«باسمه تعالی»

«پوریا دادصفا - ۴۰۱۲۰۱۳۸۱»

«تمرین سری ششم»

رمزنگاری پیشرفته -

۱) الف) مقاله چند جمله‌ای روشی است برای آنکه از خطی بودن رابطه LFSR استفاده کنیم و حمله‌ی ساده همبستگی پایه‌ای، که معمولاً رابطه بین دنباله خروجی و یکی از LFSR ها و متغیرها را مقایسه می‌کرد و سعی به ساختن دنباله‌ی درست داشت، را بهبود بخشیم چراکه آن روش تنها در حالات ε بزرگ و داشتن طول کافی بزرگ قابل اعمال است و از هیچ خاصیت LFSR استفاده نمی‌کند.

می‌دانیم برای یک LFSR با تابع مشخصه معلوم $f(x) = \sum c_i x^i = 0$ می‌توانیم روابطی معادلی برای ضرایب a_i متناظر بنویسیم $\sum a_i = 0$. مثلاً اگر $f(x) = x^{100} + x^{37} + 1$ خواهیم داشت $a_{100} + a_{37} + a_0 = 0$

اما نکته این است که از $f(x) = 0$ می‌توان به $f(x)g(x) = 0$ رسید که $g(x)$ یک چندجمله‌ای دلخواه است. مثلاً اگر خود $f(x)$ را انتخاب کنیم $f^2(x) = 0$ می‌شود و به روابط خطی جدیدی برای ضرایب a_i می‌رسیم.

این عمل به ما این امکان را می‌دهد که با داشتن همان طول قبلی از دنباله خروجی LFSR، معادلات خطی بسیار بیشتری را بنویسیم و در واقع معادل گسترش طول دنباله عادی عمل می‌کند که باعث می‌شود در حمله‌ی همبستگی به جای این که معمولاً دنباله‌ی عادی خروجی و LFSR را با احتمال P مقایسه کنیم، اکنون می‌توان دنباله‌ی خروجی LFSR را با احتمال P ساخت و تمام روابط خطی را چک کرد و در صورت برقرار بودن به سمت دنباله اولیه پیش می‌رود.

این کار دست ما را برای انتخاب $g(x)$ نیز باز می‌گذارد که به روابطی برسیم که وزن تابع را در حد خوبی نگه دارد و با گسترش خوبی از طول به ما بدهد.

ب) در این حمله مشابه روش حمله همبستگی جبران، با استفاده از گسترش طول دنباله و ایجاد روابط خطی میان بیت‌ها

تابع $f(m), g(n)$ بین خروجی LFSR ای که $P(a=z)>0.5$ باشد حمله را انجام می‌دهیم و

از آن جایی که احتمال همبستگی این دو دنباله بیش از 0.5 است، باید بیش از 50% روابط خطی به دست آمده

روی دنباله اصلی LFSR، روی دنباله خروجی z نیز صدق کند. بدین ترتیب اگر به خروجی $\{z\}$ به عنوان حالت نویزی

نگاه کنیم که با بردار نویز $\{e\}$ جمع شده، می‌خواهیم یک عمل تصحیح خطا روی آن انجام دهیم که به دنباله $\{a\}$

اصلی برسیم. پارامترهایی که در این روند استفاده می‌کنیم ترشولد α (که بیانگر میزان صدق کردن از کل روابط خطی موجود) و

β که حداکثر تعداد دورهایی که می‌خواهیم الگوریتم را تکرار کنیم که بهبود یابد می‌باشد. اکنون با دانستن کلیات این حمله

(طول دنباله خروجی)

الگوریتم آن را می‌بینیم:

گام اول) تمام روابط خطی $\{L_i\}$ را برای هر بیت دنباله $\{z_i\}_{i=0}^{N-1}$ می‌نویسیم. اگر اکثریت ($\geq \alpha\%$)

درست بودند، بیت متناظر a_i با z_i برابر است $a_i = z_i$ (روابط i ام). در غیر این صورت $a_i \neq z_i$ و بیت z_i در خروجی را

با $\bar{z_i}$ جایگزین می‌کنیم و به مرحله بعدی می‌رویم.

گام دوم) با تغییرات صورت گرفته به گام اول برمی‌گردیم؛ اگر تعداد روابط درست بالای α بود حرفی نداریم بود

و یا به حداکثر β رسیده بودیم دیگر تکرار نمی‌کنیم و به گام سوم می‌رویم.

