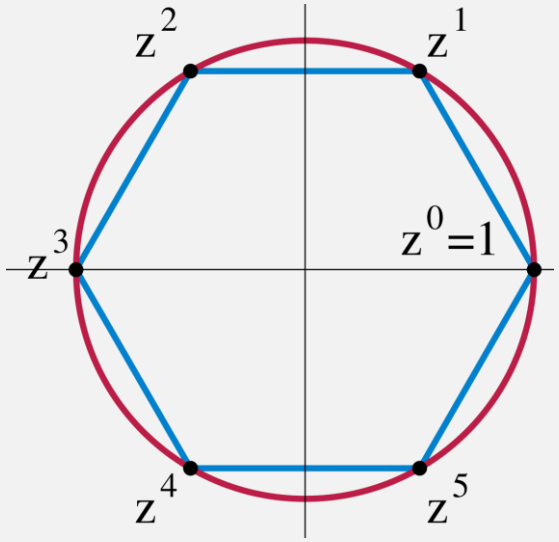
عنصر ۱ همیشه مولد است؛ چون با جمع‌کردن ۱ به‌خودی‌خود، همه‌ی کلاس‌ها را می‌سازیم.

**گروه‌های با اندازه‌ی اول** (مثلاً جمعی Z_p):

اگر p اول باشد، Z_p با جمع، گروهی دوری است که هر عنصر غیرصفر، مولد آن است.

در ادامه‌ی بخش ۹٫۳، کتاب روی گروه‌های دوری خاصی (مثل زیرگروه‌های ضربی Z_p^* و گروه‌های روی منحنی بیضوی) تمرکز می‌کند و روی آن‌ها فرض‌های سختی DLP / DH را تعریف می‌کند.

# قضیهٔ ۹٫۵۷ – گروه ضربی Z*_p




**قضیه ۹٫۵۷:**

اگر p یک عدد اول باشد، آنگاه گروه ضربی Z*_p یک گروه دوری با اندازهٔ p-۱ است.

یعنی:

عناصر گروه: p-۱,...,۱,۲

عمل: ضرب mod p

یک مولد g وجود دارد که توان‌هایش تمام این p-۱ عنصر را می‌سازند:

Z*_p = { g^۰ , g^۱ , ... , g^(p-۲) }

این دیگر از آن قضیه‌های «حقیقت از آسمان افتاده» است: اثباتش ساده نیست، ولی نتیجه‌اش پایهٔ DLP و DH است.

آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# مثال ۹٫۵۸ – گروه جمعی $Z_{15}$

اینجا با **جمع پیمانه‌ای** کار می‌کنیم، نه ضرب.

گروه: $Z_{15}$ با عمل $a + b \bmod 15$

عنصر همانی: ۰

۱. **مولد ۱**

۱, ۲, ۳, ..., ۱۴, ۰

تمام ۱۵ عنصر را می‌سازند ⇒ ۱ مولد است.

**مولد ۲**

دنبالهٔ جمعی ۲:

۰, ۲, ۴, ۶, ۸, ۱۰, ۱۲, ۱۴, ۱, ۳, ۵, ۷, ۹, ۱۱, ۱۳

باز هم تمام $Z_{15}$ را پوشش می‌دهد ⇒ ۲ هم مولد است.

**عنصر غیرمولد: ۳**

دنبالهٔ ۳:

۰, ۳, ۶, ۹, ۱۲, ۰, ...

فقط ۵ عنصر به دست می‌آید: {۰,۳,۶,۹,۱۲}

پس مرتبهٔ ۳ برابر ۵ است و فقط یک **زیرگروه** اندازهٔ ۵ می‌سازد.

# ادامه مثال ۹٫۵۸ – گروه جمعی $Z_{15}$

**عنصر غیرمولد: ۱۰**

دنبالهٔ ۱۰:

۰, ۱۰, ۵, ۰, ...

فقط ۳ عنصر {۰,۵,۱۰} ⇒ مرتبهٔ ۱۰ برابر ۳ است.

این مثال قشنگ نشان می‌دهد:

همهٔ گروه دوری است،

ولی بعضی عناصر کل گروه را تولید می‌کنند (مولد)،

بعضی‌ها فقط زیرگروه‌های کوچک‌تر می‌سازند،

و اندازهٔ زیرگروه‌ها (۵ و ۳) مقسوم‌علیهِ ۱۵ هستند (طبق گزارهٔ ۹٫۵۵).

# مثال ۹٫۵۹ – گروه ضربی $Z^*_{15}$

الان گروه **ضربی** را نگاه می‌کنیم:

$Z^*_{15}$ یعنی اعدادی بین ۱ تا ۱۴ که با ۱۵ هم‌اول‌اند:

$\{1,2,4,7,8,11,13,14\}$

اندازهٔ گروه: $\phi(15) = (5-1)*(3-1) = 8$

دنبالهٔ توان‌های ۲:

$2^1 = 2$

$2^2 = 4$

$2^3 = 8$

$2^4 = 16 \equiv 1 \bmod 15$

پس مرتبهٔ ۲ برابر ۴ است (چهار مرحله تا برسیم به ۱)

و ۴ مقسوم‌علیه ۸ است، مطابق گزارهٔ ۹٫۵۵.

# مثال ۹/۶۰ – گروه جمعی Z_p برای p اول

این مثال تأیید می‌کند که:

در گروه جمعی Z_p(پیمانهٔ عدد اول p)

**هر عنصر غیرصفر یک مولد است.**

برای هر h با ۱ <= h <= p-۱:

اگر i > ۰ داشته باشیم که:

$$i * h \equiv 0 \bmod p$$

این یعنی p، i*h را تقسیم می‌کند. از قضیهٔ تقسیم‌پذیری می‌دانیم:

یا p، h را تقسیم می‌کند

یا p , i را

چون h < p است، نمی‌تواند مضربی از p باشد؛

پس باید p، i را تقسیم کند ⇒ کوچک‌ترین ممکن همان i = p است. یعنی مرتبهٔ هر h غیرصفر برابر p است و بنابراین **همهٔ h ≠ ۰ مولد هستند**، همان چیزی که قبل‌تر در نتیجهٔ ۹/۵۶ گفته شده بود.

# مثال ۹٫۶۱ – گروه ضربی $Z^*_7$

حالا گروه ضربی $Z^*_7$ را می‌بینیم:

اعضا: {۱,۲,۳,۴,۵,۶}

اندازه: $\varphi(7) = 6$

طبق قضیهٔ ۹٫۵۷ این گروه دوری است.

توان‌های ۲:

$2^1 = 2$

$2^2 = 4$

$2^3 = 8 \equiv 1 \bmod 7$

پس مرتبهٔ ۲ برابر ۳ است ⇒ مولد **نیست**.

توان‌های ۳:

$3^1 = 3$

$3^2 = 9 \equiv 2 \bmod 7$

$3^3 = 3*2 = 6 \bmod 7$

$3^4 = 3*6 = 18 \equiv 4 \bmod 7$

$3^5 = 3*4 = 12 \equiv 5 \bmod 7$

$3^6 = 3*5 = 15 \equiv 1 \bmod 7$

پس دنبالهٔ ۳ شامل تمام عناصر {۱,۲,۳,۴,۵,۶} است.بنابراین ۳ یک **مولد** برای $Z^*_7$ است.

# مثال ۹٫۶۲ – ایزومورفیسم بین Z_n و هر گروه دوری

فرض کن: G یک گروه دوری از مرتبهٔ n باشد،

g یک مولد در G باشد.

تابع زیر را تعریف می‌کنیم:

f : Z_n -> G با فرمول f(a) = g^a

این تابع:

جمع را حفظ می‌کند:

f(a + a') = g^(a + a') = g^a * g^a' = f(a) * f(a')

یک‌به‌یک و پوشا هم هست (با استفاده از این‌که مرتبهٔ g برابر n است).

پس f یک **ایزومورفیسم** است؛ یعنی:

از دید جبر مجرد، هر گروه دوری با اندازه‌ی n

با گروه جمعی» Z_n یک‌شکل» است.

اما نکتهٔ خیلی مهم برای رمزنگاری:

این‌که یک ایزومورفیسم f وجود دارد،

**به این معنی نیست که معکوسش f^-۱ (یعنی پیدا کردن نمای x از روی g^x)**

به‌صورت کارآمد قابل محاسبه است.

از نگاه محاسباتی، همین «فاصله» است که مسألهٔ **لگاریتم گسسته (DLP)** را سخت می‌کند و تبدیلش می‌کند به یک فرض رمزنگاری.

# مدل کلی: گروه دوری + الگوریتم تولید گروه

فرض می‌کنیم یک الگوریتم تولید گروه داریم که کتاب اسمش را G گذاشته.

ایده‌اش این است:

ورودی: ۱^n (پارامتر امنیتی با طول n)

خروجی:

یک گروه دوری G

اندازهٔ گروه q (تقریباً هم‌مرتبه با n)

یک مولد g در این گروه

نمایش هر عضو گروه باید **منحصربه‌فرد** و به صورت رشتهٔ بیت باشد و: یک الگوریتم کارا داریم که تشخیص دهد یک رشته بیت واقعاً عضو G هست یا نه. یک الگوریتم کارا برای انجام عمل گروه (ضرب یا جمع) داریم.

پس می‌توانیم به‌صورت کارا:

توان‌گیری g^x

محاسبهٔ معکوس g^(q-۱) و انتخاب عضو یکنواخت h با انتخاب یکنواخت x از Z_q و گذاشتن h = g^x را انجام دهیم.

نکتهٔ مهم:

از دید **جبر**، همهٔ گروه‌های دوری با اندازهٔ برابر، «یک‌شکل» هستند؛

اما از دید **محاسباتی**، این‌که چطور عناصر را به‌صورت بیت نمایش می‌دهیم، تعیین می‌کند سختی محاسبات در آن گروه چقدر است.

# لگاریتم گسسته (Discrete Logarithm)

اگر Gگروه دوری با اندازهٔ q و مولد g باشد، آن‌وقت:

G = { g^۰ , g^۱ , ... , g^(q-۱) }

برای هر h در G دقیقاً یک x در Z_q هست که : g^x = h

به این x می‌گوییم لگاریتم گسستهٔ h نسبت به g و می‌نویسیم:

x = log_g(h)

اگر برای یک عدد صحیح دلخواه x داشته باشیم g^x = h، آن‌وقت همیشه:

x mod q] = log_g(h)]

قوانینش مثل لگاریتم معمولی است:

log_g(۱) = ۰

برای هر عدد صحیح r :

log_g(h^r) = [ r * log_g(h) mod q ]

برای هر :h۱ , h۲

log_g(h۱ * h۲) = [ log_g(h۱) + log_g(h۲) mod q ]

(عمل گروه ممکن است ضرب باشد یا جمع؛ این‌جا نماد ضربی می‌نویسیم.)


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# مسئلهٔ DLog و فرض سختی آن

مسئلهٔ لگاریتم گسسته در یک گروه دوری G با مولد g این است که:

برای یک h یکنواخت در G، عدد x = log_g(h) را پیدا کنیم.

کتاب این را با یک آزمایش رسمی تعریف می‌کند:

آزمایش :DLog_A,G(n)

اجرای G(1^n) و گرفتن (G , q , g)؛

G گروه دوری، اندازهٔ گروه q، و g مولد.

انتخاب یکنواخت h از G.

دادن (G , q , g , h) به الگوریتم A؛ خروجی A عددی x در Z_q است.

خروجی آزمایش ۱ است اگر:

$$g^x = h$$

و در غیر این صورت ۰ است.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# ادامه مسئلهٔ DLog و فرض سختی آن

**تعریف ۹٫۶۳ (فرض Dlog) :**

می‌گوییم مسئلهٔ Dlog نسبت به G سخت است اگر:

برای هر الگوریتم تصادفی چندجمله‌ای A

یک تابع ناچیز negl(n) وجود داشته باشد طوری که:

**Pr[ DLog_A,G(n) = 1 ] <= negl(n)**

یعنی هیچ الگوریتم سریع، روی گروه‌هایی که این الگوریتم تولید می‌کند،

نمی‌تواند لگاریتم گسسته را با احتمال قابل توجه حل کند.

فرض لگاریتم گسسته یعنی:

دست‌کم یک الگوریتم تولید گروه G وجود دارد

که روی گروه‌هایش مسئلهٔ DLog سخت است.

در بخش‌های بعد (۹٫۳٫۳ و ۹٫۳٫۴) مثال‌هایی از چنین گروه‌هایی می‌بینیم

(گروه‌های روی Z_p^* و منحنی‌های بیضوی).

# مسائل Diffie–Hellman : CDH و DDH

حالا می‌رویم سر سرِ اصلی این بخش: فرض‌های DH.

فرض کن:

G یک گروه دوری

g یک مولد

برای دو عنصر h۱ , h۲ در G داریم:

اگر h۱ = g^x۱ و h۲ = g^x۲ باشد،

تعریف می‌کنیم:

DH_g(h۱ , h۲) = g^(x۱ * x۲)

چون x۱ = log_g(h۱) و x۲ = log_g(h۲)، می‌توانیم بنویسیم:

DH_g(h۱ , h۲) = g^(log_g(h۱) * log_g(h۲))

یا DH_g(h۱ , h۲) = h۱^x۲ = h۲^x۱

این همان «کلید مشترک» کلاسیک Diffie–Hellman است:

هر طرف نمای خودش را دارد و محصول نمایی مشترک، برابر است.

