دانشکده مهندسی کامپیوتر

امنیت داده و شبکه

تمرین ۱

دکتر میزانیان

آریا جلالی — ۹۸۱۰۵۶۶۵

۱۳ فروردین ۱۴۰۲

فهرست مطالب

۱ سوال ۱

۱.۱ TCP Sequence Prediction Attack

در این حمله مهاجم با گوش دادن به پیام‌های رد و بدل شده بین client و server سعی می‌کند شناسه پکت‌های TCP را حدس بزند. پس از درست حدس زدن این شناسه مهاجم می‌تواند بسته‌های ساختگی برای client بفرستد و آن‌ها را طوری طراحی کند که انگار از سمت Server آمده‌اند. مهاجم می‌تواند با گوش دادن به مکالمات رد و بدل شده شناسه را حدس بزند و پس از حدس شناسه یک مسابقه بین پکت‌های ارسال شده توسط سرور و مهاجم آغار می‌شود. دقت کنید در اکثر مواقع، پکت‌های سرور زودتر از پکت‌های مهاجم می‌رسند. به همین دلیل اکثر اوقات یک حمله محروم‌سازی از سرویس (DOS) به صورت موازی نیز انجام می‌شود.

روش‌های مقابله

۱. انتخاب اولین شناسه با استفاده از الگوریتم‌های تصادفی باعث می‌شود مهاجم نتواند شناسه را حدس بزند و حمله را آغاز کند.

۲. رمزنگاری کردن اطلاعات رد و بدل شده. در صورتی که پکت‌های در حال رد و بدل را رمزنگاری کنیم، این اجازه را به مهاجم نمی‌دهیم که بتواند شناسه پکت‌ها را بخواند و حتی در صورت درست حدس زدن، امکان تغییر و دستکاری را ندارد.

۳. استفاده از دیوار آتش این اجازه را به ما می‌دهد تا جلوی بسته‌هایی که از خارج شبکه با IP داخلی را بگیریم و نگذاریم بسته‌های جعلی از طریق مهاجم به کاربر برسد.

۴. استفاده از اطلاعات لایه‌های پروتکل زیرین. با استفاده از این اطلاعات مانند متادیتا فرستادنده پکت، فاصله زمانی بین پکت‌ها می‌توانیم متوجه حقیقی یا جعلی بودن پکت فرستاده شده بشویم و آن‌ها را drop کنیم. البته دقت کنید در اکثر اوقات دسترسی به این اطلاعات ممکن نیست یا بسیار دشوار است.

در این حمله مهاجم می‌تواند پکت‌های ساختکی از طرف سرور برای کلاینت بفرستد. در این حمله اصل صحت (Integrity) و به طور خاص‌تر صحت منبع نقض شده است.
دقت کنید می‌توان ادعا کرد اصل محرمانگی (Confidentiality) نقض شده است، اما دقت کنید در این نوع حملات مهاجم پیام‌های ارسال شده توسط کلاینت را دریافت نمی‌کند و تنها می‌تواند پیام‌های جعلی برای یک گیرنده بفرستد.

۲.۱ Reflector Attack

این حمله یک حمله محروم‌سازی از سرویس توزیع‌یافته (DDOS) است. در این نوع حمله با استفاده از تکنیک IP address spoofing مهاجم از طریق چندین دستگاه، IP جعلی تولید می‌کند و درخواستی به یک سرور آسیب‌پذیر مانند یک سرور DNS میفرستد. این سرورها معمولا پاسخی به مراتب حجیم‌تر از درخواست فرستاده شده می‌فرستند. و به نحوی باعث تقویت سیگتال مزاحم مهاجم می‌شوند. با استفاده از این تکنیک مهاجم می‌تواند با فرستادن ترافیک‌های کوچک، حجم زیادی از داده را به سمت IP جعل شده هدایت کند و باعث مختل شدن آن شود.

روش‌های مقابله

۱. استفاده از load balancing. در این روش پکت‌های ارسال شده توسط سرورها بین چندین سرور پخش می‌شود و مهاجم نمی‌تواند یک سرور به خصوص را مورد حمله قرار دهد.

۲. استفاده از دیوارآتش این اجازه را به ما می‌دهد تا ترافیک‌های دریافتی از سرورهای آسیب‌پذیر را محدود کنیم و از یک حدی به بعد این پکت‌ها را drop کنیم.

۳. با استفاده از فیلترهای Regex می‌توانیم پیام‌هایی که از یک فرم خاص پیروی می‌کنند (مانند پیام‌های DNS) تشخیص دهیم و آن‌ها را drop کنیم.

همانطور که بالاتر به آن اشاره شد، این حمله یک نوع خاص حمله محروم‌سازی از سرویس است که همانطور که از اسم آن مشخص است اصل Availability را نقض می‌کند.

۳.۱ Masquerading Attack

در این حمله مهاجم با استفاده از یک هویت جعلی به سیستم‌ها یا شبکه‌های یک کاربر دسترسی پیدا می‌کند و با استفاده از این هویت به اعمالی می‌پردازد که محدود به آن کاربر است. در اکثر اوقات از این حمله برای انجام اعمال غیرقانونی با استفاده از نام و هویت فرد دیگری انجام می‌شود.

روش‌های مقابله

۱. استفاده از روش‌های شناسایی پیشرفته‌تر. به عنوان مثال می‌توانیم از روش‌های multi-factor au- thentication استفاده کنیم که فرایند جعل هویت را به مراتب دشوارتر می‌کند.

۲. استفاده از امضاهای دیجیتال باعث جلوگیری از اجرای نرم‌افزارهایی میشود که دارای یک امضا از طرف یک منبع معتبر نیستند.

۳. استفاده از اصل privilege least باعث میشود هر کاربر تنها در حد نیاز دسترسی به اطلاعات حساس داشته باشد، اینکار باعث می‌شود مهاجم نتواند با جعل هویت هرکس بتوان به مقاصد شوم خود برسد و تنها افراد خاصی دسترسی‌های موردنیاز را داشته باشند.

۴. بررسی فعالیت‌های کاربر می‌تواند باعث شناسایی رفتارهای مشکوک شود. به عنوان مثال در صورت لوگین کردن یک کاربر از یک مکان جدید با IP جدید ممکن است نشانه این نوع حمله باشد.

