
بسم الله الرحمن الرحيم

پیک تلوریم

محمد مهدی اقدسی ۴۰۰۵۲۱۰۸۱

علی شکوهی ۴۰۰۵۲۱۴۷۷

دانیال یگانه ۴۰۰۵۲۲۳۳۷

آخرین ویرایش: ۱۰ آذر ۱۴۰۳ در ساعت ۲۰ و ۱۳ دقیقه

فهرست مطالب

۱ سوالات تئوری رمزهای متقارن

۱.۱ سوال اول (فضای کلید)

فرض می‌کنیم که رمزهای استفاده شده برای یک وب‌سایت شامل ۸ کاراکتر با فرمت UTF-8 هستند. برای محاسبه فضای کلید و طول کلید بر حسب بیت، مراحل زیر را دنبال می‌کنیم:

- **تعداد کاراکترها در UTF-8:** هر کاراکتر UTF-8 بین ۱ تا ۴ بایت فضا می‌گیرد. کاراکترهای UTF-8 چهار حالت زیر را دارند:

0xxxxxxx

110xxxxx 10xxxxxx

1110xxxx 10xxxxxx 10xxxxxx

11110xxx 10xxxxxx 10xxxxxx 10xxxxxx

پس سقف تعداد کاراکترهای ممکن (بدون در نظر گرفتن قوانین مخصوص) برابر است با:

$$2^{21} + 2^{16} + 2^{11} + 2^{7} = 2,164,864 \tag{۱.۱}$$

- **فضای کلید:** فضای کلید تعداد کل ترکیب‌های ممکن برای کلید را نشان می‌دهد. با داشتن ۲،۱۶۴،۸۶۴ حالت، تعداد کل ترکیب‌ها به صورت زیر خواهد بود:

$$(2,164,864)^{8} \simeq 4.8 \times 10^{50} \tag{۲.۱}$$

- **طول کلید به بیت:** با توجه به فضای کلید، طول کلید برابر است با:

$$\log_2(4.8 \times 10^{50}) \simeq 168.37 \text{ بیت} \tag{۳.۱}$$

بنابراین:

- **فضای کلید:** 4.8×10^{50}
- **طول کلید:** 168.37 بیت

۲.۱ سوال دوم (اعداد شبه تصادفی)

تولید اعداد شبه‌تصادفی در علوم کامپیوتر یکی از زمینه‌های بسیار جالب است که تحقیقات زیادی از زمان‌های قدیم تا امروز در آن انجام شده است. در اینجا دو روش مشهور برای تولید این اعداد آورده شده است:

۱. ژنراتور خطی همگن (LCG)

توضیح

ژنراتور خطی همگن یکی از قدیمی‌ترین و ساده‌ترین روش‌ها برای تولید اعداد شبه‌تصادفی است. این روش بر اساس رابطه بازگشتی زیر عمل می‌کند:

$$X_{n+1} = (aX_n + c) \mod m$$

که در آن:

- X_n عدد تصادفی جاری است.
- a، c و m مقادیر ثابت هستند که به ترتیب ضرب‌کننده، افزونه و مدولوس هستند.
- X_{n+1} عدد تصادفی بعدی تولید شده است.

این ژنراتور با شروع از یک مقدار اولیه به نام **بذر**، دنباله‌ای از اعداد تولید می‌کند که به نظر تصادفی می‌آیند، اما کاملاً تعیین‌شده هستند.

مزایا

- **سادگی و سرعت:** LCG به راحتی پیاده‌سازی می‌شود و می‌تواند بسیار سریع باشد، به همین دلیل برای برنامه‌های ساده‌ای مانند شبیه‌سازی‌ها مناسب است.
- **شناخت خوب:** از آنجا که این روش یک روش قدیمی است، ویژگی‌ها، مزایا و معایب آن به خوبی مستند شده است.

معایب

- **رندم بودن ضعیف:** دنباله اعداد تولید شده توسط LCG می‌تواند پیش‌بینی‌پذیر باشد، به ویژه اگر پارامترهای a، c و m به درستی انتخاب نشوند. این عیب برای کاربردهای رمزنگاری مشکل‌ساز است.
- **دوره کوتاه:** اگر مدولوس m به اندازه کافی بزرگ نباشد، دنباله پس از تعداد نسبتاً کمی گام‌ها تکرار می‌شود که محدودیت‌هایی را ایجاد می‌کند.
- **همبستگی:** در برخی موارد، ممکن است همبستگی‌های قابل توجهی بین اعداد در دنباله وجود داشته باشد که منجر به ایجاد الگوهایی می‌شود که می‌توانند در آزمون‌های آماری شناسایی شوند.

منابع

- "Numerical Recipes: The Art of Scientific Computing" by William H. Press et al. (Cambridge University Press, 2007).
- "The Art of Computer Programming, Volume 2: Seminumerical Algorithms" by Donald E. Knuth (Addison-Wesley, 1997).

۲. مریسن توییستر (MT19937)

توضیح

مرسن توییستر یک ژنراتور شبه‌تصادفی است که به دلیل تولید اعداد تصادفی با کیفیت بالا بسیار شناخته شده است. این روش در سال ۱۹۹۷ توسط ماکوتو ماتسوموتو و تاکوجی نیشیمورا توسعه یافت. نام "مرسن" به این دلیل است که دوره این ژنراتور یک عدد اول مرسن است، به طور خاص $2^{19937} - 1$.

این ژنراتور از شیفت رجیستر بازخورد پیچیده (TGFSR) استفاده می‌کند که بر روی کلمات ۳۲ بیتی عملیات می‌کند. ژنراتور با انجام یک سری عملیات، شامل شیفت بیت و عملیات XOR، روی بردار وضعیت، دنباله‌ای از اعداد شبه‌تصادفی تولید می‌کند.

نکته: در ریاضیات، عدد اول مرسن (Mersenne Prime) یک عدد اول است که به فرم $n^2 - 1$ قابل نمایش است. در اینجا n یک عدد طبیعی است. برای مثال به ازای $n = 3$، خواهیم داشت $2^3 - 1 = 7$ بنابراین ۷ یک عدد اول مرسن است.

متداول‌ترین نسخه الگوریتم Mersenne Twister براساس عدد اول مرسن $2^{19937} - 1$ ساخته شده است. در نسخه استاندارد آن، یعنی MT19937، از کلمه با طول ۳۲ بیت استفاده می‌شود. پیاده سازی دیگری از این الگوریتم با نام MT19937-64 نیز وجود دارد که از کلمه با طول ۶۴ بیت استفاده می‌کند. این الگوریتم توالی متفاوتی نسبت به الگوریتم ۳۲ بیتی ایجاد می‌کند. عبارت MT مخفف Mersenne Twister است.

تولید اعداد تصادفی با Mersenne Twister براساس یک کلمه w بیتی، یک عدد تصادفی از مجموعه اعداد صحیح در بازه $[0, w^2 - 1]$ تولید می‌کند. تولید اعداد تصادفی با Mersenne Twister مبتنی بر یک ماتریس با رابطه بازگشتی خطی (Matrix Linear Recurrence) روی یک میدان دودویی F2 متناهی (finite binary field F2) است.

این الگوریتم یک ثبات جابجایی بازخورد تعمیم یافته پیچیشی (Twisted Generalized Feedback Shift Register) یا به اختصار TGFSR است که با نمایش بیت حالت (State bit) و فرم منطقی نرمال (Rational Normal Form) ماتریس یا فرم فربنیوس (Frobenius normal form) عمل می‌کند. در این حالت ثبات را به اختصار به شکل TGFSR(R) نشان می‌دهند.

نکته: در فرم فربنیوس، یک فضای برداری به زیرفضاهایی چرخشی از ماتریس A تجزیه و تفکیک می‌شود.

ایده اصلی این است که یک سری یا دنباله از x_i را به وسیله یک رابطه تکراری ساده تعریف کرده و سپس اعدادی به شکل x_iT را استخراج کرد. البته مشخص است که T یک ماتریس F2 معکوس‌پذیر بوده که معمولا از آن به عنوان ماتریس تعدیل (Tempering Matrix) نام می‌برند.

الگوریتم عمومی برای تولید اعداد تصادفی با Mersenne Twister با مقادیر زیر مشخص می‌شود:

- w: اندازه کلمه (به تعداد بیت).
- n: درجه بازگشت.
- m: کلمه میانی، پارامتری که در رابطه بازگشتی به کار رفته و برای تعریف دنباله Xها لازم است. $(m \geq 1)$
- r: نقطه جداسازی یک کلمه یا تعداد bitmask پایینی. $(0 \leq r \leq w-1)$
- a: ضرایب ماتریس پیچشی منطقی نرمال.
- b، c: تعدیل کننده (bitmask).
- s، t: تعدیل کننده بیت جابجایی (bit shifts).
- u، d، l: بیت‌های جابجایی و ماسک‌های تعدیل کننده Mersenne Twister.

اول از همه به یاد داشته باشید که $2^{nw-r}-1$ یک عدد اول مرسن است. این انتخاب، آزمون ابتدایی و آزمون توزیع k را که برای جستجوی پارامتر مورد نیاز لازم است، ساده می‌کند.

دنباله Xها به عنوان مجموعه‌ای از مقادیر w بیتی با رابطه بازگشتی زیر تعریف می‌شود:

$$x_{k+n} := x_{k+m} \oplus \left(({x_k}^u \mid\mid {x_{k+1}}^l) A \right) \qquad k = 0, 1, \ldots$$

که در آن || نشانگر الحاق بردارهای بیتی (با بیت‌های بالا در سمت چپ)، $\oplus$ عملگر (XOR)، ${x_k}^u$ به معنای $w-r$ بیت بالایی از x_k و ${x_{k+1}}^l$ نیز r بیت پایین برای x_{k+1} است.

در این حالت تبدیل پیچشی ماتریس A به فرم منطقی نرمال به صورت زیر خواهد بود.

$$A = \begin{pmatrix} 0 & I_{w-1} \\ a_{w-1} & (a_{w-2}, \ldots, a_0) \end{pmatrix}$$

نکته: به یاد داشته باشید که در اینجا ضرب ماتریسی در جبر F2 روی می‌دهد و در نتیجه عملگر بیتی XOR به عنوان جمع به کار می‌رود.

مانند A، تبدیل تعدیل‌گر را به شکلی انتخاب می‌کنیم که محاسبات به سادگی قابل انجام باشند.

در الگوریتم اولیه یا همان MT19937، ضرایب یا پارامترها به صورت زیر هستند:

- $(w, n, m, r) = (32, 624, 397, 31)$
- $a = 9908B0DF16$
- $(u, d) = (11, FFFFFFFF16)$
- $(s, b) = (7, 9D2C568016)$

- $(t, c) = (15, EFC6000016)$
- $l = 18$

مزایا

- **دوره طولانی:** رسن توییستر دوره‌ای بسیار طولانی به اندازه $2^{19937} - 1$ دارد، به این معنی که دنباله برای مدت زمان بسیار طولانی تکرار نخواهد شد و برای برنامه‌هایی که نیاز به تولید تعداد زیادی عدد تصادفی دارند مناسب است.
- **ویژگی‌های آماری خوب:** این ژنراتور آزمون‌های آماری بسیاری برای تصادفی بودن را پشت سر گذاشته است و در شبیه‌سازی‌ها، رمزنگاری (هنگامی که تنها منبع تصادفی نیست) و بازی‌ها به طور گسترده استفاده می‌شود.
- **سرعت بالا:** این ژنراتور بسیار سریع است، به‌ویژه زمانی که در سخت‌افزار یا با استفاده از کتابخانه‌های بهینه‌شده پیاده‌سازی شود.