# مسئلهٔ CDH (Computational Diffie–Hellman)

**مسئلهٔ CDH** در یک گروه دوری G با مولد g :

ورودی: g , g^x , g^y برای دو عدد تصادفی x , y از Z_q.

کار: محاسبهٔ g^(x*y).

به‌طور خلاصه:

A باید از روی g^x و g^y، عنصر DH_g(g^x , g^y) را پیدا کند.

اگر مسئلهٔ DLog سخت **نباشد** (یعنی لگاریتم را بتوانیم سریع حساب کنیم)،

آن‌وقت CDH هم سخت نخواهد بود:

کافی است x۱ = log_g(h۱) و x۲ = log_g(h۲) را حساب کنیم،

بعد g^(x۱ * x۲) را خروجی بدهیم.

اما برعکسش معلوم نیست:

سخت بودن DLog الزاماً ثابت نمی‌کند که CDH هم سخت است؛

این سؤال هنوز در حالت کلی بی‌جواب است.

# مسئلهٔ DDH (Decisional Diffie–Hellman)

اینجا سؤال کمی فرق دارد:

آیا می‌توانیم تشخیص بدهیم یک عنصر واقعاً g^(x*y) است

یا فقط یک عنصر تصادفی از گروه است؟

به‌طور دقیق:

دو توزیع را مقایسه می‌کنیم:

(G , q , g , g^x , g^y , g^z)

که در آن x , y , z یکنواخت در Z_q هستند؛ پس g^z یک عضو **کاملاً تصادفی** در G است.

(G , q , g , g^x , g^y , g^(x*y))

که در آن x , y یکنواخت‌اند و g^(x*y) همان مقدار Diffie–Hellman است.

# ادامه مسئلهٔ DDH (Decisional Diffie–Hellman)

تعریف ۹٫۶۴ (فرض DDH) :

می‌گوییم مسئلهٔ DDH نسبت به G سخت است اگر:

برای هر الگوریتم تصادفی چندجمله‌ای A

اختلاف این دو احتمال ناچیز باشد:

Pr[ A(G,q,g,g^x,g^y,g^z) = ۱ ]

و

Pr[ A(G,q,g,g^x,g^y,g^(x*y)) = ۱ ]

(در هر دو، x,y,z یکنواخت در Z_q هستند.)

یعنی هیچ الگوریتم سریع نمی‌تواند تشخیص دهد

آیا آخرین عنصر یک DH واقعی است یا فقط یک عنصر تصادفی؛

برتری‌اش نسبت به حدس تصادفی، ناچیز است.

# مثال عددی ساده برای DLog / CDH / DDH

گروه نمونه:

$G = Z^*_{11}$، اندازهٔ گروه ۱۰ است و ۲ یک مولد است.

مسئلهٔ لگاریتم گسسته (DLP) : می‌خواهیم x را پیدا کنیم به‌طوری‌که:

$2^x \equiv 9 \pmod{11}$

با امتحان کردن $x = 0$ تا ۹ می‌بینیم:

$2^6 \equiv 9 \pmod{11}$

پس $x = 6$ است.

مسئلهٔ دیفی–هلمن محاسباتی (CDH) :

آلیس عدد مخفی $a = 3$ دارد و

$A = 2^3 \equiv 8 \pmod{11}$ را می‌فرستد. باب عدد مخفی $b = 4$ دارد و

$B = 2^4 \equiv 5 \pmod{11}$ را می‌فرستد.

کلید مشترک ایده‌آل:

$K = 2^{(ab)} = 2^{12} \equiv 2^2 \equiv 4 \pmod{11}$

سؤال CDH : با داشتن فقط $g = 2$، $A$ و $B$، محاسبهٔ $K$ چقدر سخت است؟

مسئلهٔ دیفی–هلمن تصمیمی (DDH) : باید تشخیص بدهیم آیا سه‌تایی $(A, B, 4)$ از نوع واقعی $(ab)^2$ است یا سه‌تایی $(A, B, 3)$ که مؤلفهٔ سومش یک عضو تصادفی از گروه است. اگر هیچ الگوریتمی نتواند بهتر از حدس‌زدن بین این دو حالت تمایز بدهد، فرض DDH در این گروه برقرار است.

# روابط بین DLog، CDH و DDH

جمع‌بندی روابط:

اگر DLog آسان باشد ⇒

CDH هم آسان است (چون با لگاریتم گرفتن مسئله حل می‌شود).

اگر CDH آسان باشد ⇒

DDH هم آسان است (می‌توانیم g^(x*y) را خودمان حساب کنیم و مقایسه کنیم).

اما برعکس‌ها معلوم نیست:

سخت بودن CDH لزوماً به سخت بودن DLog منجر نمی‌شود.

حتی گروه‌هایی می‌شناسیم که DLog و CDH در آن‌ها سخت «فرض» می‌شوند، ولی DDH در آن‌ها آسان است (مثال‌ها بعداً در تمرین‌ها و فصل ۱۵ می‌آید).

پس از دید زنجیرهٔ سختی:

DLog آسان ⇒ CDH آسان ⇒ DDH آسان

ولی

DDH سخت سخت‌تر از بقیه نیست؛ فقط یک فرض قوی‌تر و متفاوت است.

# چرا سراغ گروه‌های با مرتبهٔ اول (prime-order) می‌رویم؟

برای DLog / DH می‌تونیم از هر گروه دوری استفاده کنیم،

ولی در عمل معمولاً گروهی رو انتخاب می‌کنیم که اندازه‌اش یک عدد اول q باشد.

سه دلیل اصلی:

۱. سختی DLog در گروه‌های prime-order بیشتر است

۲. پیدا کردن مولد در گروه prime-order خیلی ساده است

۳. وجود معکوس برای همهٔ نماها و قضیه کمکی ۹٫۶۵


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# ۱. سختی DLog در گروه‌های prime-order بیشتر است

یک الگوریتم به نام Pohlig–Hellman (تو فصل ۱۰) نشان می‌دهد:

اگر اندازهٔ گروه q ترکیبی باشد و عامل‌های اول کوچک داشته باشد،

شکستن DLog در آن گروه خیلی ساده‌تر می‌شود.

هرچه عامل‌های اول q کوچکتر و ساده‌تر باشند،

مسئله Dlog هم بیشتر تکه‌تکه و راحت‌تر می‌شود.

این لزوماً به این معنی نیست که Dlog در همهٔ گروه‌های غیر اول «آسان» است،

اما یعنی: بدترین حالت DLog معمولاً وقتی است که qخودش اول باشد.

برای DDH هم یک مشکل مشابه داریم:

اگر q عامل‌های اول کوچک داشته باشد، در بعضی گروه‌ها تشخیص DH واقعی از عنصر تصادفی، ساده می‌شود (مثال دقیقش را کتاب در تمرین ۱۵/۱۶ می‌دهد).

# ۲. پیدا کردن مولد در گروه prime-order خیلی ساده است

طبق نتیجهٔ قبلی‌مان (Corollary ۹.۵۶)

در یک گروه با اندازهٔ اول q،

همهٔ عناصر غیرهمانی مولد هستند.

یعنی اگر:

G گروهی با اندازهٔ q اول باشد،

و عنصر همانی ۱ باشد،

آن‌وقت هر g با ۱ =! g یک generator است.

پس:

برای ساخت گروه prime-order فقط کافی است یک عنصر غیرهمانی برداریم؛

با احتمال ۱، مولد است.

در مقابل، در یک گروه دوری با اندازهٔ ترکیبی،

برای پیدا کردن مولد کار راحت نیست؛

معمولاً باید عامل‌های اول اندازهٔ گروه را بدانیم تا مولد بسازیم

(این را کتاب در Appendix B.۳ توضیح می‌دهد).

# ۳. وجود معکوس برای همهٔ نماها و قضیه کمکی ۹٫۶۵

در خیلی از اثبات‌های امنیتی لازم می‌شود نمایی مثل a یا x را معکوس ضربی کنیم. اگر اندازهٔ گروه q اول باشد: هر عدد غیرصفر در Z_q وارون دارد؛ یعنی برای هر y != ۰ می‌توانیم y^(-۱) mod q را حساب کنیم. کتاب این نکته را با قضیه کمکی ۹٫۶۵ فرمال می‌کند:

قضیه کمکی ۹٫۶۵

در گروهی G با اندازهٔ اول q، و عناصر g,h در G با g != ، اگر دو جفت متمایز (x,y) و (x',y') در Z_q × Z_q داشته باشیم که:

g^x * h^y = g^x' * h^y'

آن‌وقت می‌توانیم log_g(h) را به‌صورت کارا حساب کنیم. ایدهٔ اثبات از برابری بالا شروع می‌کنیم:

g^x * h^y = g^x' * h^y'

دو طرف را تقسیم می‌کنیم (در گروه ضربی):

g^(x - x') = h^(y' - y)

بگذاریم:

alpha = x - x'

beta = y' - y

پس:

g^alpha = h^beta

# ادامه وجود معکوس برای همهٔ نماها و قضیه کمکی ۹٫۶۵

اگر beta ≡ ۰ mod q بود،

آن‌وقت (x,y) و (x',y') جفت متمایز حساب نمی‌شدند؛

پس حتماً beta != ۰ mod q است و چون q اول است،

beta در Z_q معکوس دارد:

beta_inv = beta^(-۱) mod q

حالا هر دو طرف را به توان beta_inv می‌رسانیم:

g^(alpha * beta_inv) = h

یعنی:

log_g(h) = alpha * beta_inv mod q

و این محاسبه در Z_q کاملاً کاراست.

این‌جا کل ماجرا روی این تکیه دارد که beta حتماً وارون‌پذیر باشد

و این دقیقاً به خاطر اول بودن q است.

# ربط prime-order به سختی DDH

برای سخت بودن DDH یک شرط شهودی لازم داریم: مقدار DH یعنی DH_g(h۱,h۲) خودش به‌تنهایی ، باید شبیه یک عنصر یکنواخت از گروه به نظر برسد.

در تعریف رسمی DDH ما دو توزیع را مقایسه می‌کنیم:

(h۱ , h۲ , DH_g(h۱,h۲))

(h۱ , h۲ , y) با yیکنواخت در گروه

اگر توزیع DH_g(h۱,h۲) از نظر آماری خیلی از توزیع یکنواخت فاصله داشته باشد،

یک حملهٔ آماری ساده DDH را می‌شکند.

کتاب می‌گوید (بدون اثبات دقیق) که:

وقتی اندازهٔ گروه q اول باشد،

توزیع DH_g(h۱,h۲) برای h۱,h۲ یکنواخت، تقریباً شبیه توزیع یکنواخت است (به معنایی دقیق که بعداً تعریف می‌کند).

اگر q ترکیبی باشد، این خواص لزوماً برقرار نیست و ممکن است ساختار اضافی ظاهر شود که به DDHآسیب بزند. پس برای کاربردهایی که DDH باید سخت باشد (مثل بسیاری از پروتکل‌های Diffie–Hellman و ElGamal)، استفاده از گروه prime-order بسیار طبیعی‌تر و امن‌تر است.

# کار کردن در گروه‌های $Z^*_p$

یکی از کلاس‌های مهم گروه‌های دوری که توشون DLog سخت **فرض** می‌شه، گروه‌های ضربی $Z^*_p$ هستن:

- p عدد اول n بیتی
- گروه $Z^*_p$ با اندازهٔ $q = p-1$
- g یک مولد (generator برای $Z^*_p$)

الگوریتم $G(1^n)$ این کار رو می‌کنه:

- یک عدد اول n بیتی p انتخاب می‌کند؛
- گروه را $Z^*_p$ و اندازه را $q = p-1$ می‌گیرد؛
- یک مولد g برای $Z^*_p$ پیدا می‌کند.

نمایش عناصر $Z^*_p$ خیلی ساده است: همان اعداد ۱ ... .۱-p

**فرض** رایج: مسئلهٔ لگاریتم گسسته (DLog) در این گروه **سخت** است.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# مشکل: $Z^*_p$ خودش prime-order نیست، و DDH خراب می‌شود

برای $p > ۳$، اندازهٔ $Z^*_p$ برابر $p-۱$ است که معمولاً اول نیست.

قبلاً گفتیم که از نظر رمزنگاری:

گروه‌های prime-order خوش‌دست‌ترند؛

و برای DDH (تصمیمی)، گروه‌های با اندازهٔ ترکیبی می‌توانند مشکل‌ساز شوند.

در واقع:

در بسیاری از حالات، DDH در گروه کامل $Z^*_p$ سخت نیست

(یعنی می‌شود با اطلاعات ساختاری، DH واقعی را از عنصر تصادفی تشخیص داد).

برای کاربردهایی که بر پایهٔ فرض DDH هستند (مثل ElGamal روی DH)،

استفادهٔ مستقیم از کل $Z^*_p$ قابل قبول نیست.

راه‌حل: به‌جای کل $Z^*_p$، از یک زیرگروهِ prime-order داخل آن استفاده کنیم.

# ساخت زیرگروهِ prime-order داخل Z*_p

- ایده : انتخاب کنیم که: p = r*q + ۱
- p و q هر دو اول باشند؛
- r را به این می‌گوییم.cofactor
- در این حالت، r*q = p-۱ است و می‌توانیم یک زیرگروه با اندازهٔ q بسازیم.
- تعریف گروه جدید:

$$\{[h^r \bmod p] \mid h \in \mathbb{Z}_p^*\}$$

یعنی G مجموعهٔ تمام «توان‌های r ام» در Z*_p است

)به این‌ها می‌گویند r-th residues)

- **قضیه ۹٫۶۶ (راست‌به‌اصل):**

اگر p = r*q + ۱ با p,q اول باشد،

آن‌وقت همین مجموعهٔ بالا:

G = { h^r mod p }

یک زیرگروه از Z*_p با اندازهٔ دقیقاً q است.