در این حمله چون هویت یک فرد جعل می‌شود و دسترسی به اطلاعات کاربر به افراد غیرمجاز داده می‌شود هم اصل Integrity و هم اصل Confidentiality نقض می‌شود.

۴.۱ Fault Injection Attack

در حملات FIA مهاجم سعی می‌کند به صورت نرم‌افزاری یا سخت‌افزاری حالت خطا در سیستم ایجاد کند تا سیستم به صورت غیرمنتظره رفتار کند. در این بازه مهاجم می‌تواند با بررسی نوع رفتار سیستم ضعف‌هایی در آن پیدا کند و از آن‌ها سواستفاده کند.
در نوع فیزیکی FIA مهاجم می‌تواند با تغییر سیگنال کلاک یا تغییر ولتاژ پردازنده یا استفاده از تشعشعات متفاوت خطا وارد سیستم کند و سیستم را وادار به رفتارهای غیرمنتظره کند.

در نوع نرم‌افزاری این حمله مهاجم با استفاده از تکنیک‌های متفاوت مانند اختلال حافظه با Overflow کردن بافر سیستم با استفاده از فرستادن مقدار زیادی اطلاعات یا اجرای کد مخرب با گول زدن پردازنده، سیستم را وادار به رفتار غیرمنتظره می‌کند.

روش‌های مقابله

۱. استفاده از تصادفی سازی فضای حافظه یا ،(ASLR) در این روش فضای حافظه به صورت تصادفی چیده می‌شود و مهاجم نمی‌تواند به راحتی فضای ادرس را پیشبینی کند.

۲. محدود کردن دسترسی به سخت‌افزار سیستم به افراد معدود.

۳. استفاده از نرم‌افزارهای متفاوت برای اعمال Code review به صورت روتین برای تشخیص آسیب‌پذیری‌های ممکن.

در این نوع حملات اصل Integrity به دلیل رفتار غیرمنتظره سیستم که ممکن است باعث تخریب داده شود و اصل Availability نقض میشوند.

۵.۱ Man In The Middle Attack

در این نوع حملات یک فرد سوم بین ارتباط دو نفر قرار می‌گیرد و آن‌ها را فریب می‌دهد که به طور مستقیم در حال ارتباط با یکدیگر هستند. یک نوع محبوب از این حمله eavesdropping است که در آن مهاجم دو ارتباط مستقل به ۲ طرف برقرار می‌کند. هر دو طرف ارتباط فکر می‌کنند به طور مستقیم با یکدیگر در حال ارتباط هستند، در صورتی که تمام این ارتباط توسط فرد سوم که مهاجم است کنترل می‌شود. مهاجم می‌تواند تنها بسته‌های ارسالی را شنود کند و یا آن‌ها را دستکاری کند و با جعل هویت به فرد دیگر بفرستد.

روش‌های مقابله

۱. استفاده از ارتباط رمزنگاری شده

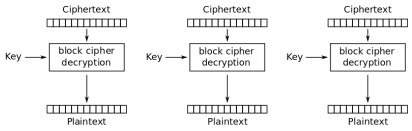
۲. استفاده از گواهی دیجیتال. استفاده از این گواهی‌ها می‌تواند به تایید صحت هویت فرد مقابل کمک کند و نگذارد فرد سومی هویت او را جعل کند.

۳. استفاده از کلیدهای عمومی که توسط سازمان‌های معتبر صادر شده‌اند. این روش جلوی استفاده از کلیدهای قلابی توسط افراد مهاجم را می‌گیرد.

در این نوع حمله هم شاهد جعل هویت هستیم و هم شاهد تغییر و دستکاری داده که نشان می‌دهد اصول Integrity و Confidentiality در آن نقض می‌شوند.

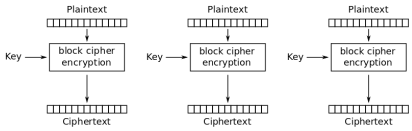
۲ سوال ۲

۱.۲ ECB



Electronic Codebook (ECB) mode decryption

شکل ۲: فرایند رمزگشایی



Electronic Codebook (ECB) mode encryption

شکل ۱: فرایند رمزگذاری

الف) همانطور که از تصویر بالا مشخص است، خطا در یک بیت از ciphertext تنها تاثیر مستقیم بر روی plaintext بلوک دارای آن بیت خواهد داشت. و تاثیری روی بقیه بلوک‌ها نخواهد گذاشت.

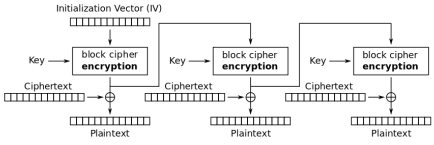
ب) از آنجایی که وابستگی‌ای بین بلوک‌ها چه در زمان رمزگذاری و رمزگشایی وجود ندارد، از دست رفتن یک بلوک تاثیری بر روی بلوک‌های دیگر ندارد.

پ) همانطور که پیشتر به آن اشاره شد به دلیل عدم وجود وابستگی بین بلاک‌ها، امکان رمزگشایی و رمزگذاری موازی وجود دارد.

ت) به همان دلایلی که در بخش قبل به آن‌ها اشاره شد(عدم وجود وابستگی بین بلوک‌ها) امکان رمزگشایی موازی وجود دارد.

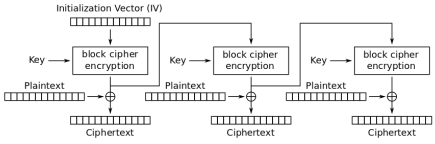
ث) چون وابستگی‌ای بین بلوک‌ها وجود ندارد، امکان رمزگشایی و رمزگذاری یک بلوک دلخواه وجود دارد.

۲.۲ OFB



Output Feedback (OFB) mode decryption

شکل ۴: فرایند رمزگشایی



Output Feedback (OFB) mode encryption

شکل ۳: فرایند رمزگذاری

الف) ciphertext هر بلوک تنها تاثیر مستقیم بر روی plaintext همان بلوک دارد و خطا در انتقال یک بیت از متن رمز شده تنها متن رمزگشایی شده همان بلوک را تحت تاثیر قرار می‌دهد.