گام سوم) دنباله $\{a_i\}$ همان $\{z_i\}$ به دست آمده می‌باشد. البته طبیعتاً یک احتمال درستی دارد که

به تعداد روابط بستگی دارد و اگر بیشتر باشد، احتمال درستی این ساختار بیشتر است.

$\Leftarrow$ اکنون با داشتن $\{a_i\}_{0}^{N-1}$ می‌توانیم شرایط اولیه و LFSR و رابطه را استخراج کنیم

و تابع مشخصه آن.

ج) ابتدا احتمال همبستگی خروجی به هر یک از lfsr ها را می‌سنجیم:

$$f(z_1, z_2, z_3) = z_1z_2 \oplus z_2z_3 \oplus z_1z_3$$

همان‌گونه که می‌بینیم رابطه f تابع اکثریت می‌باشد بنابراین داریم:

z_1	z_2	z_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$$\Rightarrow Pr\{f = z_1\} = Pr\{f = z_2\} = Pr\{f = z_3\} = \frac{2}{3}.$$

* برای حمله به این Z_sequence داده شده، کد پایتون حمله همبستگی پایه را نوشته و اجرا می‌کنیم [که در کنار پاسخ ضمیمه شده است.] و نتایج زیر را برای شرط‌های اولیه ۳ lfsr به دست می‌دهد:

Initial Sequence 1 : [1,0,1,0,0] — $p^* = 0.539$

Initial Sequence 2 : [0,1,1,1,1,1,1,0] — $p^* = 0.58$

Initial Sequence 3 : [0,1,1,1,1,1,1,1,1] — $p^* = 0.5859$

د) تابع مولد خروجی چندجمله‌ای شبیه آن اثر مستقیمی روی احتمال موفقیت حمله و پیچیدگی آن دارد؛ به طور مثال هرچه درجه تابع کمتر باشد حمله همبستگی را می‌توان اجرا کرد چرا که لیست‌های طول بیشتری با دنباله خروجی یکسان طول می‌توان اجرا کرد و به روابط خطی بیشتری رسید. همچنین مدل تابع نیز اگر کمتر باشد یافتن رابطه بین ضرایب و روابط خطی ساده‌تر شده و در واقع قابلیت تصحیح خطا بهتر صورت می‌گیرد.

۲)

الف) این مولد از ۲ ثبات R_1 ، R_2 تشکیل می‌شود که ثبات R خروجی ثبات R_2 را کنترل می‌کند. بدین صورت که هر دو R_1 ، R_2 انتقال می‌یابند و درصورتی که $R_1=1$ باشد، خروجی R_2 یک بیت از دنباله کلید اجرایی خواهد بود در غیر این صورت از این خروجی R_2 صرف‌نظر می‌شود.

ب) برای حمله باید شرایط اولیه دو LFSR، SRA و SRS را بدست آوریم

$(a_0, \dots, a_{A-1})$
$(s_0, \dots, s_{S-1})$

حمله‌ای برای این کار در مقاله New attack strategy for the Shrinking Generator معرفی شده که از دو فاز اصلی استفاده می‌کند، قدم اول شرایط اولیه SRA را استخراج کرده و در قدم بعد براساس آن، شرایط اولیه متناظر SRS استخراج می‌شود.

در قدم اول براساس A بیت ابتدایی دنباله خروجی تعداد ستون اول ماتریس تبدیل IC استخراج کرده سپس n, i, e یعنی اعضای متناظر برابر با $a_0 \dots a_{A-1}$ به دست آورد (اثبات در مقاله)

برای SRS نیز یک Submatrix SxA از IC را استخراج کرده و با ستون‌های باقی‌مانده شرایط اولیه را استخراج می‌نماییم.

③ اثبات ویژگی تمامیت: اگر P، V به درستی عمل کنند در نهایت اثبات کننده مقدار $z = we + r$

را برای وارسی کننده ارسال می‌کند و با محاسبه g^z داریم: $g^z = g^{we+r} = (g^w)^e \times g^r = h^e \times a$

و وارسی کننده با محاسبه و ضرب کردن دو مقدار g^z و $a h^e$ به درستی آن پی می‌برد در صورت درستی، احتمال تقلب اثبات کننده $\frac{1}{q} = \frac{1}{|G|}$ است و با احتمال $1 - \frac{1}{q}$ صادق است.

ویژگی صحت: باید نشان داد احتمال موفقیت اثبات کننده‌ای که w را نمی‌داند ناچیز است.