# ایدهٔ اثبات قضیه ۹٫۶۶

هدف: نشان بدهیم G زیرگروه است و اندازه‌اش q است. زیرگروه بودن (بسته‌بودن، وجود معکوس، همانی) ساده است و کتاب می‌گوید «اثباتش سرراست است». برای اندازه، این ترفند را می‌زنیم:

یک مولد g برای کل Z*_p در نظر می‌گیریم؛

پس عناصر Z*_p این هستند:

g^۰ , g^۱ , ... , g^(p-۲)

تابع f_rرا تعریف می‌کنیم:

f_r(g^i) = g^(i * r) mod p

این تابع از کل Z*_p به G می‌فرستد. نشان می‌دهیم این تابع r- به ۱- است: یعنی هر خروجی، دقیقاً از r ورودی متفاوت به‌دست می‌آید.

g^(i*r) = g^(j*r) اگر و تنها اگر ((i-j*r ≡ ۰ mod (p-۱))

چون p-۱ = r*q، این یعنی q، (i-j) را تقسیم می‌کند. پس برای هر j، دقیقاً r تا مقدار i داریم (در فاصله‌های qتایی) که همان خروجی را می‌دهند. از آن‌جا که تعداد کل عناصر Z*_p برابر p-۱ = r*q است، و هر خروجی G از r ورودی می‌آید،

نتیجه می‌شود:

G| = (p-۱) / r = q|

پس G یک زیرگروهِ prime-order داخل Z*_p است؛ همان چیزی که برای DLog / DDH می‌خواستیم.

# تولید عنصر یکنواخت در زیرگروه G

طبق قضیهٔ قبل، زیرگروه ما اینه:

G = {h^r mod p | h در Z*_p}

پس برای این‌که یک عنصر **یکنواخت** از G بسازیم، کافی است:

یک h یکنواخت از Z*_p انتخاب کنیم،

مقدار g = h^r mod p را حساب کنیم.

به‌خاطر این‌که در اثبات قضیه نشان داده شد نگاشت h -> h^r یک تابع r- به ۱- است،

اگر h یکنواخت باشد، توزیع h^r mod p هم یکنواخت روی تمام اعضای G می‌شود.

# تست عضویت: آیا h داخل G است؟

برای یک h در Z*_p می‌خواهیم بفهمیم آیا h داخل زیرگروهِ G هست یا نه.

کتاب می‌گوید کافی است این را چک کنیم:

h^q ?= ۱ mod p

ایدهٔ استدلال:

فرض کن g مولد کل Z*_p باشد، پس هر h را می‌شود به شکل h = g^i نوشت.

آن وقت:

h^q = (g^i)^q = g^(i*q)

شرط h^q = ۱ mod p یعنی g^(i*q) = ۱ mod p.

با توجه به این‌که مرتبهٔ g برابر p-۱ است، این معادل است با:

(p-۱) | (i*q)

حالا چون p-۱ = r * q است، بالا یعنی:

r * q | i * q ⇒ r | i

پس i = c * r و:

h = g^i = g^(c*r) = (g^r)^c

یعنی h توانِr ام یک عنصر است و دقیقاً داخل همان زیرگروهِ G قرار می‌گیرد.

خلاصه: **اگر و فقط اگر** h^q = ۱ mod p باشد، آن‌وقت h عضو G است.

# تولید گروه و زیرگروه‌های مرتبهٔ اول (۹.۳.۳)

از اینجا به بعد هدف این است که یک گروه امن بسازیم که:

محاسبات داخلش سریع باشد (ضرب/توان‌رسانی)

ولی حل Dlog / CDH / DDH داخلش سخت باشد

چرا زیرگروه مرتبهٔ اول (prime-order) مهم است؟

چون حمله‌های زیرگروه کوچک و مشکلات cofactor کمتر می‌شود

و تحلیل امنیتی تمیزتر و استانداردتر است

ایده‌ی کلیدی: از داخل $Z^*_p$ یا $E(Z_p)$، با یک تبدیل/توان‌رسانی به زیرگروه prime-order می‌رویم

# الگوریتم ۹٫۶۷ – الگوریتم تولید گروه G

الان کتاب همهٔ بحث بالا را در یک الگوریتم G جمع می‌کند که خروجی‌اش یک گروه دوری با مرتبهٔ اول q و یک مولد g است.

ورودی‌ها:

پارامتر امنیتی n^۱ (تعداد بیت‌های q)

یک تابع lambda = lambda(n) که طول p را مشخص می‌کند (تعداد بیت‌های p)

خروجی:

مدول p،

مرتبهٔ گروه q،

یک مولد g برای زیرگروهِ مرتبهٔ q

روند کار:

یک عدد اول p با طول ambda بیت و یک عدد اول q با طول n بیت انتخاب می‌کنیم،

طوری که q و p-۱ را تقسیم کند:

q | (p-۱)

(یعنی p-۱ = r * q برای یک r)

# ادامه الگوریتم ۹٫۶۷ – الگوریتم تولید گروه G

***ALGORITHM 9.67***
**A group-generation algorithm $\mathcal{G}$**

**Input:** Security parameter $1^n$, parameter $\ell = \ell(n)$
**Output:** Cyclic group $\mathbb{G}$, its (prime) order $q$, and a generator $g$

**choose** $\ell$-bit prime $p$ and $n$-bit prime $q$ such that $q \mid (p-1)$
  // we omit the details of how this is done
**until** $g \neq 1$ **do:**
  **choose** uniform $h \in \mathbb{Z}_p^*$
  **set** $g := [h^{(p-1)/q} \bmod p]$
**return** $p$, $q$, $g$  // $\mathbb{G}$ is the order-$q$ subgroup of $\mathbb{Z}_p^*$ generated by $g$

یک حلقه اجرا می‌کنیم تا وقتی g =! ۱ بشود:

یک h یکنواخت از Z*_p انتخاب می‌کنیم،

عدد

g = h^((p-۱)/q) mod p

را حساب می‌کنیم؛

اگر g = ۱ شد، دوباره از اول یک h جدید امتحان می‌کنیم.

در پایان، p , q , g را برمی‌گردانیم.

زیرگروهِ کاری ما همان زیرگروهِ مرتبهٔ q است که با g تولید می‌شود.

نکتهٔ نام‌گذاری پارامترها:

|q| = n تعداد بیت‌های مرتبهٔ گروه است،

|p| = lambda تعداد بیت‌های مدول است،

رابطهٔ دقیق بین این دو (برای توازن امنیت و کارایی) در ادامهٔ فصل توضیح داده می‌شود.

# انتخاب اندازهٔ $p$ و $q$

اینجا کتاب می‌خواهد بگوید طول بیت‌های $p$ و $q$ را چطور انتخاب کنیم که هم امن باشیم هم الکی کند نشویم.

بگذاریم

$|q| = n \rightarrow$ طول بیت‌های مرتبهٔ گروه (زیرگروهِ prime-order)،

$|p| = \lambda \rightarrow$ طول بیت‌های مدول $p$.

برای شکستن Dlog در زیرگروهِ مرتبهٔ $q$ دو نوع الگوریتم شناخته شده است:

الگوریتم‌های «جنریک گروهی» با زمان حدودی $\sqrt{q} \approx 2^{n/2}$.

الگوریتم‌های نوع index-calculus با زمان حدودی

$$2^{\mathcal{O}((\log p)^{1/3} \cdot (\log \log p)^{2/3})} = 2^{\mathcal{O}(\ell^{1/3} \cdot (\log \ell)^{2/3})}$$

وقتی یک سطح امنیتی مثلاً «امنیت در حد $2^n$ را هدف می‌گیریم، باید $\lambda$ را طوری انتخاب کنیم که **هیچ‌کدام** از این دو خانوادهٔ حمله خیلی سریع نشوند؛ یعنی سختی هر دو تقریباً در یک سطح باشد.

اگر $\lambda$ را **خیلی کوچک** بگیریم → حملهٔ نوع دوم سریع می‌شود و امنیت کم می‌شود.

اگر $\lambda$ را **خیلی بزرگ** بگیریم → محاسبات روی $G$ بی‌دلیل کند می‌شود،

در حالی که سختی حمله تقریباً تغییر خاصی نمی‌کند. در عمل، معمولاً کسی خودش این پارامترها را از صفر نمی‌سازد؛ استانداردها مثل (NIST) مقادیر آماده برای $p$، $q$ و یک مولد $g$ می‌دهند و پیاده‌سازی‌ها همان‌ها را استفاده می‌کنند.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# مثال ۹٫۶۸ – پیدا کردن زیرگروه‌های اول‌مرتبه در $Z^*_{11}$(۱)

| Powers of 2: | $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^7$ | $2^8$ | $2^9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Values: | 1 | 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 |

گروه ضربی $Z^*_{11}$ اندازهٔ ۱۰ دارد. می‌خواهیم ببینیم کدام اعداد، مولد کل گروه یا زیرگروه‌های آن هستند.

۱. امتحان عدد ۲

(اینجا جدول توان‌های ۲^ ۰... ۹۸۲ و مقادیرشان mod ۱۱ را که فرستادی قرار بده.) از جدول می‌بینیم: توان‌های ۲، تمام ۱۰ عنصر $Z^*_{11}$ را تولید می‌کنند. پس ۲ یک مولد کل گروه $Z^*_{11}$ است.

۲. امتحان عدد ۳

در کتاب جدول توان‌های ۳^ ۰... را می‌آورد. از آن جدول می‌بینیم: دنبالهٔ مقادیر فقط پنج عنصر را تولید می‌کند: {۱, ۳, ۹, ۵, ۴}

پس ۳ مولد کل گروه نیست، بلکه زیرگروهی با مرتبهٔ ۵ می‌سازد:

G = {۱, ۳, ۴, ۵, ۹}

۳. امتحان عدد ۱۰

جدول توان‌های ۱۱^۱۰ k mod نشان می‌دهد: فقط دو مقدار {۱, ۱۰} تکرار می‌شوند، پس ۱۰ زیرگروهی با مرتبهٔ ۲ می‌سازد: {۱, ۱۰}.

# مثال ۹٫۶۸ – استفاده از قضیهٔ ۹٫۶۶ روی (۲)۱ + ۲.۵ = ۱۱

| Powers of 2: | $2^0$ | $2^1$ | $2^2$ | $2^3$ | $2^4$ | $2^5$ | $2^6$ | $2^7$ | $2^8$ | $2^9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Values: | 1 | 2 | 4 | 8 | 5 | 10 | 9 | 7 | 3 | 6 |

برای کاربرد رمزنگاری، می‌خواهیم در یک **گروهِ** prime-order کار کنیم.

این‌جا داریم: ۱۱ = ۲ · ۵ + ۱ پس p = ۱۱ و ۲ · ۵ = ۱۰ = p−۱ است. می‌توانیم قضیهٔ ۹٫۶۶ را با دو انتخاب مختلف به کار ببریم:

**حالت اول: q = ۵ و r = ۲**

قضیه می‌گوید: **توان دوم همهٔ عناصر Z*_۱۱** (یعنی مربع‌ها) یک زیرگروهِ مرتبهٔ ۵ می‌سازند.

جدول «مربع‌ها» در کتاب نشان می‌دهد که مجموعه‌ی زیر دقیقاً همان زیرگروه است:

{۱, ۳, ۴, ۵, ۹}

بالاتر دیدیم که ۳ مولد این زیرگروه است؛ چون گروه مرتبهٔ ۵ (اول) است، هر عضو غیر از ۱ در این زیرگروه یک مولد است.

**حالت دوم: q = ۲ و r = ۵**

این‌بار قضیه می‌گوید: **توان پنجم عناصر Z*_۱۱** یک زیرگروهِ مرتبهٔ ۲ می‌سازند. اگر مقادیر h^۵ mod ۱۱ را حساب کنیم، می‌بینیم فقط {۱, ۱۰} به‌دست می‌آید؛ یعنی همان زیرگروه مرتبهٔ ۲ که ۱۰ مولد آن بود.

# توان‌های ۳ را پیمانه‌ی ۱۱ حساب می‌کنیم

| Powers of 3: | $3^0$ | $3^1$ | $3^2$ | $3^3$ | $3^4$ | $3^5$ | $3^6$ | $3^7$ | $3^8$ | $3^9$ |
|---|---|---|---|---|---|---|---|---|---|---|
| Values: | 1 | 3 | 9 | 5 | 4 | 1 | 3 | 9 | 5 | 4 |

دنباله‌ی مقادیر می‌شود: ۱, ۳, ۹, ۵, ۴, ۱, ۳, ۹, ۵, ۴, ...

قبل از این‌که همه‌ی ۱۰ عنصر $Z^*_{11}$ را ببینیم، مقدار ۱ دوباره تکرار می‌شود.

پس:

مرتبه‌ی ۳ برابر **۵** است؛

یعنی ۳ فقط مجموعه‌ی {۱, ۳, ۴, ۵, ۹} را تولید می‌کند؛

در نتیجه ۳ **مولد کل گروه** $Z^*_{11}$ نیست،

بلکه مولد یک **زیرگروهِ مرتبه‌ی ۵** داخل آن است.