معایب

- **امنیت ضعیف در رمزنگاری:** در حالی که مرسن توییستر برای شبیه‌سازی‌ها و استفاده‌های عمومی عالی است، برای کاربردهای رمزنگاری امن نیست. اعداد تولید شده توسط آن تعیین‌شده هستند و اگر وضعیت داخلی ژنراتور مشخص باشد، می‌توان آن‌ها را پیش‌بینی کرد.
- **استفاده از حافظه زیاد:** این ژنراتور به مقدار زیادی حافظه برای ذخیره وضعیت خود نیاز دارد (معمولاً ۶۲۴ عدد ۳۲ بیتی)، که ممکن است برای برنامه‌هایی با محدودیت حافظه بهینه نباشد.
- **شروع کند:** فرآیند شروع به کار می‌تواند کند باشد، به‌ویژه زمانی که با یک وضعیت آماده نشده بذر زده شود.

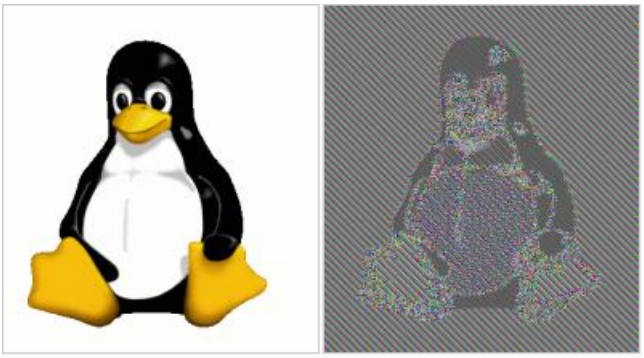
منابع

- Matsumoto, M., & Nishimura, T. (1998). Mersenne Twister: A 623-Dimensional Equidistributed Uniform Pseudo-Random Number Generator. ACM Transactions on Modeling and Computer Simulation, 8(1), 3–30.
- "Numerical Recipes: The Art of Scientific Computing" by William H. Press et al. (Cambridge University Press, 2007).

۳.۱ سوال سوم (مدهای رمزهای قالبی)

الف) به دلایل زیر نباید از مد ECB استفاده کرد:

- نشت الگو: از آنجایی که بلوک‌های متن یکسان به بلوک‌های متن رمز یکسان رمزگذاری می‌شوند، الگوهای موجود در متن اصلی حفظ می‌شوند. این می‌تواند به یک حمله‌کننده اجازه دهد تا اطلاعاتی را در مورد ساختار یا محتوای متن اصلی استنباط کند، به خصوص اگر همان داده‌ها چندین بار رمزگذاری شده باشند.

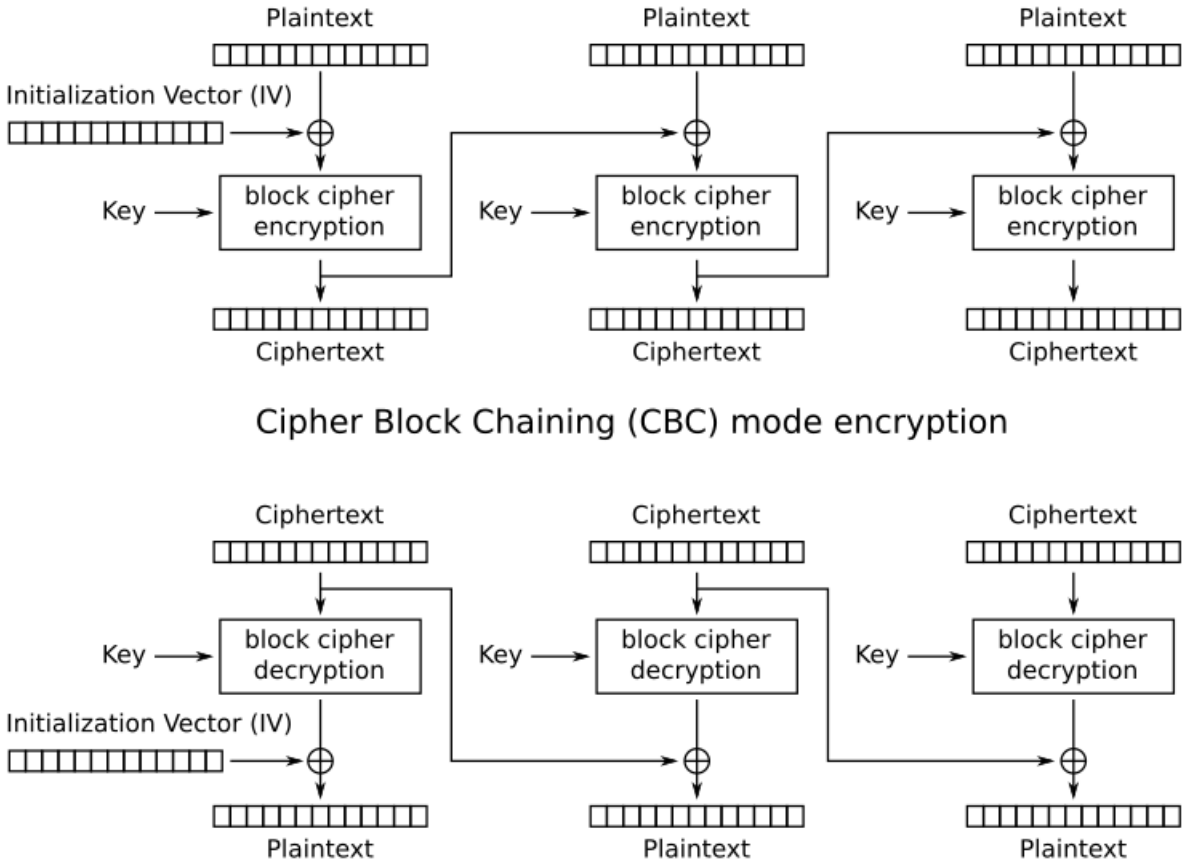


شکل ۱.۱: نشت الگو در ECB

- حفاظت نکردن از یکپارچگی: مد ECB دارای هیچ مکانیسمی برای حفظ یکپارچگی متن رمز شده نمی‌باشد. این بدین معناست که یک مهاجم می‌تواند یک بلوک از متن رمز شده را با بلوک رمز دیگری بدون شناسایی عوض کند، که منجر به دستکاری غیرقابل کشف متن رمزگشایی شده می‌شود.
- ضعف در برابر حملات دستکاری بلوک: یک مهاجم می تواند حملات دستکاری بلوک را آسانتر انجام دهد زیرا مهاجم می‌تواند پیش‌بینی کند که چگونه تغییرات در متن رمزگذاری شده روی متن رمزگشایی شده تأثیر می گذارد.
- ضعف در برابر حملات تکرار: از آنجایی که بلوک‌های متن یکسان، متن‌های رمزی یکسانی تولید می‌کنند مهاجم می‌تواند متن رمز را دستکاری کرده و بلوک‌های متن رمز معتبر را دوباره پخش کند. این می‌تواند منجر به اقدامات غیرمجاز یا بازسازی بخشهایی از متن رمزگشایی شده شود.

ب) سه مورد از مدهای مطرح دیگر به صورت زیر می‌باشند:

- مد Cipher Block Chaining یا به اصطلاح CBC یکی از رایج‌ترین مدها در رمزنگاری‌های بلوکی به حساب می‌آید. نحوه عملکرد این مد بدین صورت است:
  ۱. برای رمز کردن یک متن، اولین بلوک از متن اصلی با یک رشته از بیت‌های تصادفی به نام Initialization Vector یا به اصطلاح XOR IV می‌شود.
  ۲. سپس حاصل بدست آمده با کلید مورد نظر رمز می‌شود تا اولین بلوک متن رمز تولید شود.
  ۳. در قدم بعدی اولین بلوک رمز شده با بلوک متن اصلی بعدی XOR می‌شود و حاصل آن با کلید مورد نظر رمز می‌شود تا بلوک متن رمز بعدی نیز بدست آید.
  ۴. این فرآیند برای بلوک‌های بعدی نیز تکرار می‌شود تا هر بلوک از متن رمز بر روی حاصل رمز هر بلوک از متن اصلی تاثیر بگذارد.
  ۵. برای رمزگشایی نیز برعکس این روند طی می‌شود (شکل ۲.۱).



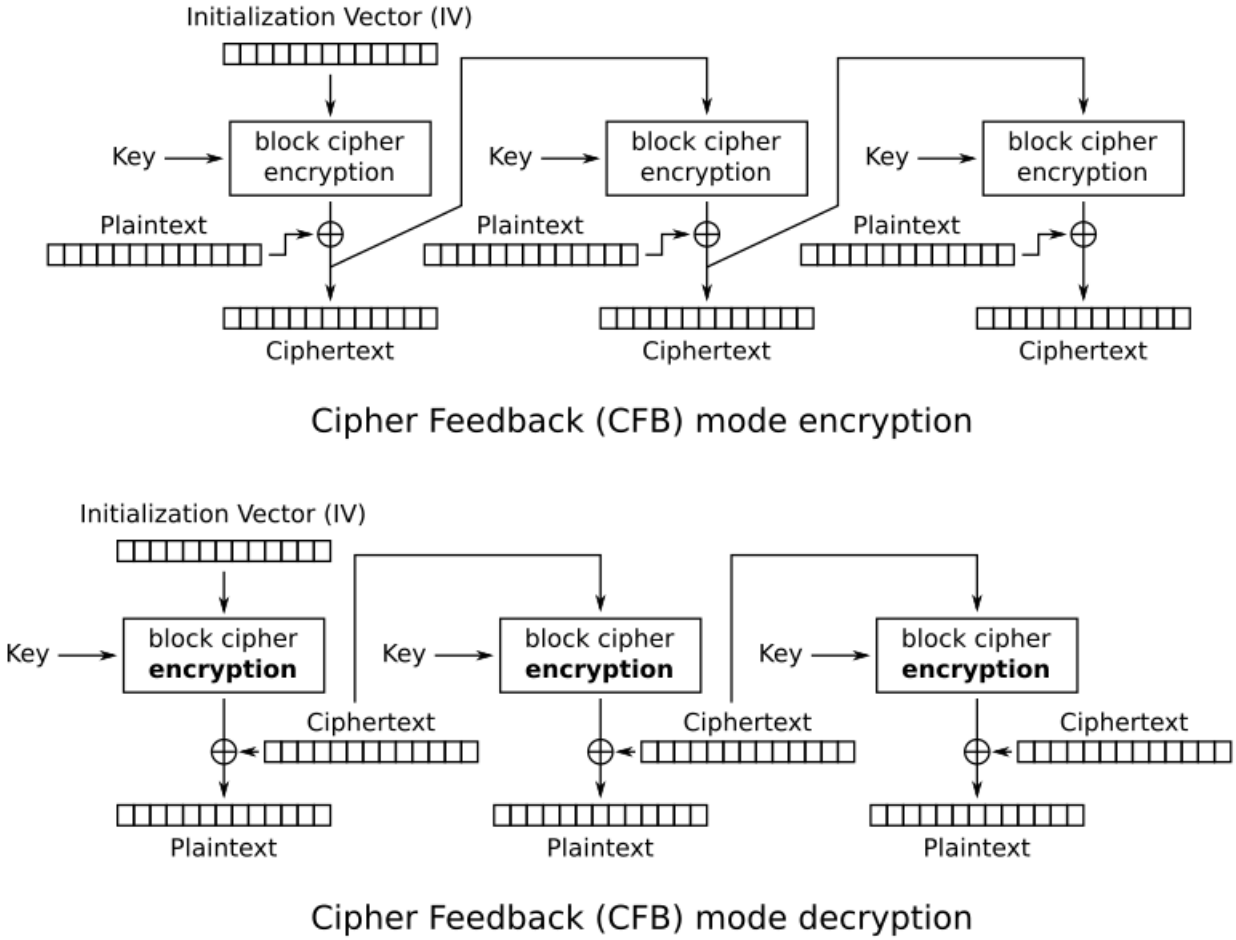
شکل ۲.۱: نحوه عملکرد مد CBC

چند مورد از ویژگی‌های این مد به صورت زیر می‌باشد:

- وابستگی زنجیره‌ای: هر بلوک متن رمز به تمام بلوک‌های متن اصلی قبلی خود وابستگی دارد. این تضمین می‌کند که حتی اگر بلوک‌های متن اصلی یکسانی رمزگذاری شوند، به بلوک‌های متن رمزی متفاوتی منجر خواهند شد. این مورد باعث حفاظت از یکپارچگی متن می‌باشد که آن را در برابر حملات دستکاری بلوک مقاوم می‌کند. همچنین این مورد باعث می‌شود تا عملیات رمزنگاری نتواند به صورت موازی انجام بگیرد و باعث کندتر شدن آن نسبت به مدهای دیگر می‌شود، هرچند که می‌توان عملیات رمزگشایی را به صورت موازی انجام داد.
- بردار اولیه سازی (IV): استفاده از یک IV به ما اطمینان می‌دهد که متن اصلی با همان کلید متن‌های رمزی متفاوتی را تولید می‌کند، که این یک لایه امنیتی اضافی ایجاد می‌کند. این مورد مانع شناسایی الگوهای موجود در داده‌های رمزگذاری شده توسط مهاجمان می شود.
- انتشار محدود خطا: در حالی که یک خطا در یک بلوک به بلوک بعدی منتشر می‌شود، این به بلوک بلافاصله بعد محدود می‌شود و امکان بازیابی بخش زیادی از متن اصلی را فراهم می‌کند.