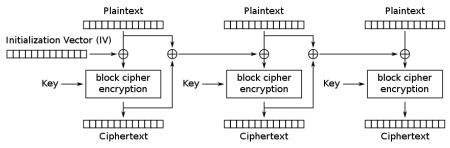
ب) همانطور که در تصویر مشخص است، یک بلوک متن رمزشده تنها در همان بلوک استفاده می‌شود و حذف آن تاثیری روی فرایند رمزگشایی بقیه بلوک‌ها نمی‌گذارد.

پ) در صورتی که از قبل vector Intialization با استفاده از key به تعداد دفعات لازم encrypt شده باشد، می‌توانیم با آمدن متن ساده آن‌ها را به صورت موازی رمزگذاری کنیم.

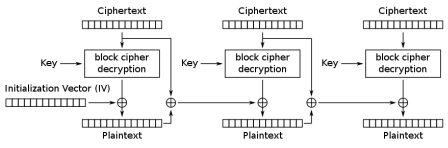
ث) از آنجایی که فرایند رمزگشایی و رمزگذاری دقیقا عین هم است، می‌توان با انجام عملیات رمزگذاری vector Initialization در ابتدا با استفاده از کلید فعالیت رمزگشایی را با آمدن متن‌های رمز به صورت موازی انجام داد.

ث) با توجه به توضیحات بالا، اگر از قبل vector Initialization با استفاده از کلید رمزگشایی شده باشد (به تعداد دفعات لازم) می‌توان بلوک دلخواه را رمزگشایی کرد.

PCBC ۳.۲



شکل ۵: فرایند رمزگذاری



شکل ۶: فرایند رمزگشایی

الف) همانطور که از شکل مشخص است خطا در بیت یک متن رمز به طور وضوح بر روی متن عادی همان بلاک تاثیر خواهد داشت، ولی چون از این متن رمز در رمزگشایی مرحله بعد نیز استفاده می‌شود، متن بلوک بعدی نیز تحت تاثیر قرار خواهد گرفت. با تحت تاثیر قرار گرفتن بلوک بعدی، به دلیل XOR شدن متن ساده با متن رمز و استفاده از نتیجه در بلوک بعد، این خطا تا انتها propagate می‌شود و تمامی بلوک‌های پس از بلوک دارای متن رمز خطا دار دچار مشکل می‌شوند.

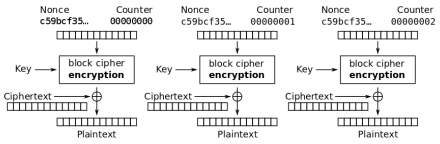
ب) به دلیل وابستگی متن‌های رمزگشایی شده هر بلوک به متن‌های رمزدار بلوک‌های قبل، با از دست دادن یک بلوک فرایند رمزگشایی تمام بلوک‌ها دچار اختلال می‌شود.

پ) همانطور که از تصویر مشخص است، هر بلاک برای شروع فرایند رمزگذاری نیاز به XOR متن بلوک قبل و نتیجه رمزگذاری آن دارد، این وابستگی اجازه رمزگذاری موازی را به ما نمی‌دهد.

ت) در فرایند رمزگشایی نیز برای بدست آوردن نتیجه نهایی نیاز به Plaintext بلوک قبل داریم و نمی‌توانیم عملیات را به صورت موازی انجام دهیم.

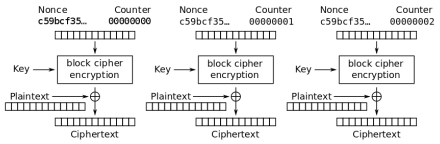
ث) به دلیل وابستگی نتیجه رمزگشایی یک بلوک دلخواه به نتیجه تمامی بلوک‌های قبل از خود، نمی‌توان یک بلوک دلخواه را بدون رمزگشایی بلوک‌های قبل از آن رمزگشایی کرد.

CTR ۴.۲



Counter (CTR) mode decryption

شکل ۸: فرایند رمزگشایی



Counter (CTR) mode encryption

شکل ۷: فرایند رمزگذاری

الف) از آنجایی که در فرایند رمزگشایی نتیجه بردار Initialization تنها در انتها با متن رمز XOR می‌شود، تنها بیت متناظر با بیت خطا دچار خطا می‌شود.

ب) به دلیل عدم وجود وابستگی بین بلوک‌ها چه در رمزگشایی و چه در رمزگذاری از دست دادن یک بلوک متن رمزشده تنها همان بلوک را تحت تاثیر قرار خواهد داد.

پ) به دلیل عدم وجود وابستگی بین Plaintext ها در فرایند رمزگذاری و دانستن مقدار (offset(counter و initialization (Nonce) می‌توانیم عملیات رمزگذاری را به صورت موازی انجام دهیم.

ت) همانند بخش قبل وابستگی‌ای بین متون رمز وجود ندارد و می‌توان عملیات رمزگشایی را به صورت موازی انجام داد.

ث) چون رمزگشایی هر بلوک وابستگی خارجی‌ای ندارد و تنها نیاز به دانستن Offset (Counter) و Nonce داریم، می‌توانیم بلوک دلخواه را رمزگشایی کنیم.

۳ سوال ۳

الف) این رمزنگاری در دسته رمزهای جانشینی چندالفبایی قرار می‌گیرد، زیرا چند حرف (در این الگوریتم ۲) با چند حرف دیگر (در این الگوریتم باز ۲) جایگزین می‌شوند.

ب) مزایا

۱. این الگوریتم به دلیل سادگی طراحی و رمزگذاری و رمزگشایی در مقاصد زیادی استفاده می‌شود.
۲. این الگوریتم در دسته رمزهای جانشینی چندالفبایی قرار می‌گیرد و هر دو حرف را جایگزین می‌کند که آن را نسبت به الگوریتم‌های جانشینی تک‌الفبایی امن‌تر می‌کند.

معایب

۱. در این الگوریتم نمی‌توان white space و یا نشانه‌های نگارشی را رمزنگاری کرد و خواندن متن رمزگشایی شده سخت است.

۲. هر حرف فقط می‌تواند با حروفی که در ستون یا سطر با آن اشتراک دارند جایگزین شود، که باعث می‌شود فضای حالت به اندازه کافی بزرگ نشود و بتوان با حملاتی مانند Brute Force یا تحلیل فرکانسی آن را رمزگشایی کرد.

