ساختار ۸-۸۲

یک تابع چکیده ساز با طول ثابت (Gen, H) را به‌صورت زیر در نظر بگیرید:

(الف) Gen: با دریافت ورودی 1^n، $\mathcal{G}(1^n)$ را اجرا کنید تا $(\mathbb{G}, q, h_1)$ به دست آید و سپس $h_2, \ldots, h_t \leftarrow \mathbb{G}$ را انتخاب کنید. خروجی $s := \langle \mathbb{G}, q, (h_1, \ldots, h_t) \rangle$ را به‌عنوان کلید ارائه کنید.

(ب) H: با داشتن یک کلید $s = \langle \mathbb{G}, q, (h_1, \ldots, h_t) \rangle$ و ورودی $(x_1, \ldots, x_t)$ با $x_i \in \mathbb{Z}_q$ خروجی $H^s(x_1, \ldots, x_t) := \prod_i h_i^{x_i}$ را ارائه کنید.

(الف) اثبات کنید که اگر مسئله‌ی لگاریتم گسسته نسبت به $\mathcal{G}$ سخت باشد و q عددی اول باشد، آنگاه برای هر $t = poly(n)$، این ساختار یک تابع چکیده ساز مقاوم در برابر برخورد با طول ثابت خواهد بود.

(ب) این مسئله را موردبحث قرار دهید که این ساختار را چگونه می‌توان برای به دست آوردن «فشرده‌سازی» مورداستفاده قرار داد بدون توجه به تعداد بیت‌های موردنیاز برای نمایش عناصر $\mathbb{G}$ (مادامی‌که در n، چندجمله‌ای است).

فصل نهم: * الگوریتم‌هایی برای تجزیه و محاسبه‌ی لگاریتم‌های گسسته

در فصل قبلی، چندین مسئله‌ی نظریه‌ی اعداد را معرفی کردیم- مهم‌تر از همه، مسئله‌ی تجزیه‌ی ضرب دو عدد اول بزرگ و مسئله‌ی محاسبه‌ی لگاریتم‌های گسسته در گروه‌های خاص- که باور عمومی بر سختی آن‌ها است. همان‌طور که در آنجا تعریف کردیم، این مقوله به این معنا است که فرض کرده‌ایم هیچ الگوریتم زمان- چندجمله‌ای برای این مسائل وجود ندارد. بااین‌حال، این مفهوم «جانبی» برای سختی، اطلاعات اندکی در مورد چگونگی مشخص کردن پارامتر امنیتی- که گاهی «طول کلید» نامیده می‌شود هرچند این عبارت‌ها را نمی‌توان به‌جای هم به کار برد- برای رسیدن به یک سطح مطلوب «عینی» از امنیت در عمل، می‌دهد. یک درک درست از این موضوع دارای اهمیت بسیار زیادی برای استفاده از سامانه‌های رمزنگاری مبتنی بر این مسائل، در جهان واقعی است. تعیین یک پارامتر امنیتی که بیش‌ازحد پایین باشد به این معنا است که سیستم رمزنگاری ممکن است در برابر حملاتی که کاراتر از حد مورد انتظار هستند، آسیب‌پذیر باشند؛ بیش‌ازحد محافظه‌کار بودن و تعیین یک پارامتر امنیتی که بیش‌ازحد بالا باشد، امنیت مناسبی را می‌دهد ولی کارایی برای کاربران صادق را با چالش روبه‌رو می‌کند. دشواری نسبی مسائل مختلف نظریه‌ی اعداد همچنین می‌تواند در تعیین این مسئله‌ایفای نقش نماید که کدام مسائل را باید به‌عنوان پایه و اساس برای ساختن سامانه‌های رمزنگاری مورداستفاده قرار داد.

البته مسئله‌ی بنیادی آن است که یک جستجوی فراگیر ممکن است بهترین الگوریتم برای حل یک مسئله‌ی مفروض نباشد؛ بنابراین، استفاده از طول کلید n، امنیتی علیه حملات اجراشده در مدت 2^n واحد زمانی، به‌طورکلی ارائه نمی‌کند. این امر در تضاد با وضعیت کلید خصوصی است که در آن، بهترین حملات علیه رمزهای قالبی موجود، تقریباً دارای پیچیدگی جستجوی فراگیر (با نادیده‌گیری پیش‌پردازش) هستند. درنتیجه، طول کلیدهای استفاده‌شده در حالت کلید عمومی معمولاً به شکل چشم‌گیری بزرگ‌تر از طول کلیدهای استفاده‌شده در وضعیت کلید خصوصی هستند.

برای درک بهتر این نکته، در این فصل، چندین الگوریتم زمان-چندجمله‌ای را برای تجزیه و محاسبه‌ی لگاریتم‌های گسسته موردبررسی قرار می‌دهیم که بسیار بهتر از جستجوی فراگیر هستند. هدف در

اینجا تنها ارائه‌ی مقدمه‌ای بر الگوریتم‌های موجود برای این مسائل و همین‌طور ارائه‌ی راهنمایی ابتدایی برای تعیین پارامترها در عمل است. تمرکز ما بر روی ایده‌های سطح بالا است و عمداً به بسیاری از جزئیات مهم پیاده‌سازی که در صورت استفاده از این الگوریتم‌ها در عمل دارای اهمیت حیاتی هستند، نمی‌پردازیم. همچنین به‌صورت اختصاصی بر الگوریتم‌های «کلاسیک» تمرکز می‌کنیم و برای بحث در خصوص الگوریتم‌های زمان-چندجمله‌ای (!) «کوانتومی» برای تجزیه و محاسبه‌ی لگاریتم‌های گسسته، خواننده می‌تواند به منابع دیگر مراجعه کند. (این، یک تصمیم عمدی دیگر بود هم به این دلیل که نمی‌خواستیم داشتن پیش‌زمینه‌ی خواننده در ریاضیات کوانتومی را فرض کنید و هم به این دلیل که به نظر نمی‌رسد رایانه‌های کوانتومی به‌زودی در دسترس باشند.)

خواننده همچنین ممکن است متوجه شود که ما تنها الگوریتم‌هایی برای تجزیه و محاسبه‌ی لگاریتم‌های گسسته را توصیف می‌کنیم و نه الگوریتم‌هایی برای مثلاً حل مسائل *RSA* یا دیفی-هلمن تصمیمی. انتخاب ما به این دلیل توجیه می‌شود که بهترین الگوریتم‌های شناخته‌شده برای حل *RSA* نیازمند تجزیه‌ی پیمانه هستند و (در گروه‌های بحث شده در بخش‌های ۸-۳-۳ و ۸-۳-۴)، بهترین رویکردهای شناخته‌شده برای حل مسئله‌ی دیفی-هلمن تصمیمی نیازمند محاسبه‌ی لگاریتم‌های گسسته هستند.

۹-۱- الگوریتم‌هایی برای تجزیه

در سراسر بحث، فرض می‌کنیم که $N = pq$ حاصل‌ضرب دو عدد اول متمایز با $p < q$ است. ما به حالتی علاقه‌مند هستیم که در آن، p و q دارای طول (معلوم) یکسان n باشند و بنابراین $n = \Theta(\log N)$.

به‌صورت مکرر از قضیه‌ی باقیمانده‌ی چینی به همراه نمادهای توسعه‌یافته در بخش ۸-۱-۵ استفاده خواهیم کرد. قضیه‌ی باقیمانده‌ی چینی بیان می‌کند که:

$$\mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^* \quad \text{و} \quad \mathbb{Z}_N \simeq \mathbb{Z}_p \times \mathbb{Z}_q$$

با یکریختی داده‌شده توسط $f(x) \stackrel{\text{def}}{=} ([x \bmod p], [x \bmod q])$. این واقعیت که f یک یکریختی است به این معنا است که به شکل خاص، یک نگاشت دوسویی بین عناصر $x \in \mathbb{Z}_N$ و زوج‌های $(x_p, x_q) \in \mathbb{Z}_p \times \mathbb{Z}_q$ می‌دهد. برای نشان دادن این نگاشت دوسویی می‌نویسیم $x \leftrightarrow (x_p, x_q)$ با $x_p = [x \bmod p]$ و $x_q = [x \bmod q]$

از بخش ۸-۲ به یاد داریم که «محک تقسیم»- یک روش تجزیه‌ی فراگیر بدیهی- یک عامل از یک عدد مفروض N را در زمان $O(N^{1/2} \cdot \text{polylog}(N))$ می‌دهد. (این الگوریتم یک الگوریتم زمان

نمایی است ازآنجاکه اندازه‌ی ورودی عبارت است از طول نمایش دودویی N، یعنی $\|N\| = O(\log N)$.[۴۶] ما سه الگوریتم تجزیه با عملکرد بهتر را بررسی می‌کنیم:

- روش $p-1$ پلارد در صورتی مؤثر است که $p-1$ تنها دارای عوامل اول «کوچک» باشد.
- روش رو پلارد برای N دلخواه اعمال می‌شود. (در این حالت، آن را الگوریتم تجزیه‌ی هدف عام می‌نامند.) زمان اجرای آن برای N تعریف‌شده در ابتدای این بخش برابر با $O(N^{1/4} \cdot polylog(N))$ است. توجه کنید که این زمان همچنان در n نمایی است که n همان طول N است.
- الگوریتم غربال مربعی یک الگوریتم تجزیه‌ی هدف عام است که در زمان زیرنمایی بر اساس طول N اجرا می‌شود. ما مروری سطح بالا بر چگونگی کارکرد این الگوریتم خواهیم داشت ولی جزئیات آن تا حدودی پیچیده و فراتر از گستره‌ی این کتاب هستند.

سریع‌ترین الگوریتم تجزیه‌ی هدف عام شناخته‌شده عبارت است از «غربال میدان اعداد عام». به شکل تجربی، این الگوریتم ورودی N را در زمان مورد انتظار $2^{O((\log N)^{1/3} \cdot (\log\log N)^{2/3})}$ تجزیه می‌کند که بر اساس طول N، زیرنمایی است.