- مد Cipher FeedBack یا به اصطلاح CFB مدی مشابه مد CBC می‌باشد، که رمزنگاری‌های بلوکی را تبدیل به رمزنگاری‌های جویباری می‌کند. این مد بدین صورت عمل می‌کند:

۱. اول رشته‌ای از بیت‌های تصادفی به نام Initialization Vector توسط کلید رمز می‌شود.
۲. سپس حاصل این رمز با اولین بلوک از متن اصلی XOR می‌شود تا اولین بلوک از متن رمز بدست آید.
۳. در قدم بعدی بلوک رمز تولید شده دوباره رمز می‌شود و حاصل آن با بلوک بعدی از متن اصلی XOR می‌شود.
۴. این فرآیند برای بلوک‌های بعدی نیز تکرار می‌شود تا متن رمز بدست آید.
۵. برای رمزگشایی نیز برعکس این روند طی می‌شود (شکل ۳.۱).



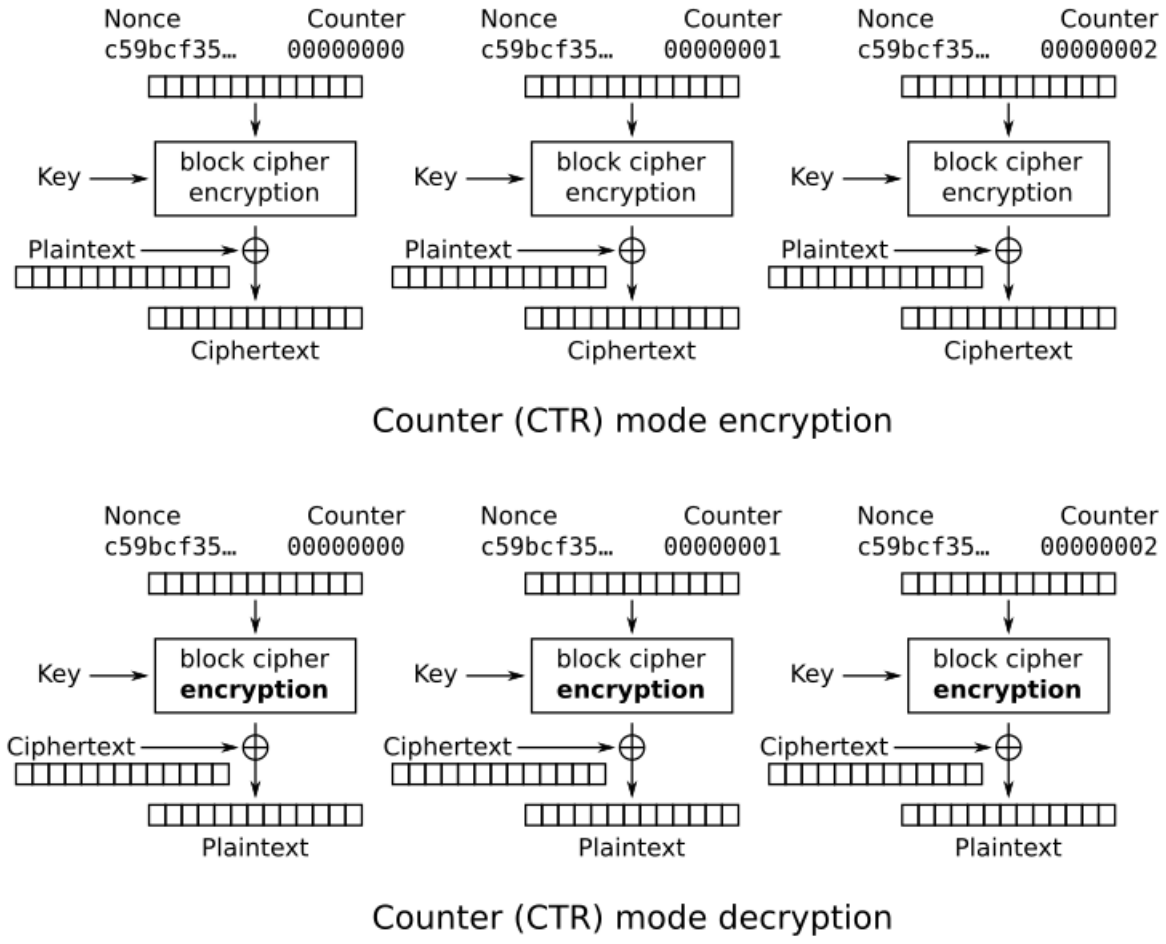
شکل ۳.۱: نحوه عملکرد مد CFB

چند مورد از ویژگی‌های این مد به صورت زیر می‌باشد:

- جوییاری بودن: برخلاف مد CBC الزامی برای یکسان بودن اندازه بلوک الگوریتم رمزنگاری استفاده شده و اندازه بلوک آخر نمی‌باشد (می‌توان خروجی آخرین بلوک رمزنگاری را زودهنگام قطع کرد)، بنابراین نیازی به padding نبوده که این موضوع این مد را تبدیل به یک رمزنگاری جوییاری می‌کند.
- بردار اولیه سازی (IV): همانند مد CBC، مد CFB برای شروع فرآیند رمزگذاری به یک IV نیاز دارد. بردار اولیه سازی تضمین می‌کند که بلوک‌های متن یکسان منجر به بلوک‌های متن رمزی متفاوت می‌شوند و امنیت را افزایش می‌دهند.
- خود همگام سازی: حالت CFB خود همگام است، یعنی اگر بخشی از متن رمزی گم شود یا خراب شود، رمزگشایی همچنان می‌تواند پس از چند بلوک همگام شود و ادامه پیدا کند.

- مد CTR نیز مد محبوب دیگری برای رمزنگاری‌های بلوکی می‌باشد که همانند مد CFB رمزنگاری‌های بلوکی را تبدیل به رمزنگاری‌های جویباری می‌کند. این مد بدین صورت عمل می‌کند:

۱. اول رشته‌ای از بیت‌های تصادفی به نام Initialization Vector یا nonce تولید می‌شود، سپس این رشته با یک شمارنده توسط یک عمل برگشت‌پذیر (مانند XOR کردن یا چسباندن) ترکیب می‌شود و سپس رمز می‌شود.
۲. سپس حاصل این رمز با اولین بلوک از متن اصلی XOR می‌شود تا اولین بلوک از متن رمز بدست آید.
۳. در قدم بعدی، شمارنده یکی زیاد شده و دوباره با nonce ترکیب می‌شود. سپس این ترکیب رمز شده و با بلوک بعدی از متن اصلی XOR می‌شود.
۴. این فرآیند برای بلوک‌های بعدی نیز تکرار می‌شود.
۵. برای رمزگشایی نیز برعکس این روند طی می‌شود (شکل ۴.۱).



شکل ۴.۱: نحوه عملکرد مد CTR

چند مورد از ویژگی‌های این مد به صورت زیر می‌باشد:

- جویباری بودن: همانند مد CFB الزامی برای یکسان بودن اندازه بلوک الگوریتم رمزنگاری استفاده شده و اندازه بلوک آخر نمی‌باشد، بنابراین نیازی به padding نبوده و این موضوع این مد را تبدیل به یک رمزنگاری جویباری می‌کند.
- موازی‌سازی: برخلاف دیگر مدهای اشاره شده، مد CTR قابلیت رمزنگاری به صورت موازی را دارا می‌باشد، یعنی

می‌توان بلوک‌های متن اصلی را به صورت موازی با یکدیگر رمزنگاری کرد. این امر باعث می‌شود تا این مد برای کاربردهای پرسرعت و سیستم‌های چند پردازنده‌ای مناسب باشد.

- دسترسی تصادفی: از آنجا که هر بلوک متن اصلی با یک مقدار شمارنده منحصر به فرد رمزگذاری شده است، هر بلوکی را می‌توان مستقل از بقیه رمزگذاری یا رمزگشایی کرد. این ویژگی امکان دسترسی تصادفی کارآمد به داده‌های رمزگذاری شده را فراهم می کند.
- انتشار خطای محدود: برخلاف برخی مدهای دیگر (مانند CBC)، یک خطا در یک بلوک متن رمزی در طول رمزگشایی به بلوک‌های دیگر منتشر نمی‌شود. این بدان معنی است که یک خطای تک بیتی در متن رمزی تنها منجر به از دست رفتن یک بلوک از متن اصلی می‌شود.

۴.۱ سوال چهارم (رمز جایگشتی)

برای حل این سوال، از الگوریتم رمزنگاری جایگشتی با کلید داده شده استفاده می‌کنیم. ابتدا قسمت آ را حل می‌کنیم و سپس به قسمت ب می‌پردازیم.

آ) رمزنگاری و رمزگشایی یک رشته ۱۶ کاراکتری به انتخاب خود

فرض کنید رشته‌ای با ۱۶ کاراکتر به صورت زیر داریم:

$$R = \text{"ABCDEFGHIJKLMNOP"}$$

و طبق جدول داده شده، تابع $\pi(x)$ به صورت زیر است:

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi(x)$	12	5	4	15	2	6	14	7	8	9	13	16	3	10	11	1

رمزنگاری: برای رمزنگاری رشته R، جایگشت $\pi(x)$ را بر روی آن اعمال می‌کنیم. بنابراین، هر کاراکتر در موقعیت x به موقعیت $\pi(x)$ منتقل می‌شود.

رشته رمز شده $R_{\text{encrypted}}$ به صورت زیر خواهد بود:

$$R_{\text{encrypted}} = \text{"PEMCBFHIJNOAKGDL"}$$

رمزگشایی: برای بازگرداندن رشته $R_{\text{encrypted}}$ به حالت اولیه، به جایگشت معکوس $\pi^{-1}(x)$ نیاز داریم. این جایگشت معکوس، مکان‌های اصلی هر کاراکتر را بازیابی می‌کند.

جدول جایگشت معکوس $\pi^{-1}(x)$ به صورت زیر است:

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\pi^{-1}(x)$	16	5	13	3	2	6	8	9	10	14	15	1	11	7	4	12

با استفاده از $\pi^{-1}(x)$، می‌توانیم $R_{\text{encrypted}}$ را به رشته اصلی R بازگردانیم.