پ) در ابتدا یک حرف از حروف انگلیسی حذف می‌شود تا ۲۵ حرف باقی بماند. در ادامه یک ماتریس ۵ در ۵ ساخته می‌شود که در ابتدا حرف‌های یکتا کلید به ترتیب در آن وارد می‌شوند و در ادامه حروف استفاده نشده در کلید به ترتیب الفبایی نوشته می‌شوند. معمولا حرف J حذف می‌شود.
در ادامه plaintext به صورت لیستی از جفت حرف‌ها تقسیم می‌شود.

۱. در صورتی که دو حرف یک جفت یکسان باشند یک حرف bogus که معمولا x است بین آن‌ها در plaintext اصلی قرار می‌دهیم.

۲. در صورتی که تعداد حروف plaintext فرد باشد یک حرف اضافه که معمولا z است در انتهای متن اصلی قرار می‌دهیم.

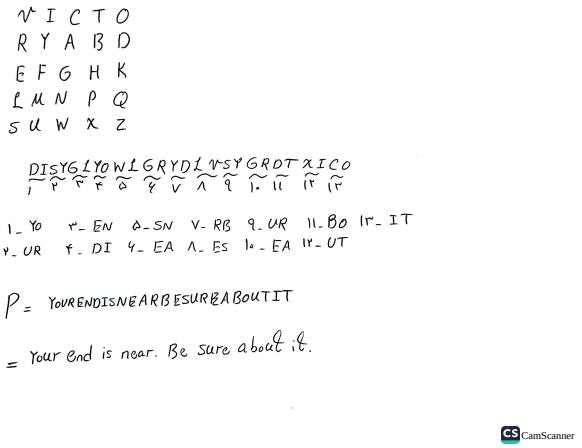
در ادامه هر دو حرف به صورت مقابل رمزگذاری می‌شوند

۱. در صورتی که دو حرف روی یک سطر باشند حرف راست هر حرف (چپ برای رمزگشایی) را جای هر حرف قرار می‌دهیم.

۲. در صورتی که دو حرف روی یک ستون باشند حرف پایین هر حرف (بالا برای رمزگشایی) را جای هر حرف قرار می‌دهیم.

۳. در صورتی که دو شرط بالا برقرار نباشد یک مستطیل محاط بین ۲ حرف تشکیل می‌دهیم و حرفی که روی راس مخالف افقی هر حرف قرار دارد را جای حرف اصلی قرار می‌دهیم.

ت) فرایند رمزگشایی در عکس زیر قابل مشاهده است.



ث) از آنجایی که این الگوریتم عنصر تصادفی‌ای در خود ندارد، امنیت بالایی نسبت به حملات فرکانسی ندارد. یکی از این دلایل این مشکل این است که هر حرف را تنها می‌توان با حروفی که در سطر یا ستون با آن اشتراک دارن جابجا کرد که باعث می‌شود فضای مسئله به اندازه کافی بزرگ نباشد. از طرفی کران بالا کلیدها !25 است که با استفاده از کامپیوترهای امروزی می‌توان کلید را با استفاده از Brute

Force در زمان نسبتا کمی بدست آورد. دقت کنید !25 کران بالا است و تعداد کلیدهای خاص بسیار کمتر است. از طرفی ویژگی‌های خاص این الگوریتم تحلیل فرکانسی را آسان‌تر می‌کند، به عنوان مثال اگر AB به LU تبدیل شود، BA به UL تبدیل می‌شود.
ترکیب ویژگی‌های این الگوریتم با نمودار فرکانسی حروف دوتایی یک زبان شکستن آن را با تحلیل فرکانسی ممکن می‌سازد، اما به مراتب شکستن آن از شکستن الگوریتم‌های جانشینی تک‌الفبایی سخت‌تر است.

۴ سوال ۴

$$C' = K' \oplus P = \neg K \oplus P = \neg(K \oplus P) = \neg C$$

همانطور که رابطه بالا نشان می‌دهد پیام رمزنگاری شده با کلید نقیض، نقیض پیام را تولید می‌کند. همانطور که در درس به آن اشاره شد OTP دارای امنیت مطلق است و داشتن نقیض پیام رمزشده کمکی به ما برای رمزگشایی نمی‌کند.

۵ سوال ۵

۱.۵ خاصیت بهمنی اکید

خاصیت بهمنی به خاصیت دلخواه الگوریتم‌های رمزنگاری اشاره می‌کند که در آن یک تغییر کوچک در ورودی (تغییر یک یا چند بیت) باعث تغییر قابل توجه در خروجی می‌شود. خاصیت بهمنی اکید یا Strict avalanche criterion فرم فرمول‌سازی شده خاصیت بهمنی است. می‌گوییم یک الگوریتم دارای خاصیت بهمنی اکید است اگر تغییر یک بیت در ورودی باعث تغییر تمام بیت‌های خروجی با احتمال ۵۰ درصد شود.

۲.۵ خاصیت تمامیت

در رمزنگاری می‌گوییم یک تابع بولین کامل است اگر هر بیت خروجی به تمام بیت‌های ورودی وابسته باشد. در صورتی که به طور متوسط با تغییر هر بیت ورودی هر بیت خروجی با احتمال ۵۰ درصد تغییر پیدا کند استفاده از این تابع باعث وجود خاصیت بهمنی اکید در الگوریتم رمزنگاری می‌شود.

۳.۵ Random Cipher

مدل Random Cipher یک مدل تئوری است که در آن فرض می‌شود الگوریتم رمزگذاری یک تابع تصادفی است که پیام آشکار داده شده را با احتمال برابر به یک متن‌رمز نگاشت می‌کند. در صورتی که احتمال تمام متن رمزها برای یک متن آشکار یکسان باشد ارتباطی بین متن آشکار و متن رمز وجود نخواهد داشت و الگوریتم دارای امنیت مطلق خواهد بود.