۹-۱-۱- الگوریتم $p-1$ پلارد

اگر $N = pq$ و $p-1$ تنها دارای عوامل اول «کوچک» باشد، الگوریتم $p-1$ پلارد را می‌توان برای تجزیه‌ی کارای N به کار گرفت. ایده‌ی اصلی، ساده است. فرض کنید B یک عدد صحیح باشد که برای آن، $(p-1) | B$ و $(q-1) \nmid B$؛ جزئیات چگونگی یافتن چنین B را در ادامه بیان می‌کنیم. فرض کنید $B = \gamma \cdot (p-1)$ برای یک عدد صحیح مفروض γ. یک $x \in \mathbb{Z}_N^*$ یکنواخت را انتخاب کرده و $y := [x^B \bmod N]$ را محاسبه کنید. (توجه کنید که y را می‌توان با استفاده از الگوریتم توان رسانی کارا از ضمیمه‌ی ب-۲-۳ محاسبه نمود.) ازآنجاکه $(1,1) \leftrightarrow 1$ داریم:

$$\begin{aligned} y = [x^B - 1 \bmod N] &\leftrightarrow (x_p, x_q)^B - (1,1) \\ &= (x_p^B - 1 \bmod p, x_q^B - 1 \bmod q) \\ &= \left((x_p^{p-1})^\gamma - 1 \bmod p, x_q^B - 1 \bmod q\right) \\ &= (0, [x_q^B - 1 \bmod q]) \end{aligned}$$

با استفاده از قضیه‌ی ۸-۱۴ و این واقعیت که مرتبه‌ی $\mathbb{Z}_p^*$ برابر با $p-1$ است. در ادامه نشان می‌دهیم که با احتمال بالا، $x_q^B \neq 1 \bmod q$. با فرض اینکه این حالت برقرار است، یک عدد صحیح y را به دست آورده‌ایم که برای آن

---

[۴۶] بنابراین، یک زمان اجرای برابر با $N^{O(1)} = 2^{O(\|N\|)}$ نمایی است، یک زمان اجرای $2^{o(\log N)} = 2^{o(|N|)}$ زیرنمایی است و یک زمان اجرای برابر با $\log^{O(1)} N = \|N\|^{O(1)}$ چندجمله‌ای است.

$$y \neq 0 \bmod q \quad \text{ولی} \quad y = 0 \bmod p$$

یعنی، $p|y$ ولی $q \nmid y$. این امر به‌نوبه‌ی خود نشان می‌دهد که $\gcd(y, N) = p$ است. بنابراین، یک محاسبه‌ی ساده‌ی gcd (که می‌توان آن را به‌صورت کارا به شکل توصیف‌شده در ضمیمه‌ی ب-۱-۲ انجام داد) یک عامل اول برای N را می‌دهد.

الگوریتم ۹-۱

الگوریتم $p - 1$ پلارد برای تجزیه
ورودی: عدد صحیح N
خروجی: یک عامل غیربدیهی برای N
$x \leftarrow \mathbb{Z}_N^*$
$y := [x^B - 1 \bmod N]$
B // در متن تعریف شده است.
$p := \gcd(y, N)$
if $p \notin \{1, N\}$ **return** p

اجازه دهید در ابتدا استدلال کنیم این الگوریتم، عمل می‌کند (با احتمال بالا). فرض کنید $B!(p-1)$ ولی $B \nmid (q-1)$. در این حالت، مادامی‌که $x_q^B \stackrel{\text{def}}{=} [x \bmod q]$ یک مولد برای $\mathbb{Z}_q^*$ باشد، داریم $x_q^B \neq 1 \bmod q$. (این امر از گزاره‌ی ۸-۵۲ نتیجه می‌شود.) چیزی که باقی می‌ماند عبارت از تحلیل احتمال اینکه x_q یک مولد باشد. در اینجا ما به تعدادی از نتایج اثبات‌شده در ضمیمه‌ی ب-۳-۱ تکیه می‌کنیم. ازآنجاکه q عددی اول است، $\mathbb{Z}_q^*$ یک گروه دوری از مرتبه‌ی $q - 1$ است که دقیقاً دارای $\phi(q-1)$ مولد است (ن.ک. قضیه‌ی ب-۱۶). اگر x به‌صورت یکنواخت از $\mathbb{Z}_N^*$ انتخاب شود، آنگاه x_q دارای توزیع یکنواخت در $\mathbb{Z}_q^*$ است. (این، یکی از نتایج این واقعیت است که قضیه‌ی باقیمانده‌ی چینی یک نگاشت دوسویی بین $\mathbb{Z}_N^*$ و $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ را می‌دهد.) بنابراین، احتمال اینکه x_q یک مولد باشد برابر است با $\frac{\phi(q-1)}{q-1} = \Omega(1/\log q) = \Omega(1/n)$ (ن.ک. قضیه‌ی ب-۱۵). چندین مقدار از x را می‌توان برای افزایش احتمال موفقیت، انتخاب نمود.

تنها چیزی که باقی مانده است مسئله‌ی یافتن B به صورتی است که $B \mid (p-1)$ ولی $B \nmid (q-1)$ باشد. یک روش عبارت است از انتخاب $B = \prod_{i=1}^{k} p_i^{\lfloor n/\log p_i \rfloor}$ برای یک k مفروض، که در آن، p_i نشان‌دهنده‌ی عدد اول iام (یعنی $p_1 = 2, p_2 = 3, p_3 = 5, \ldots$) و n برابر با طول p است. (توجه کنید که $p_i^{\lfloor n/\log p_i \rfloor}$ برابر است بزرگ‌ترین توان p_i که می‌تواند $p - 1$ را تقسیم کند.) اگر $p - 1$ را بتوان به‌صورت $\prod_{i=1}^{k} p_i^{e_i}$ با $e_i \geq 0$ نوشت (یعنی، اگر بزرگ‌ترین عامل اول $p - 1$ کوچک‌تر از

p_k باشد)، داریم $B \mid (p-1)$. در مقابل، اگر $q - 1$ دارای یک عامل اول بزرگ‌تر از p_k باشد، آنگاه $B \nmid (q-1)$ خواهد بود.

انتخاب یک مقدار بزرگ‌تر برای k باعث افزایش B می‌شود و بنابراین زمان اجرای الگوریتم را افزایش می‌دهد (که یک توان رسانی پیمانه‌ای را تا توان B اجرا می‌کند). یک مقدار بزرگ‌تر برای k همچنین این احتمال را بالا می‌برد که $B \mid (p-1)$ باشد ولی درعین‌حال، احتمال اینکه $B \nmid (q-1)$ باشد را کاهش می‌دهد. البته می‌توان الگوریتم را به شکل مکرر با انتخاب‌های مختلف برای k اجرا نمود.

الگوریتم $p - 1$ پلارد عقیم خواهد شد اگر $p - 1$ و $q - 1$ دارای عوامل اول بزرگ‌تری باشند. (به لحاظ دقیق‌تر، الگوریتم همچنان کار می‌کند ولی تنها برای B که به حدی بزرگ باشد که الگوریتم، غیرعملی شود.) اگر p و q اعداد اول n-بیتی یکنواخت باشند، احتمال اینکه $p - 1$ یا $q - 1$ تنها دارای عوامل اول کوچک باشند، کاهش خواهد یافت. بااین‌حال، زمان تولید یک پیمانه‌ی $N = pq$ برای کاربردهای رمزنگاری، p و q گاهی به‌صورت اعداد اول قوی، انتخاب می‌شوند. (یادآوری می‌کنیم که p یک عدد اول قوی خواهد بود اگر $(p-1)/2$ نیز اول باشد.) انتخاب p و q به این صورت به لحاظ چشم‌گیری کارایی کمتری نسبت به انتخاب p و q به‌عنوان اعداد اول دلخواه (تصادفی) دارد. به این دلیل که الگوریتم‌های بهتری برای تجزیه در دسترس هستند (همان‌طور که در ادامه خواهیم دید) و به دلیل نتیجه‌ی بالا، اجماع کنونی آن است که هزینه‌ی محاسباتی اضافی تولید p و q به‌صورت اعداد اول قوی، هیچ بهره‌ی امنیتی مفیدی را ارائه نمی‌کند.

۹-۱-۲- الگوریتم رو پلارد

الگوریتم رو پلارد را می‌توان برای تجزیه‌ی یک عدد صحیح دلخواه $N = pq$ استفاده نمود؛ در این حالت، این الگوریتم یک الگوریتم تجزیه‌ی هدف عام است. به شکل تجربی، الگوریتم، N را با احتمال ثابت در $O(N^{1/4} \cdot polylog(N))$ واحد زمانی، تجزیه می‌کند؛ این زمان همچنان نمایی است ولی بهبود چشم‌گیری نسبت به محک تقسیم می‌دهد.

ایده‌ی اصلی این رویکرد عبارت است از یافتن مقادیر متمایز $x, x' \in \mathbb{Z}_N^*$ که به پیمانه‌ی p باهم برابرند (یعنی برای آن‌ها، $x = x' \bmod p$)؛ چنین زوجی را «خوب» می‌نامیم. توجه کنید که برای یک زوج خوب x, x' داریم $\gcd(x - x', N) = p$ (چراکه $x \neq x' \bmod N$)، بنابراین محاسبه‌ی gcd یک عامل غیربدیهی برای N می‌دهد.

