دانشکده مهندسی کامپیوتر

تمرین ایتریم

امنیت سیستم‌های کامپیوتری

مدرس: دکتر ابوالفضل دیانت

محمدحسین عباسپور، فرزان رحمانی

شماره دانشجویی: ۹۹۵۲۱۴۳۳، ۹۹۵۲۱۲۷۱

نیم سال دوم
سال تحصیلی ۱۴۰۳-۱۴۰۲

سوال ۱

در الگوریتم RSA، لازم نیست gcd(m،n)=۱. در اینجا، m پیامی را که باید رمزگذاری شود، نشان می‌دهد و n ماژول است، که حاصل ضرب دو عدد اول بزرگ p و q است. شرط لازم برای RSA برای درست کار کردن این است که m باید در محدوده صفر تا n باشد.

با این حال، در طول مرحله تولید کلید، بسیار مهم است که توان رمزگذاری انتخاب شده e و $\phi(n)$ (که در آن $\phi(n) = (p-1)(q-1)$) coprime باشند، یعنی، $\gcd(e, \phi(n)) = 1$. این تضمین می کند که e دارای معکوس d است که برای رمزگشایی استفاده می شود.

در اینجا نیز یک استدلال کوتاه آمده است: اگر فاکتور مشترکی با داشته باشد، سیستم رمزنگاری RSA همچنان کار خواهد کرد. برای دیدن این، فرض کنید که $p|m$. سپس $c \equiv m^e \equiv 0 \pmod p$ و $c^d \equiv 0 \equiv m \pmod p$. می بینیم که خاصیت $(m^d)^e \equiv m$ برای همه عوامل اول n صرف نظر از اینکه m را عاد می کنند یا نه برقرار است و بنابراین $(m^d)^e \mod n$ صرف نظر از اینکه m نسبت به n اول باشد یا نه صادق است. البته در رمزنگاری بسیار بعید است که m نسبت به n اول نباشد. (پیوند به اثبات دقیق تر و کامل تر)

سوال ۲

در الگوریتم RSA بهتر است پارامتر e فرد باشد تا زوج. این به این دلیل است که e باید نسبت به $\phi(n)$ اول باشد، که در آن $\phi(n) = (p-1)(q-1)$. از آنجایی که $\phi(n)$ همیشه زوج است (چرا که هر دو $p-1$ و $q-1$ زوج هستند)، e زوج نسبت به $\phi(n)$ اول نخواهد بود، مگر اینکه ۱ باشد، که برای رمزگذاری مناسب نیست. به طور معمول، مقادیر کوچک فرد مانند ۳، ۱۷، یا ۶۵۵۳۷ برای e انتخاب می‌شوند، زیرا برای رمزگذاری کارآمد هستند و اطمینان می‌دهند که بزرگترین مقسوم‌کننده مشترک با $\phi(n)$ برابر با ۱ است. بنابراین در حالی که الگوریتم RSA برای مقادیر زوج e نیز به صورت ریاضی کار می‌کند، انتخاب یک نمای عمومی فرد e ویژگی‌های امنیتی و مزایای محاسباتی بهتری را فراهم می‌کند. مقادیر رایج مورد استفاده برای e در عمل ۳، ۱۷، و ۶۵۵۳۷ هستند که همگی فرد هستند.

سوال ۳

اعداد فرما دنباله خاصی از اعداد هستند که با فرمول تعریف می شوند:

$$F_n = 2^{2^n} + 1$$

که در آن n یک عدد صحیح غیر منفی است. چند عدد اول فرما عبارتند از:

- $F_0 = 3$
- $F_1 = 5$
- $F_2 = 17$
- $F_3 = 257$
- $F_4 = 65537$

نقش اعداد فرما در تولید پارامتر RSA در الگوریتم RSA، انتخاب توان عمومی e به عنوان عدد فرما، به ویژه $F_4 = 65537$، رایج است. این انتخاب مطلوب است زیرا:

۱. کارایی(Efficiency): استفاده از ۶۵۵۳۷ (که $2^{16}+1$ است) به عنوان e رمزگذاری کارآمد و تأیید امضا را به دلیل وزن کم همینگ(Hamming weight) آن تضمین می کند (یعنی فقط دو بیت ۱ در نمایش باینری خود دارد).

۲. امنیت(Security): اعداد فرما که اعداد اول نسبتاً بزرگ هستند، خاصیت لازم را حفظ می کنند که e نسبت به $\phi(n)$ هم اول باشد.

۳. ضرورت(Necessity): اعداد اول هستند که لازمه الگوریتم RSA است.

۴. اعداد فرد(Odd numbers): آنها اعداد فرد هستند و ما ترجیح می دهیم e فرد باشد همان طور که در سوال ۲ بیان شد.