در ابتدا اثبات کننده r را انتخاب و $g^r = a$ را ارسال می‌کند. در مرحله بعد با دریافت e، مقدار $z = we + r$ را دستکاری و نهایتاً وارسی کننده درستی $g^z = a.h^e$ را بررسی می‌کند. حال اگر اثبات کننده یک z دلخواه را انتخاب کند و $a = \frac{g^z}{h^{e'}}$ را ارسال کند، در صورت خوش شانسی که وارسی کننده $e' = e$ را انتخاب کند که با احتمال $\frac{1}{q}$ اتفاق می‌افتد، موفق می‌شود.

$$g^z = a \times h^{e'} \overset{e'=e}{=} a \times h^e \quad Pr = \frac{1}{q}$$

پس با n بار اجرای این کار احتمال موفقیت او $\frac{1}{q^n}$ می‌باشد.

ناگزاوایی: باید یک شبیه‌ساز بتواند از پروتکل و متن، وارسی کننده را بسازد و transcripts واقعی و شبیه سازی قابل تشخیص نباشد. اکنون روند شبیه ساز را بیان می‌کنیم.

* ابتدا e و z دلخواه تصادفی انتخاب شود. * $\frac{g^z}{h^e} = a$ حساب شود.

* معادله را به این صورت پیاده کند

P^* $\xrightarrow{\alpha}$ V

$\xleftarrow{e}$

$\xrightarrow{z}$ $\Rightarrow g^z = a.h^e$ مشابه

* با توجه به وجود این شبیه سازی و معادله واقعی قابل تشخیص نیستند، اما وارسی کننده باید این باشد که e تصادفی انتخاب کند تا موفقیت باشد.

ب) یک وارسی‌کننده غیرامین می‌تواند e را غیرتکرارمت و adaptive انتخاب کند تا به اطلاعات مورد نیاز برای

ما برسد. مثلاً می‌توانیم هر ZK نیاز به یک شبیه‌ساز دارد و ۲ بار Schnorr شبیه‌سازی در حالت

malicious verifier نداریم:

هر شبیه‌ساز، بعد از ارسال e_1 و دریافت $z_1 = we_1 + r$ می‌تواند اثبات کند که در حالت ۱ و با همان r

بازگرداند تا با $z_2 = we_2 + r$ داریم $\frac{z_2 - z_1}{e_2 - e_1} = w$ پس در این حالت نشت اطلاعات داریم

⇐ Schnorr خاصیت witness hiding را رعایت می‌کند ZK دارد و چون شبیه‌ساز در حالت

malicious verifier کار می‌کند، HVZK است.

برای عیب قبلی می‌توان e را از گروه کوچک $\{0,1\}$ یا تقسیم داده در سایز mod گروه $\{0, \ldots, \frac{q}{2}\}$

استفاده کنیم تا ZK شود.

--- --- ---

۲- روند پروتکل برای صحت باشد:

$r \in_R Z_q$

$a = g^r, b = s^r \xrightarrow{a,b}$

$e \in_R Z_q$

$\xleftarrow{e}$

$z = we + r$

$\xrightarrow{z} \{g^z = h^e \times a\} / \{s^z = t^e \times b\}$

اکنون ۳ ویژگی را برای آن بررسی می‌کنیم. ← صفحه بعد

* ویژگی تمامیت :

$g^{z} = g^{we+r} = h^{e} \times a$, $s^{z} = s^{we+r} = t^{e} \times b$

بنابراین پس از محاسبه g^z برابر $h^e \times a$ بوده و s^z برابر $t^e \times b$ بود، درست بود (والبته با احتمال کمی اثبات کننده تقلب کرده است)

* صحت : مانند قسمت الف سوال قبل، برای P^* که w را نمی‌داند، e' را انتخاب می‌کند و چالش (Prover) را حدس می‌زند. $a = \frac{g^z}{h^{e'}}$ و $b = \frac{s^z}{t^{e'}}$ را ارسال می‌کند. در صورت انتخاب e' توسط اثبات کننده موفق می‌شود

احتمال این اتفاق اگر فضای e، D باشد برابر $\frac{1}{D}$ می‌شود که اینجا برابر $\frac{1}{q}$ می‌شود.

* ناتراوایی : شبیه‌ساز این روند را طی می‌کند : * e, z دلخواه را انتخاب کند * $a = \frac{g^z}{h^e}$, $b = \frac{s^z}{t^e}$ محاسبه می‌شود.