# در این مرحله عدد ۱۰ را امتحان می‌کنیم.

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Powers of 10:** | $10^0$ | $10^1$ | $10^2$ | $10^3$ | $10^4$ | $10^5$ | $10^6$ | $10^7$ | $10^8$ | $10^9$ |
| **Values:** | 1 | 10 | 1 | 10 | 1 | 10 | 1 | 10 | 1 | 10 |

با نگاه به جدول توان‌های ۱۰ پیمانهٔ ۱۱ می‌بینیم که دنبالهٔ مقادیر فقط بین دو عدد ۱ و ۱۰ می‌چرخد و خیلی زود به ۱ برمی‌گردد.

نتیجه:

توان‌های ۱۰ فقط مجموعهٔ {۱ , ۱۰} را تولید می‌کنند.

پس ۱۰ مولد کل گروه $Z^*_{11}$ نیست،

بلکه یک زیرگروه از مرتبهٔ ۲ را می‌سازد.

برای کاربردهای رمزنگاری معمولاً می‌خواهیم در یک گروه با مرتبهٔ اول کار کنیم.

| Element: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Square: | 1 | 4 | 9 | 5 | 3 | 3 | 5 | 9 | 4 | 1 |

در این مثال داریم:

۱۱ = ۲ * ۵ + ۱

پس می‌توانیم قضیهٔ ۹٫۶۶ را به دو شکل به‌کار ببریم:

حالت اول: q = ۵ و r = ۲

قضیه می‌گوید مربعِ همهٔ عناصر $Z^*_{11}$ یک زیرگروهِ مرتبهٔ ۵ می‌سازد. جدولElement / Square این را نشان می‌دهد و می‌بینیم مجموعهٔ حاصل همان {۱,۳,۴,۵,۹} است.

از قبل دیدیم که ۳ مولد این زیرگروه است؛ چون مرتبهٔ زیرگروه ۵ (اول) است، هر عضو غیر از ۱ هم مولد همان زیرگروه خواهد بود.

حالت دوم: q = ۲ و r = ۵

این‌بار قضیهٔ ۹٫۶۶ می‌گوید توان‌های پنجم عناصر، یک زیرگروهِ مرتبهٔ ۲ می‌سازد.

با محاسبهٔ h^۵ mod ۱۱ درمی‌یابیم که این زیرگروه همان {۱ , ۱۰} است که قبلاً دیدیم ۱۰ مولد آن است. این مثال نشان می‌دهد چطور با تجزیهٔ p-۱ و استفاده از توان‌ها می‌توانیم از داخل $Z^*_p$ زیرگروه‌های prime-order مناسب برای DLog/DH بسازیم.

# زیرگروه‌های میدان‌های متناهی

تا این‌جا فقط روی زیرگروه‌های داخل $Z^*_p$ کار کردیم، که در واقع حالت خاصی از یک داستان کلی‌تر است.

کتاب می‌گوید می‌توانیم به‌جای فقط $Z^*_p$ از میدان‌های متناهی هم استفاده کنیم.

برای هر عدد اول $p$ و هر عدد صحیح $k \geq 1$ یک میدان متناهی یکتا وجود دارد که معمولاً با $F_{p^k}$ نشان می‌دهند و تعداد عناصرش $p^k$ است.

گروه ضربی این میدان، یعنی $F_{p^k}^*$، یک گروه دوری با اندازهٔ $p^k - 1$ است.

اگر:

$q$ یک عامل اول بزرگ از $p^k - 1$ باشد،

آن‌وقت طبق همان قضیهٔ ۹٫۶۶:

گروه $F_{p^k}^*$ یک زیرگروهِ دوری با مرتبهٔ $q$ دارد.

نکتهٔ مهم:

در اثبات قضیهٔ ۹٫۶۶ تنها چیزی که از $Z^*_p$ استفاده کردیم این بود که آن گروه دوری است؛

این خاصیت دربارهٔ $F_{p^k}^*$ هم صدق می‌کند، پس همان استدلال این‌جا هم کار می‌کند.

# ادامه زیرگروه‌های میدان‌های متناهی

نتیجهٔ عملی:

زیرگروه‌های prime-order فقط از $Z^*_p$ نمی‌آیند؛

می‌توانیم از زیرگروه‌های prime-order داخل $F_{p^k}^*$ هم استفاده کنیم،

و در آن‌ها هم DLog و Diffie–Hellman سخت **فرض** می‌شود.

تمام بحثی که در این بخش برای $Z^*_p$ داشتیم، در واقع **حالت ویژهٔ $k = 1$** از همین چارچوب کلی است.

انتخاب پارامترهای مناسب وقتی $k > 1$ باشد (برای کاربردهای واقعی) موضوع پیچیده‌ای است و کتاب می‌گوید خارج از محدودهٔ این فصل است.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# چرا منحنی بیضوی برای رمزنگاری جذابه؟

تا الان همهٔ گروه‌هامون از جنس حساب پیمانه‌ای معمولی بودن؛ مثل $Z^*_p$ یا زیرگروه‌هایش.

حالا یک کلاس جدید داریم: گروه نقاط روی منحنی‌های بیضوی.

نکته‌ی مهم و عملی:

برای $Z^*_p$ و میدان‌های متناهی معمولی، الگوریتم‌های زیرنمایی برای DLog وجود داره (یعنی از $2^n$ بهتر عمل می‌کنن).

برای گروه‌های مناسب روی منحنی بیضوی، تا الان هیچ الگوریتم زیرنمایی شناخته نشده.

نتیجهٔ رمزنگاری:

برای یک سطح امنیت ثابت،

استفاده از منحنی بیضوی ⇒ *گروه کوچک‌تر، اعداد کوتاه‌تر،*

| Category Type | ECC Key Size | RSA Key Size | Key Size Ratio |
|---|---|---|---|
| A | 112 | 512 | 1:5 |
| B | 163 | 1024 | 1:6 |
| C | 192 | 1536 | 1:8 |
| D | 224 | 2048 | 1:9 |
| E | 256 | 3072 | 1:12 |
| F | 384 | 7680 | 1:20 |
| G | 512 | 15,360 | 1:30 |

⇒ محاسبه و مخصوصاً حجم پیام‌ها خیلی کم‌تر از زیرگروه‌های $Z^*_p$ می‌شود.

برای پروتکل‌های مبتنی بر DLog / Diffie–Hellman، نسخهٔ بیضوی معمولاً از نظر ارتباطی به‌صرفه‌تر است.

# تعریف کلی منحنی بیضوی روی Z_p

در این بخش فرض می‌کنیم:

p ≥ ۵ یک عدد اول است.

برای ما، یک منحنی بیضوی روی Z_p با یک معادلهٔ درجهٔ سه در دو متغیر xو y تعریف می‌شود.

یک نمونهٔ استاندارد (فرم وایرستراس) این است:

$$y^2 = x^3 + Ax + B \pmod p$$

که در آن:

A و B عددهایی در Z_p هستند،

و باید شرط زیر برقرار باشد:

$$4A^3 + 27B^2 \equiv /0 \pmod p$$

این شرط یعنی معادلهٔ x^۳ + A x + B = ۰ (mod p) ریشهٔ تکراری نداشته باشد؛

بدون این شرط، ساختار گروهی قشنگی که می‌خواهیم خراب می‌شود.

به این نمایش می‌گویند نمایش وایرستراس (Weierstrass form) و کتاب می‌گوید هر منحنی بیضوی را می‌توان با یک تبدیل خطی معکوس روی x و y به این شکل نوشت.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# مجموعهٔ نقاط روی منحنی

حالا مجموعهٔ نقاط روی این منحنی را تعریف می‌کنیم.

می‌نویسیم: E(Z_p) = مجموعهٔ همهٔ زوج‌های (x, y) در Z_p × Z_p که معادلهٔ بالا را ارضا کنند:

$$y^2 = x^3 + Ax + B \bmod p,$$

به‌علاوه یک نقطهٔ ویژه با نام O.

پس:

$$E(\mathbb{Z}_p) \stackrel{\text{def}}{=} \{(x, y) \mid x, y \in \mathbb{Z}_p \text{ and } y^2 = x^3 + Ax + B \bmod p\} \cup \{\mathcal{O}\}.$$

به هر (x,y) که معادله را ارضا کند می‌گوییم یک نقطهٔ منحنی بیضوی.

O را نقطهٔ بی‌نهایت (point at infinity)می‌نامند؛

بعداً می‌بینیم نقش این نقطه این است که عضو همانی گروه باشد.

فعلاً همین سه ایده مهم است:

منحنی بیضوی = معادلهٔ y^۲ = x^۳ + A x + B (mod p) با شرط ۴A^۳ + ۲۷B^۲ ≠ ۰

نقاط منحنی = تمام (x,y) های حل این معادله روی Z_p.

یک نقطهٔ اضافی O هم اضافه می‌کنیم که بعداً در قانون جمع، نقش «صفر» گروهی را بازی می‌کند.

# باقیماندهٔ درجه‌دو و مثال روی Z_p

- برای p اولِ فرد، y یک QR است اگر x وجود دارد که. x^۲ ≡ y (mod p)
- در Z_p^* (یعنی عناصر غیرصفر)، دقیقاً نصف عناصر QR هستند.
- هر QR غیرصفر دقیقاً دو ریشه دارد : x و p-x.
- ۰هم QR است، ولی فقط یک ریشه دارد: ۰.
- پس وقتی می‌گوییم "نصف"، منظور معمولاً Z_p^* است.
- تست QR بودن در عمل با نماد لژاندر انجام می‌شود.
- محاسبه ریشه مربع (مثل Tonelli–Shanks) زمان چندجمله‌ای دارد.
- در ساخت نقطه روی منحنی، همین محاسبه لازم است.
- اگر f(x) غیر QR- باشد، آن x رد می‌شود و دوباره تلاش می‌کنیم.
- به همین دلیل "چند تلاش کوتاه" معمولاً کافی است.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# مثال ۹٫۶۹ – پیدا کردن نقاط روی منحنی روی Z_۷

تابع زیر را تعریف می‌کنیم:

$f(x) = x^{۳} + ۳x + ۳$

و منحنی زیر را در نظر می‌گیریم:

$E : y^{۲} = f(x) \pmod{۷}$

برای هر مقدار x از ۰ تا ۶، مقدار f(x) را پیمانهٔ ۷ حساب می‌کنیم و می‌بینیم آیا باقیماندهٔ درجه‌دو است یا نه:

→ $f(۰) = ۳ \pmod{۷}$ عدد ۳ در Z_۷ باقیماندهٔ درجه‌دو **نیست** → برای x = ۰ هیچ نقطه‌ای روی منحنی نداریم.

→ $f(۱) = ۰ \pmod{۷}$ تنها ریشهٔ دومش y = ۰ است → نقطه‌ی (۱,۰) روی منحنی است.

→ $f(۲) = ۳ \pmod{۷}$ دوباره ۳، که باقیماندهٔ درجه‌دو **نیست** → برا x = ۲ نقطه‌ای نداریم.

→ $f(۳) = ۴ \pmod{۷}$ یک باقیماندهٔ درجه‌دو است و دو ریشهٔ دوم y = ۲ و y = ۵ دارد → دو نقطه‌ی (۳,۲) و (۳,۵) روی منحنی هستند.

# ادامه مثال ۹٫۶۹ – پیدا کردن نقاط روی منحنی روی $Z_7$

$f(۴) = ۲ \pmod ۷ \rightarrow ۲$ هم باقیماندهٔ درجه‌دو است با ریشه‌های دوم $y = ۳$ و $y = ۴ \rightarrow$ دو نقطه‌ی (۴,۳) و (۴,۴) روی منحنی هستند.

$f(۵) = ۳ \pmod ۷ \rightarrow$ دوباره ۳، غیر درجه‌دو $\rightarrow$ نقطه‌ای نداریم.

$f(۶) = ۶ \pmod ۷ \rightarrow ۶$ هم غیر درجه‌دو $\rightarrow$ نقطه‌ای نداریم.

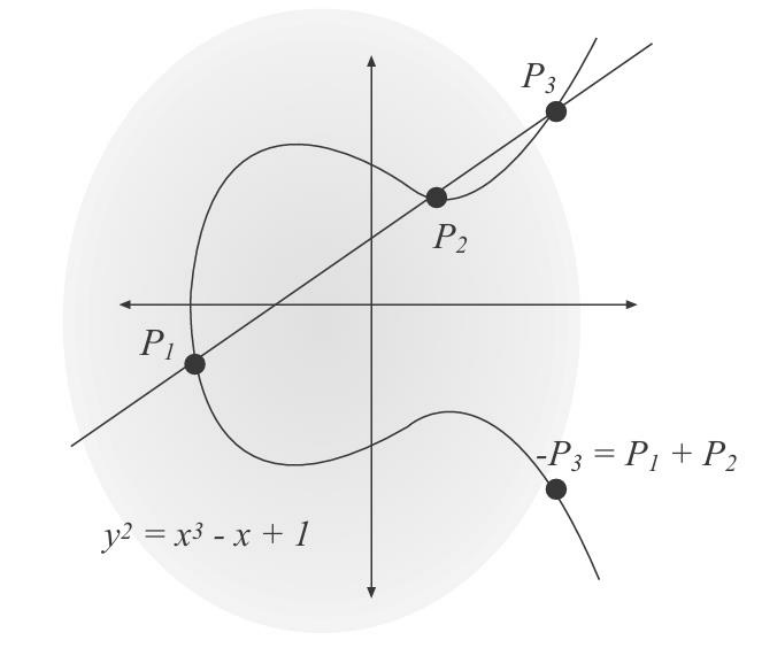
پس تا این‌جا پنج نقطه‌ی «واقعی» روی منحنی داریم:

(۱,۰) , (۳,۲) , (۳,۵) , (۴,۳) , (۴,۴)

اگر نقطهٔ خاص O (نقطهٔ بی‌نهایت) را هم اضافه کنیم، مجموعه‌ی نقاط روی این منحنی روی $Z_7$ دقیقاً ۶ عضو دارد.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# شهود هندسی – نگاه کردن به نمودار روی اعداد حقیقی




این شکل یک منحنی بیضوی روی اعداد حقیقی را نشان می‌دهد، مثلاً معادله‌ای مثل
y^۲ = x^۳ - x + ۱
خطی که از دو نقطه‌ی P۱ و P۲ می‌گذرد، معمولاً منحنی را در یک نقطه‌ی سوم هم قطع می‌کند؛ این نقطه را P۳ برای تعریف جمع، مختصات P۳ را نسبت به محور x قرینه می‌کنیم و نقطه‌ی جدید را P۱ + P۲می‌گیریم؛ روی شکل این نقطه با -P۳ = P۱ + P۲ نشان داده شده است .می‌نامیم .نقطه‌ی بی‌نهایت O را می‌توان مثل نقطه‌ای روی بالای محور y تصور کرد که همه‌ی خط‌های عمودی بعد از قرینه‌سازی به آن ختم می‌شوند و نقش عضو همانی گروه را دارد.