ب) رمزگشایی رشته "Tom Marvolo Riddle"

برای رمزگشایی عبارت "Tom Marvolo Riddle" به کمک جایگشت معکوس $\pi^{-1}(x)$، مراحل زیر را طی می‌کنیم:

رشته رمزگذاری شده $R_{\text{encrypted}}$ به صورت زیر است:

$$R_{\text{encrypted}} = \text{"Tom Marvolo Riddle"}$$

با اعمال جایگشت معکوس $\pi^{-1}(x)$، مکان‌های کاراکترها را به حالت اولیه بازمی‌گردانیم و رشته اصلی را بازیابی می‌کنیم. عبارت داده‌شده، "Tom Marvolo Riddle"، دقیقاً ۱۶ کاراکتر دارد، بنابراین می‌توانیم مستقیماً از جدول جایگشت استفاده کنیم.

ترتیب‌بندی کاراکترها با استفاده از $\pi^{-1}(x)$

با اعمال جایگشت $\pi^{-1}(x)$، هر حرف از موقعیت فعلی خود به مکان جدید منتقل می‌شود:

- مکان T: $1 \to 16$
- مکان o: $2 \to 5$
- مکان m: $3 \to 13$
- مکان M: $4 \to 3$
- مکان a: $5 \to 2$
- مکان r: $6 \to 6$
- مکان v: $7 \to 8$
- مکان o: $8 \to 9$
- مکان l: $9 \to 10$
- مکان o: $10 \to 14$
- مکان R: $11 \to 15$
- مکان i: $12 \to 1$
- مکان d: $13 \to 11$
- مکان d: $14 \to 7$

- مکان l: $4 \rightarrow 15$
- مکان e: $12 \rightarrow 16$

نتیجه نهایی

با انتقال کاراکترها به موقعیت‌های جدیدشان، عبارت "I Am Lord Voldemort" به دست می‌آید.

۵.۱ سوال پنجم(کلید شماره r لو رفت)

آ) از آنجایی که کلید ۲۵ در سمت راست کلید ۰ قرار دارد، بنابراین می‌توانیم به راحتی تمام عناصر زیر کلید ۱ را در نظر بگیریم. با همان منطق (اما با والد متفاوت)، می‌توانیم ۶ و ۱۱ را نیز در نظر بگیریم. در برگ‌های باقی‌مانده تنها چیزی که باید در نظر گرفته شود، ۲۶ است. پس جواب خواهد بود: ۱،۶،۱۱،۲۶

ب) پاسخ $\log_2(n)$ است چون عمق درخت باینری با n گره، این مقدار خواهد بود. علت اینکه عمق درخت پاسخ ماست هم این است که مسیر از ریشه تا گره مورد نظر خراب شده و در این مسیر به ازای هر گره باید گره سمت دیگرش را برای رمز کردن استفاده کنیم.

ج) با ۶ کلید : ۱۵،۱۷،۴،۱۱،۲۶،۶

۶.۱ سوال ششم (قدرت پردازشی)

برای به دست آوردن مدت زمان مورد نیاز برای شکستن یک الگوریتم رمزنگاری مانند AES با کلید ۱۲۸ بیتی، نیاز داریم تا سه مولفه را بدانیم:

- تعداد عملیات‌های مورد نیاز برای بررسی یک کلید
- تعداد عملیات‌های محاسباتی که کامپیوتر ما می‌تواند در یک مدت زمان مشخص انجام دهد
- تعداد کلیدهایی که باید بررسی شود

فرض می‌کنیم که حمله کننده به ۱۰ تا از قوی‌ترین پردازنده‌های موجود در بازار یعنی کارت گرافیک H100 دسترسی دارد. این کارت گرافیک در محک FP64 یا عملیات‌های اعداد اعشاری ۶۴ بیتی قدرتی معادل 60 TFLOPS یا 6×10^{13} عملیات اعشاری در ثانیه را دارا می‌باشد (شکل ۵.۱).

NVVIDIA H100		vs A100
FP8	4,000 TFLOPS	6x
FP16	2,000 TFLOPS	3x
TF32	1,000 TFLOPS	3x
FP64	60 TFLOPS	3x
HBM3	3 TB/s	1.5X
PCI Gen5	128 GB/s	2x
4TH Gen NVLink	900 GB/s	1.5X
TDP	700W (SXM)	

شکل ۵.۱: محک‌های مختلف برای کارت گرافیک H100 و مقایسه آن با کارت گرافیک A100

در خوشبینانه‌ترین حالت فرض می‌کنیم تعداد عملیات‌های مورد نیاز برای بررسی یک کلید AES با طول کلید ۱۲۸ بیت معادل ۱۰۰۰ عملیات اعداد اعشاری می‌باشد. پس می‌توان گفت که هر کارت گرافیک H100 می‌تواند در ثانیه $6 \times 10^{13}/1000 = 6 \times 10^{10}$ کلید را بررسی کند.

فرض می‌کنیم طول کلید ما ۱۲۸ بیت می‌باشد، بنابرین تعداد حالات ممکن کلید برابر با $2^{128} \simeq 3.4 \times 10^{38}$ می‌باشد. پس مدت زمان لازم برای شکستن این رمز توسط حمله کننده برابر است با:

$$\text{Time} = \frac{3.4 \times 10^{38}}{10 \times 6 \times 10^{10}} \text{ Seconds} \simeq \frac{5.7 \times 10^{26} \text{ Seconds}}{31,536,000 \text{ Seconds / Year}} \simeq 1.8 \times 10^{19} \text{ Years}$$

بنابراین مدت زمان لازم برای شکستن یک رمز AES با کلید ۱۲۸ بیتی برابر است با 1.8×10^{19} یا ۱۸ میلیارد میلیارد سال!!

حال اگر فرض کنیم که حمله کننده دارای دسترسی به قوی‌ترین ابرکامپیوتر موجود که کامپیوتر Frontier با قدرت 1.2 exaFLOPS یا 1.2×10^{18} عملیات اعداد اعشاری بر ثانیه (یا $1.2 \times 10^{1}8/1000 = 1.2 \times 10^{15}$ عملیات رمز شکاندن در ثانیه) می‌باشد، آنگاه خواهیم داشت:

$$\text{Time} = \frac{3.4 \times 10^{38}}{1.2 \times 10^{15}} \text{ Seconds} \simeq \frac{2.8 \times 10^{23} \text{ Seconds}}{31,536,000 \text{ Seconds / Year}} \simeq 8.9 \times 10^{15} \text{ Years}$$

بنابراین قوی‌ترین ابرکامپیوتر موجود برای شکاندن این رمز به 8.9×10^{15} یا ۸/۹ میلیون میلیارد سال زمان نیاز دارد!!

۷.۱ سوال هفتم (رمز آلمان‌ها)

الگوریتم رمزگذاری Double Transposition یکی از رمزهای دستی ایمن بود که در دوران جنگ جهانی اول مورد استفاده قرار می‌گرفت. این روش توسط هر دو طرف جنگ، متفقین و قدرت‌های مرکزی، به کار گرفته می‌شد و کارایی بالایی داشت. با این حال، نقطه ضعف اصلی آن این بود که اگر مهاجم به دو یا چند پیام با همان طول و کلید دست پیدا می‌کرد، می‌توانست با استفاده از یک فرآیند زمان‌بر به نام "آناگرام‌گیری چندگانه" به رمزگشایی آنها بپردازد. همچنین، اجرای صحیح این الگوریتم به دقت بالایی نیاز داشت و اگر در نقطه‌ای حساس اشتباهی رخ می‌داد، رمزگشایی بسیار دشوار می‌شد. الگوریتم Double Transposition از دو مرحله جایگشت ستونی برای یک پیام تشکیل شده است. این دو مرحله می‌توانند با همان کلید یا با کلیدهای متفاوت انجام شوند. به عنوان مثال می‌خواهیم متن attackxatxdawn را رمزگذاری کنیم (x نشان‌دهنده فاصله بین کلمات است):

۱. ابتدا یک جدول اولیه با ابعاد مشخص درست میکنیم؛ سپس حروف متن را به ترتیب داخل جدول قرار میدهیم:

	col 1	col 2	col 3
row 1	a	t	t
row 2	a	c	k
row 3	x	a	t
row 4	x	d	a
row 5	w	n	x

شکل ۶.۱: مرحله اول Double Transposition

۲. در مرحله بعد یک جایگشت از ستون‌ها و یک جایگشت از ردیف‌ها را انتخاب میکنیم. به عنوان مثال برای:

ستون‌ها (۲، ۳، ۱) و برای ردیف‌ها (۲، ۴، ۱، ۵، ۳)

	col 1	col 3	col 2
row 3	x	t	a
row 5	w	x	n
row 1	a	t	t
row 4	x	a	d
row 2	a	k	c

شکل ۷.۱: مرحله دوم Double Transposition

۳. در مرحله آخر حروف را به ترتیب کنار هم مینویسیم. متن رمزگذاری شده برابر است با:

xtawxnattxadakc

۸.۱ سوال هشتم(انواع حملات)

شبکه GSM یک استاندارد بین‌المللی برای ارتباطات سلولی است که برای اولین بار در دهه ۱۹۹۰ معرفی شد. GSM به عنوان یک فناوری نسل دوم (2G) در ارتباطات سلولی شناخته می‌شود و هنوز تا به امروز در بسیاری از کشورها برای ارتباطات سلولی استفاده می‌شود.

ویژگی‌های GSM

ویژگی‌ها و عملکرد شبکه GSM عبارتند از:

- استفاده از فرکانس‌های تقسیم شده : GSM از تکنیک تقسیم فرکانس برای تقسیم باند فرکانسی استفاده می‌کند. با استفاده از FDMA، فرکانس‌های موجود در یک منطقه جغرافیایی به صورت تقسیم شده بین کاربران تقسیم می‌شوند. این به شبکه GSM امکان ارائه خدمات به چندین کاربر به صورت همزمان را می‌دهد.
- استفاده از TDMA: GSM از TDMA برای تقسیم زمانی فرکانس‌های تقسیم شده برای ارسال اطلاعات استفاده می‌کند. با استفاده از TDMA، زمان ارسال اطلاعات بین کاربران تقسیم می‌شود، به طوری که هر کاربر در یک زمان مشخص قادر به ارسال و دریافت اطلاعات است.
- استفاده از SIM: GSM از کارت SIM برای شناسایی و تأیید هویت کاربران استفاده می‌کند. کارت SIM شامل اطلاعات شبکه و کاربر می‌باشد و در دستگاه تلفن همراه قرار می‌گیرد. با استفاده از کارت SIM، کاربران می‌توانند به شبکه GSM متصل شوند و خدمات مخابراتی را دریافت کنند.
- پشتیبانی از خدمات صوتی و داده: GSM امکان ارائه خدمات صوتی (مکالمات) و خدمات داده (از جمله پیامک‌های کوتاه - SMS و ارسال داده‌ها) را فراهم می‌کند. این خدمات به کاربران امکان ارتباط و تبادل اطلاعات را می‌دهند.

شبکه GSM به عنوان یک استاندارد بین‌المللی، امکان اتصال و تبادل اطلاعات بین اپراتورهای مختلف را فراهم می‌کند و اجازه می‌دهد تا کاربران در سراسر جهان با هم ارتباط برقرار کنند.

GSM Active Sys

شبکه GSM را می‌توان با استفاده از دستگاهی به نام GSM Active Sys شنود کرد. این دستگاه که در حالت عادی به عنوان IMSI Catcher نیز شناخته می‌شود، یک دستگاه مخصوص است که برای شنود و در برخی موارد تغییر و مداخله در ارتباطات شبکه GSM استفاده می‌شود. این دستگاه قادر است ترافیک بی سیم مربوط به تلفن همراه‌های در حال مکالمه در یک شبکه GSM را شنود کند و حتی می‌تواند خود را در میان ارتباط کاربران قرار دهد.