۴.۵ بررسی خواص برای مدل داده شده

۱.۴.۵ خاصیت بهمنی اکید

دقت کنید پس از اجرای یک راند الگوریتم (همانطور که در تصویر نشان داده شده است) وابستگی بین بلوک‌های مختلف وجود ندارد و تنها بلوک $y3$ به بلوک $x1$ وابسته است. پس از ۴ بار تکرار این ساختار هر بلوک اولیه یک شیفت کامل می‌خورد و به بلوک سمت راست خود وابسته است. اگر خروجی‌های راند ۴ را با P_i نشان دهیم داریم

1. $P_0 = g(X_1, X_0)$
2. $P_1 = g(X_2, X_1)$
3. $P_2 = g(X_3, X_2)$
4. $P_3 = g(X_0, X_3)$

با انجام دادن این ساختار ۴ بار دیگر خواهیم داشت

1. $P_0^{'} = g(g(X_1), P_1) = g^{'}(X_1, X_2)$
2. $P_1^{'} = g(g(X_2), P_2) = g^{'}(X_2, X_3)$
3. $P_2^{'} = g(g(X_3), P_3) = g^{'}(X_3, X_0)$
4. $P_3^{'} = g(g(X_4), P_0) = g^{'}(X_0, X_1)$

و در نهایت پس از ۱۲ راند خواهیم داشت

1. $P_0^{''} = g^{''}(X_1, X_2, X_3)$
2. $P_1^{''} = g^{''}(X_2, X_3, X_0)$
3. $P_2^{''} = g^{''}(X_3, X_0, X_1)$
4. $P_3^{''} = g^{''}(X_0, X_1, X_2)$

تکرار ساختار داده شده به اندازه حداقل ۱۲ بار شرط لازم خاصیت تمامیت و خاصیت بهمنی اکید است.
برای برقراری خاصیت بهمنی اکید و تمامیت اثبات می‌شود تابع F باید یک تابع bent باشد که در ادامه به تعریف آن خواهیم پرداخت:

Bent Function

تابع bent یک تابع بولین است که فاصله Hamming جدول صحت آن از تمامی تابع‌های خطی و affine بیشترین فاصله را دارد.
تبدیل Walsh یک تابع بولین به صورت مقابل تعریف می‌شود.

$$f^a = \sum_{x \in \mathcal{Z}_2^n} (-1)^{f(x)+a.x}$$

به طور دقیق‌تر تابع bent تابعی است که تبدیل walsh آن یک تابع ثابت باشد.

۲.۴.۵ خاصیت تمامیت

همانطور که در بالا اشاره شد برای داشتن خاصیت تمامیت لازم است ساختار داده شده را حداقل ۱۲ بار تکرار کنیم و از تابع‌های خاصی برای F استفاده کنیم.

۳.۴.۵ Random Cipher

دقت کنید در مدل Random Cipher هر متن رمز برای یک متن آشکار مشخص احتمال برابری دارد و باید یک کلید مخصوص به خود داشته باشد. از آن‌جایی که CAST یک الگوریتم بلاکی است، طول کلید از قبل در آن فیکس می‌شود و سایز آن تغییر نمی‌کند، در صورتی که طول متن آشکار دست ما نیست. از آنجایی که فضای حالت متن آشکار بزرگتر از فضای حالت کلیدها است احتمال متن رمزها نمی‌تواند برای هر متن آشکار برابر باشد و این ساختار از مدل Random Cipher پیروی نمی‌کند.

۶ سوال ۶

بله. یکی از مشکلات اصلی رمزنگاری DES سایز کلید ان است. این کلید ۵۶ بیتی حمله آزمون جامع را ممکن می‌سازد. به همین دلیل یکی از ویژگی‌های اصلی کلید که سایز آن است تاثیرگذار است.
عامل دیگری که در رمزگذاری DES حائز اهمیت است نوع کلید است. هر کلید ۵۶ بیتی در ۱۶ دور با استفاده از یک Key Scheduler به یک کلید ۴۸ بیتی تبدیل می‌شود. در رمزنگاری هر الگوریتم ممکن است تعدادی کلید ضعیف داشته باشد که باعث می‌شود الگوریتم به درستی عمل نکند. در DES از بین 2^{64} چهار کلید ضعیف داریم. در صورتی که این کلیدها در بدترین حالت انتخاب شوند در تمام ۱۶ مرحله یک کلید تولید می‌شود.

کلیدهای ضعیف الگوریتم DES

1. 0x0101010101010101
2. 0xFEFEFEFEFEFEFEFE
3. 0xE0E0E0E0F1F1F1F1
4. 0x1F1F1F1F0E0E0E0E

دقت کنید کلیدهای بالا پس از Permuation اولیه به کلیدهای ۵۶ بیتی مقابل تبدیل می‌شوند

1. 0x0000000 0000000
2. 0x0000000 FFFFFFF
3. 0xFFFFFFF 0000000
4. 0XFFFFFFF FFFFFFF

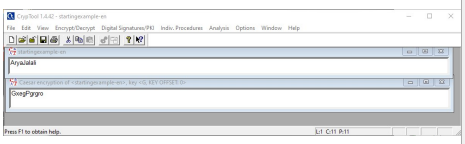
در صورتی که کلیدهای بالا به عنوان کلید ۶۴ بیتی انتخاب شوند، مهاجم اکنون تنها به دنبال یک کلید ۴۸ بیتی است و ۸ بیت از طول کلید حذف شده است.
از طرفی دقت کنید الگوریتم DES به صورت عمومی در دسترس است و نحوه decipher کردن یک متن در الگوریتم DES به صورت وارون cipher کردن آن است، اگر ترتیب کلیدها را برعکس کنیم. حال در حالتی

که تمام کلیدها یکسان باشند عملیات decipher با cipher یکسان خواهد بود و مهاجم می‌تواند با یکبار رمزنگاری کردن پیام رمزشده به پیام اصلی دست پیدا کند، یعنی