# شهود هندسی – نگاه کردن به نمودار روی اعداد حقیقی

برای فهم شهودی $E(Z_p)$، کتاب پیشنهاد می‌کند به نمودار همان معادله روی اعداد حقیقی نگاه کنیم:

معادله‌ی حقیقی:

$$y^2 = x^3 + A x + B$$

(اینجا در مثال‌مان $A = 3$ و $B = 3$ است.)

روی اعداد حقیقی، این معادله یک منحنی نرم و پیوسته می‌دهد با بی‌نهایت نقطه؛

روی $Z_p$ ما فقط تعداد محدودی نقطه داریم، اما شکل کلی منحنی واقعی کمک می‌کند رفتار گروه را تجسم کنیم.

نقطه‌ی O ( نقطهٔ بی‌نهایت) را می‌توان به‌صورت شهودی این‌گونه دید:

روی شکل حقیقی، O انگار بالای محور y نشسته است،

و «روی همه‌ی خط‌های عمودی» قرار دارد؛ یعنی هر خط عمودی را طوری در نظر می‌گیریم که علاوه بر دو نقطهٔ معمولش، در نقطهٔ بی‌نهایت هم آن را قطع می‌کند.

# تعریف عمل جمع روی نقاط منحنی بیضوی

کتاب حالا روی $E(Z_p)$ یک عمل دوتایی تعریف می‌کند که به آن «جمع» (addition) می‌گوید و با + می‌نویسد، طوری که نقاط منحنی با این عمل، یک گروه تشکیل بدهند. قواعد جمع:

عنصر واحد (identity) :

نقطه‌ی O نقش صفر را بازی می‌کند. برای هر نقطه P روی منحنی داریم:

$$P + O = O + P = P$$

جمع دو نقطه‌ی معمولی:

دو نقطه‌ی P۱ و P۲ روی منحنی را در نظر بگیر (هیچ‌کدام O نباشند). خطی که از P۱ و P۲ می‌گذرد رسم می‌کنیم. اگر P۱ = P۲ باشد، به‌جای خط بین دو نقطه، خط مماس منحنی در P۱ را رسم می‌کنیم. این خط منحنی را در یک نقطهٔ سوم هم قطع می‌کند؛ آن را P۳ بنام. اگر خط عمودی باشد، این نقطه‌ی سوم همان O است. اگر P۳ = (x,y) یک نقطهٔ معمولی باشد، جمع را این‌طور تعریف می‌کنیم:

$P_1 + P_2 = (x, -y)$ (منظور از -y وارونهٔ y پیمانه‌ی p است.)

اگر P۳ = O باشد، تعریف می‌کنیم:

$$P_1 + P_2 = O$$

به‌صورت تصویری: خط را می‌کشی، نقطهٔ سوم P۳ را پیدا می‌کنی، بعد آن را نسبت به محور x قرینه می‌کنی؛ نتیجه می‌شود P۱ + P۲.

# معکوس هر نقطه روی منحنی

اگر $P = (x,y)$ یک نقطه‌ی غیر از $O$ روی منحنی باشد، نقطه‌ی

$-P = (x, -y)$

هم روی منحنی است و نقش **معکوس جمعی** را دارد؛ یعنی:

$P + (-P) = O$

چرا؟

خطی که از $(x,y)$ و $(x,-y)$ می‌گذرد عمودی است.

طبق قانون جمع، وقتی خط عمودی باشد، نقطه‌ی سوم تقاطع را همان $O$ می‌گیریم.

پس جمع این دو نقطه $O$ می‌شود.

اگر $y = ۰$ باشد، آن‌وقت $P = (x,y)$ و $-P = (x,-y)$ در واقع یکسان می‌شوند (چون $-۰ = ۰$ است). در این حالت، خط مماس در $P$ عمودی می‌شود و باز هم:

$P + P = O$

و طبق تعریف گروه، $O$ خودش معکوس خودش است:

$O + O = O$

# محاسبهٔ جمع P۱ + P۲ به‌صورت فرمولی

دو نقطه روی منحنی داریم:

$P۱ = (x۱, y۱)$ و $P۲ = (x۲, y۲)$ در $E(Z_p)$

فرض می‌کنیم:

P۱و P۲ هر دو غیر از O هستند،

و $x۲ \neq x۱$ (حالت $x۱ = x۲$ یعنی جمع یک نقطه با خودش، جداگانه و کمی سخت‌تر است و کتاب بعداً برایش فرمول می‌دهد).

۱. شیب خط بین دو نقطه

شیب خطی که از این دو نقطه می‌گذرد:

$$s = (y۲ - y۱) * (x۲ - x۱)^{(-۱)} \pmod p$$

اینجا $(x۲ - x۱)^{(-۱)}$ یعنی معکوس ضربی عدد $x۲ - x۱$ پیمانهٔ p.

چون $x۲ \neq x۱$ و p اول است، این معکوس وجود دارد.

۲. معادلهٔ خط بین دو نقطه

معادلهٔ این خط روی $Z_p$ :

$$y = s * (x - x۱) + y۱ \pmod p$$ ... (معادل فرمول ۹.۳ کتاب)

# پیدا کردن نقطهٔ سوم و نتیجهٔ جمع

برای پیدا کردن نقطهٔ سوم تقاطع این خط با منحنی، این معادلهٔ خط را در معادلهٔ منحنی جای‌گذاری می‌کنیم: منحنی در فرم وایرستراس:

$$y^2 = x^3 + A * x + B \pmod p$$

به‌جای y بنویس:

$$y = s * (x - x_1) + y_1$$

بعد از بسط و ساده‌سازی (که کتاب می‌گوید کارش خسته‌کننده است)، به این نتیجه می‌رسیم که سه جواب برای x وجود دارد:

$x_1$

$x_2$

و یک مقدار جدید $x_3$ که همان مختصات x نقطهٔ سوم است. این $x_3$ به‌صورت زیر به‌دست می‌آید:

$$x_3 = s^2 - x_1 - x_2 \pmod p$$

حالا که $x_3$ را داریم، مختصات y- نقطهٔ سوم را از معادلهٔ خط حساب می‌کنیم:

$$y_3 = s * (x_3 - x_1) + y_1 \pmod p$$

اینجا $P_3 = (x_3, y_3)$ همان نقطه‌ای است که خط، برای بار سوم منحنی را قطع کرده است.ولی طبق قانون جمع روی منحنی، جمع $P_2 + P_1$ برابر **قرینهٔ عمودی** $P_3$ است.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# قضیهٔ ۹٫۷۰ – قانون جمع روی منحنی بیضوی

منحنی بیضوی روی پیمانهٔ اول $p \geq 5$ به صورت

$$y^2 = x^3 + A x + B \pmod p$$

با شرط $4A^3 + 27B^2 \neq 0 \pmod p$ در نظر گرفته می‌شود. دو نقطهٔ $P_1 = (x_1, y_1)$ و $P_2 = (x_2, y_2)$ روی $E(Z_p)$ (و هر دو $\neq O$) داریم: اگر $x_2 \neq x_1$ باشد، جمع آن‌ها $P_1 + P_2 = (x_3, y_3)$ است که در آن:

$$s = (y_2 - y_1) * (x_2 - x_1)^{-1} \pmod p$$

$$x_3 = s^2 - x_1 - x_2 \pmod p$$

$$y_3 = s * (x_1 - x_3) - y_1 \pmod p$$

اگر $x_1 = x_2$ و $y_1 \neq y_2$ باشد، دو نقطه روی یک خط عمودی‌اند، بنابراین

$$P_1 + P_2 = O.$$

اگر $P_1 = P_2$ و $y_1 \neq 0$ باشد (دو برابر کردن نقطه)،

$P_1 = (x_3, y_3)$ با:

$$s = (3 x_1^2 + A) * (2 y_1)^{-1} \pmod p$$

$$x_3 = s^2 - 2 x_1 \pmod p$$

$$y_3 = s * (x_1 - x_3) - y_1 \pmod p$$

اگر $P_1 = P_2$ و $y_1 = 0$ باشد، دو برابر کردن نقطه به نقطهٔ بی‌نهایت می‌رسد و

$$P_1 = O. \quad ۲$$

# مثال ۹٫۷۱ – بررسی شرکت‌پذیری (۱)

روی منحنی مثال ۹٫۶۹ در پیمانهٔ ۷، سه نقطه داریم:

$P_1 = (0, 1)$ و $P_2 = P_3 = (3, 4)$.

گام‌های محاسبهٔ $P_3 = (P_1 + P_2) + P_3$:

برای $P_1 + P_2$ :

$s = (0 - 3) * (1 - 4)^{(-1)} \pmod 7 = 1$

$x_3 = 1^2 - 1 - 4 = 3 \pmod 7$

$y_3 = 1 * (3 - 1) - 0 = -2 = 5 \pmod 7$

پس $Q = P_1 + P_2 = (5, 3)$.

برای $Q + P_3$ :

$s = (5 - 3) * (3 - 4)^{(-1)} \pmod 7 = 5$

$x_3 = 5^2 - 3 - 4 = 4 \pmod 7$

$y_3 = 5 * (4 - 3) - 5 = -1 - 5 = -6 = 4 \pmod 7$

در نتیجه $P_3 = (P_1 + P_2) + P_3 = (4, 4)$.

# مثال ۹٫۷۱ – بررسی شرکت‌پذیری (۲)

حالا گام‌های محاسبهٔ :(P۳ + P۲) + P۱

ابتدا : P۲ + P۳ = ۲ P۲

s = (۳ * ۴^۲ + ۳) * (۲ * ۳)^(-۱) (mod ۷) = ۵

x۳ = ۵^۲ - ۲ * ۴ = ۳ (mod ۷)

y۳ = ۵ * (۴ - ۳) - ۳ = ۲ (mod ۷)

پس .(۳ , ۲) = P۲ + P۳ = Q'

سپس : P۱ + Q'

s = (۲ - ۰) * (۳ - ۱)^(-۱) (mod ۷) = ۱

x۳ = ۱^۲ - ۱ - ۳ = ۴ (mod ۷)

y۳ = ۱ * (۱ - ۴) - ۰ = -۳ = ۴ (mod ۷)

بنابراین .(۴ , ۴) = P۱ + (P۲ + P۳)

نتیجه می‌شود:

(P۱ + P۲) + P۳ = P۱ + (P۲ + P۳)

و در این مثال خاصیت شرکت‌پذیری جمع روی منحنی بیضوی دیده می‌شود.

# مسئلهٔ لگاریتم گسسته و دیفی–هلمن روی منحنی بیضوی

در گروه‌های منحنی بیضوی نمایش به صورت جمعی است و xP یعنی جمع‌کردن نقطهٔ P با خودش x بار.

مسئلهٔ لگاریتم گسسته:

داده‌ها: یک نقطهٔ پایه P و نقطهٔ Q = xP.

هدف: پیدا کردن عدد صحیح x.

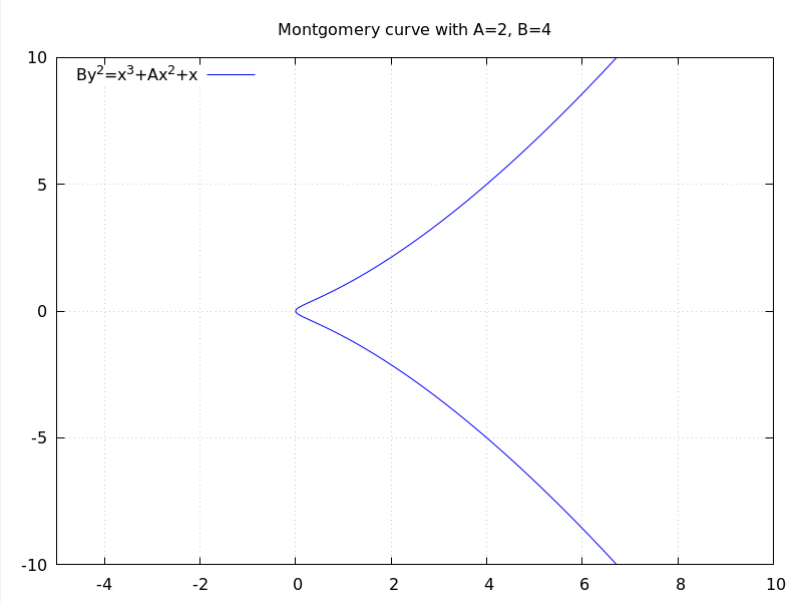
مسئلهٔ دیفی–هلمن تصمیمی روی منحنی بیضوی:

داده‌ها: سه نقطهٔ (aP , bP , T)که در آن a و b و c به‌صورت یکنواخت از Z_q انتخاب می‌شوند.