چگونه می‌توانیم یک زوج خوب را پیدا کنیم؟ فرض کنید ما مقادیر $x^{(1)}, \ldots, x^{(k)}$ را به‌صورت یکنواخت از $\mathbb{Z}_N^*$ انتخاب کرده‌ایم که در آن، $k = 2^{n/2} = O(\sqrt{p})$. با در نظر گرفتن این مقادیر در نمایش باقیمانده‌ی چینی آن‌ها به‌صورت $(x_p^{(1)}, x_q^{(1)}), \ldots, (x_p^{(k)}, x_q^{(k)})$ داریم که هر $x_p^{(i)} \stackrel{\text{def}}{=} [x^{(i)} \bmod p]$ در $\mathbb{Z}_p^*$، یکنواخت است. (این امر از نگاشت دوسویی بین $\mathbb{Z}_N^*$ و $\mathbb{Z}_p^* \times \mathbb{Z}_q^*$ نتیجه می‌شود.) بنابراین، با استفاده از کران روز تولد لم الف-۱۶، مشاهده می‌کنیم که با احتمال بالا،

i,j متمایز با $x_p^{(i)} = x_p^{(j)}$ یا به شکل برابر، $x^{(i)} = x^{(j)} \bmod p$ وجود دارند. به‌علاوه، لم الف-۱۵ نشان می‌دهد که $x^{(i)} \neq x^{(j)}$ مگر با احتمال ناچیز. بنابراین، با احتمال بالا، یک زوج خوب $x^{(i)}, x^{(j)}$ را به دست می‌آوریم که می‌توان برای یافتن یک عامل غیر بدیهی برای N، به شکل بحث شده در بالا، از آن استفاده نمود.

الگوریتم ۹-۲

الگوریتم رو پلارد برای تجزیه

ورودی: عدد صحیح N که ضرب دو عدد اول n-بیتی است.

خروجی: یک عامل غیربدیهی برای N

$x^{(0)} \leftarrow \mathbb{Z}_N^*, x' := x := x^{(0)}$

for $i = 1$ to $2^{n/2}$:

$x := F(x)$

$x' := F(F(x'))$

$p := \gcd(x - x', N)$

if $p \notin \{1, N\}$***return*** p ***and stop***

می‌توانیم $k = O(\sqrt{p})$ عنصر یکنواخت از $\mathbb{Z}_N^*$ را در زمان $O(\sqrt{p}) = O(N^{1/4})$ تولید کنیم. بااین‌حال، آزمودن تمام زوج عناصر به‌منظور شناسایی یک زوج خوب نیازمند زمان $\binom{k}{2} = O(k^2)$ خواهد بود! (توجه کنید که ازآنجاکه p مجهول است، نمی‌توانیم $O(p) = O(N^{1/2})$ را به‌صورت آشکار محاسبه نموده و سپس $x_p^{(1)}, \ldots, x_p^{(k)}$ را مرتب کنیم تا یک زوج خوب را بیابیم. به‌جای این کار، برای تمام زوج‌های متمایز i,j باید $\gcd(x^{(i)} - x^{(j)}, N)$ را محاسبه کنیم تا ببینیم آیا این مقدار یک عامل غیربدیهی برای N می‌دهد یا خیر.) بدون بهینه‌سازی‌های بیشتر، این مقوله چیزی بهتر از محک تقسیم نخواهد بود.

ایده‌ی پلارد استفاده از روشی بود که در بخش ۵-۴-۲ در رابطه با حملات روز تولد فضای کوچک مشاهده کردیم. به‌صورت خاص، ما دنباله‌ی $x^{(1)}, x^{(2)}, \ldots$ را با قرار دادن هر مقدار به‌عنوان تابعی از مقدار قبلی خود، محاسبه می‌کنیم؛ یعنی، یک تابع مفروض $F: \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ را در نظر می‌گیریم، یک $x^{(0)} \in \mathbb{Z}_N^*$ یکنواخت را در نظر می‌گیریم و سپس برای $i = 1, \ldots, k$ قرار می‌دهیم $x^{(i)} := F(x^{(i-1)})$. ملزم می‌کنیم که F دارای این ویژگی باشد که اگر $x = x' \bmod p$ آنگاه $F(x) = F(x') \bmod p$؛ این امر تضمین می‌کند که زمانی که تساوی به پیمانه‌ی p رخ می‌دهد، تساوی حفظ شود. (یک انتخاب استاندارد برابر است با $F(x) = [x^2 + 1 \bmod N]$، ولی هرچندجمله‌ای F دارای این ویژگی خواهد بود.) اگر ما F را به‌عنوان یک تابع تصادفی مدل‌سازی کنیم (که به‌صورت تجربی عمل می‌کند)، آنگاه با احتمال بالا، یک زوج خوب در اولین k عنصر این دنباله وجود خواهد داشت.

با ادامه‌ی کار به شکلی تقریباً مشابه با الگوریتم ۵-۹ از بخش ۵-۴-۲، می‌توانیم یک زوج خوب (اگر وجود داشته باشد) را با استفاده۹ از تنها $O(k)$ محاسبه‌ی gcd بیابیم؛ ن.ک. الگوریتم ۹-۲. علاوه بر بهبود زمان اجرا، ایده‌ی پلارد همچنین باعث کاهش چشم‌گیر میزان حافظه‌ی موردنیاز خواهد شد.

۹-۱-۳- الگوریتم غربال مربعی

الگوریتم رو پلارد بهتر از محک تقسیم است ولی همچنان در زمان نمایی اجرا می‌شود. الگوریتم غربال مربعی در زمان زیرنمایی اجرا می‌شود. تا اوایل دهه‌ی ۱۹۹۰، این الگوریتم سریع‌ترین الگوریتم تجزیه‌ی شناخته شده بود و همچنان الگوریتم تجزیه‌ی انتخابی برای اعداد با طول تا حدود ۳۰۰ بیت است. ما اصول کلی این الگوریتم را توصیف می‌کنیم ولی به خواننده هشدار می‌دهیم که چندین نکته‌ی جزئی مهم حذف شده‌اند.

یک عنصر $z \in \mathbb{Z}_N^*$ یک «مانده‌ی درجه دوم به پیمانه‌ی N» است اگر یک $x \in \mathbb{Z}_N^*$ وجود داشته باشد به صورتی که $x^2 = z \bmod N$؛ در این حالت، می گوییم x جذر z است. مشاهدات زیر به‌عنوان نقطه‌ی شروع ما عمل می‌کنند:

- اگر N حاصل‌ضرب دو عدد اول متمایز فرد باشد، آنگاه هر مانده‌ی درجه دوم به پیمانه‌ی N دقیقاً دارای چهار جذر (ریشه دوم) است. (ن.ک. بخش ۱۳-۴-۲.)
- با داشتن x, y با $x^2 = y^2 \bmod N$ و $x \neq \pm y \bmod N$ می‌توان یک عامل غیربدیهی از N را در زمان چندجمله‌ای، محاسبه نمود. این امر به دلیل این واقعیت است که $x^2 = y^2 \bmod N$ نشان می‌دهد که

$$0 = x^2 - y^2 = (x - y)(x + y) \bmod N$$

و بنابراین $N \mid (x - y)(x + y)$. بنابراین، $N \nmid (x - y)$ و $N \nmid (x + y)$ به این دلیل که $x \neq \pm y \bmod N$. بنابراین، این حالت باید برقرار باشد که $\gcd(x - y, N)$ برابر با یکی از عوامل اول N است. (ن.ک. لم ۱۳-۳۵.)

الگوریتم غربال مربعی سعی می‌کند x, y را با $x^2 = y^2 \bmod N$ و $x \neq \pm y \bmod N$ تولید کند. یک روش ابتدایی برای انجام این کار-که پایه و اساس یک الگوریتم تجزیه‌ی قدیمی‌تر ساخته‌شده توسط فرمات را شکل می‌دهد- عبارت است از انتخاب $x \in \mathbb{Z}_N^*$، محاسبه‌ی $q := [x^2 \bmod N]$ و سپس بررسی اینکه آیا q یک مربع روی اعداد صحیح هست یا خیر (یعنی، بدون کاهش به پیمانه‌ی N). اگر باشد، آنگاه $q = y^2$ برای یک عدد صحیح مفروض y و بنابراین $x^2 = y^2 \bmod N$ متأسفانه، احتمال اینکه $[x^2 \bmod N]$ یک مربع باشد به‌اندازه‌ای پایین است که این فرآیند را باید به‌دفعات نمایی تکرار نمود.

یک بهبود چشم‌گیر با تولید یک دنباله از مقادیر $q_1 := [x_1^2 \bmod N], \ldots$ و شناسایی یک زیرمجموعه از این مقادیر به دست می‌آید که «حاصل‌ضرب» آن، یک مربع روی اعداد صحیح است. در الگوریتم غربال مربعی، این امر با استفاده از دو گام زیر انجام می‌شود:

گام اول

یک کران B را در نظر بگیرید. فرض کنید یک عدد صحیح B-هموار است اگر تمام عوامل اول آن کوچک‌تر مساوی B باشند. در مرحله‌ی اول این الگوریتم، ما به دنبال اعداد صحیح به شکل $q_i = [x_i^2 \bmod N]$ می‌گردیم که B-هموار باشند و آن‌ها را تجزیه می‌کنیم. (هرچند تجزیه سخت است، یافتن و تجزیه‌ی اعداد B-هموار هنگامی‌که B به‌اندازه‌ی کافی کوچک باشد، عملی خواهد بود.) این $\{x_i\}$ با امتحان کردن $x = \sqrt{N}+1, \sqrt{N}+2, \ldots$ انتخاب می‌شوند؛ این امر یک کاهش غیربدیهی به پیمانه‌ی N (ازآنجاکه $x > \sqrt{N}$) را تضمین می‌کند و دارای این مزیت است که $q \overset{\text{def}}{=} [x^2 \bmod N] = x^2 - N$ «کوچک» است به صورتی که q به‌احتمال بیشتری، B-هموار خواهد بود.

فرض کنید $\{p_1, \ldots, p_k\}$ عبارت باشد از مجموعه اعداد اول کوچک‌تر مساوی B. هنگامی‌که $\{q_i\}$ B-هموار را به‌صورت توصیف‌شده در بالا یافته و تجزیه کردیم، یک مجموعه معادله به شکل زیر خواهیم داشت:

$$\begin{aligned} q_1 &= [x_1^2 \bmod N] = \prod_{i=1}^{k} p_i^{e_{1,i}} \\ &\vdots \\ q_\ell &= [x_\ell^2 \bmod N] = \prod_{i=1}^{k} p_i^{e_{\ell,i}} \end{aligned} \tag{۹.۱}$$

(توجه کنید که معادلات بالا روی اعداد صحیح هستند.)