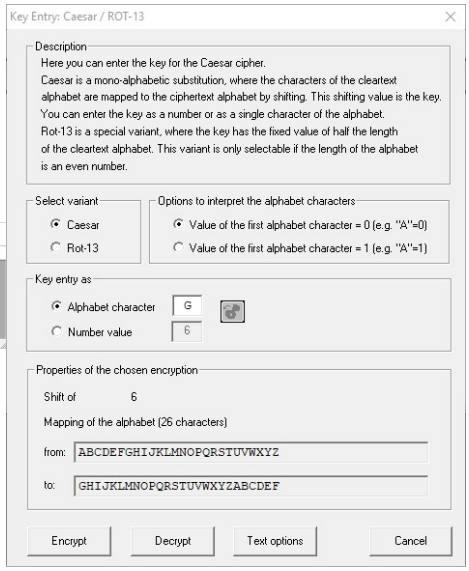
$$E_{WK}(E_{WK}(P)) = P$$

۷ سوال ۷

۱.۷ رمزگذاری با Caesar



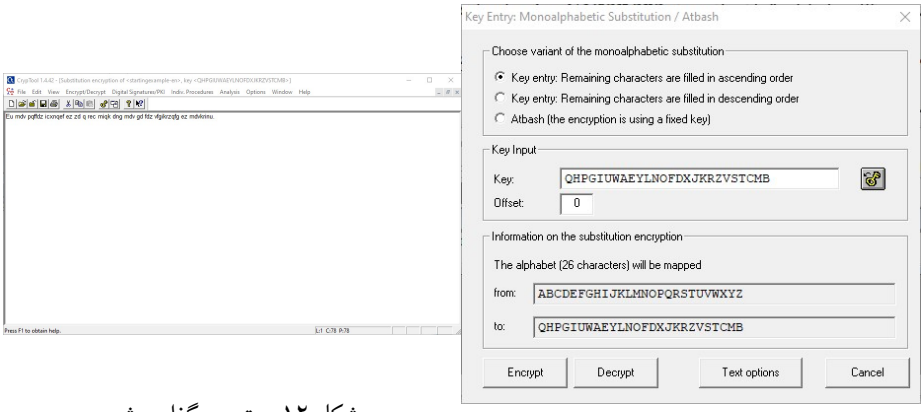
شکل ۱۰: متن اصلی و متن رمزگذاری شده متناظر



شکل ۹: فرایند رمزگذاری

متن رمزشده‌ی ’AryaJalali’ با استفاده از روش سزار برابر است با .’GxegPgrgro’

۲.۷ رمزگذاری با رمز جایگزینی

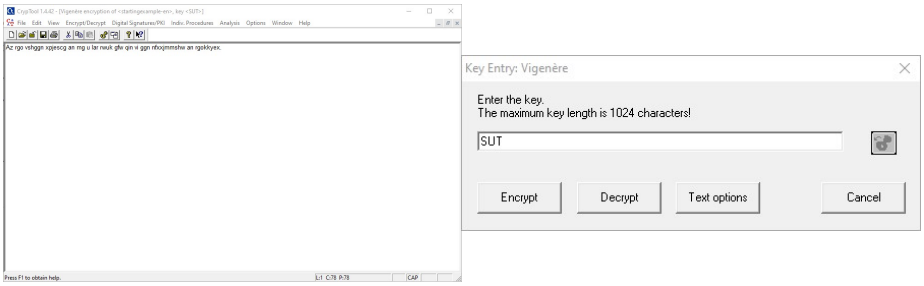


شکل ۱۱: فرایند رمزگذاری

شکل ۱۲: متن رمزگذاری شده

۳.۷ رمزگذاری با Vigenere

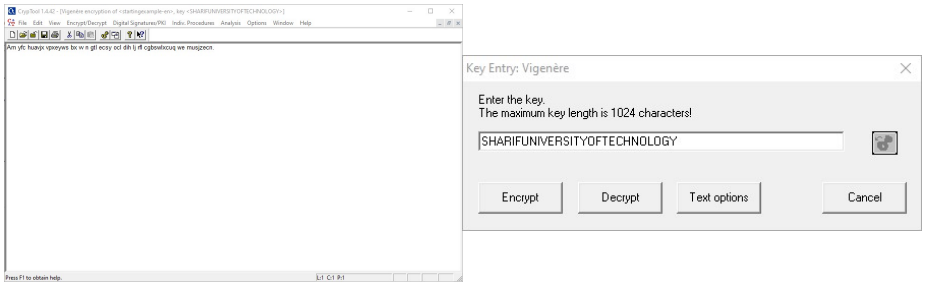
۱.۳.۷ کلید ’SUT’



شکل ۱۳: فرایند رمزگذاری

شکل ۱۴: متن رمزگذاری شده

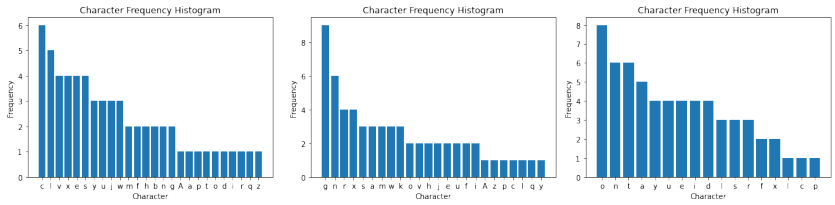
۲.۳.۷ کلید ’SHARIFUNIVERSITYOFTECHNOLOGY’



شکل ۱۵: فرایند رمزگذاری

شکل ۱۶: متن رمزگذاری شده

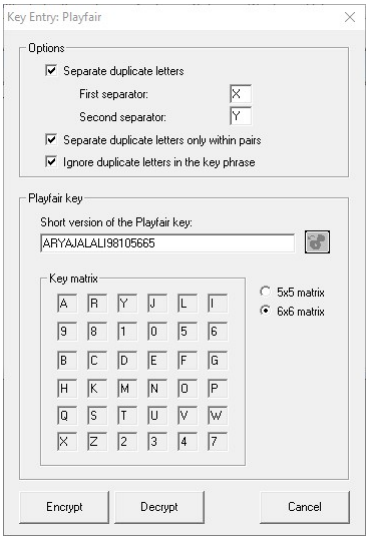
۳.۳.۷ تاثیر طول کلید



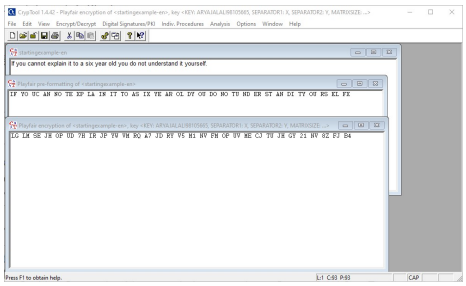
شکل ۱۷: متن اصلی شکل ۱۸: کلید ’SUT’ شکل ۱۹: کلید ’SHARIF’

همانطور که از نمودارهای فرکانسی بالا مشخص است، استفاده از کلیدهای طولانی‌تر یکتایی کلمات را کاهش می‌دهد و احتمال اینکه حروف یکسان با یک حرف رمز شوند کاهش پیدا می‌کند و بررسی جایگشت حروف در نمودار فرکانسی به مراتب دشوارتر است به دلیل تعداد حالات بیشتر.