* این مکالمه انجام می‌شود :

$P \xrightarrow{a,b} V$

$\xleftarrow{e}$

$\xrightarrow{z}$

و تمایزی بین مکالمه واقعی ندارد و با دانش w هم همچنان وارسی g^z و s^z

نیازی به وارسی کننده امین ندارد

--- --- ---

۵ـ رمز جانشینی تک الفبایی :

۱ـ الگوریتم تولید کلید : یک جایگشت تصادفی از الفبای زبان (انگلیسی) انتخاب می‌کنیم. به عبارتی یک جایگشت (مثلاً)
(Gen(A))

دلخواه از مجموعه $\{0, 1, ..., 25\}$ را از A می‌سازیم و K می‌نامیم.

$i \in [0,25]$ ، K(i) را برابر مقداری تصادفی از مجموعه A که از قبل هم برابر هم‌چنین قبلاً برای عنصری ثابت نشده
مقادیر قبلی انتخاب نشده باشد

۲_ الگوریتم رمزکردن: به عنوان ورودی رشته‌ای به طول L مثل $M = m_1 m_2 \cdots m_L$ که m_i در الفبا هستند را دارد ($Enc(M, K)$)

و کلید K: $C_i = K(m_i)$, $C = c_1 c_2 \cdots c_L$.
$\forall\ 1 \leq i \leq L$

۳_ الگوریتم رمزگشایی: ورودی رشته به طول L $c_1 c_2 \cdots c_L$ که c_i در الفبا است و کلید جایگشت K.
$Dec(C, K)$

قرار می‌دهیم $m_i = K^{-1}(c_i)$ و $M = m_1 m_2 \cdots m_L$. که در آن K^{-1} جایگشت وارون K است.
$\forall\ 1 \leq i \leq L$

ب) رمز Vigenere:

۱_ الگوریتم تولید کلید: باید مجموعه‌ای از جایگشت‌ها را بسازیم تولید شود. بنابراین یک عدد تصادفی t به عنوان (Gen)
سوره شده انتخاب می‌کنیم، و t عدد تصادفی در الفبا (مثلاً بین ۰ تا 25 برای انگلیسی) انتخاب می‌کنیم.
$K_i \in_R \{0, \cdots, 25\}$ $\quad 0 \leq i \leq t-1$

کلید برابر $K = k_0, \cdots, k_{t-1}$ می‌باشد.

۲_ الگوریتم رمزکردن: ورودی رشته پیام طول L، $m_1 \cdots m_L$ و $K = k_0 \cdots k_{t-1}$ گرفته و
$Enc(M, K)$
$$c_i = (m_i + K_{i \bmod t}) \bmod 26 \quad 1 \leq i \leq L$$

را تولید و $C = c_1 \cdots c_L$ می‌باشد.

۳_ الگوریتم رمزگشایی: ورودی رشته طول L $c_1 \cdots c_L$، $K = k_0 \cdots k_{t-1}$:
$Dec(C, K)$
$$m_i = (c_i - K_{i \bmod t}) \bmod 26 \quad 1 \leq i \leq L$$

و $M = m_1 \cdots m_L$.

Scanned by CamScanner

(۶)

سزار: تنها با داشتن یک متن رمز و پیام اصلی شکسته می‌شود ، $k = c - p \mod[26]$

\# حروف الفبا

$\Rightarrow$ $N_0 = 1$

جانشینی: با یک متن اصلی شامل همه حروف الفبا نیاز داریم تا جانشینی کامل کلید را بفهمیم .

$k(P_i) = C_i$

Vignerer: اگر دوره تناوب (t) را بدانیم ، یک متن رمز شده به طول t و متن اصلی نظیر لازم است

$P = P_0 \cdots P_{t-1}$

$C = C_0 \cdots C_{t-1}$

$K_i = C_i - P_i \mod[26]$ (# الفبا), $0 \le i \le t-1$ $\Rightarrow K = K_0 \cdots K_{t-1}$

درصورت نداشتن t ولی دانستن ماکسیمم مقدار ممکن t ، (t_{max}) نیز می‌توان با پیام به طول t_{MAX} شکست

با این حساب که دوره تناوب را $t = 1$ قرار داده ، کلید متن را تست کنیم درصورت عدم صحت ، ۱ واحد

زیاد کنیم و دوباره تست کنیم ؛ تا جایی که بازهمه P و C درست کار کند .