GSM Active Sys چگونه کار می‌کند؟

نحوه کار این دستگاه به صورت کلی بدین صورت است که:

- شبیه‌سازی یک ایستگاه پایه سلولی: دستگاه GSM Active Sys قادر است به عنوان یک ایستگاه پایه سلولی عمل کند و اطلاعات لازم برای شبیه‌سازی ایستگاه پایه را در اختیار دارد. این ایستگاه پایه سلولی می‌تواند به تلفن‌های همراه در محدوده خود سرویس دهد و به آنها ارتباطی تقلبی ارائه دهد.
- تعامل با تلفن‌های همراه: هنگامی که تلفن همراه‌ها در محدوده تحت پوشش دستگاه GSM Active Sys قرار می‌گیرند، آنها سعی می‌کنند به شبکه ایستگاه پایه متصل شوند. در این مرحله، دستگاه GSM Active Sys به نماینده ایستگاه پایه سلولی می‌نماید و اطلاعات مورد نیاز برای برقراری ارتباط را از تلفن همراه دریافت می‌کند.
- شنود ترافیک ارتباطی: پس از برقراری ارتباط تقلبی با تلفن همراه، دستگاه GSM Active Sys قادر است ترافیک ارتباطی بین تلفن همراه و ایستگاه پایه را شنود کند. این شامل مکالمات صوتی، پیامک‌ها، داده‌ها و سایر ارتباطات است.
- تغییر و مداخله در ارتباطات: علاوه بر شنود، دستگاه GSM Active Sys در برخی موارد می‌تواند به صورت فعال در میان ارتباط قرار گیرد و تغییراتی در ارتباطات اعمال کند. به عنوان مثال، می‌تواند تماس‌ها را قطع کند، پیامک‌ها را مسدود کند یا دستکاری در داده‌های انتقالی انجام دهد.

به طور کلی، دستگاه GSM Active Sys با تقلید از ایستگاه پایه سلولی و ایجاد یک ارتباط تقلبی با تلفن همراه‌ها، قادر است ترافیک ارتباطی را شنود کند و در برخی موارد تغییراتی در ارتباطات اعمال کند. با این کار، قادر است به صورت غیرمجاز به اطلاعات حساس کاربران دسترسی پیدا کند.

راه‌های مقابله با GSM Active Sys

برای جلوگیری از این نوع حملات، سازمان 3GPP (سازمان مشترک تلفن همراه) تلاش کرده است تا استانداردها و روش‌های امنیتی را در شبکه‌های نسل سوم (3G) و نسل چهارم (4G) بهبود بخشد. این تلاش‌ها عمدتاً برای محدود کردن قابلیت ایجاد ارتباطات تقلبی و جعلی و تشخیص و جلوگیری از حملات IMSI Catcher صورت گرفته است. به طور کلی، این تلاش‌ها شامل موارد زیر است:

- استفاده از رمزنگاری: شبکه‌های نسل سوم و چهارم از رمزنگاری قوی تری نسبت به GSM استفاده می‌کنند. این رمزنگاری باعث کاهش امکان شنود و تقلب در ارتباطات می‌شود.
- استفاده از الگوریتم‌های امنیتی: استفاده از الگوریتم‌های امنیتی مانند A5/3 در شبکه‌های 3G و الگوریتم‌های امنیتی مبتنی بر AES در شبکه‌های 4G، امکان تقلب و شنود ارتباطات را به شدت کاهش می‌دهند.
- تشخیص تقلب: سازمان 3GPP روش‌های تشخیص و جلوگیری از IMSI Catcher را در استانداردها و امکانات شبکه‌های شبکه همراه تعبیه کرده است. این روش‌ها شامل تشخیص تغییرات ناگهانی در پارامترهای شبکه، تشخیص ارتباطات تقلبی و تشخیص تغییرات ناگهانی در مسیرهای ارتباطی هستند. با تشخیص اینگونه حملات، شبکه توانایی جلوگیری از ادامه عملیات تقلبی را دارد.
- استفاده از شبکه‌های همراه نسل پنجم (5G): شبکه‌های 5G از تکنولوژی‌ها و استانداردهای امنیتی پیشرفته‌تری نسبت به نسل‌های قبلی استفاده می‌کنند. این تکنولوژی‌ها شامل شناسایی ارتباطات تقلبی، رمزنگاری قوی‌تر، مدیریت دسترسی پیشرفته و امکانات امنیتی دیگر می‌شوند.

اگرچه تلاش‌های بسیاری برای جلوگیری از حملات IMSI Catcher انجام شده است، اما همچنان امکان وقوع این نوع حملات وجود دارد. بنابراین، شرکت‌های تولیدکننده شبکه و اپراتورهای تلفن همراه نیز باید بهبودهای امنیتی مستمری را در شبکه‌های خود اعمال کنند تا از حملات احتمالی جلوگیری کنند.

۹.۱ سوال نهم (حمله به DES)

این سوال ترجمه‌ای از مقاله زیر می‌باشد:

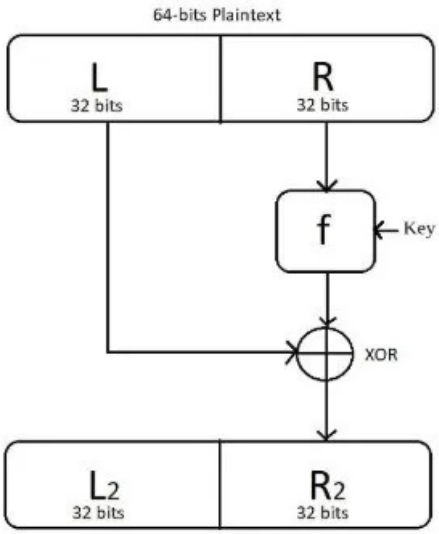
https://medium.com/@jnaman806/breaking-des-using-differential-cryptanalysis-958e8118ff41

تحلیل رمزی تفاضلی روشی است که تأثیر تفاوت‌های خاص در جفت‌های متن ساده بر تفاوت‌های جفت‌های متن رمزی حاصل را تحلیل می‌کند. از این تفاوت ها می توان برای تخصیص احتمالات به کلیدهای ممکن و یافتن محتمل ترین کلید استفاده کرد. در این پست خواهیم دید که چگونه می توان این تفاوت ها را تجزیه و تحلیل کرد و از آن برای شکستن DES به ۶ دور استفاده کرد.

این پست فرض می کند که خواننده استاندارد رمزگذاری داده ها (DES) را درک می کند. رمزگشایی دیفرانسیل معمولاً روی بسیاری از جفت‌های متن ساده با همان تفاوت خاص فقط با استفاده از جفت‌های متن رمزی حاصل کار می‌کند. برای سیستم‌های رمزنگاری مشابه ،DES این تفاوت به عنوان یک مقدار XOR ثابت دو متن ساده انتخاب می‌شود. اجازه دهید به سرعت ساختار DES Feistel را مرور کنیم.

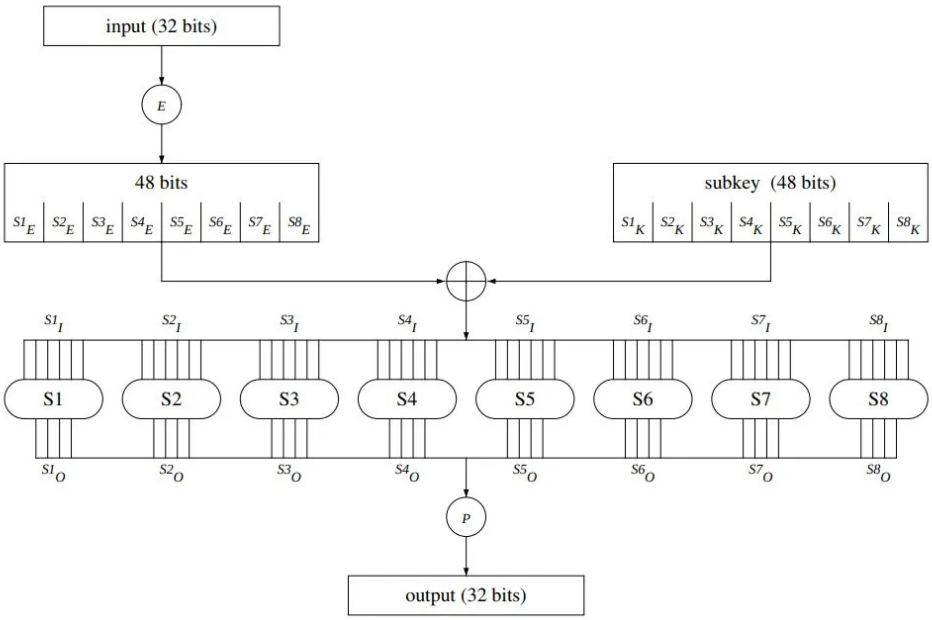
ساختار فایستلی

این یک مدل طراحی است که رمزهای بلوکی مختلفی از آن مشتق شده است. DES یکی از این رمزارزهای بلوکی است. مدل Feistel برای DES ۶۴ بیت متن ساده را می گیرد و آن را به دو نیم تقسیم می کند، L و R هر کدام ۳۲ بیت. R۲ به صورت R، Key) L محاسبه می شود و L۲ همان R است. در اینجا، Key یک کلید ۴۸ بیتی است که از الگوریتم زمان بندی کلید مشتق شده است. این مدل در شکل زیر نشان داده شده است.



در یک الگوریتم رمزگذاری، این روش تبدیل متن ساده را می توان برای هر تعداد بار استفاده کرد. خروجی یک دور به عنوان ورودی دور بعدی در نظر گرفته می شود. این کار برای ۱۶ دور در DES استاندارد انجام می شود.

در داخل، f دارای ساختار زیر در DES است.



نمادها

n_x: یک عدد هگزادسیمال با زیرنویس x نشان داده می شود

X, X': در هر مرحله میانی در طول رمزگذاری جفت پیام، X۱ و X۲ مقادیر میانی متناظر دو اجرای الگوریتم هستند. X به X' تعریف شده است = X1 XOR X2

P: متن ساده با P نشان داده می شود.

T: متن رمز شده با T نشان داده می شود.

P(X): جایگشت P با P(X) نشان داده می شود. توجه داشته باشید که P به عنوان یک متغیر متن ساده را نشان می دهد.

E(X): بسط E با E(X) نشان داده می شود.

IP(X): جایگشت اولیه.

(L, R): نیمه چپ و راست متن ساده P (پس از جایگشت اولیه) به ترتیب با L و R نشان داده می شوند.

(l, r): نیمه چپ و راست متن رمزی T (قبل از جایگشت نهایی) به ترتیب با l و r نشان داده می شوند.

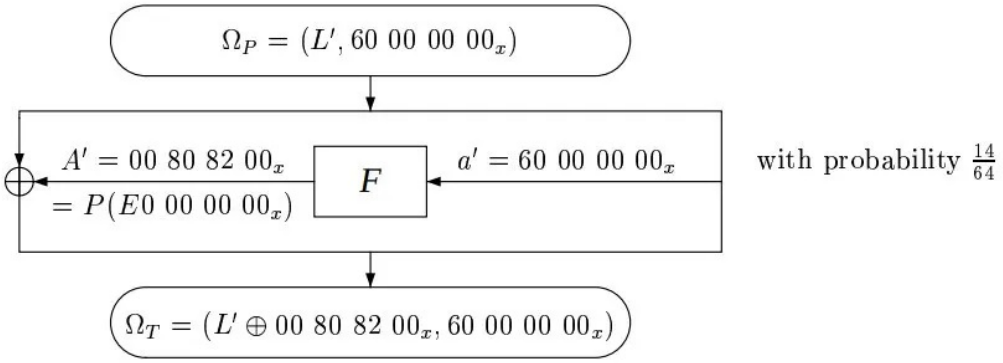
a,..., j: ورودی های ۳۲ بیتی تابع f در دورهای مختلف

A,..., J: خروجی های ۳۲ بیتی تابع f در دورهای مختلف

مشخصه

با هر جفت از رمزگذاری‌های ممکن، مقدار XOR دو متن اصلی آن، XOR متن رمزی آن، XOR ورودی هر دور در دو اجرا، و XOR خروجی هر دور در دو اجرا مشخصی وجود دارد. این مقادیر XOR یک مشخصه n دور را تشکیل می دهند. یک مشخصه یک احتمال دارد، که برابر است با احتمال اینکه یک جفت تصادفی با متن اصلی XOR انتخاب شده دارای XORهای دور و متن رمزی مشخص شده در مشخصه باشد. ما متن های اصلی XOR یک مشخصه را با Ω_p و متن های رمزی آن XOR را با Ω_t نشان می دهیم. توجه داشته باشید که این احتمال به دلیل وجود جفت ورودی های مختلف با XOR یکسان ممکن است به خروجی XOR متفاوت منجر شود.