بنابراین به طور خلاصه، در حالی که مورد نیاز نیست، استفاده از یک عدد اول فرما مانند ۶۵۵۳۷ به عنوان توان عمومی e در RSA یک روش معمول توصیه شده برای تولید پارامترهای RSA قوی و قابل همکاری است. ۶۵۵۳۷ تعادلی بین کارایی محاسباتی و امنیت برقرار می کند و آن را به یک انتخاب محبوب در پیاده سازی RSA تبدیل می کند.

سوال ۴

مقاله RSA چند الگوریتم کارآمد را برای انجام عملیات توان ماژولار $M^e \mod n$ و $C^d \mod n$ در طول رمزگذاری و رمزگشایی به ترتیب ذکر می‌کند. چند نمونه از الگوریتم های بهینه مورد بحث عبارتند از:

۱. توان با مربع و ضرب مکرر(Exponentiation by repeated squaring and multiplication) (بخش VII.A) این یک الگوریتم اساسی است که در خود مقاله ارائه شده است. زمانی که بیت متناظر در نمایش باینری e برابر با ۱ است، $M^e \mod n$ را با مجذور کردن مکرر M و ضرب در M محاسبه می‌کند. پیچیدگی زمانی آن $O(\log e)$ است.

۲. رویه های کارآمدتر (More efficient procedures) (بخش VII.A) این مقاله ذکر می‌کند که روش‌های کارآمدتری نسبت به روش مربع‌سازی مکرر پایه شناخته شده‌اند، بدون اینکه وارد جزئیات شوند. چند نمونه عبارتند از:

- الف) توان پنجره کشویی(Sliding Window Exponentiation) - توان های کوچک را از قبل محاسبه کرده و مجدداً برای سرعت بخشیدن به قدرت استفاده می کند.
- ب) توان زنجیره جمع(Addition Chain Exponentiation) - M^e را با بهره برداری از زنجیره جمع برای e محاسبه می کند.

۳. الگوریتم های مطالعه شده توسط Knuth (بخش VII.A) این مقاله به کتاب اصلی کنوت "هنر برنامه نویسی کامپیوتری" اشاره می کند که الگوریتم های قدرت را با جزئیات زیاد مورد مطالعه قرار می دهد، از جمله:

- الف) توان دودویی (Binary exponentiation)
- ب) استفاده از زنجیر اضافه (Use of addition chains)
- ج) بهره برداری از الگوهای توان ویژه (Exploiting special exponent patterns)
- د) ضرب های معامله ای برای تربیع (Trading multiplications for squarings)

۴. الگوریتم ضرب مونتگومری (Montgomery multiplication algorithm) در حالی که به صراحت در مقاله اصلی RSA ذکر نشده است، روش ضرب مونتگومری یک تکنیک بهینه برای انجام ضرب های ماژولار در طول توان است، و از عملیات تقسیم پرهزینه جلوگیری می کند.

۵. کاهش بارت(Barrett : Reduction) این برای بهبود کارایی مرحله کاهش ماژولار در طول توان استفاده می شود. مقادیری را از پیش محاسبه می کند که به جای آن با استفاده از ضرب و تفریق، امکان کاهش ماژولار را بدون تقسیم انجام می دهد.

این تکنیک‌ها و سایر تکنیک‌های پیشرفته مانند استفاده از سخت‌افزار محاسباتی ماژولار می‌توانند به طور قابل‌توجهی عملیات قدرت‌یابی ماژولار هسته را در RSA، که محاسباتی‌ترین بخش‌های آن هستند، بهینه کنند. الگوریتم‌های توان بهینه برای پیاده‌سازی RSA با کارایی بالا حیاتی هستند.

سوال ۵

مقایسه سطح امنیتی اندازه های کلید RSA اغلب با اندازه های الگوریتم های کلید متقارن برای رمزهای بلوکی(symmetric key sizes for block : ciphers)

- یک کلید RSA ۱۰۲۴ بیتی از نظر امنیت تقریباً معادل یک کلید متقارن ۸۰ بیتی نظیر AES است.
- یک کلید RSA ۲۰۴۸ بیتی از نظر امنیت تقریباً معادل یک کلید متقارن ۱۱۲ بیتی نظیر AES است.

این مقایسه‌ها ایده‌ای از چگونگی مقایسه تلاش محاسباتی مورد نیاز برای شکستن رمزگذاری RSA با تلاش‌های لازم برای شکستن الگوریتم‌های رمزگذاری متقارن مانند AES ارائه می‌کنند. با این حال، توجه به این نکته مهم است که اینها تقریب های تقریبی هستند. مفروضات پیچیدگی محاسباتی و مدل‌های حمله برای سیستم‌های رمزنگاری کلید عمومی مانند RSA کاملاً متفاوت از موارد مربوط به رمزهای بلوکی است. اما این تخمین یک حس کلی از سطوح امنیتی هدفمند برای اندازه‌های مختلف کلید RSA در مقایسه با رمزهای بلوکی به دست می‌دهد.