باید تشخیص داده شود که T = abP است (یعنی سه‌تایی از شکل (aP , bP , abP) است) یا این‌که T = cP بوده و سه‌تایی از شکل (aP , bP , cP) است.

این دو مسئله در گروه‌ها یا زیرگروه‌های منحنی بیضوی با مرتبهٔ اول بزرگ، تحت چند شرط فنی روی پارامترها، سخت فرض می‌شوند.

# نمایش مونتگمری

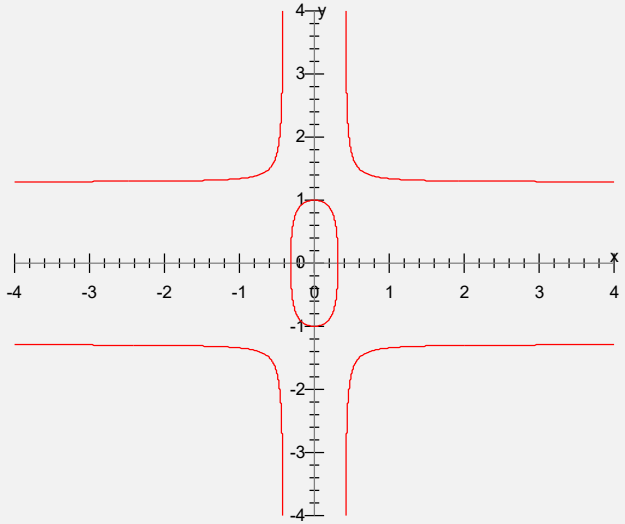



نمایش وایرشتراس تنها شکل نوشتن منحنی بیضوی نیست. برای کارایی و امنیت پیاده‌سازی (مثلاً مقاومت در برابر حملات کانال جانبی)، نمایش مونتگمری هم استفاده می‌شود. در این نمایش، منحنی روی پیمانهٔ pبه صورت زیر است:

B y^۲ = x^۳ + A x^۲ + x (mod p)

که در آن : B صفر نیست پیمانهٔ p، و A نیز برابر ۲ یا -۲ پیمانهٔ p نیست. مجموعهٔ نقاط شامل تمام زوج‌های (x , y) در Z_p × Z_p است که این معادله را ارضا می‌کنند، به‌اضافهٔ نقطهٔ بی‌نهایت O قانون جمع نقاط همان ایدهٔ هندسی قبلی را دارد، ولی فرمول‌هایش متفاوت از فرم وایرشتراس است. هر منحنی را نمی‌توان به فرم مونتگمری نوشت؛ مرتبهٔ هر گروه منحنی بیضوی در این فرم، مضربی از ۴ است.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش




# نمایش ادواردز پیچ‌خورده

نمایش ادواردز پیچ‌خورده از معادله‌ای به شکل زیر استفاده می‌کند:

$$a\,x^2 + y^2 = 1 + d\,x^2\,y^2 \pmod p$$

که در آن:

a و d هر دو ناصفر پیمانهٔ p هستند، و a با d برابر نیست (پیمانهٔ p) حالت خاص a = ۱ نمایش ادواردز نامیده می‌شود. این نمایش همان خانوادهٔ منحنی‌هایی را پوشش می‌دهد که با نمایش مونتگمری هم قابل بیان‌اند. در این‌جا هم $E(Z_p)$ مجموعهٔ نقاط روی منحنی است، ولی دیگر نیازی به نقطهٔ ویژهٔ بی‌نهایت نیست؛ نقطهٔ (۰ , ۱) خودش عنصر همانی گروه است. وقتی a یک باقی‌ماندهٔ مربعی و d یک نامربعی مربعی پیمانهٔ p باشد، قانون جمع ساده است. برای

$$P_1 = (x_1 , y_1)$$ و $$P_2 = (x_2 , y_2)$$:

$$x_3 = (x_1\,y_2 + x_2\,y_1) / (1 + d\,x_1\,x_2\,y_1\,y_2) \pmod p$$

$$y_3 = (y_1\,y_2 - a\,x_1\,x_2) / (1 - d\,x_1\,x_2\,y_1\,y_2) \pmod p$$

و $P_1 + P_2 = (x_3 , y_3)$ به‌دست می‌آید. در این فرم، جمع با همین دو کسر انجام می‌شود و نیازی به بررسی حالت‌های متعدد مثل قضیهٔ ۹٫۷۰ نیست، و نوشتن و پیاده‌سازی عملیات منحنی بیضوی ساده‌تر می‌شود.

آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# انتخاب گروه منحنی بیضوی و حد هاسه

برای کاربردهای رمزنگاری، نیاز به گروه منحنی بیضوی با مرتبهٔ بزرگ است. پرسش مهم این است که اندازهٔ |E(Z_p)|چقدر است.

در نمایش وایرشتراس، معادلهٔ y^۲ = f(x) (mod p) تقریباً برای نصف مقادیر x دو جواب y و برای بعضی x ها یک جواب دارد. شهودی انتظار می‌رود حدود p + ۱نقطه (همراه با O) روی منحنی باشد.

حد هاسه این شهود را دقیق می‌کند و می‌گوید برای هر عدد اول p و منحنی E روی Z_p داریم:

$$p + ۱ - ۲ * \sqrt{p} \le |E(Z_p)| \le p + ۱ + ۲ * \sqrt{p}$$

یعنی می‌توان نوشت $|E(Z_p)| = p + ۱ - t$ که در آن $|t| \le ۲ * \sqrt{p}$ است. عدد t را ردّ منحنی (trace) می‌نامند.

الگوریتم‌های کارای محاسبهٔ |E(Z_p)| (یا همان t) وجود دارد.

از حد هاسه نتیجه می‌شود پیدا کردن یک نقطه روی منحنی ساده است:

# ادامه انتخاب گروه منحنی بیضوی و حد هاسه

یک x یکنواخت از Z_p انتخاب می‌شود،

مقدار f(x) حساب می‌شود،

اگر صفر یا باقی‌ماندهٔ مربعی بود، یکی از ریشه‌های مربعی آن به عنوان y گرفته می‌شود و نقطهٔ (x , y) به‌دست می‌آید.

چون تعداد نقاط روی منحنی زیاد است، معمولاً به تلاش‌های کمی نیاز است.

برای رمزنگاری، معمولاً در یک (زیر)گروه با مرتبهٔ اول کار می‌شود:

اگر |E(Z_p)| خودش عدد اول باشد، مستقیماً در گروه E(Z_p) کار می‌کنیم.

اگر |E(Z_p)| اول نباشد ولی یک عامل اول بزرگ q داشته باشد، در زیرگروهی با مرتبهٔ q کار می‌کنیم.

اگر

$|E(Z_p)| = r * q$ با q اول و $r < q$

باشد (که r را ضریب یا cofactor می‌نامند)، آنگاه مجموعهٔ

$G = \{r P \mid P \text{ در } E(Z_p)\}$

یک زیرگروه از E(Z_p) با مرتبهٔ q می‌سازد و برای سازوکارهای دیفی–هلمن و امضاهای مبتنی بر منحنی بیضوی استفاده می‌شود.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# حد هاسه و ردّ منحنی

قضیهٔ ۹٫۷۲ (حد هاسه) می‌گوید اگر p عدد اول و E یک منحنی بیضوی روی $Z_p$ باشد، تعداد نقاط آن چنین محدود است:

$$p + ۱ - ۲\sqrt{p} \le |E(Z_p)| \le p + ۱ + ۲\sqrt{p}.$$

بنابراین همیشه می‌توان نوشت:

$|E(Z_p)| = p + ۱ - t$، که در آن $|t| \le ۲\sqrt{p}.$

به t می‌گویند «ردّ منحنی» (trace) لگوریتم‌های کارای محاسبهٔ $|E(Z_p)|$ یا همان t وجود دارند، ولی وارد جزئیاتشان در این فصل نمی‌شویم.

# پیامد حد هاسه: پیدا کردن یک نقطه روی منحنی

چون |E(Z_p)|تقریباً برابر p است، نقطه روی منحنی کم نیست. بنابراین پیدا کردن یک نقطه آسان است:

۱. یک x یکنواخت از Z_p انتخاب کن.

۲. مقدار f(x) را در معادلهٔ y^۲ = f(x) (mod p) حساب کن.

۳. اگر f(x) صفر یا باقیماندهٔ مربعی پیمانهٔ p بود، یکی از ریشه‌های مربعی آن را به‌عنوان y بگیر.

نقطهٔ (x , y) روی منحنی است. به‌طور متوسط با چند تلاش کوتاه به نقطه می‌رسیم.

(آزمون مربعی بودن و محاسبهٔ ریشهٔ مربعی پیمانهٔ p در فصل ۱۵ توضیح داده شده‌اند.)


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# زیرگروهِ مرتبهٔ اول و cofactor

برای استفادهٔ رمزنگاری، دوست داریم در یک (زیر)گروه با مرتبهٔ اول (prime) کار کنیم.
اگر خودِ |E(Z_p)|عدد اول باشد، مستقیماً از گروه کامل E(Z_p) استفاده می‌کنیم.
اگر نه، ولی |E(Z_p)| یک عامل اول بزرگ داشته باشد، در زیرگروه متناظر آن عامل کار می‌کنیم. اگر

$$|E(Z_p)| = r * q$$

با q عدد اول و r < q باشد (به r می‌گویند cofactor و داریم gc(r , q) = ۱)، آنگاه مجموعهٔ

$$G = \{ rP \mid P \text{ در } E(Z_p) \}$$

یک زیرگروه با مرتبهٔ q می‌سازد. این ساخت دقیقاً شبیه ساخت زیرگروه‌های مرتبهٔ اول در Z_p بود، فقط این‌جا E(Z_p) لزوماً چرخه‌ای کامل نیست.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# منحنی امن یعنی جایی که لگاریتم گسسته «واقعاً سخت» باشد

هدف در رمزنگاری این است که منحنی‌ای انتخاب شود که بهترین الگوریتم شناخته‌شده برای لگاریتم گسسته روی آن، از نوع عمومی و نمایی باشد؛ یعنی هیچ الگوریتم زیرنمایی ویژه‌ای برای آن گروه وجود نداشته باشد.

چند خانواده منحنی از این نظر ضعیف‌اند و باید کنار گذاشته شوند:

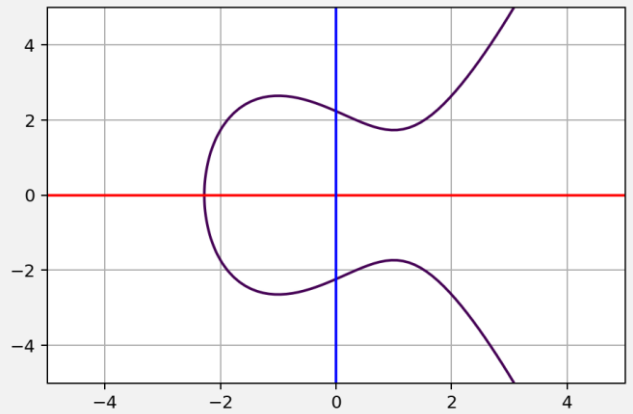
منحنی‌هایی که |E(Z_p)| = p دارند؛ در این حالت لگاریتم گسسته را می‌توان در زمان چندجمله‌ای حل کرد.

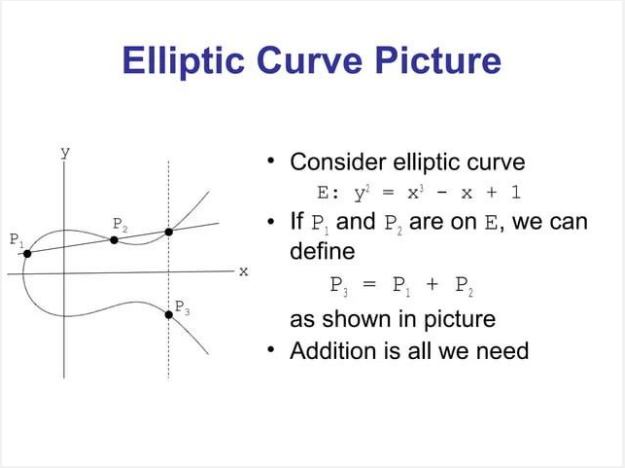
منحنی‌هایی که |E(Z_p)| مقسوم‌علیه p^k − ۱ رای k کوچک است؛ در این حالت مسئلهٔ لگاریتم گسسته روی E(Z_p) به لگاریتم گسسته در میدان F_{p^k} کاهش می‌شود، و برای آن میدان الگوریتم‌های زیرنمایی شناخته شده‌اند.