برای درک بهتر، اجازه دهید به یک مثال نگاه کنیم.



a’ = 0110 0000 0000 0000 0000 0000 0000 0000 0000

E(a’) = 001100 000000 000000 000000 000000 000000 000000 000000

در حال حاضر، برای S۲-S۸ به عنوان ورودی XOR ۰ است، بنابراین خروجی XOR ۰ خواهد بود. اما، برای S۱، به عنوان ورودی XOR ۰۰۱۱۰۰، توزیع زیر از خروجی XOR بوجود می آید. بنابراین، احتمال به دست آوردن "E" ۷/۳۲ (یعنی ۱۴/۶۴) است. جایگشت P، هنگامی که به خروجی S-box اعمال می شود، به A’ = (00 80 82 00)x منجر می شود.

Output XOR ($S1'_O$)	Possible Input Pairs ($S1_I$)
3	(10,1C), (14,18), (24,28). (31,3D)
5	(00,0C), (15,19), (16,1A)
6	(07,0B), (20,2C), (33,3F)
9	(05,09), (11,1D), (35,39)
10	(22,2E), (30,3C), (34,38)
11	(23,2F), (27,2B)
12	(02,0E), (25,29), (32,3E)
13	(01,0D), (12,1E), (36,3A)
E	(03,0F), (06,0A), (13,1F), (17,1B), (21,2D), (26,2A), (37,3B)
F	(04,08)

بنابراین، مشخصه ۱ دور را به صورت Ωt = (L’ ⊕ 00 80 82 00, 60 00 00 00)x با احتمال ۱۴/۶۴ دریافت می کنیم.

شکستن DES به ۶ دور کاهش یافته

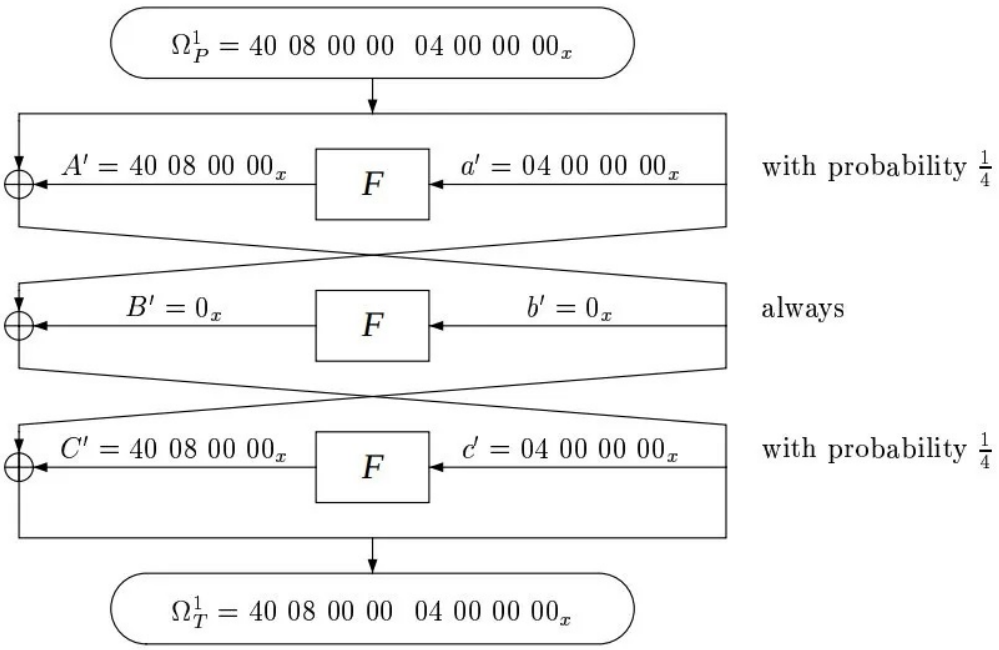
پس از تعریف مفهوم جفت ها و ویژگی ها، توضیح می دهیم که چگونه می توان از آن برای شکستن DES به ۶ دور استفاده کرد. ما از دو مشخصه ۳ دور استفاده می کنیم، هر دو با احتمال ۱/۱۶ و کلید را انتخاب می کنیم که اغلب شمارش می شود. هر یک از مشخصه ها به ما کمک می کند تا ۳۰ بیت کلید دور ۶ را پیدا کنیم. با این حال، ۳ تا از جعبه های S رایج هستند، بنابراین ما فقط ۴۲ بیت داریم. بقیه ۱۴ بیت را می توان با جستجوی جامع پیدا کرد.

در این نوع رویکرد، هدف ما این است که هر دو ورودی S-box و خروجی XOR از S-box را برای برخی دور بدست آوریم (دوره آخر راحت تر است زیرا خروجی را مستقیماً از متن رمز می دانیم). سپس، می‌توانیم روی کلیدهای ممکن تکرار کنیم تا به کلید برسیم.

فرض کنید دارای یک اوراکل می‌باشیم، که متن رمزی متن ساده داده شده را به ما می گوید (به یاد داشته باشید که در حال تلاش برای حمله متن ساده انتخابی هستیم). اوراکل می داند که DES ۶ دور است و همچنین کلید برابر است با:

111011110011001101110110110111100011010001010111111100010001 0011

اولین ویژگی این است:



ما متن‌های ساده را طوری انتخاب می‌کنیم که ورودی XOR بعد از جایگشت اولیه (IP) به صورت x(00 00 00 04 00 00 00 08 40) باشد. توجه داشته باشید که P1⊕ P2 با مشخصه یکسان نیست، اما IP(P1)⊕ IP(P2) برابر با مشخصه است.

P_1=1101010100100010110100110111001011100001101111000011001000101010

P_2=1101010100100010010100110110001011100001101111000111001000101010

Characteristic:

0100000000001000000000000000000000000010000000000000000000000000

P_1 after IP:

0001110101101101001000010001010100110101111110101010000011001110

P_2 after IP:

0101110101100101001000010001010100110001111110101010000011001110

پنج S-box (S2, S5, S6, S7, S8) در دور چهارم دارای XOR ورودی صفر هستند و بنابراین XOR خروجی آنها صفر است.

$d' = (40\ 08\ 00\ 00)_x$

d' = 0100 0000 0000 1000 0000 0000 0000 0000

E(d') = 001000 000000 000001 010000 000000 000000 000000 000000

XOR های خروجی مربوطه در دور ششم را می توان با $F' = c' \oplus l'$ پیدا کرد.

$l' = F' \oplus e'$

$e' = D' \oplus c'$

$F' = D' \oplus c' \oplus l'$

Hence, $F' = c' \oplus l'$ for 5 of the S-boxes

اوراکل متن های رمزی مربوط به P۱ و P۲ را به ما می دهد.

T_1=0101010011000110011001011101011110010001110101110111011001001110

T_2=1110010000110100001111111000001001010110011100110101000011010010

IP را برای خنثی سازی اثر جایگشت نهایی (FP) اعمال شده در دور آخر اعمال کنید.

After application of IP:

T_1=1110111101111001111011110011110000111010010001001000000011101010

T_2=1111000111110110000101110010010010001001001001110000010010111100

حالا ۳۲ بیت اول را به صورت f و ۳۲ بیت آخر را به صورت l استخراج کنید.

f_1 = 0011 1010 0100 0100 1000 0000 1110 1010

f_2 = 1000 1001 0010 0111 0000 0100 1011 1100

l_1 = 0011 1010 0100 0100 1000 0000 1110 1010

l_2 = 1000 1001 0010 0111 0000 0100 1011 1100 Input of S-box:

$E(f_1)$ = 000111 110100 001000 001001 010000 000001 011101 010100

$E(f_2)$ = 010001 010010 100100 001110 100000 001001 010111 111001

به دلیل ماهیت احتمالی مشخصه، هر جفت ورودی کلید صحیح را پیشنهاد نمی کند. ما یکی را می گیریم که اغلب در بسیاری از جفت های ورودی رخ می دهد. با تکرار بر روی تمام ۶۴ مقدار ممکن یک کلید برای هر یک از پنج S-box متناظر، تعداد کلیدهایی را افزایش دهید که مقادیر f بالا منجر به XOR شده با F' =c'⊕l' می شود. XOR خروجی S-box P_inverse (F') است.

c' = 0000 0100 0000 0000 0000 0000 0000 0000

l' = 1011 0011 0110 0011 1000 0100 0101 0110

F' = 1011 0111 0110 0011 1000 0100 0101 0110

Output of S-box:

1100 1101 0100 0100 0100 1101 0001 1011

برای S2، جفت های ورودی داده شده تعداد کلیدهای '111101', '111011', '011101', '011011' را افزایش می دهند. اجازه دهید این را برای کلید «011011» تأیید کنیم (همه مقادیر در کادر زیر فقط مربوط به S۲ هستند). E(f1)[S2] نشان دهنده ورودی S۲ در دور ششم است.

$E(f_1)[S2] \oplus 011011 = 110100 \oplus 011011 = 101111$

$E(f_2)[S2] \oplus 011011 = 010010 \oplus 011011 = 001001$S2(101111) = 0010

S2(001001) = 11110010 ⊕ 1111 = 1101 = Output XOR of S-box 2

با تجزیه و تحلیل ۲۵۰ جفت ورودی، بیت های کلید زیر مربوط به ۵ جعبه S را دریافت می کنیم:

Key bits corresponding to the blocks:

S2: 111101

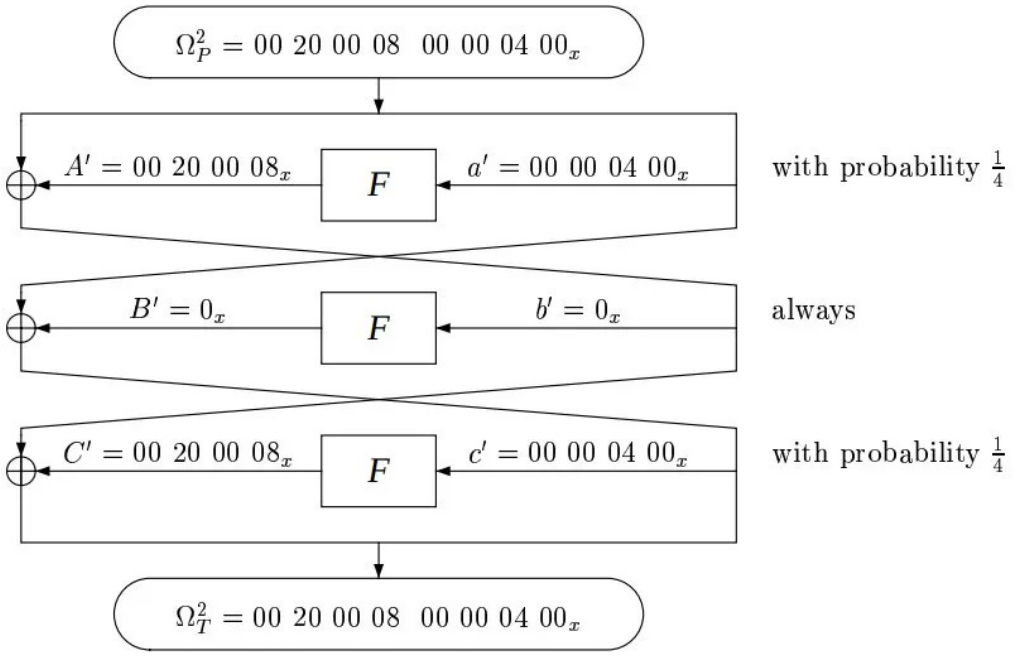
S5: 011010

S6: 101100

S7: 111011

S8: 010011

به طور مشابه، ۳۰ بیت دیگر ،S۱ ،S۲ ،S۴ ،S۵ S۶ را با استفاده از مشخصه دوم پیدا می‌کنیم.