گام دوم

در ادامه می‌خواهیم یک زیرمجموعه از $\{q_i\}$ را بیابیم که ضرب آن، یک مربع باشد. اگر یک زیرمجموعه‌ی S از $\{q_i\}$ را ضرب کنیم، مشاهده می‌کنیم که نتیجه

$$z = \prod_{j \in S} q_j = \prod_{i=1}^{k} p_i^{\sum_{j \in S} e_{j,i}}$$

یک مربع خواهد بود اگر و تنها اگر توان هر عدد اول p_i، زوج باشد. این امر نشان می‌دهد که ما در مورد توانهای $\{e_{j,i}\}$ در معادله‌ی (۹.۱) تنها به پیمانه‌ی ۲ اهمیت می‌دهیم؛ به‌علاوه، می‌توانیم از حبر خطی برای یافتن یک زیرمجموعه از $\{q_i\}$ استفاده کنیم که جمع «بردارهای نمایی» آن برابر با بردار-۰ به پیمانه‌ی ۲ باشد.

با جزئیات بیشتر: اگر توانها را در معادله‌ی (۹.۱) را به پیمانه‌ی ۲ کاهش دهیم، ماتریس-۰/۱ ۲ داده‌شده به‌صورت زیر را به دست می‌آوریم:

$$\begin{pmatrix} \gamma_{1,1} & \gamma_{1,2} & \cdots & \gamma_{1,k} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{\ell,1} & \gamma_{\ell,2} & \cdots & \gamma_{\ell,k} \end{pmatrix} \overset{\text{def}}{=} \begin{pmatrix} [e_{1,1} \bmod 2] & [e_{1,2} \bmod 2] & \cdots & [e_{1,k} \bmod 2] \\ \vdots & \vdots & \ddots & \vdots \\ [e_{\ell,1} \bmod 2] & [e_{\ell,2} \bmod 2] & \cdots & [e_{\ell,k} \bmod 2] \end{pmatrix}$$

اگر $l = k + 1$ باشد، آنگاه Γ دارای تعداد ردیف بیشتر از ستون خواهد بود و باید یک زیرمجموعه‌ی غیرتهی S دارای ردیف‌هایی وجود داشته باشد که مجموع آن‌ها برابر با بردار-۰ به پیمانه‌ی ۲ خواهد بود. چنین زیرمجموعه را می‌توان به‌صورت کارا با استفاده از جبر خطی یافت. آنگاه:

$$z \overset{\text{def}}{=} \prod_{j \in S} q_j = \prod_{i=1}^{k} p_i^{\sum_{j \in S} e_{j,i}} = \left(\prod_{i=1}^{k} p_i^{(\sum_{j \in S} e_{j,i})/2}\right)^2,$$

با استفاده از این واقعیت که همه‌ی $\{\sum_{j \in S} e_{j,i}\}$ زوج هستند. ازآنجاکه

$$z = \prod_{j \in S} q_j = \prod_{j \in S} x_j^2 = \left(\prod_{j \in S} x_j\right)^2 \bmod N,$$

ما دو جذر (به پیمانه‌ی N) برای z یافته‌ایم. هرچند هیچ تضمینی وجود ندارد که این جذرها اجازه‌ی تجزیه‌ی N را بدهند (به دلایل بحث شده در ابتدای این بخش)، ازلحاظ تجربی، با احتمال ثابت. این امکان را به وجود می‌آورند. با در نظر گرفتن $l > k + 1$، می‌توانیم چندین زیرمجموعه‌ی S با ویژگی مطلوب به دست آوریم و سعی کنیم N را با استفاده از هر گزینه، تجزیه کنیم.

مثال ۹-۳

$N = ۳۷۷۷۵۳$ را در نظر بگیرید. داریم $[۶۲۰^2 \bmod N] = ۶۶۴۷$ و می‌توانیم ۶۶۴۷ را (روی اعداد صحیح بدون هیچ کاهش پیمانه‌ای) به‌صورت زیر، تجزیه کنیم:

$$[۶۲۰^2 \bmod N] = ۶۶۴۷ = ۱۷^2 . ۲۳.$$

به شکل مشابه،

$$[۶۲۱^2 \bmod N] = ۲^4 . ۱۷ . ۲۹$$
$$[۶۴۵^2 \bmod N] = ۲^7 \cdot ۱۳ \cdot ۲۳$$

$$[۶۵۵^{۲} \bmod N] = ۲^{۲} \cdot ۱۳ \cdot ۱۷ \cdot ۲۹.$$

زیرمجموعه‌ی S را شامل هر چهار معادله‌ی بالا دانستن نشان می‌دهد که

$$۶۲۰^{۲} \cdot ۶۲۱^{۲} \cdot ۶۴۵^{۲} \cdot ۶۵۵^{۲} = ۲^{۱۴} \cdot ۱۳^{۲} \cdot ۱۷^{۴} \cdot ۲۳^{۲} \cdot ۲۹^{۲} \bmod N$$
$$\Rightarrow [۶۲۰ \cdot ۶۲۱ \cdot ۶۴۵ \cdot ۶۵۵ \bmod N]^{۲} = [۲^{۷} \cdot ۱۳ \cdot ۱۷^{۲} \cdot ۲۳ \cdot ۲۹ \bmod N]^{۲} \bmod N$$
$$\Rightarrow ۱۲۷۱۹۴^{۲} = ۴۵۳۳۵^{۲} \bmod N,$$

با $۱۲۷۱۹۴ \neq \pm ۴۵۳۳۵ \bmod N$. محاسبه‌ی $\gcd(۱۲۷۱۹۴ - ۴۵۳۳۵, ۳۷۷۷۵۳) = ۷۵۱$ یک عامل غیربدیهی برای N می‌دهد. ◊

زمان اجرا

انتخاب یک مقدار بزرگ‌تر برای B این احتمال را بالا می‌برد که یک مقدار یکنواخت $q = [x^{۲} \bmod N]$، B-هموار باشد؛ از سوی دیگر، این امر به این معنا است که ما باید برای شناسایی و تجزیه‌ی اعداد B-هموار، سخت‌تر کارکنیم و باید تعداد بیشتری از آن‌ها را بیابیم (چراکه ملزم می‌کنیم $l > k$ باشد که در آن، k عبارت است از تعداد اعداد اول کوچک‌تر مساوی B). این امر همچنین به این معنا است که ماتریس Γ بزرگ‌تر خواهد بود و بنابراین، گام جبر خطی، کندتر خواهد بود. انتخاب مقدار بهینه‌ی B الگوریتمی را می‌دهد که (حداقل به شکل تجربی) N را در زمان $۲^{O(\sqrt{\log N \log\log N})}$ تجزیه می‌کند. (درواقع، جمله‌ی ثابت در توان را می‌توان به شکل نسبتاً دقیق تعیین نمود.) نکته‌ی مهم برای اهداف ما این است که این مقوله زیرنمایی بر اساس طول N است.

۹-۲- الگوریتم‌هایی برای محاسبه‌ی لگاریتم‌های گسسته

فرض کنید $\mathbb{G}$ یک گروه با مرتبه‌ی معلوم q باشد. یک نمونه از مسئله‌ی لگاریتم گسسته در $\mathbb{G}$ یک پایه‌ی $g \in \mathbb{G}$ (که نیازی نیست یک مولد برای $\mathbb{G}$ باشد) و یک عنصر $h \in \langle g \rangle$، یعنی زیرگروه تولیدشده توسط h، را مشخص می‌کند. هدف عبارت است از یافتن x به صورتی که $g^{x} = h$ باشد (ن.ک. بخش ۸-۳-۲.) راه‌حل x را «لگاریتم گسسته‌ی h نسبت به g» می‌نامند. یک جستجوی فراگیر بدیهی برای x را می‌توان در زمان $O(|\langle g \rangle| \leq q)$ (با امتحان کردن تمام مقادیر ممکن) انجام داد و بنابراین ما تنها علاقه‌مند به الگوریتم‌هایی خواهیم بود که زمان اجرای آن‌ها بهتر از این باشد.

الگوریتم‌های ارائه‌شده برای حل مسئله‌ی لگاریتم گسسته در دو دسته قرار می‌گیرند: آن‌هایی که «عام» بوده و برای گروه‌های دلخواه اعمال می‌شوند و آن‌هایی که به‌گونه‌ای تغییر یافته‌اند که برای یک دسته‌ی «خاص» از گروه‌ها کار کنند. ما با بحث پیرامون سه الگوریتم عام، شروع می‌کنیم:

- هنگامی‌که مرتبه‌ی گروه، q، عددی اول نیست و تجزیه‌ی q مشخص است یا تعیین آن ساده است، الگوریتم پولیگ-هلمن مسئله‌ی یافتن لگاریتم‌های گسسته در $\mathbb{G}$ را به مسئله‌ی یافتن لگاریتم‌های گسسته در زیرگروه‌های مرتبه اول $\mathbb{G}$ کاهش می‌دهد. تقریباً، نتیجه آن است

که پیچیدگی حل لگاریتم گسسته در یک گروه با مرتبه‌ی q بیشتر از پیچیدگی حل لگاریتم گسسته در یک گروه با مرتبه‌ی q' نیست که در آن، q' بزرگ‌ترین عدد اول تقسیم‌کننده‌ی q است. این امر تبیین‌کننده‌ی ترجیح استفاده از گروه‌های مرتبه اول است (ن.ک. بخش ۸-۳-۲).

- روش قدم کوچک-قدم بزرگ، ارائه‌شده توسط شانکس، لگاریتم گسسته را در یک گروه با مرتبه‌ی q در زمان $O(\sqrt{q}.\text{polylog}(q))$ و با ذخیره کردن $O(\sqrt{q})$ عنصر گروهی، محاسبه می‌کند.
- الگوریتم رو پلارد نیز اجازه‌ی محاسبه‌ی لگاریتم‌های گسسته را در زمان $O(\sqrt{q}.polylog(q))$ ولی با استفاده از حافظه‌ی «ثابت» می‌دهد. این الگوریتم را می‌توان به این صورت در نظر گرفت که از ارتباط بین مسئله‌ی لگاریتم گسسته و چکیده‌سازی مقاوم در برابر برخورد که در بخش ۸-۴-۲ مشاهده کردیم، استفاده می‌کند. یک الگوریتم متفاوت را توصیف می‌کنیم که به شکلی واضح‌تر این ارتباط را نشان می‌دهد.