۴.۷ رمزگذاری با Playfair



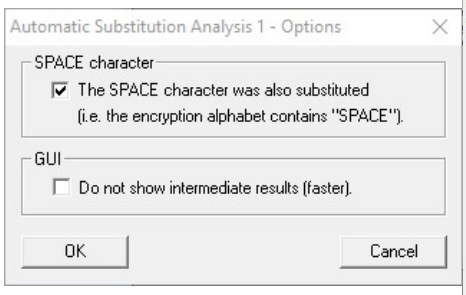
شکل ۲۰: فرایند رمزگذاری

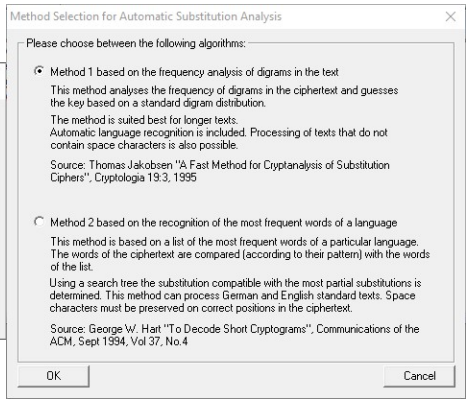


شکل ۲۱: متن رمزگذاری شده

۵.۷ رمزگشایی جایگشتی

۱.۵.۷ تنظیمات



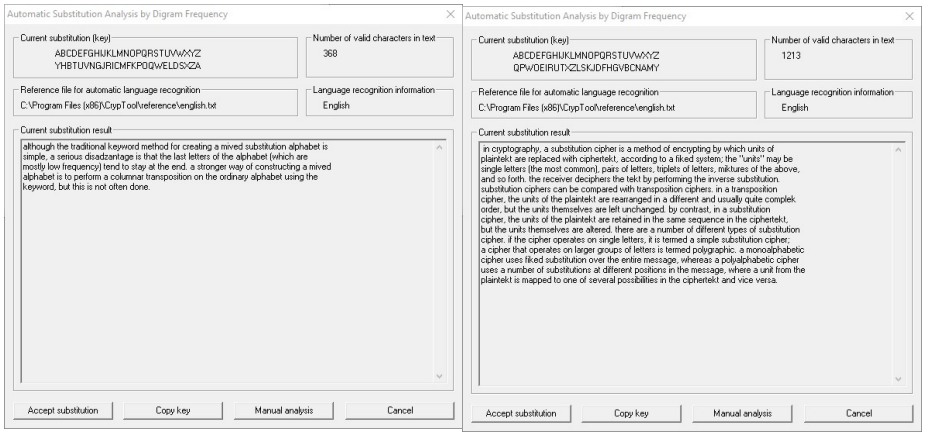


شکل ۲۳: تنظیمات رمزگشایی

شکل ۲۲: انتخاب روش رمزگشایی

دو روش رمزگشایی پیشنهاد شد که برای این سوال از روش اول که روش default است استفاده کردیم و در ادامه space را به عنوان حروف الفبا متن در نظر گرفتیم(در صورت عدم این فرض، خروجی معنی‌دار نمی‌شود)

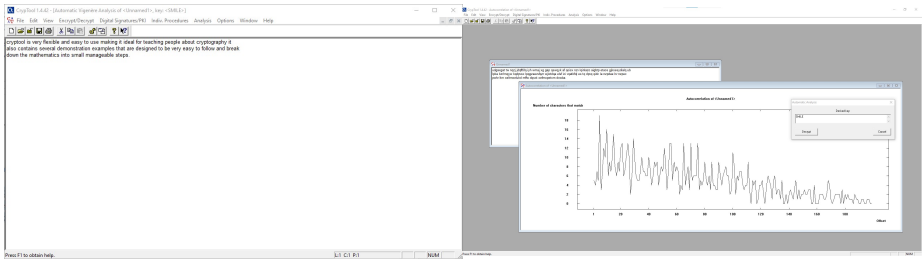
۲.۵.۷ رمزگشایی



شکل ۲۴: متن اول شکل ۲۵: متن دوم

در متن اول حرف X به اشتباه به A مپ شده است، در متن دوم این اتفاق برای حروف X و V رخ داده است. یکی از دلایلی که می‌تواند بانی این اتفاق باشد طول بیشتر متن اول است. در صورتی که طول متنی بزرگتر باشد، تحلیل فرکانسی رمز جایگشتی آن با احتمال بیشتری به فرکانس زبان انگلیسی نزدیک‌تر خواهد بود و رمزگشایی آن آسان‌تر خواهد بود. به همین دلیل متن ۱ به زبان طبیعی نزدیک‌تر است.

۶.۷ رمزگشایی vigenere



شکل ۲۶: کلید و نمودار Autocorrelation شکل ۲۷: رمزگشایی شده متن رمز

علاوه بر روش Kasiski روش دیگری که برای بدست آوردن طول کلید استفاده می‌شود روش Autocorrelation است. در این روش متن رمز را مقداری شیفت می‌دهیم و تعداد کاراکترهایی که عین هم هستند را بدست می‌آوریم، به احتمال بالا زمانی که به اندازه طول کلید شیفت می‌دهیم این مقدار ماکسیمم خود را می‌گیرد که در نمودار بالا برابر با ۵ است که طول کلید ما یعنی ’SMILE’ است. ایده پشت این روش این است که پس از شیفت

دادن به اندازه طول کلید، بقیه متن با یک الگو رمزگذاری شده‌اند و احتمال برابری زیاد است، در صورتی که کمتر از طول کلید شیفت دهیم حروف یکسان به احتمال بالا با حروف مختلف کلید رمزگذاری شده‌اند و تعداد کاراکترهای یکسان کمتر از حالت گفته شده است. منطقی هست که هرچه بیشتر شیفت دهیم به دلیل اشتراک کمتر بین دو متن تعداد کاراکترها به صورت متوسط کاهش پیدا می‌کند.

$$c_i - c_j = p_i - p_j + k_{i \bmod \mathcal{L}} - k_{j \bmod \mathcal{L}}$$

برای 0 شدن عبارت بالا که به معنای برابری کاراکترهای c_i و c_j در ciphertext است، نیاز داریم عبارت سمت راست 0 شود. اگر فرض کنیم که $i \equiv j \pmod{\mathcal{L}}$ رابطه بالا به صورت مقابل در خواهد آمد

$$c_i - c_j = p_i - p_j$$

به دلیل نامنظم بودن توزیع آماری متن اصلی، انتظار داریم عبارت سمت راست با احتمال بالاتری نسبت به بقیه حالات 0 شود که نتیجه می‌دهد اگر عبارت را به اندازه طول کلید شیفت دهیم احتمال دیدن حروف برابر بیشتر است.