با استفاده از مشخصه دوم تجزیه و تحلیل می کنیم:

Key bits corresponding to the blocks:

S1: 110010

S2: 111101

S4: 100110

S5: 011010

S6: 101100

مقادیر کلید محاسبه شده مربوط به S۲، S۵، S۶ باید با استفاده از هر دو ویژگی یکسان باشند. در غیر این صورت جفت های ورودی بیشتری را باید تحلیل کنیم. اکنون ۴۲ بیت از کلید ۵۶ بیتی داریم. موقعیت آنها را می توان با استفاده از یک الگوریتم زمان بندی کلیدی تعیین کرد. ۱۴ بیت باقی مانده از کلیدها را می توان با استفاده از brute force پیدا کرد.

Key guessed after analyzing using 2 characteristics:

x11011xxx011001x0xx1011xx10x111xx01xx10x01x1011x11x1000xx001x01x

Key guessed after brute force on remaining 14 bits:

1110111x0011001x0111011x1101111x0011010x0101011x1111000x0001001x

Actual Key:

11101111001100110111011011011110001101000101011111110001000100 11

توجه داشته باشید که تنها ۵۶ بیت از ۶۴ بیت در کلیدهای مختلف توسط الگوریتم زمان بندی کلید استفاده می شود. هر ۸ بیت قبل از اعمال الگوریتم زمان‌بندی کلید حذف می‌شود.

۱۰.۱ سوال دهم(همراه با Shannon)

اطلاعات متقابل و آنتروپی

آنتروپی

آنتروپی، در نظریه اطلاعات، معیار عدم قطعیت یا تصادفی بودن مرتبط با یک متغیر تصادفی است. مقدار اطلاعات مورد نیاز برای توصیف وضعیت متغیر را کمیت می کند. برای یک متغیر تصادفی X با مقادیر ممکن $x1, x2, \ldots, x_n$ و احتمالات مربوطه $p(x1), p(x2), \ldots, p(x_n)$ ، آنتروپی H(X) به صورت زیر تعریف می شود:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$$

آنتروپی اساساً نشان دهنده مقدار متوسط اطلاعات تولید شده توسط یک منبع تصادفی داده است.

اطلاعات متقابل

اطلاعات متقابل (MI) مقدار اطلاعاتی را که یک متغیر تصادفی در مورد متغیر تصادفی دیگر در خود دارد اندازه گیری می کند. این نشان می دهد که دانستن یکی از این متغیرها تا چه اندازه عدم اطمینان در مورد دیگری را کاهش می دهد. برای دو متغیر تصادفی X و Y مقدار MI یا I(X; Y) برابر است با:

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

که در آن H(X،Y) آنتروپی مشترک X و Y است. همچنین می توان آن را به صورت زیر بیان کرد:

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \left(\frac{p(x,y)}{p(x)p(y)} \right)$$

سیستم مخابراتی در مقابل سیستم رمزگذاری

سیستم مخابرات

هدف: انتقال اطلاعات قابل اعتماد

۱. جریان اطلاعات

- در یک سیستم مخابراتی، هدف اصلی انتقال دقیق اطلاعات از فرستنده به گیرنده است.
- اطلاعات متقابل در اینجا نشان دهنده مقدار اطلاعاتی است که با موفقیت از طریق کانال منتقل شده است.

۲. آنتروپی و نویز

- آنتروپی $H(X)$ میزان عدم قطعیت یا اطلاعات را در منبع اندازه گیری می کند.
- نویز یا آنتروپی زیاد در کانال می تواند اطلاعات متقابل را کاهش دهد زیرا خطاهایی را ایجاد می کند و بازیابی دقیق پیام اصلی را دشوارتر می کند.

۳. ظرفیت

- ظرفیت کانال حداکثر اطلاعات متقابل در تمام توزیع های ورودی ممکن است. این نشان دهنده بالاترین نرخی است که در آن اطلاعات می تواند با خطای خودسرانه کم منتقل شود.

سیستم رمزنگاری

هدف: انتقال امن اطلاعات

۱. جریان اطلاعات

- در یک سیستم رمزگذاری، هدف این است که اطلاعات را از دسترسی غیرمجاز ایمن نگه دارد.
- اطلاعات متقابل بین متن ساده و متن رمزی باید به حداقل برسد تا استنتاج متن ساده از متن رمز برای مهاجم دشوار شود.

۲. آنتروپی و امنیت

- آنتروپی متن ساده $H(X)$ باید زیاد باشد و پیش‌بینی آن را سخت‌تر می‌کند.
- اطلاعات متقابل $I(X;Y)$ بین متن ساده X و متن رمزی Y باید نزدیک به صفر باشد، که نشان می دهد دانستن Y اطلاعات کمی در مورد X ارائه می دهد یا هیچ اطلاعاتی را ارائه نمی دهد.

۳. تحلیل رمز

- حملاتی مانند تجزیه و تحلیل رمزنگاری تفاضلی و خطی سعی می کنند از اطلاعات متقابل غیر صفر بین متن ساده، متن رمزی و کلید برای کاهش امنیت سیستم رمزگذاری بهره برداری کنند.

خلاصه

- سیستم مخابراتی
  - هدف: به حداکثر رساندن اطلاعات متقابل برای اطمینان از ارتباط قابل اعتماد.
  - نقش آنتروپی: مقدار اطلاعات موجود در سیگنال منبع را منعکس می کند و به ارزیابی ظرفیت کانال کمک می کند.
  - اطلاعات متقابل مطلوب: اطلاعات متقابل بالا نشان دهنده انتقال موثر اطلاعات است.
- سیستم رمزنگاری
  - هدف: به حداقل رساندن اطلاعات متقابل برای اطمینان از ارتباط ایمن.
  - نقش آنتروپی: آنتروپی بالا در متن ساده، پیش بینی را دشوار می کند و امنیت را تضمین می کند.
  - اطلاعات متقابل مورد نظر: اطلاعات متقابل کم نشان دهنده رمزگذاری قوی و امنیت در برابر حملات است.

۲ سوالات عملی رمزهای متقارن

۱.۲ سوال اول(تحلیل فرکانسی)

در این سوال از الگوریتم ژنتیک برای کشف رمز استفاده شده است. این شامل اولیه سازی جمعیتی از کلیدها، ارزیابی تناسب آنها بر اساس فرکانس های n-gram یا چند حرفی ها بر اساس فرکانس آنها در زبان انگلیسی، انتخاب بهترین کلیدها، و تولید کلیدهای جدید از طریق متقاطع و جهش و بهبود مکرر جمعیت برای یافتن بهترین کلید رمزگشایی است.

۱. کلید شکسته‌شده: PGKUWQIYFXRCNAJDZHBMSTOVLE

۲. متن استخراج‌شده:

julius caesar was a celebrated roman general and statesman the conqueroo of gaul victor in the civil war and dicattor who was launching a series of political and social reform n when he was asasasinatd by a group of noble n in the nenate house on the ides of march he is one of the major figuren of clasisical antiuqity caesar changed the couore of the history of the grecoroman world deciviely and irreversbily the grecoroman society has been extinct for so long that most of the names of its great men mean little to the average educatde modern person but caesars name like alexanders is still on peoples lips throughout the christian and islamic worlds even people who know nothing of caesar as a historic personality are familiar with his family name as a title signifying a ruler who is in some sense uniquely supreme or paramount the meaing of kaiser in german tsar in the slavonic langauges and qayjar in the langauges of the islamic world

۲.۲ سوال دوم(رمز One-Time pad)

در این سوال باید از حمله گهواره کشیدن یا crib dragging استفاده شود. حمله گهواره کشیدن یک تکنیک تحلیل رمزنگاری است که برای شکستن رمزگذاری زمانی که یک کلید چندین بار استفاده می شود، مانند many time pad استفاده می شود (همچنین به عنوان one time pad استفاده مجدد نیز شناخته می شود).

نحوه کار

۱. انتخاب گهواره: گهواره دنباله کوتاهی از متن است که احتمالا در متن اصلی ظاهر می‌شود. کلمات رایج عبارتند از "the" یا "and" که اغلب به عنوان گهواره استفاده می‌شوند.

۲. عملیات XOR : گهواره با متن رمز در موقعیت های مختلف XOR می شود تا ببیند آیا متن قابل خواندن ظاهر می شود یا خیر. این به این دلیل است که XOR کردن کلید یکسان با متن رمزی و گهواره، کلید را خنثی می‌کند و بخش‌هایی از متن ساده را آشکار می‌کند.

۳. گهواره در سراسر متن رمزگذاری شده "کشیده می شود"، به این معنی که در نقاط شروع مختلف XOR می شود تا متن قابل خواندن را بررسی کند. هنگامی که نتیجه عملیات XOR متن قابل خواندن تولید می کند، نشان می دهد که گهواره به درستی در آن موقعیت قرار گرفته است.

۴. نگامی که یک متن قابل خواندن پیدا شد، می توان گهواره را گسترش داد تا متن ساده بیشتری را کشف کند. این روند تا زمانی تکرار می شود که متن ساده کافی برای درک پیام آشکار شود.

مثال

فرض کنید دو متن رمزگذاری شده با یک کلید داریم:

متن رمز ۱: '3b101c091d53320c000910'

متن رمز ۲: '271d154502010a04000419'

ما گمان می کنیم که کلمه "the" (که در هگز "746865" است) در هر دو متن اصلی ظاهر می شود. ما دو متن رمزی را XOR می کنیم:

$$\text{متن رمز ۱} \oplus \text{متن رمز ۲} = 3c0d094c1f523808000d09$$

سپس، گهواره ('746865') را در موقعیت های مختلف در نتیجه XOR می کنیم:

$$3c0d094c1f523808000d09746865$$

وقتی نتیجه متن قابل خواندن باشد، می دانیم که موقعیت صحیح گهواره را پیدا کرده ایم. سپس می توانیم از این اطلاعات برای کشف بیشتر متن ساده استفاده کنیم.

چرا کار می کند

حمله کشیدن گهواره کار می کند زیرا استفاده مجدد از همان کلید، اصل اساسی پد یکبار مصرف را نقض می کند، که نیاز به یک کلید واقعا تصادفی دارد که فقط یک بار استفاده شود. هنگامی که کلید دوباره استفاده می شود، الگوها را می توان مورد سوء استفاده

قرار داد و به مهاجمان اجازه می دهد بخش هایی از متن ساده را بازیابی کنند.

۳.۲ سوال سوم(رمز Vigenère)

مقدمه

رمزنگاری vigenere یکی از روش‌های رمزنگاری کلاسیک است که به دلیل استفاده از کلید چندحرفی، نسبت به روش‌های ساده‌تر نظیر سزار امنیت بیشتری دارد. با این حال، این روش در برابر تحلیل‌های آماری قابل شکستن است. هدف این پروژه پیاده‌سازی فرآیندی است که به صورت خودکار رمز vigenere را رمزگشایی کرده و متن اصلی را بازگرداند. مراحل کار به شرح زیر است:

۱. تعیین طول کلید با استفاده از **شاخص تطابق (Index of Coincidence)**.

۲. بازسازی کلید با تحلیل فراوانی حروف.

۳. رمزگشایی متن رمزگذاری شده با استفاده از کلید.