می‌توان نشان داد که تا آنجایی که با الگوریتم‌های عام سروکار داریم، پیچیدگی زمانی دو الگوریتم آخر «بهینه» است. بنابراین، به‌منظور امید داشتن برای عملکردی بهتر، باید الگوریتم‌های ارائه‌شده برای گروه‌های خاص را در نظر بگیریم که از «نمایش» اعضای گروه در آن گروه‌ها استفاده می‌کنند، که در آن منظور ما از «نمایش»، شیوه‌ی کدگذاری اعضای گروه به شکل رشته‌های بیتی است.

این نکته نیازمند اندکی بحث است. از دیدگاه ریاضیاتی، هر دو گروه دوری دارای مرتبه‌ی یکسان، یکریخت هستند به این معنا که این گروه‌ها یکسان هستند تا حد «تغییر نام» اعضای گروه. بااین‌حال، از دیدگاه محاسباتی/الگوریتمی، این «تغییر نام» می‌تواند تأثیر چشم‌گیری داشته باشد. برای مثال، گروه دوری $\mathbb{Z}_q$ از اعداد صحیح $\{۰, \ldots, q-۱\}$ با عمل جمع به پیمانه‌ی q را در نظر بگیرید. محاسبه‌ی لگاریتم‌های گسسته در این گروه، بدیهی است. فرض کنید ما $g, h \in \mathbb{Z}_q$ را دریافت کرده‌ایم که gف یک مولد است (بنابراین، $g \neq ۰$) و می‌خواهیم x را به شکلی بیابیم که $x.g = h \bmod q$ باشد. ازآنجاکه $g \neq ۰$، داریم $\gcd(g, q) = ۱$ و بنابراین، g دارای یک وارون ضربی $g^{-۱}$ به پیمانه‌ی q است. به‌علاوه، $g^{-۱}$ را می‌توان به‌صورت کارا، به شکل نشان داده‌شده در ضمیمه‌ی ب-۲-۲، محاسبه نمود. ولی در این حالت، $x = h.g^{-۱} \bmod q$ راه‌حل مطلوب است. توجه کنید که به شکل رسمی، x در اینجا نشان‌دهنده‌ی یک عدد صحیح است و نه یک عنصر گروهی- به‌هرحال، عمل گروه، جمع است و نه ضرب. بااین‌حال، در حل مسئله‌ی لگاریتم گسسته در $\mathbb{Z}_q$ می‌توانیم از این واقعیت بهره ببریم که یک عمل دیگر (یعنی، ضرب) را می‌توان بر روی عناصر آن گروه، تعریف نمود.

در رابطه با گروه‌های دارای اهمیت رمزنگاری، توجه خود را بر (زیرگروه‌های) $\mathbb{Z}_p^*$ برای p اول، متمرکز می‌کنیم. (ن.ک. بخش ۸-۳-۳.) مروری سطح بالا بر «الگوریتم حسابان نمایه‌ای» برای حل مسئله‌ی لگاریتم گسسته در چنین گروه‌هایی در زمان زیرنمایی را ارائه می‌کنیم. در حال حاضر، بهترین

الگوریتم شناخته‌شده برای این دسته از گروه‌ها، «غربال میدان اعداد»[۳۷] است که به‌صورت تجربی در زمان $2^{O((\log p)^{1/3}\cdot(\log\log p)^{2/3})}$ اجرا می‌شود. الگوریتم‌های زیرنمایی برای محاسبه‌ی لگاریتم‌های گسسته در زیرگروه‌های ضربی میدان‌های متناهی دارای مشخصه‌ی بزرگ نیز شناخته شده‌اند. در اوایل سال ۲۰۱۳، پیشرفت‌های چشمگیری در الگوریتم‌ها برای محاسبه‌ی لگاریتم‌های گسسته در زیرگروه‌های ضربی میدان‌های متناهی با مشخصه‌ی «کوچک» به وجود آمد؛ به نظر منطقی می‌رسد که از استفاده از چنین گروه‌هایی برای کاربردهای رمزنگاری، اجتناب کنیم.

نکته‌ی مهم این است که هیچ الگوریتم زیرنمایی برای محاسبه‌ی لگاریتم‌های گسسته در بعضی گروه‌های خم بیضوی خاص، شناخته نشده است. این امر به این معنا است که برای یک سطح امنیتی مفروض، می‌توانیم از گروه‌های با مرتبه‌ی کوچک‌تر هنگام کار کردن در گروه‌های خم بیضوی در مقایسه با مثلاً کار کردن در $\mathbb{Z}_p^*$ استفاده کنیم که منجر به طرح‌های رمزنگاری با کارایی مجانبی بهتر خواهد شد.

۹-۲-۱- الگوریتم پولیگ-هلمن

الگوریتم پولیگ-هلمن را می‌توان برای سرعت بخشیدن به محاسبه‌ی لگاریتم‌های گسسته در یک گروه $\mathbb{G}$ هنگامی استفاده نمود که هر تعداد از عوامل غیربدیهی مرتبه‌ی گروه، یعنی q، معلوم هستند. یادآوری می‌کنیم که مرتبه‌ی یک عنصر g، که در اینجا با $ord(g)$ نشان می‌دهیم عبارت است از کوچک‌ترین عدد صحیح مثبت i که برای آن، $g^i = 1$ باشد. ما به لم زیر نیز خواهیم داشت:

لم ۹-۴

فرض کنید $ord(g) = q$ و $p|q$. آنگاه $\text{ord}(g^p) = q/p$.

اثبات

ازآنجاکه $(g^p)^{q/p} = g^q = 1$، مرتبه‌ی g^p حداکثر برابر با q/p خواهد بود. فرض کنید $i > 0$ به صورتی باشد که $(g^p)^i = 1$. آنگاه، $g^{pi} = 1$ و ازآنجاکه q مرتبه‌ی g است، باید داشته باشیم $pi \geq q$ یا به شکل برابر $i \geq q/p$. بنابراین، مرتبه‌ی g^p دقیقاً برابر با q/p خواهد بود. ■

همچنین از یک تعمیم از قضیه‌ی باقیمانده‌ی چینی استفاده می‌کنیم: اگر $q = \prod_{i=1}^{k} q_i$ و $\gcd(q_i, q_j) = 1$ برای همه‌ی $i \neq j$، آنگاه

$$\mathbb{Z}_q \simeq \mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_k} \qquad \text{و} \qquad \mathbb{Z}_q^* \simeq \mathbb{Z}_{q_1}^* \times \cdots \times \mathbb{Z}_{q_k}^*$$

[۳۷] تصادفی نیست که نام این الگوریتم و زمان اجرای آن مشابه با غربال میدان اعداد عام برای تجزیه هستند به این دلیل که این الگوریتم‌ها دارای تعداد زیادی گام مبنای یکسان هستند.

(این را می‌توان *از طریق استقرا بر اساس* k با استفاده از قضیه‌ی باقیمانده‌ی چینی اساسی برای $k = ۲$ اثبات نمود.) به‌علاوه، با تعمیم الگوریتم در بخش ۸-۱-۵، امکان تبدیل کارا بین نمایش یک عنصر به‌عنوان عنصری از $\mathbb{Z}_q$ و نمایش آن به‌عنوان عنصری از $\mathbb{Z}_{q_1} \times \cdots \times \mathbb{Z}_{q_k}$ وجود دارد.

اکنون الگوریتم پولیگ-هلمن را توصیف می‌کنیم. یک مولد g و یک عنصر h را دریافت کرده‌ایم و می‌خواهیم یک x را بیابیم به صورتی که $g^x = h$ باشد. فرض کنید یک تجزیه‌ی $q = \prod_{i=1}^{k} q_i$ معلوم است به صورتی که $\{q_i\}$ دوبدو نسبت به هم اول هستند. (نیازی نیست که این، تجزیه‌ی اول کامل q باشد.) می‌دانیم که

$$(g^{q/q_i})^x = (g^x)^{q/q_i} = h^{q/q_i} \text{ for } i = 1, \ldots, k \qquad (۹.۲)$$

با قرار دادن $g_i \overset{\text{def}}{=} g^{q/q_i}$ و $h_i \overset{\text{def}}{=} h^{q/q_i}$ دارای k نمونه از یک مسئله‌ی لگاریتم گسسته در k گروه کوچک‌تر خواهیم بود. به‌صورت خاص، هر مسئله‌ی $g_i^x = h_i$ یک زیرگروه با اندازه‌ی $ord(g_i) = q_i$ (بر اساس لم ۹-۴) خواهد بود. (توجه کنید که هرکدام از چنین مسائلی تنها $[x \bmod q_i]$ را تعیین می‌کند؛ این امر از گزاره‌ی ۸-۵۳ نتیجه می‌شود.)

می‌توانیم هرکدام از k نمونه‌ی به‌دست‌آمده را با هر الگوریتم ارائه‌شده برای حل مسئله‌ی لگاریتم گسسته، حل کنیم. حل کردن این نمونه‌ها یک مجموعه از پاسخ‌های $\{x_i\}_{i=1}^{k}$ با $x_i \in \mathbb{Z}_{q_i}$ را می‌دهد که برای آن، $g_i^{x_i} = h_i = g_i^x$ است. گزاره‌ی ۸-۵۳ به این معنا است که $x = x_i \bmod q_i$ برای همه‌ی i. بر اساس قضیه‌ی باقیمانده‌ی چینی تعمیم‌یافته که قبلاً بحث شد، محدودیت‌های

$$\begin{aligned} x &= x_1 \bmod q_1 \\ &\vdots \\ x &= x_k \bmod q_k \end{aligned}$$

به شکل منحصربه‌فرد x به پیمانه‌ی q را تعیین می‌کند و راه‌حل مطلوب x را می‌توان به‌صورت کارا از روی $\{x_i\}$ بازسازی نمود.