سوال ۶

تولید اعداد اول مناسب برای RSA شامل چندین مرحله برای اطمینان از ایمن بودن و مناسب بودن اعداد برای استفاده رمزنگاری است:

۱. انتخاب تصادفی نامزدها

- طول بیت: طول بیت اول را انتخاب میکنیم (به عنوان مثال، ۱۰۲۴ بیت برای کلیدهای ۲۰۴۸ بیتی RSA).
- Random Number : Generation یک عدد تصادفی از طول بیت مورد نظر را ایجاد کنید. مطمئن شوید که فرد است (زیرا اعداد زوج > ۲ اول نیستند).

۲. تست اولیه

- تست های پایه: بررسی های اولیه مانند تقسیم پذیری بر اعداد اول کوچک را انجام دهید تا سریعاً اعداد غیر اول را رد کنیم.
- تست های اولیه احتمالی: از آزمون هایی مانند آزمون Miller-Rabin یا آزمون Baillie-PSW برای تعیین اینکه آیا عددی با احتمال زیاد اول است یا خیر، استفاده میکنیم. چندین دور را تکرار میکنیم تا شانس مثبت کاذب را کاهش دهیم.

۳. تضمین امنیت رمزنگاری

- راندهای کافی(Sufficient : Rounds) برای Miller-Rabin، از دورهای کافی (مثلاً ۴۰ برای امنیت بالا) استفاده میکنیم تا به سطح اطمینان مطلوبی دست یابید.

- اجتناب از عوامل کوچک(Small Avoiding : Factors) اطمینان حاصل میکنیم که اعداد اول خیلی به توان های اعداد اول کوچک نزدیک نیست تا از حملات رمزنگاری خاص جلوگیری شود.

۴. تایید و اعتبار سنجی

- منحصر به فرد و به اندازه کافی بزرگ: مطمئن میشویم که اعداد اول p و q متمایز و به اندازه کافی بزرگ هستند تا حاشیه امنیتی لازم را فراهم کنند.
- معیارهای اضافی: به صورت اختیاری، ویژگی های اضافی مانند p-۱ یا q-۱ را که دارای فاکتورهای اصلی بزرگ برای افزایش امنیت هستند، بررسی میکنیم.

مثالی از این فرایند:

۱. ایجاد نامزد:

- یک عدد فرد تصادفی ۱۰۲۴ بیتی ایجاد میکنیم.

۲. بررسی اولیه:

- بررسی میکنیم که آیا عدد بر هر عدد اول کوچک بخش پذیر است (مثلاً تا ۱۰۰۰).

۳. تست اولیه:

- آزمون Miller-Rabin را برای ۴۰ تکرار اعمال میکنیم.

۴. تکرار:

- اگر عدد تمام تست ها را پشت سر بگذارد، احتمالاً اول است. در غیر این صورت، یک نامزد جدید ایجاد میکنیم و تکرار میکنیم.

تضمین امنیت تولید اعداد اول:

- از مولدهای اعداد تصادفی امن رمزنگاری شده (CSPRNG) استفاده کنیم.
- اعتبار اجرا را بر اساس استانداردهای شناخته شده (به عنوان مثال، FIPS ۱۸۶-۴).

کتابخانه ها و ابزارها:

- کتابخانه های رمزنگاری مانند OpenSSL و GNU MP (GMP) توابع داخلی را برای تولید و آزمایش اعداد اول بزرگ ارائه می کنند و از تولید اعداد اول قابل اعتماد و کارآمد برای RSA اطمینان حاصل می کنند.

همچنین باید در نظر داشت که در حالت کلی الگوریتم های تولید اعداد اول به دو دسته تقسیم می شوند: الگوریتم های احتمالی و قطعی الگوریتم های احتمالی تولید اعداد اول احتمال زیادی برای تولید یک عدد اول در یک محدوده مشخص ارائه می کنند، اما ممکن است گاهی اوقات اعداد ترکیبی تولید کنند. این الگوریتم ها اغلب برای آزمایش اولیه یا زمانی که سرعت بر قطعیت مطلق اولویت دارد استفاده میشود. الگوریتم های رایج احتمالی تولید اعداد اول عبارتند از:

- Miller-Rabin

- Lucas Probabilistic
- ...

الگوریتم های تولید اعداد اول قطعی تضمین می کنند که خروجی یک عدد اول است، اما آنها اغلب کندتر از الگوریتم های احتمالی هستند. این الگوریتم ها معمولا بر ویژگی های ریاضی خاص اعداد اول تکیه می کنند و ممکن است محاسبات پیچیده‌تر و زمان طولانی تری را شامل شوند. الگوریتم های متداول تولید اعداد اول قطعی عبارتند از:

- algorithm rho Pollard's
- Eratosthenes of Sieve