بخش اول: شاخص تطابق

شاخص تطابق معیاری آماری است که احتمال تکرار حروف در یک متن را محاسبه می‌کند. این مقدار برای متون انگلیسی معمولاً نزدیک به ۷.۱ است، اما برای متون تصادفی مقدار آن نزدیک به ۱ است. از این ویژگی برای تعیین طول کلید استفاده می‌شود. فرآیند کار به این صورت است:

۱. متن رمزگذاری شده به بخش‌هایی با طول‌های مختلف تقسیم می‌شود.

۲. شاخص تطابق برای هر بخش محاسبه می‌شود.

۳. طولی که مقدار میانگین شاخص تطابق آن نزدیک به ۷.۱ باشد به عنوان طول احتمالی کلید انتخاب می‌شود.

کد زیر این فرآیند را پیاده‌سازی می‌کند:

```
def index_of_coincidence(text):
    counts = Counter(text)
    total = sum(counts.values())
    ioc = sum(freq * (freq - 1) for freq in counts.values()) / (total * (total - 1))
    return 26 * ioc

def find_key_length(ciphertext, max_len=20):
    probable_lengths = []
    for key_len in range(1, max_len + 1):
        slices = [''] * key_len
```

```
        for i, char in enumerate(ciphertext):
            slices[i % key_len] += char
        avg_ioc = sum(index_of_coincidence(slice) for slice in slices) / key_len
        if avg_ioc > 6.1:
            probable_lengths.append((key_len, avg_ioc))
    return probable_lengths
```

در این کد، متن رمزگذاری شده برای طول‌های مختلف کلید بررسی می‌شود و طول‌هایی که شاخص تطابق آن‌ها مناسب است به‌عنوان طول احتمالی کلید برگردانده می‌شوند.

بخش دوم: یافتن کلید

پس از تعیین طول کلید، حروف کلید باید بازسازی شوند. برای این کار، هر بخش از متن رمزگذاری شده که مربوط به یک حرف از کلید است، تحلیل شده و با مقایسه فراوانی حروف آن با فراوانی حروف زبان انگلیسی، بهترین تطابق پیدا می‌شود.

```
def get_key(ciphertext, key_len):
    key = ''
    english_freqs = [
        082.0, 015.0, 028.0, 043.0, 13.0, 022.0, 02.0, 061.0, 07.0, 0015.0,
        0077.0, 04.0, 024.0, 067.0, 075.0, 019.0, 00095.0, 06.0, 063.0, 091.0,
        028.0, 0098.0, 024.0, 0015.0, 02.0, 00074.0
    ]
    for i in range(key_len):
        slice = ciphertext[i::key_len]
        counts = Counter(slice)
        total = sum(counts.values())
        scores = []
        for shift in range(26):
            shifted_freqs = [counts.get(ALPHABET[(j + shift) % 26], 0) / total for j in range(26)]
            scores.append(sum(f * e for f, e in zip(shifted_freqs, english_freqs)))
        key += ALPHABET[scores.index(max(scores))]
    return key
```

در این کد، هر بخش از متن رمز با شیفت‌های مختلف بررسی می‌شود و شیفتی که بهترین تطابق را با فراوانی حروف زبان انگلیسی داشته باشد به عنوان حرف کلید انتخاب می‌شود.

بخش سوم: رمزگشایی متن

پس از یافتن کلید، رمزگشایی متن با استفاده از معکوس فرمول رمزنگاری vigenere انجام می‌شود:

$$p_i = (c_i - k_{i\%L}) \mod 26 \tag{۱.۲}$$

کد مربوط به این مرحله به صورت زیر است:

```
def decrypt(ciphertext, key):
    plaintext = ''
    key_indices = [ALPHABET.index(k) for k in key]
    key_pos = 0

    for char in ciphertext:
        if char in ALPHABET:
            shift = key_indices[key_pos % len(key)]
            plaintext += ALPHABET[(ALPHABET.index(char) - shift) % 26]
            key_pos += 1
        else:
            plaintext += char
    return plaintext
```

نتیجه‌گیری

این پروژه با تحلیل آماری رمز vigenere را شکسته و کلید و متن اصلی را بازسازی می‌کند. مراحل به ترتیب شامل تعیین طول کلید، یافتن کلید و رمزگشایی متن است. این روش نشان می‌دهد که رمزنگاری کلاسیک در برابر تحلیل‌های مدرن آسیب‌پذیر است.

۱. کلید شکسته‌شده: CAKE

۲. متن استخراج‌شده:

ALAN MATHISON TURING WAS AN ENGLISH MATHEMATICIAN، COMPUTER SCIENTIST، LOGICIAN، CRYPTANALYST، PHILOSOPHER AND THEORETICAL BIOLOGIST.HE WAS HIGHLY INFLUENTIAL IN THE DEVELOPMENT OF THEORETICAL COMPUTER SCIENCE، PROVIDING A FORMALISATION OF THE CONCEPTS OF ALGORITHM AND COMPUTATION WITH THE TURING

MACHINE، WHICH CAN BE CONSIDERED A MODEL OF A GENERAL-PURPOSE COMPUTER.TURING IS WIDELY CONSIDERED TO BE THE FATHER OF THEORETICAL COMPUTER SCIENCE.DURING THE SECOND WORLD WAR، TURING WAS A LEADING PARTICIPANT IN THE BREAKING OF GERMAN CIPHERS AT BLETCHLEY PARK. THE HISTORIAN AND WARTIME CODEBREAKER ASA BRIGGS HAS SAID، "YOU NEEDED EXCEPTIONAL TALENT، YOU NEEDED GENIUS AT BLETCH-LEY AND TURING'S WAS THAT GENIUS."ON ٨ JUNE ،١٩٥٤ AT HIS HOUSE AT ۴٣ ADLINGTON ROAD، WILMSLOW، TURING'S HOUSEKEEPER FOUND HIM DEAD.A POST MORTEM WAS HELD THAT EVENING، WHICH DETERMINED THAT HE HAD DIED THE PREVIOUS DAY AT AGE ۴١ WITH CYANIDE POISONING CITED AS THE CAUSE OF DEATH.WHEN HIS BODY WAS DISCOV-ERED، AN APPLE LAY HALF-EATEN BESIDE HIS BED، AND ALTHOUGH THE APPLE WAS NOT TESTED FOR CYANIDE، IT WAS SPECULATED THAT THIS WAS THE MEANS BY WHICH TURING HAD CONSUMED A FATAL DOSE.

۳ سوالات تئوری نهان‌نگاری

۱.۳ سوال اول(روش LSB)

۱۰۲۴ پیکسل در تصویر وجود دارد. برای هر پیکسل می توانیم ۳ بیت را مخفی کنیم. بنابراین، تعداد کل بیت هایی که می توانیم در تصویر مخفی کنیم عبارتند از:

$$3 * 1024 = 3072\ bits$$

از آنجایی که در یک بایت ۸ بیت وجود دارد، می توانیم تعداد کل بیت ها را به بایت تبدیل کنیم:

$$3072/8 = 384\ bytes$$

۲.۳ سوال دوم(تعریف برخی مفاهیم)

مقدمه

نهان‌سازی اطلاعات یکی از حوزه‌های مهم در علم امنیت داده‌ها است که در آن هدف اصلی، پنهان کردن اطلاعات محرمانه در داده‌های بی‌ضرر به منظور حفظ حریم خصوصی یا جلوگیری از دسترسی غیرمجاز است. در این گزارش، به تعریف و توضیح مختصر چهار مفهوم کلیدی در این زمینه می‌پردازیم.

(آ) خطای نوع اول و خطای نوع دوم

در تحلیل آماری و آزمون فرضیه‌ها، دو نوع خطای اساسی تعریف می‌شود:

۱. **خطای نوع اول (Type I Error):** زمانی رخ می‌دهد که فرض صفر (H_0) رد شود، در حالی که در واقع درست است. به عبارتی، نتیجه آزمون به اشتباه وجود نشان یا اثر را تأیید می‌کند. احتمال وقوع این خطا با سطح معناداری α مشخص می‌شود.

۲. **خطای نوع دوم (Type II Error):** زمانی رخ می‌دهد که فرض صفر پذیرفته شود، در حالی که در واقع نادرست است. به عبارتی، سیستم نمی‌تواند نشان موجود را تشخیص دهد. احتمال وقوع این خطا با β مشخص می‌شود و قدرت آزمون برابر با

$1 - \beta$ است.

(ب) نمودار ROC (Receiver Operating Characteristics)

نمودار ROC ابزاری گرافیکی است که برای ارزیابی عملکرد سیستم‌های دسته‌بندی باینری یا مدل‌های تشخیص استفاده می‌شود. این نمودار نرخ تشخیص درست (True Positive Rate یا TPR) را در برابر نرخ مثبت کاذب (False Positive Rate یا FPR) برای آستانه‌های مختلف نشان می‌دهد.

- **حساسیت (Sensitivity یا TPR):** نسبت نمونه‌های مثبت درست تشخیص داده‌شده به کل نمونه‌های مثبت واقعی:

$$TPR = \frac{\text{Positives True}}{\text{Positives True} + \text{Negatives False}}$$

- **نرخ مثبت کاذب (FPR):** نسبت نمونه‌های منفی که اشتباهاً مثبت تشخیص داده‌شده‌اند به کل نمونه‌های منفی واقعی:

$$FPR = \frac{\text{Positives False}}{\text{Positives False} + \text{Negatives True}}$$

- **مساحت زیر نمودار (AUC):** مقدار AUC به عنوان معیار کلی عملکرد مدل در نظر گرفته می‌شود. مقدار AUC نزدیک به ۱ نشان‌دهنده عملکرد بسیار خوب مدل است.

(ج) سیگنال پوشش (Cover Signal)

سیگنال پوشش به داده یا سیگنال اصلی (مانند تصویر، صوت، ویدئو یا متن) اشاره دارد که برای پنهان کردن اطلاعات استفاده می‌شود. ویژگی‌های سیگنال پوشش عبارت‌اند از:

- باید به اندازه کافی پیچیدگی داشته باشد تا مخفی کردن اطلاعات در آن غیرقابل شناسایی باشد.
- سیگنال‌های پوشش معمولاً شامل داده‌های چندرسانه‌ای مانند تصاویر یا فایل‌های صوتی هستند.

مثال: در نهان‌نگاری تصویر، یک عکس معمولی به عنوان سیگنال پوشش استفاده می‌شود تا پیام مخفی در آن تعبیه شود.

(د) نشان‌گذاری کور (Blind Watermarking)

نشان‌گذاری کور یکی از روش‌های نهان‌نگاری دیجیتال است که در آن نیازی به نسخه اصلی سیگنال برای بازیابی نشان وجود ندارد. این ویژگی نشان‌گذاری کور را به گزینه‌ای مناسب برای کاربردهای مختلف تبدیل کرده است.

- **ویژگی‌ها:**
  - مقاوم بودن در برابر تغییرات یا حملات (مانند نویز، فشرده‌سازی یا ویرایش).
  - امکان استخراج نشان بدون نیاز به نسخه اصلی سیگنال.
- **کاربردها:**

- حفاظت از حق کپی‌رایت دیجیتال.
- تشخیص تغییرات غیرمجاز در داده‌ها.

مثال: اگر لوگویی به صورت نامرئی در یک تصویر ذخیره شود و بتوان آن را بدون دسترسی به تصویر اصلی بازیابی کرد، از نشان‌گذاری کور استفاده شده است.

نتیجه‌گیری

مفاهیم مطرح‌شده ابزارها و روش‌های کلیدی در علم نهان‌سازی اطلاعات هستند که درک آنها برای طراحی، پیاده‌سازی و ارزیابی سیستم‌های نهان‌نگاری ضروری است. هر یک از این مفاهیم نقش مهمی در تضمین امنیت و کارایی سیستم‌های مربوطه دارند.