مثال ۹-۵

مسئله‌ی محاسبه‌ی لگاریتم‌های گسسته در $\mathbb{Z}_{۳۱}^*$، یک گروه با مرتبه‌ی $q = ۳۰ = ۵.۳.۲$ را در نظر بگیرید. فرض کنید $g = ۳$ و $h = ۲۶ = g^x$ با x مجهول باشد. داریم:

$$(g^{۳۰/۵})^x = h^{۳۰/۵} \Rightarrow (۳^۶)^x = ۲۶^۶ \Rightarrow ۱۶^x = ۱$$
$$(g^{۳۰/۳})^x = h^{۳۰/۳} \Rightarrow (۳^{۱۰})^x = ۲۶^{۱۰} \Rightarrow ۲۵^x = ۵$$
$$(g^{۳۰/۲})^x = h^{۳۰/۲} \Rightarrow (۳^{۱۵})^x = ۲۶^{۱۵} \Rightarrow ۳۰^x = ۳۰$$

(تمام معادلات بالا به پیمانه‌ی ۳۱ هستند.) داریم $ord(16) = 5$، $ord(25) = 3$ و $ord(30) = 2$. با حل هر معادله، به دست می‌آوریم

$$x = 0 \bmod 5, x = 2 \bmod 3, \text{ and } x = 1 \bmod 2,$$

و بنابراین $x = 5 \bmod 30$ درواقع، $2^5 = 26 \bmod 31$. ◊

اگر q دارای تجزیه‌ی اول (معلوم) $q = \prod_{i=1}^{k} p_i^{e_i}$ باشد، آنگاه، با استفاده از الگوریتم پولیگ-هلمن، زمان موردنیاز برای محاسبه‌ی الگوریتم‌های گسسته در یک گروه با مرتبه‌ی q از طریق محاسبه‌ی یک لگاریتم گسسته در یک زیرگروه با اندازه‌ی $\max_i\{p_i^{e_i}\}$ تعیین می‌شود. این مقوله را می‌توان به محاسبه‌ی یک لگاریتم گسسته در یک زیرگروه با اندازه‌ی $\max_i\{p_i\}$ کاهش داد؛ ن.ک. تمرین ۹-۵.

۹-۲-۲- الگوریتم قدم کوچک-قدم بزرگ

الگوریتم قدم کوچک-قدم بزرگ لگاریتم‌های گسسته را در یک گروه با مرتبه‌ی q با استفاده از $O(\sqrt{q})$ عملیات گروهی، محاسبه می‌کند. این ایده، ساده است. با داشتن یک مولد $g \in \mathbb{G}$، می‌توانیم توان‌های g را شکل‌دهنده‌ی یک چرخه در نظر بگیریم:

$$1 = g^0, g^1, g^2, \ldots, g^{q-2}, g^{q-1}, g^q = 1.$$

می‌دانیم که h باید جایی در این چرخه قرار داشته باشد. محاسبه‌ی تمام نقاط در این چرخه نیازمند $\Omega(q)$ زمان خواهد بود. به‌جای این کار، ما چرخه را در دامنه‌های به‌اندازه‌ی $t \stackrel{\text{def}}{=} \lfloor\sqrt{q}\rfloor$ «علامت‌گذاری» می‌کنیم؛ به شکل دقیق‌تر، ما $\lfloor q/t \rfloor + 1 = O(\sqrt{q})$ عنصر زیر را محاسبه و ذخیره می‌کنیم:

$$g^0, g^t, g^{2t}, \ldots, g^{\lfloor q/t \rfloor \cdot t}.$$

(این‌ها، «قدم‌های بزرگ» هستند.) توجه کنید که شکاف بین «علامت‌های» متوالی حداکثر برابر با t خواهد بود. به‌علاوه، می‌دانیم که $h = g^x$ در یکی از این شکاف‌ها قرار می‌گیرد. بنابراین، اگر در ادامه، «قدم‌های کوچک» برداریم و t عنصر زیر را محاسبه کنیم:

$$h \cdot g^1, \ldots, h \cdot g^t,$$

که هرکدام متناظر با یک «انتقال» h است، می‌دانیم که یکی از این مقادیر برابر با یکی از نقاطی خواهد بود که علامت‌گذاری کرده‌ایم. فرض کنید ما درمی‌یابیم که $h \cdot g^i = g^{k \cdot t}$. در ادامه به‌سادگی می‌توانیم $\log_g h := [(kt - i) \bmod q]$ را محاسبه کنیم. شبه کد برای این الگوریتم در ادامه آورده شده است.

الگوریتم ۹-۶

الگوریتم قدم کوچک-قدم بزرگ

ورودی: عناصر $g, h \in \mathbb{G}$؛ مرتبه q برای $\mathbb{G}$

خروجی: $\log_g h$

$t := \lfloor\sqrt{q}\rfloor$
for $i = 0$ to $\lfloor q/t \rfloor$:
compute $g_i := g^{i \cdot t}$
زوج‌های (i, g_i) را بر اساس مؤلفه‌ی دوم، مرتب کنید.
for $i = 1$ to t:
compute $h_i := h \cdot g^i$
اگر برای k مفروض، داشته باشیم $h_i = g_k$،
مقدار $[kt - i \bmod q]$ را بازگردان.

این الگوریتم نیازمند $O(\sqrt{q})$ توان رسانی/ضرب در $\mathbb{G}$ است. (درواقع، به‌غیراز اولین مقدار $g_1 = g^t$، هر مقدار g_i را می‌توان با استفاده از یک ضرب یکتا به‌صورت $g_i := g_{i-1} \cdot g_1$ محاسبه نمود. به شکل مشابه، هر h_i را می‌توان به‌صورت $h_i := h_{i-1} \cdot g$ محاسبه نمود.) مرتب کردن $O(\sqrt{q})$ زوج $\{(i, g_i)\}$ نیازمند زمان $O(\sqrt{q} \cdot \log q)$ خواهد بود و در ادامه می‌توانیم از جستجوی دودویی برای بررسی این امر استفاده کنیم که آیا هر h_i برابر با یک g_k مفروض در زمان $O(\log q)$ است یا خیر. بنابراین، کل الگوریتم در زمان $O(\sqrt{q}.polylog(q))$ اجرا می‌شود.

مثال ۹-۷

یکی از کاربردهای این الگوریتم را در گروه دوری $\mathbb{Z}_{29}^*$ با مرتبه‌ی $q = 29 - 1 = 28$ نشان می‌دهیم. قرار دهید $g = 2$ و $h = 17$. قرار می‌دهیم $t = 5$ و محاسبه می‌کنیم:

$$2^0 = 1, \quad 2^5 = 3, \quad 2^{10} = 9, \quad 2^{15} = 27, \quad 2^{20} = 23, \quad 2^{25} = 11.$$

(باید بدانیم که تمام عملیات در $\mathbb{Z}_{29}^*$ هستند.) سپس محاسبه می‌کنیم:

$$17.2^1 = 5, \quad 17.2^2 = 10, \quad 17.2^3 = 20, \quad 17.2^4 = 11,$$

و توجه کنید که $17.2^4 = 11 = 2^{25}$. بنابراین، داریم $\log_2 17 = 25 - 4 = 21$ ◊

۹-۲-۳- لگاریتم‌های گسسته از روی برخوردها

یک نقطه‌ضعف الگوریتم قدم کوچک-قدم بزرگ این است که از مقدار زیادی حافظه استفاده می‌کند چراکه نیازمند ذخیره‌ی $O(\sqrt{q})$ نقطه است. می‌توانیم یک الگوریتم را به دست آوریم که از حافظه‌ی ثابت استفاده می‌کند- و دارای زمان اجرای مجانبی یکسان است- به این صورت که از رابطه‌ی بین

مسئله‌ی لگاریتم گسسته و چکیده‌سازی مقاوم در برابر برخورد، نشان داده شده در بخش ۸-۴-۲ و یادآوری حمله‌ی روز تولد فضای کوچک برای یافتن برخوردها از بخش ۵-۴-۲ استفاده کنیم.

ایده‌ی سطح بالا را توصیف می‌کنیم. یک پایه‌ی $g \in \mathbb{G}$ و یک عنصر مفروض $h \in \langle g \rangle$ را در نظر بگیرید. از نتایج بخش ۸-۴-۲ به یاد داریم که اگر تابع چکیده ساز $H_{g,h}: \mathbb{Z}_q \times \mathbb{Z}_q \to \mathbb{G}$ را توسط $H_{g,h}(x_1, x_2) = g^{x_1} h^{x_2}$ تعریف کنیم، آنگاه یافتن یک برخورد در $H_{g,h}$ به‌صورت ضمنی نشان‌دهنده‌ی توانایی برای محاسبه‌ی $\log_g h$ است. (ن.ک. اثبات قضیه‌ی ۸-۷۹.) بنابراین، مسئله‌ی محاسبه‌ی $\log_g h$ را به مسئله‌ی یافتن یک برخورد در یک تابع چکیده ساز تقلیل داده‌ایم، یعنی کاری که می‌دانیم چگونه در زمان $O(\sqrt{|\mathbb{G}|}) = O(\sqrt{q})$ با استفاده از یک حمله‌ی روز تولد، انجام دهیم! به‌علاوه، یک حمله‌ی روز تولد فضای کوچک یک برخورد را در زمان یکسان و فضای ثابت، می‌دهد.

تنها چیزی که باقی می‌ماند پرداختن به تعدادی جزئیات فنی است. یکی از این جزئیات این است که حمله‌ی روز تولد فضای کوچک توصیف‌شده در بخش ۵-۴-۲ فرض می‌کند که برد تابع چکیده ساز یک زیرمجموعه از دامنه‌ی آن است؛ در اینجا این امر برقرار نیست و درواقع (بسته به نمایش استفاده‌شده برای عناصر $\mathbb{G}$) ممکن است $H_{g,h}$ فشرده‌سازی نکند. یک مسئله‌ی دوم این است که تحلیل در بخش ۵-۴-۲ تابع چکیده ساز را به‌عنوان یک تابع تصادفی در نظر گرفت درحالی‌که $H_{g,h}$ دارای مقدار زیادی ساختار جبری است.