۸ سوال ۸

۱.۸ پیدا کردن طول کلید

```
import math

words = CFG.encrypted.split(" ")
repeated_words = {}
gcd = 0

current_length = 0
for word in words:
    if len (word) < 3:
        current_length += len(word)
        continue
    if word not in repeated_words:
        repeated_words[word] = [current_length]
    else:
        for length in repeated_words[word]:
            if gcd == 0:
                gcd = current_length - length
            else:
                gcd = math.gcd(gcd, current_length - length)
        repeated_words[word].append(current_length)
    current_length += len(word)

possible_key_lengths = [i for i in range(2, gcd+1) if gcd % i == 0]
```

با استفاده از یک دیکشنری کلمات را ذخیره می‌کنیم و در صورتی که طول آن‌ها بزرگتر از ۳ باشد و قبلا در متن با آن‌ها برخورد کرده باشیم، ب.م.م فاصله‌های کلمه‌های یکسان را محاسبه می‌کنیم و این فرایند را تا پایان متن ادامه می‌دهیم و در نهایت مقسوم‌علیه‌های ب.م.م را به عنوان طول‌های ممکن کلید در لیست possible_key_lengths ذخیره می‌کنیم.

۲.۸ بدست آوردن کلید

```
import numpy as np

def get_distribution(text):
    freq = [text.count(char) for char in CFG.alphabet_frequencies.keys()]
    freq = [f / sum(freq) for f in freq]
    return freq

def get_current_char(distribution):
    word_freq = np.array(list(CFG.alphabet_frequencies.values())) / 100
    mse_list = [np.mean((np.roll(word_freq, shift) - distribution) ** 2) for shift \
    in range(1, len(distribution) + 1)]
    best_shift = np.argmin(mse_list)
    return list(CFG.alphabet_frequencies.keys())[(best_shift + 1) % len(distribution)]

clean_text = CFG.encrypted.replace(" ", "")

keys = []
for key_length in possible_key_lengths:
    cur_key = ""
    for i in range(key_length):
        distribution = get_distribution(clean_text[i::key_length])
        cur_key += get_current_char(distribution)
    keys.append(cur_key)
```

در این بخش برای هر طول کلید ممکن حروفی که توسط یک حرف کلید رمزگذاری شده‌اند را جدا می‌کنیم و توزیع آماری تکرار حروف را بدست می‌آوریم و با معیار MSE مقدار شیفتی را پیدا می‌کنیم تا این معیار کمینه شود و با توجه به مقدار شیفت، حرف متناظر را به عنوان حرف i ام کلید برمی‌گردانیم. در نهایت کلیدها را در آرایه keys ذخیره می‌کنیم.
*کلیدهای بدست آمده 'دروغ' و 'مف' هستند.

۳.۸ رمزگشایی

```
def decode_words(text, key):
    spaces = [i for i, letter in enumerate(text) if letter == " "]
    text = text.replace(" ", "")
    letters = list(CFG.alphabet_frequencies.keys())
```

```
    decoded = ""
    for i, letter in enumerate(text):
        decoded += letters[(letters.index(letter) - \
        letters.index(key[i % len(key)])) % len(letters)]

    for index in spaces:
        decoded = decoded[:index] + ' ' + decoded[index:]
    return decoded

for key in keys:
    print(f"Key = {key}\n", decode_words(CFG.encrypted, key))
```

در کد بالا ابتدا پس از ذخیره کردن مکان white space ها، حروف کلید را به ترتیب از حروف متن رمز کم می‌کنیم تا به متن اصلی برسیم و در نهایت با استفاده از لیست spaces فاصله‌ها را در مکان خود قرار می‌دهیم.

۱.۳.۸ متن با کلید 'دروغ'

لقمان حکیم پسر را گفت امروز طعام مخور و روزه دار و هرچه بر زبان راندی بنویس شبانگاه همه انچه را که نوشتی بر من بخوان انگاه روزهات را بگشا و طعام ختج شبانگاه پسر هر چه نوشته بود خواند دیروقت شد و طعام نتوانست خورد روز دوم نیز چنین شد و پسر هیچ طعام نخورد روز سوم باز هرچه گفته بود نوشت و تا نوشته را بر خواند افتاب روز چهارم طلوع کرد و او هیچ طعام نخورد روز چهارم هیچ نگفت شب پدر از او خواست که کاغذها بیاتجد و نوشتهها بخواند پسر گفت امروز هیچ نگفتهام تا برخوانم لقمان گفت پس بیا و از این نان که بر سفره است بخور و بدان که روز قیامت انان که کم گفتهاند چنان حال خوشی دارند که اکنون تو داری

۴.۸ متن با کلید 'مف'

خزویذ نلهد قصذ گغ مغض غوذرب عظسص وحرا ی ذربا خسا ی وگما اگ بتیذ اپمقع تمرعص سشغهکسظ الز غهجز اپ قز ضیسضع تذ دض تحرغه یذسپو گطسوسک ژی شسضی ر حفید وجث وفپمحغا بنا اذ عظ هنوکا اره ذنسضر خژایفض ثر ن ادپل ذکییذتج حرار ذرب رند ضبر عضبم وه ی بنا اهع حفید ضذنگه ژنل تیل شغس وگما کثکا اره هنوک ی پس ضیسضظ ژی شا ذنسضر یثکپا گطس جزغژل اشیظ چار ن سط اهع حفید ضذنگه ژنل مایگص اهع ضمغض ثت بقا پر سط ذنستج قز ژپعکظپ اژغجثق ط هنوکاوس فذنسضر بنا مغض غوذرب اهع ضمغضظپل ضغ تذفطپمد شگلسض مغض قص اژغ ی یل غبم ذغه قز فژ ژثاا ینک تحرا ی اقغه قز ایر جعپلض غهیذ ژا قد سکپزغهخ عضپم غغن حرثب خساهخ چظ پقذطه پر هپذژ

همانطور که واضح است، کلید واقعی 'دروغ' با طول ۴ است.