برای همین در عمل معمولاً منحنی‌های استانداردی را که توسط NIST یا سایر نهادها پیشنهاد شده‌اند انتخاب می‌کنند و ساخت «منحنی شخصی» توصیه نمی‌شود.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# فشرده‌سازی نقطه (Point Compression)




نمایش مستقیم نقطه P = (x , y) روی منحنی، دو عدد پیمانهٔ p می‌خواهد. می‌توان تقریباً نصف این فضا را صرفه‌جویی کرد برای هر x در Z_p حداکثر دو y ممکن است معادلهٔ y^۲ = f(x) (mod p) را ارضا کنند: دو مقدار y و −y

پس برای مشخص‌کردن نقطه کافی است:

عدد x را ذخیره کنیم،

یک بیت b نگه داریم که تعیین کند «کدام» از این دو y مدنظر است (برای مثال: اگر y زوج باشد b = ۰، اگر فرد باشد b = ۱) بازسازی نقطه از روی (x , b) این‌گونه است:

۱. دو ریشهٔ مربعی y۱ و y۲ از معادلهٔ y^۲ = f(x) (mod p) را حساب می‌کنیم.

۲. یکی از آن‌ها زوج و دیگری فرد است (یا هر دو صفر).

۳. بسته به مقدار b، ریشهٔ زوج یا فرد را به‌عنوان y انتخاب کرده و نقطهٔ (x , y) را بازسازی می‌کنیم.

به این ترتیب، هزینهٔ ارسال/ذخیره‌سازی نقاط تقریباً نصف می‌شود.

# مختصات پروژکتیو: عوض‌کردن نمایش برای حذف معکوس‌گیری

تا این‌جا نقطه‌ها را به‌صورت زوج $(x , y)$ نمایش می‌دادیم؛ به این می‌گویند «مختصات افاین».

در «مختصات پروژکتیو» هر نقطه $P = (x , y) \neq O$ را با یک سه‌تایی $(X , Y , Z)$ نمایش می‌دهیم که:

$$x = X / Z \pmod p$$

$$y = Y / Z \pmod p$$

و $Z \neq ۰$ است. نقطهٔ بی‌نهایت $O$ با هر سه‌تایی از شکل $( , ۰ , ۰Y)$ و $Y \neq ۰$ نمایش داده می‌شود. تبدیل بین این دو نمایش ساده است:

افاین به پروژکتیو: $(x , y) \neq (x , y , ۱) \mapsto O$

پروژکتیو به افاین: $(X , Y , Z)$ با $Z \neq ۰$ $\mapsto (X/Z , Y/Z)$ پیمانهٔ $p$

مزیت اصلی این نمایش: در فرمول‌های جمع دو نقطه در مختصات افاین، باید معکوس $(x۲ - x۱)$ پیمانهٔ $p$ را حساب کنیم؛ معکوس‌گیری گران‌تر از جمع و ضرب است. در مختصات پروژکتیو، با استفاده از این که هر نقطه نمایش‌های متعددی دارد، فرمول‌های جمع را طوری بازنویسی می‌کنند که همه‌چیز فقط با جمع و ضرب انجام شود و هیچ معکوس‌گیری صریحی لازم نباشد. کتاب این کار را به‌صورت نمادین انجام می‌دهد:

# ادامه مختصات پروژکتیو: عوض‌کردن نمایش برای حذف معکوس‌گیری

مزیت اصلی این نمایش:

در فرمول‌های جمع دو نقطه در مختصات افاین، باید معکوس (x۲ − x۱) پیمانهٔ p را حساب کنیم؛ معکوس‌گیری گران‌تر از جمع و ضرب است.

در مختصات پروژکتیو، با استفاده از این که هر نقطه نمایش‌های متعددی دارد، فرمول‌های جمع را طوری بازنویسی می‌کنند که همه‌چیز فقط با جمع و ضرب انجام شود و هیچ معکوس‌گیری صریحی لازم نباشد.

کتاب این کار را به‌صورت نمادین انجام می‌دهد:

دو نقطه P۱ = (X۱ , Y۱ , Z۱) و P۲ = (X۲ , Y۲ , Z۲) را در نظر می‌گیرد،

s را به‌صورت «اختلاف y ها ضرب در معکوس اختلافxها» می‌نویسد،

سپس هر سه مختصات نتیجه را در یک ضریب مشترک مثل Z۱ * Z۲ * (X۲ Z۱ − X۱ Z۲)^۳ ضرب می‌کند تا معکوس از فرمول حذف شود،

و در نهایت به فرمول‌هایی برای X۳ , Y۳ , Z۳ می‌رسد که فقط شامل جمع و ضرب هستند.

نکتهٔ اصلی: در این دستگاه مختصات می‌توان جمع نقاط را بدون هیچ معکوس‌گیری پیمانه‌ای انجام داد که از نظر اجرا بسیار ارزان‌تر است.

# چند نمایش، چند ظرافت امنیتی

چون در مختصات پروژکتیو یک نقطه نمایش‌های متعددی دارد، شکل ظاهری (X , Y , Z) ممکن است اطلاعاتی در مورد نحوهٔ محاسبهٔ آن نقطه بدهد؛ این گاهی می‌تواند در حملات کانال جانبی یا نشت اطلاعات به کار مهاجم بیاید.

برای کاهش این خطر:

برای ذخیره و ارسال، معمولاً از مختصات افاین (یا نسخه‌های فشرده‌شدهٔ آن) استفاده می‌شود.

مختصات پروژکتیو فقط به‌عنوان نمایش داخلی موقتی حین محاسبات به کار می‌رود و بعد دوباره نتیجه به شکل استاندارد برگردانده می‌شود.

# منحنی‌های بیضوی مشهور در عمل

به‌جای این‌که هر کس منحنی خودش را طراحی کند، عملاً از منحنی‌های استاندارد استفاده می‌شود که سال‌ها بررسی شده‌اند و هم از نظر امنیت و هم از نظر سرعت، وضعیت خوبی دارند. چند نمونه:

- خانوادهٔ P-۲۵۶، P-۳۸۴، P-۵۲۱

P-۲۵۶ یا (secp۲۵۶r۱) روی $Z_p$ تعریف شده که p یک عدد اول ۲۵۶ بیتیِ خاص است:

$p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$.

این شکلِ خاصِ p باعث می‌شود محاسبات پیمانهٔ p سریع پیاده‌سازی شوند.

معادلهٔ منحنی: $y^2 = x^3 - 3x + B \pmod p$ با یک ثابت مشخص B.

انتخاب $A = -3$ (یا معادل آن ۳) برای ساده‌شدن فرمول‌های جمع و دوبرابرکردن است.

مرتبهٔ این منحنی یک عدد اول بزرگ است؛ پس خودِ $E(Z_p)$ همان گروه هدف است.

P-۳۸۴ و P-۵۲۱ نسخه‌های مشابه روی اعداد اول ۳۸۴ و ۵۲۱ بیتی هستند.

- Curve۲۵۵۱۹ و Ed۲۵۵۱۹

# ادامه منحنی‌های بیضوی مشهور در عمل

روی $Z_p$ با $p = ۲^{۲۵۵} - ۱۹$ تعریف می‌شود؛ باز هم شکل p برای پیاده‌سازی سریع ضرب و کاهش پیمانه‌ای انتخاب شده است. این منحنی در فرم مونتگمری به نام Curve۲۵۵۱۹ و در فرم ادواردز پیچ‌خورده به نام Ed۲۵۵۱۹ استفاده می‌شود.

گروه کامل مرتبهٔ اول نیست، اما زیرگروهی با مرتبهٔ اول بزرگ دارد که برای تبادل کلید دیفی–هلمن و برای امضاها استفاده می‌شود. در بسیاری از پروتکل‌های مدرن (مانند TLS جدیدتر و انواع پیام‌رسان‌ها) از این منحنی استفاده می‌شود.

- secp۲۵۶k۱

منحنی‌ای با مرتبهٔ اول روی $Z_p$ که در آن:

$p = ۲^{۲۵۶} - ۲^{۳۲} - ۲^{۹} - ۲^{۸} - ۲^{۷} - ۲^{۶} - ۲^{۴} - ۱.$

معادلهٔ آن: $y^۲ = x^۳ + ۷ \pmod p$

این یک منحنی Koblitz است؛ ساختار جبری خاصی دارد که به بهینه‌سازی پیاده‌سازی کمک می‌کند. معروف‌ترین کاربردش: به‌عنوان منحنی اصلی در سامانهٔ رمزنگاری بیت‌کوین برای کلیدهای عمومی و امضاها.

# شروع ۹٫۴ – کاربردهای رمزنگاری

تا این‌جا در فصل ۹، روی دو چیز خیلی وقت گذاشتیم:

از یک طرف، ابزارهای ریاضی مثل نظریهٔ اعداد و گروه‌ها را ساختیم؛

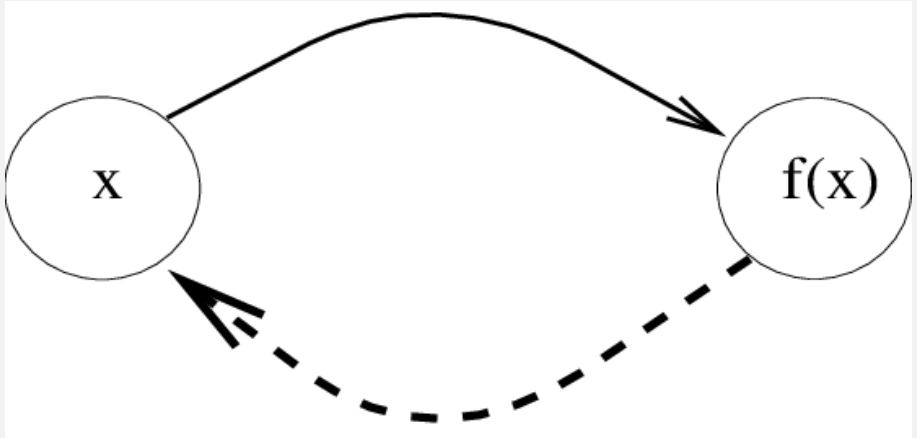
از طرف دیگر، چند فرض محاسباتی مهم تعریف کردیم (سختی فاکتورگیری، مسئلهٔ RSA، لگاریتم گسسته، دیفی–هلمن، گروه‌های اول، منحنی‌های بیضوی و ...) که «باور عمومی» این است که حل‌شان در عمل غیرممکن یا بسیار سخت است.

در ادامهٔ فصل و بقیهٔ کتاب، همین فرض‌ها تبدیل می‌شوند به اجزای واقعی رمزنگاری: تابع‌های یک‌طرفه، توابع هش، مولدهای شبه‌تصادفی، طرح‌های رمزنگاری کلید-عمومی، امضاها و...

این بخش ۹٫۴ یک شروع کوتاه و نمونه‌وار است تا ببینیم این سنگ‌بنای ریاضی چطور به «پروتکل و سیستم واقعی» تبدیل می‌شود؛ جزئیات کامل‌تر به فصل‌های بعد موکول می‌شود.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# تابع‌های یک‌طرفه – ایده و نقششان




تابع‌های یک‌طرفه (one-way functions) ساده‌ترین آجر رمزنگاری‌اند؛
هم «لازم»اند و هم «کافی» برای رمز متقارن و MAC.
ایدهٔ شهودی: یک تابع f داریم که
محاسبهٔ f(x) از روی x سریع است،
ولی برعکسش، پیدا کردن یک x که f(x) = y شود، از روی y در عمل غیرممکن باشد.
فصل ۸ نقش کلی آن‌ها را باز می‌کند؛ این‌جا فقط تعریف دقیق و چند ساخت مبتنی بر سختی عددی را می‌خواهیم.

آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH

# آزمایش معکوس‌گیری و تعریف رسمی تابع یک‌طرفه

برای فرمال‌کردن «سختی معکوس‌گیری»، آزمایش زیر تعریف می‌شود:

۱. یک x یکنواخت از بین رشته‌های دودویی طول n انتخاب می‌شود و y = f(x) حساب می‌شود.

۲. الگوریتم حریف A ورودی ۱^n و y را می‌گیرد و رشته‌ای x′ برمی‌گرداند.

۳. خروجی آزمایش برابر ۱ است اگر f(x′) = y باشد، وگرنه ۰.

می‌گوییم f : {۰,۱}* → {۰,۱}* یک‌طرفه است اگر:

۱) «محاسبهٔ آسان»: یک الگوریتم چندجمله‌ای وجود دارد که با گرفتن x، مقدار f(x) را برگرداند.

۲) «معکوس‌گیری سخت»: برای هر الگوریتم تصادفی چندجمله‌ای A، یک تابع ناچیز negl(n) وجود دارد به‌طوری‌که احتمال موفقیت A در آزمایش بالا (یعنی این‌که f(x′) = y شود) حداکثر negl(n) باشد.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# ساخت fGen از سختی فاکتورگیری – تعریف تولیدکننده

فرض کن Gen یک الگوریتم چندجمله‌ای است که روی ورودی ۱^n یک N به‌صورت N = p * q تولید می‌کند،

که p و q دو عدد اول n بیتی‌اند (به‌جز با احتمال ناچیز).

این همان تولیدکننده‌ای است که در بخش فاکتورگیری داشتیم، فقط این‌جا اسمش را Gen می‌گذاریم.

الگوریتم Gen تصادفی است و برای ورودی ۱^n حداکثر n بیت تصادفی مصرف می‌کند.

برای ساخت یک تابع قطعی fGen، ورودی تابع را به‌عنوان همان رشتهٔ تصادفی Gen استفاده می‌کنیم:

روی ورودی x (یک رشتهٔ n بیتی)،

الگوریتم Gen(۱^n ; x) را اجرا می‌کنیم تا (N , p , q) را به‌دست آوریم

و سپس فقط N را به‌عنوان خروجی fGen(x) برمی‌گردانیم.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# چرا fGen یک تابع یک‌طرفه است؟

دو توزیع زیر روی N دقیقاً یکسان‌اند:

۱) x یکنواخت در رشته‌های n بیتی، و سپس N = fGen(x)

۲) اجرای Gen(۱^n) به‌صورت تصادفی و گرفتن N خروجی.