الگوریتم رو پلارد یک روش را برای حل مسائل بالا ارائه می‌کند. ما یک الگوریتم متفاوت را توصیف می‌کنیم که می‌توان آن را به‌عنوان یک پیاده‌سازی مستقیم‌تر از ایده‌های بالا در نظر گرفت. (در عمل، الگوریتم پلارد، کاراتر خواهد بود هرچند هر دو الگوریتم تنها از $O(\sqrt{q})$ عملیات گروهی استفاده می‌کنند.) فرض کنید $F: \mathbb{G} \to \mathbb{Z}_q \times \mathbb{Z}_q$ نشان‌دهنده‌ی یک تابع چکیده ساز رمزنگاری به‌دست‌آمده توسط برای مثال یک تغییر مناسب در $SHA-1$ باشد. $H: \mathbb{G} \to \mathbb{G}$ را توسط $H(k) \stackrel{\text{def}}{=} H_{g,h}(F(k))$ تعریف کنید. می‌توانیم از الگوریتم ۵-۹ با تغییرات طبیعی برای یافتن یک برخورد در H با استفاده از $O(\sqrt{|\mathbb{G}|}) = O(\sqrt{q})$ ارزیابی از H در مقدار مورد انتظار (و حافظه‌ی ثابت) بهره ببریم. با احتمال بسیار بالا، این مقوله یک برخورد را در $H_{g,h}$ می‌دهد. در تمرین ۹-۶ از شما خواسته شده است تا جزئیات این روش را مشخص کنید.

در اینجا شایان توجه است که یک اثبات امنیت بر اساس سختی مسئله‌ی لگاریتم گسسته- یعنی اینکه نشان‌دهنده‌ی یک تابع چکیده ساز مقاوم در برابر برخورد است- منجر به یک الگوریتم بهتر برای حل همان مسئله می‌شود! اندکی تأمل ما را قانع می‌کند که این مسئله‌ای تعجب‌آور نیست: یک اثبات از طریق تقلیل نشان می‌دهد که یک حمله علیه یک ساختار مفروض (در این حالت، یافتن برخوردها در تابع چکیده ساز) به‌صورت مستقیم یک حمله علیه فرض مبنا (در اینجا، سختی مسئله‌ی لگاریتم گسسته) را می‌دهد، که دقیقاً همان ویژگی است که الگوریتم بالا از آن استفاده می‌کند.

۹-۲-۴- الگوریتم حسابان نمایه‌ای

این بحث را با نگاهی مختصر به «الگوریتم حسابان نمایه‌ای» (غیرعام) برای محاسبه‌ی لگاریتم‌های گسسته در گروه دوری $\mathbb{Z}_p^*$ (برای p اول) به پایان می‌بریم. برخلاف الگوریتم‌های (عام) قبلی، این رویکرد دارای زمان اجرای زیرنمایی است. این الگوریتم تا حدودی مشابه با الگوریتم غربال مربعی معرفی‌شده در بخش ۹-۱-۳ است و فرض می‌کنیم خوانندگان با بحث آنجا آشنایی دارند. مشابه با آن بخش، ما ایده‌های اصلی روش حسابان نمایه‌ای را موردبحث قرار می‌دهیم ولی تحلیل دقیق را مورداشاره قرار نمی‌دهیم. همچنین، تعدادی ساده‌سازی برای شفاف نمودن بحث ارائه می‌شوند.

روش حسابان نمایه‌ای از یک فرآیند دومرحله‌ای استفاده می‌کند. نکته‌ی مهم آن است که مرحله‌ی اول تنها نیازمند دانستن پیمانه‌ی p و پایه‌ی g است و بنابراین آن را می‌توان به‌عنوان یک گام پیش‌پردازش قبل از دانستن h- مقداری که لگاریتم گسسته‌ی آن را می‌خواهیم حساب کنیم- اجرا نمود. به همین دلیل، کافی است گام اول را تنها یک‌بار اجرا کنیم تا چندین نمونه از مسئله‌ی لگاریتم گسسته را حل کنیم (مادامی‌که تمام این نمونه‌ها دارای p و g یکسان باشند).

گام اول

یک کران B مفروض را در نظر گرفته و فرض کنید $\{p_1, \ldots, p_k\}$ عبارت باشد از مجموعه‌ی اعداد اول کوچک‌تر مساوی B. در این گام، ما $\ell \geq k$ مقدار متمایز $x_1, \ldots, x_\ell \in \mathbb{Z}_{p-1}$ را می‌یابیم که برای آن‌ها $g_i \stackrel{\text{def}}{=} [g^{x_i} \bmod p]$ به‌صورت B-هموار باشد. این امر با انتخاب $\{x_i\}$ یکنواخت تا زمان یافتن مقادیر مناسب، انجام می‌شود.

با تجزیه‌ی اعداد B-هموار به‌دست‌آمده، l معادله‌ی زیر را خواهیم داشت:

$$g^{x_1} = \prod_{i=1}^{k} p_i^{e_{1,i}} \bmod p$$
$$\vdots$$
$$g^{x_\ell} = \prod_{i=1}^{k} p_i^{e_{\ell,i}} \bmod p$$

با گرفتن لگاریتم‌های گسسته، می‌توانیم این معادلات را به معادلات خطی ذیل تبدیل کنیم:

$$x_1 = \sum_{i=1}^{k} e_{1,i} \cdot \log_g p_i \bmod (p-1) \qquad (۹.۳)$$

$$\vdots$$

$$x_\ell = \sum_{i=1}^{k} e_{\ell,i} \cdot \log_g p_i \bmod (p-1) \qquad (۹.۴)$$

توجه کنید که $\{x_i\}$ و $\{e_{i,j}\}$ معلوم هستند درحالی‌که $\{\log_g p_i\}$ مجهول است.

گام دوم

اکنون یک عنصر h به ما داده می‌شود و می‌خواهیم $\log_g h$ را محاسبه کنیم. در اینجا، یک مقدار $x \in \mathbb{Z}_{p-1}$ را می‌یابیم که برای آن، $[g^x \cdot h \bmod p]$، B-هموار باشد. (بازهم این امر با انتخاب x به‌صورت یکنواخت انجام می‌شود.) فرض کنید

$$g^x \cdot h = \prod_{i=1}^{k} p_i^{e_i} \bmod p$$

$$\Rightarrow x + \log_g h = \sum_{i=1}^{k} e_i \cdot \log_g p_i \bmod (p-1),$$

که در آن، x و $\{e_i\}$ معلوم هستند. در ترکیب با معادلات (۹.۴) − (۹.۳)، تعداد $\ell + 1 \geq k + 1$ معادله‌ی خطی در $k + 1$ مجهول $\{\log_g p_i\}_{i=1}^{k}$ و $\log_g h$ داریم. با استفاده از روش‌های جبر خطی[۳۸] (و با فرض اینکه سیستم معادلات کم تعریف نیست)، می‌توانیم معادلات را برای یافتن هرکدام از مجهولات حل کنیم و علی‌الخصوص، راه‌حل مطلوب $\log_g h$ را به دست آوریم.

مثال ۹-۸

فرض کنید $p = ۱۰۱$، $g = ۳$ و $h = ۸۷$. داریم $[۳^{۱۰} \bmod ۱۰۱] = ۶۵ = ۵.۱۳$. به شکل مشابه، $[۳^{۱۲} \bmod ۱۰۱] = ۸۰ = ۲^۴.۵$ و $[۳^{۱۴} \bmod ۱۰۱] = ۱۳$. بنابراین، معادلات خطی زیر را داریم:

$$۱۰ = \log_۳ ۵ + \log_۳ ۱۳ \bmod ۱۰۰$$

$$۱۲ = ۴۰ \cdot \log_۳ ۲ + \log_۳ ۵ \bmod ۱۰۰$$

$$۱۴ = \log_۳ ۱۳ \bmod ۱۰۰.$$

[۳۸] به شکل فنی، وضعیت اندکی پیچیده‌تر است چراکه معادلات خطی همگی به پیمانه‌ی $p - ۱$ هستند که عددی اول نیست. بااین‌حال، روش‌هایی برای حل این مشکل وجود دارد.

همچنین داریم $۳^۵ \cdot ۸۷ = ۳۲ = ۲^۵ \bmod ۱۰۱$، یا

$$۵ + \log_۳ ۸۷ = ۵ \cdot \log_۳ ۲ \bmod ۱۰۰ \qquad (۹.۵)$$

با افزودن معادلات دوم و سوم و تفریق معادله‌ی اول، $۴.\log_۳ ۲ = ۱۶ \bmod ۱۰۰$ را به دست می‌آوریم. این رابطه، $\log_۳ ۲$ را به‌صورت منحصربه‌فرد، تعیین نمی‌کند (ازآنجاکه ۴ وارون‌پذیر به پیمانه‌ی ۱۰۰ نیست)، ولی به ما می‌گوید که $\log_۳ ۲ = ۴, ۲۹, ۵۴, or\ ۷۹$ (ن.ک. تمرین ۹-۳). امتحان کردن تمام گزینه‌ها می‌دهد $\log_۳ ۲ = ۲۹$. با قرار دادن این مقدار در معادله‌ی (۹.۵) خواهیم داشت $\log_۳ ۸۷ = ۴۰$. ◊

زمان اجرا

انتخاب یک مقدار بزرگ‌تر برای B این احتمال را بالا می‌برد که یک مقدار یکنواخت در $\mathbb{Z}_p^*$، B-هموار باشد؛ بااین‌حال، این امر به این معنا است که باید برای شناسایی و تجزیه‌ی اعداد B-هموار سخت‌تر کارکنیم و باید تعداد بیشتری از آن‌ها را بیابیم. به این دلیل که سیستم معادلات، بزرگ‌تر خواهد بود، حل سیستم زمان بیشتری طول می‌کشد. انتخاب مقدار بهینه‌ی B الگوریتمی را می‌دهد که (حداقل به‌صورت تجربی) لگاریتم‌های گسسته در $\mathbb{Z}_p^*$ را در زمان $۲^{O(\sqrt{\log p \log\log p})}$ محاسبه می‌کند. (درواقع، عبارت ثابت در توان را می‌توان به‌صورت نسبتاً دقیق تعیین نمود.) نکته‌ی مهم برای اهداف ما این است که این مقدار به‌صورت زیرنمایی بر اساس طول p خواهد بود.

۹-۳- طول کلیدهای پیشنهادی

دانستن بهترین الگوریتم‌های در دسترس برای حل مسائل رمزنگاری مختلف دارای اهمیت حیاتی برای تعیین یک طول کلید مناسب برای رسیدن به یک سطح امنیت موردنظر است. جدول زیر به‌صورت خلاصه، طول کلیدهایی را که در حال حاضر توسط موسسه‌ی ملی استانداردها و فناوری ایالات‌متحده[۳۹] (*NIST*) پیشنهاد می‌شوند، نشان می‌دهد [۱۳]:

طول کلید مؤثر	*RSA*	لگاریتم گسسته	
	طول پیمانه	زیرگروه مرتبه-q از $\mathbb{Z}_p^*$	مرتبه گروه خم بیضوی، q
۱۱۲	۲۰۴۸	p: ۲۰۴۸, q: ۲۲۴	۲۲۴
۱۲۸	۳۰۷۲	p: ۳۰۷۲, q: ۲۵۶	۲۵۶
۱۹۲	۷۶۸۰	p: ۷۶۸۰, q: ۳۸۴	۳۸۴
۲۵۶	۱۵۳۶۰	p: ۱۵۳۶۰, q: ۵۱۲	۵۱۲

تمام ارقام در این جدول بر اساس بیت اندازه‌گیری می‌شوند. «طول کلید مؤثر» یک مقدار n است به صورتی که بهترین الگوریتم شناخته‌شده برای حل یک مسئله، نیازمند زمان تقریبی ۲^n است؛ یعنی، دشواری محاسباتی حل یک مسئله تقریباً برابر با دشواری اجرای یک جستجوی فراگیر علیه یک طرح

[۳۹] گروه‌های دیگر هم پیشنهادهای خود را مطرح کرده‌اند؛ ن.ک. http://keylength.com

کلید متقارن با یک کلید n-بیتی یا زمان لازم برای یافتن برخوردها در یک تابع چکیده ساز با طول خروجی $2n$-بیتی است. *NIST* باور دارد که یک طول کلید ۱۱۲-بیتی برای امنیت تا سال ۲۰۳۰ قابل‌قبول خواهد بود ولی طول کلیدهای ۱۲۸-بیتی یا طولانی‌تر را برای کاربردهایی پیشنهاد می‌دهد که امنیت فراتر از آن تاریخ برای آن‌ها موردنیاز است.

با توجه به مطالبی که در این فصل یاد گرفتیم، بهتر است نگاهی دقیق‌تر به بعضی از مقادیر در این جدول بیندازیم. اولین چیزی که متوجه می‌شویم آن است که گروه‌های خم بیضوی را می‌توان برای به دست آوردن هر سطح مفروض امنیت با پارامترهای کوچک‌تری در مقایسه با پارامترهای موردنیاز برای *RSA* یا زیرگروه‌های $\mathbb{Z}_p^*$ مورداستفاده قرار داد. این امر به این دلیل است که هیچ الگوریتم زیرنمایی برای حل مسئله‌ی لگاریتم گسسته در چنین گروه‌هایی (هنگامی‌که به‌درستی انتخاب شده باشند) شناخته نشده است. بااین‌حال، رسیدن به امنیت n-بیتی نیازمند یک گروه خم بیضوی است که مرتبه‌ی q آن دارای طول $2n$ بیت باشد. این امر یکی از نتایج الگوریتم‌های عامی است که در این فصل مشاهده کرده‌ایم که مسئله‌ی لگاریتم گسسته (در هر گروهی) را در زمان $O(\sqrt{q}) \approx 2^n$ حل می‌کنند.

با بازگشت به حالت $\mathbb{Z}_p^*$ مشاهده می‌کنیم که در اینجا هم یک مقدار $2n$-بیتی برای q برای امنیت n-بیتی موردنیاز است. بااین‌حال، طول p باید به شکل چشم‌گیری بزرگ‌تر باشد به این دلیل که غربال میدان اعداد را می‌توان برای محاسبه‌ی لگاریتم‌های گسسته در $\mathbb{Z}_p^*$ در زمان زیرنمایی بر اساس طول p استفاده نمود. (یعنی، p و q به‌گونه‌ای انتخاب می‌شوند که زمان اجرای غربال میدان اعداد که به طول p بستگی دارد و زمان اجرای یک الگوریتم عام که به طول q بستگی دارد تقریباً باهم برابر باشند.) نتایج عملی این مسئله برای هر سطح امنیت مطلوبی عبارت‌اند از سیستم‌های رمزنگاری خم بیضوی که می‌توانند از پارامترهای بسیار کوچک‌تر با عملیات گروهی ازلحاظ مجانبی سریع‌تر برای طرفین صادق در مقایسه با سیستم‌های رمزنگاری مبتنی بر زیرگروه‌های $\mathbb{Z}_p^*$ استفاده کنند.

مراجع و مطالعه‌ی بیشتر

الگوریتم $p-1$ پلارد در سال ۱۹۷۴ منتشر شد [۱۳۹] و روش رو وی برای تجزیه سال بعد ارائه شد [۱۴۰]. الگوریتم غربال مربعی توسط پومرانس [۱۴۲] بر اساس ایده‌های قبلی دیکسون [۶۰] ارائه‌شده است.

الگوریتم قدم کوچک-قدم بزرگ توسط شانکس [۱۵۳] بیان شده است. الگوریتم پولیگ-هلمن در سال ۱۹۷۸ منتشر شد [۱۳۸] و همین‌طور الگوریتم رو پلارد برای محاسبه‌ی لگاریتم‌های گسسته [۱۴۱] در همین سال ارائه شد. الگوریتم حسابان نمایه‌ای به شکل توصیف‌شده در اینجا توسط ادلمن [۴] ارائه‌شده است.

متون واگستاف [۱۷۳]، شوپ [۱۵۹]، کراندال و پومرانس [۵۱]، جوکس [۹۶] و گالبریث [۶۹] اطلاعات بیشتری در خصوص الگوریتم‌های مناسب برای تجزیه و محاسبه‌ی لگاریتم‌های گسسته، شامل توصیف‌های غربال میدان اعداد (عام) به دست می‌دهند. اخیراً، یک الگوریتم بهبودیافته برای مسئله‌ی لگاریتم گسسته در میدان‌های متناهی با مشخصه‌ی کوچک، اعلام شد [۱۱].

کران‌های پایین‌تر بر روی زمان اجرای الگوریتم‌های عام برای محاسبه‌ی لگاریتم‌های گسسته، که ازلحاظ مجانبی منطبق بر زمان اجرای الگوریتم‌های توصیف‌شده در اینجا هستند، توسط نچاف [۱۳۳] و شوپ [۱۵۵] ارائه‌شده‌اند.

لنسترا و فرهول [۱۱۳] یک بحث جامع در خصوص چگونگی تأثیر الگوریتم‌های شناخته‌شده برای تجزیه و محاسبه‌ی لگاریتم‌های گسسته بر انتخاب پارامترهای رمزنگاری در عمل، ارائه می‌کنند.

تمرین‌ها

۹-۱: به‌منظور سرعت بخشیدن به الگوریتم تولید کلید برای *RSA*، پیشنهاد شده است که یک عدد اول را با تولید تعداد زیادی عدد اول تصادفی کوچک، ضرب کردن آن‌ها در هم و اضافه کردن یک (البته سپس بررسی اینکه نتیجه، اول است یا خیر) تولید کنیم. با نادیده گرفتن مسئله‌ی احتمال اینکه چنین مقداری واقعاً یک عدد اول باشد، در مورد این روش چه فکری می‌کنید؟

۹-۲: در یک اجرای الگوریتم ۹-۲، $x^{(i)} \stackrel{\text{def}}{=} F^{(i)}(x^{(\cdot)})$ را تعریف کنید. نشان دهید که اگر، در یک اجرای مفروض از این الگوریتم، $i,j \leq 2^{n/2}$ وجود داشته باشند به صورتی که $x^{(i)} \neq x^{(j)}$ باشد ولی $x^{(i)} = x^{(j)} \bmod p$، آنگاه این اجرای الگوریتم دارای خروجی p خواهد بود. (تحلیل در اینجا اندکی متفاوت از تحلیل الگوریتم ۵-۹ است چراکه الگوریتم‌ها-و اهداف آن‌ها- اندکی متفاوت هستند.)

۹-۳: (الف) نشان دهید که اگر $ab = c \bmod N$ و $\gcd(b,N) = d$، آنگاه:

I. $d|c$؛

II. $a \cdot (b/d) = (c/d) \bmod (N/d)$ و

III. $\gcd(b/d, N/d) = 1$.

(ب) چگونگی استفاده از مطلب بالا برای محاسبه‌ی لگاریتم‌های گسسته در $\mathbb{Z}_N$ به‌صورت کارا را توصیف کنید حتی زمانی که پایه‌ی g یک مولد برای $\mathbb{Z}_N$ نیست.

۹-۴: در اینجا نشان می‌دهیم که چگونه می‌توان مسئله‌ی لگاریتم گسسته را در یک گروه دوری با مرتبه‌ی $q = p^e$ در زمان $O(polylog(q).\sqrt{p})$ حل نمود. با داشتن یک مولد g با مرتبه‌ی $q = p^e$ و یک مقدار h به‌عنوان ورودی، می‌خواهیم $x = \log_g h$ را محاسبه کنیم.