اگر سختی فاکتورگیری نسبت به Gen برقرار باشد،

یعنی فاکتورگیری N هایی که با حالت دوم تولید شده‌اند برای هر حریف چندجمله‌ای سخت است.

چون توزیع N در هر دو حالت یکی است،

پس پیدا کردن یک پیش‌تصویر x برای N نسبت به fGen نیز به‌همان اندازه سخت است.

نکتهٔ کلیدی:

اگر حریفی xای پیدا کند که fGen(x) = N،

ما می‌توانیم دوباره Gen(۱^n ; x) را اجرا کنیم و (N , p , q) را به‌دست بیاوریم،

پس از هر پیش‌تصویر x می‌شود فاکتورها را بیرون کشید.

بنابراین، معکوس‌کردن fGen از نظر محاسباتی به‌سختی فاکتورگیری N است.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

# مثال عددی ساده برای fGen و فاکتورگیری

- یک ورودی خیلی کوچک فرضی برای Gen انتخاب کن؛ مثلاً بنویس:
- فرض کنید Gen با یک رشته‌ی تصادفی ساده، این دو عدد اول را بسازد:

$$p = ۱۱ , q = ۱۳ \rightarrow N = ۱۴۳$$

- بعد توضیح بده:
- حساب کردن $N = p \cdot q$ خیلی آسان است (ضرب ساده)
- اما اگر فقط $N = ۱۴۳$ را به حریف بدهیم و p و q را مخفی کنیم، کار حریف دقیقاً همان مسئله‌ی فاکتورگیری است«.
- در یک باکس کوچک جمع‌بندی بنویس:
- ورودی تابع fGen → رشته‌ی تصادفی x
- خروجی → فقط N
- برای معکوس‌کردن باید برگردی به p و q → یعنی باید N را فاکتور بگیری → این همان سختی‌ای است که فرض کرده‌ایم.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# خانوادهٔ جایگشت‌ها – Gen / Samp / f

برای تابع‌های یک‌طرفه، گاهی به‌جای «هر تابع»، خانواده‌ای از «جایگشت‌ها» (permutations) را می‌سازیم.

یک سه‌تایی Π = (Gen , Samp , f) خانواده‌ای از جایگشت‌ها است اگر:

۱) Gen(۱^n) پارامترهایی I تولید کند؛ هر ایک مجموعهٔ D_I را تعریف می‌کند که domain و range یک جایگشت f_I : D_I → D_I است.

۲) Samp(I) روی ورودی I یک عنصر یکنواخت از D_I برمی‌گرداند.

۳) f یک الگوریتم قطعی است که روی ورودی او x ∈ D_I ، مقدار y = f_I(x) را می‌دهد.

برای چنین خانواده‌ای، آزمایش معکوس‌گیری هم شبیه قبل است:

I ← Gen(۱^n)، سپس x یکنواخت از D_I با Samp،

بعد y = f_I(x) و حریف A با داشتن y , I باید یک x' پیدا کند که f_I(x') = y.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# جایگشت یک‌طرفه و ساخت مبتنی بر RSA

می‌گوییم خانواده‌ی Π = (Gen , Samp , f) یک جایگشت یک‌طرفه است اگر برای هر حریف چندجمله‌ای A، احتمال موفقیت او در آزمایش معکوس‌گیری حداکثر یک تابع ناچیز negl(n) باشد.

اکنون ساخت ۹.۷۷ (Construction) RSA را در نظر می‌گیریم:

Gen : الگوریتم GenRSA را روی ورودی 1^n اجرا می‌کنیم تا سه‌تایی (N , e , d) به‌دست آید. سپس شاخص I = (N , e) را خروجی می‌دهیم و دامنه‌ی متناظر با آن را مجموعه‌ی Z*_N اعداد وارون‌پذیر پیمانه‌ی N در نظر می‌گیریم.

Samp : یک عنصر را به‌طور یکنواخت از Z*_N انتخاب می‌کند (یا معادل آن، عنصری را از Z_N انتخاب می‌کند و این کار را تا زمانی که gcd آن با N برابر ۱ شود تکرار می‌کند).

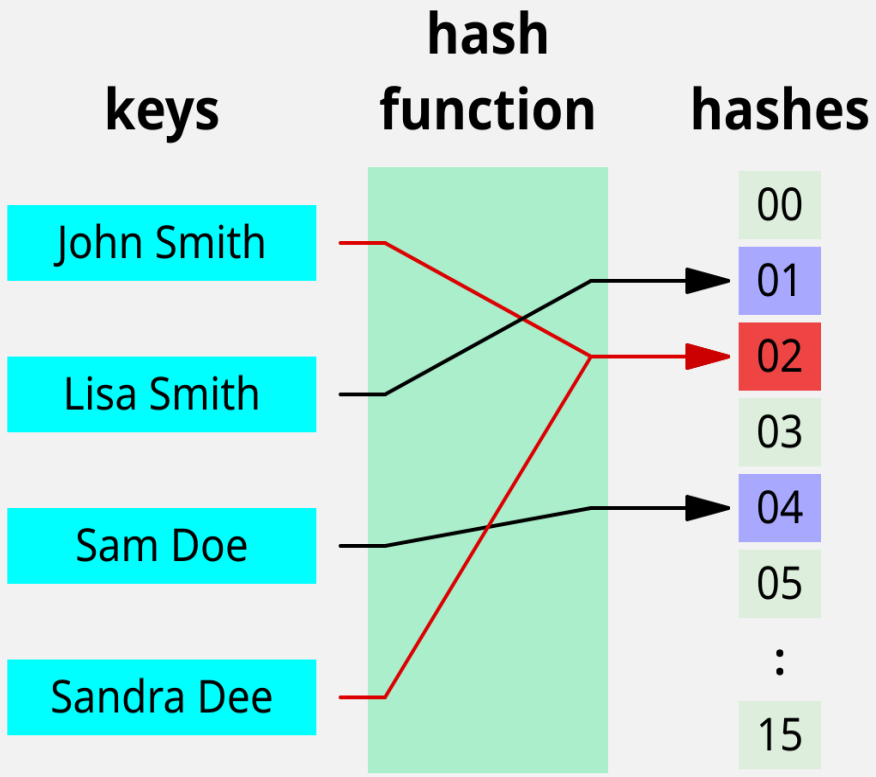
f : برای شاخص I = (N , e) و ورودی x از Z*_N، مقدار f_I(x) = x به توان e پیمانه‌ی N را برمی‌گرداند.

این تابع برای هر (N , e) یک جایگشت روی Z_N است، زیرا e نسبت به ϕ(N) هم‌اول است و در نتیجه نگاشت توان‌رسانی روی گروه Z_N وارون‌پذیر می‌باشد.

اگر مسئله‌ی RSA نسبت به الگوریتم GenRSA سخت باشد، آنگاه معکوس‌کردن این جایگشت‌ها (یعنی پیدا کردن ریشه‌ی eامِ y بدون داشتن d برای هر حریف چندجمله‌ای سخت است و در نتیجه این ساخت یک خانواده‌ی جایگشت یک‌طرفه می‌باشد.

به‌طور مشابه، با تکیه بر سختی مسئله‌ی لگاریتم گسسته در گروه‌های مناسب (مانند Z*_p)، می‌توان خانواده‌های دیگری از جایگشت‌های یک‌طرفه ساخت (رجوع شود به فصل ۸).

# هش مقاوم در برابر برخورد – ایده کلی




تابع‌های هش «مقاوم در برابر برخورد» (collision-resistant hash)در فصل‌های ۶ و ۷ معرفی شدند؛

در فصل ۷ نمونه‌های عملی (مثل SHAها) را دیدیم، اما بدون اثبات از فرض‌های ساده‌تر.

اینجا یک ساخت تئوریک می‌دهیم که برخوردناپذیری آن روی فرض سختی لگاریتم گسسته در گروه‌های مرتبهٔ اول سوار است.

فرض: G یک الگوریتم تولید گروه است که روی n^۱ یک گروه چرخه‌ای G،

یک عدد اول q (مرتبهٔ گروه) و یک مولد g تولید می‌کند.

می‌خواهیم از این G یک خانوادهٔ تابع هش ثابت‌طول بسازیم که ورودی دو «اُس» (x۱ , x۲) است و خروجی یک عنصر از گروه G.

# ساخت تابع H_s(x۱ , x۲) = g^(x۱) * h^(x۲)

کلید تابع هش یک توصیف گروه به‌همراه دو مولد است:

Gen : روی ۱^n، ابتدا G(۱^n) را اجرا می‌کنیم تا (G , q , g) به‌دست آید،

سپس یک h یکنواخت از G انتخاب می‌کنیم

و s = (G , q , g , h) را به‌عنوان کلید برمی‌گردانیم.

H : روی ورودی s = (G , q , g , h) و زوج ورودی (x۱ , x۲) با x۱ , x۲ ∈ Z_q،

مقدار H_s(x۱ , x۲) = g^(x۱) * h^(x۲) در گروه G خروجی می‌شود.

توجه: برای اسلایدها می‌توان ورودی‌ها را به‌صورت دو عدد صحیح در, [۰ , q-۱] نمایش داد و خروجی را به‌صورت یک نقطه/عنصر از گروه G (مثلاً روی منحنی بیضوی).


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

# نکات فنی دامنه، برد و فشرده‌سازی

۱) دامنهٔ تابع:

برای این‌که تابع را روی «بیت‌استرینگ» تعریف کنیم، هر ورودی طول ۲(n-۱) بیت را به دو قسمت x۱ و x۲ (هرکدام n-۱بیت) می‌بریم و هرکدام را به‌صورت طبیعی به یک عدد در Z_q تفسیر می‌کنیم.

۲) برد تابع:

خروجی H_s یک عنصر از G است؛ ولی در عمل می‌توانیم آن را به‌صورت یک رشتهٔ بیتی با طول ثابت (وابسته به n) نمایش دهیم،

و در صورت نیاز، کمی padding کنیم تا با تعریف رسمی «هش ثابت‌طول» جور شود.

۳) فشرده‌سازی واقعی:

این ساخت فقط وقتی واقعاً «فشرده»تر از ورودی است که تعداد بیت‌های لازم برای نمایش یک عنصر G کمتر از ۲(n-۱) باشد. در غیر این صورت، یک تعمیم از همین ایده لازم است (کتاب به تمرین ۹٫۲۸ ارجاع می‌دهد) تا با تکرار یا ترکیب، ورودی بلندتر را به خروجی کوتاه‌تر نگاشت کنیم.


آزمون میلر–رابین
فرض فاکتورگیری
RSA و رابطه‌ها
گروه‌ها و بیضوی
DLP/DH
یک‌طرفه و هش

# ایدهٔ اثبات برخوردناپذیری بر پایهٔ لگاریتم گسسته

قضیهٔ ۹٫۷۹ می‌گوید:

اگر G گروه‌های با مرتبهٔ اول تولید کند

و مسئلهٔ لگاریتم گسسته در این گروه‌ها سخت باشد،

آن‌وقت ساخت بالا یک خانوادهٔ هش ثابت‌طول «مقاوم در برابر برخورد» است.

طرح اثبات:

فرض کن حریف A الگوریتمی است که با احتمال ε(n)

یک برخورد برای H_s پیدا می‌کند؛

یعنی با داشتن کلید s = (G , q , g , h)،

دو ورودی متفاوت (x۱ , x۲) و (x۱' , x۲') پیدا می‌کند

به‌طوری‌که: g^(x۱') * h^(x۲') = g^(x۱) * h^(x۲) در G

و (x۱' , x۲') ≠ (x۱ , x۲)

# ادامه ایدهٔ اثبات برخوردناپذیری بر پایهٔ لگاریتم گسسته

از این تساوی می‌توان نوشت:

$g^{(x_1 - x_1')} = h^{(x_2' - x_2)}$

که یعنی h یک توان شناخته‌شده از g است.قضیه کمکی ۹٫۶۵ (روی گروه‌های مرتبهٔ اول) دقیقاً می‌گوید چطور از دو رابطهٔ این‌شکلی، log_g(h) را به‌صورت کارا محاسبه کنیم. پس می‌توانیم از Aیک الگوریتم A’ بسازیم که: از $G(1^n)$ یک (G , q , g) و سپس یک h یکنواخت می‌گیرد، همان s = (G , q , g , h) را به A می‌دهد، اگر A یک برخورد بدهد، A' با استفاده از فرمول قضیه کمکی ۹٫۶۵،

log_g(h) را محاسبه می‌کند و مسئلهٔ لگاریتم گسسته را حل می‌کند. احتمال موفقیت A’ در حل لگاریتم گسسته دقیقاً برابر ε(n) است. چون فرض کردیم لگاریتم گسسته سخت است، این احتمال باید ناچیز باشد؛ پس ε(n) هم ناچیز است، یعنی هیچ الگوریتم کارایی نمی‌تواند با احتمال غیرناچیز برای H_s برخورد تولید کند. در نتیجه، ساخت H یک تابع هش ثابت‌طول «مقاوم در برابر برخورد» است.


آزمون میلر–رابین

فرض فاکتورگیری

RSA و رابطه‌ها

گروه‌ها و بیضوی

DLP/DH

یک‌طرفه و هش

# با تشکر از توجه همه شما عزیزان