هوالحق

تمرین کامپیوتری شماره ۱
مبانی امنیت شبکه، بهار ۱۴۰۱

در این تمرین قصد داریم یک نسخه شخصی از رمزکننده دنباله‌ای A5-1 را پیاده سازی کنیم.

همانطور که در درس با این رمزکننده آشنا شده‌اید، این رمزکننده یک نمونه از ساختار های رمزکننده دنباله‌ای می باشد. از این ساختار برای ایجاد ارتباط امن بی‌سیم استفاده می شود که به طور گسترده در نسل دوم ارتباط های تلفنی مورد استفاده قرار گرفته است که البته به صورت عمومی معرفی نشده بود و از طریق روش‌های مهندسی معکوس و نشت اطلاعات توانستند این ساختار را در ابتدا بازسازی کنند. این ساختار ضعف های امنیتی بسیاری دارد که سبب شده است در نسل های جدید شبکه های همراه مورد استفاده قرار نگیرد.

ساختار این رمزکننده در فایل شماتیک داده شده قرار دارد که در این تمرین شما باید سعی کنید این رمزکننده را پیاده سازی کنید. این ساختار با ترکیب تعدادی شیفت رجیستر فیدبک‌دار خطی و با استفاده از ساختار کلاک نامنظم ایجاد شده است.

کد تحویلی شما می‌بایست شامل ۳ فایل مجزا به شرح ذیل باشد:

۱) فایل keystream_gen.py

- در این قسمت شما می‌بایست با دریافت کلید جلسه[1] از کاربر با اندازه ۶۴ بیت بتوانید کلیدی به اندازه طول مورد نیاز تولید کنید.
- برنامه شما در قدم اول باید کلید جلسه ۶۴ بیتی را از کاربر دریافت کند و صحت سنجی کند که کلید حتما از کاراکتر های ۰ و ۱ ایجاد شده باشد و اندازه آن نیز ۶۴ بیت باشد و در صورت عدم احراز شرایط گفته شده با پیغام مناسب از اجرای برنامه جلوگیری کند.
- در قدم بعدی شما باید با استفاده از کلید داده شده بتوانید طبق الگوریتم مورد نظر برای رمزکننده A5-1، سه رجیستر LFSR[2] موجود در ساختار را به طور صحیح مقداردهی اولیه کنید. هر یک از این سه رجیستر به ترتیب ۱۹، ۲۲، ۲۳ بیت از کلید جلسه را در خود نگهداری می کنند که به این ترتیب ۶۴ بیت مورد نظر در رجیستر ها ذخیره شده است.
- در هر یک از رجیسترهای موجود یک بیت به عنوان بیت مربوط به کلاک قرار داده شده است که در شماتیک داده شده با رنگ زرد نمایش داده شده‌اند. نحوه عملکرد کلاک[3] در این ساختار به این صورت است که ما از قاعده اکثریت استفاده می کنیم (با استفاده از سه بیت مشخص شده با رنگ زرد). به طور دقیق‌تر، هر کدام از رجیسترها به این شرط کلاک خواهند زد که بیت کلاک آن ها برابر باشد با مقدار حاصل از قاعده اکثریت بر روی سه بیت مشخص شده. به طور مثال اگر به ترتیب سه بیت کلاک به صورت ۰، ۱ و ۱ باشند با توجه به قاعده اکثریت، بیت انتخاب شده برابر است با ۱ و رجیسترهای شماره دو و سه کلاک خواهند زد و رجیستر شماره یک کلاک نخواهد زد. به این روش کلاک‌زنی نامنظم[4] گفته می‌شود.

---

[1] Session Key
[2] Linear-feedback Shift Register
[3] Clock
[4] Irregular Clocking

- پس از پیاده سازی ساختار کلاک گفته شده، شما باید با مشاهده شماتیک داده شده بتوانید ساختار موجود که شامل LSFRها و XORها می‌باشد را پیاده‌سازی کنید که بتواند کلید مناسب را برای ما ایجاد کند.
- این فایل باید به این صورت عمل کند که با دریافت کلید ۶۴ بیتی و متن مورد نظر برای رمزگذاری یا رمزگشایی، بتواند کلید رشته‌ای مورد نظر را ایجاد کند.
- تعداد دفعات اجرای الگوریتم بالا را طول متن داده شده برای رمزگذاری یا رمزگشایی مشخص می‌کند.

۲) فایل encrypt.py

- این فایل به انجام رمزنگاری روی فایل ورودی با نام input.* پرداخته و با استفاده از رشته کلید تولید شده توسط توابع موجود در فایل keystream_gen.py رمزنگاری را انجام می‌دهد. به این صورت که نتیجه XOR بیت‌های متناظر از key_stream و متن داده شده را محاسبه و ذخیره می‌کند.
- فایل ورودی باید به صورت باینری خوانده شود (و نه متنی).
- نتیجه حاصل از رمزنگاری را در فایل با نام input.*.enc ذخیره کنید. با توجه به الگوریتم استفاده شده برای رمزنگاری، باید اندازه فایل حاصل از رمزنگاری با فایل ابتدایی برابر باشد.

۳) فایل decrypt.py

- این فایل به انجام رمزگشایی روی فایل ورودی با نام input.*.enc پرداخته و با استفاده از رشته کلید تولیدشده توسط توابع موجود در فایل key_stream.py رمزگشایی را انجام می‌دهد؛ به این صورت که نتیجه XOR بیت‌های متناظر از key_stream و متن داده شده را محاسبه و ذخیره می‌کند.
- نتیجه حاصل از رمزگشایی را در فایل با نام output.* ذخیره کنید. نتیجه حاصل باید عیناً شبیه به فایل ابتدایی داده شده با نام input.* باشد.

هوالحق

تمرین کامپیوتری شماره ۱
مبانی امنیت شبکه، بهار ۱۴۰۱

نکات

* برنامه شما باید علاوه بر رمزنگاری و رمزگشایی فایل‌های متنی، توانایی انجام عملیات روی هر نوع فایلی را داشته باشد. این امر با برآورد شرط مشابهت صد درصدی بایت به بایت فایل‌های *.input و *.output تضمین خواهد شد (* به معنی هر نوع فرمت دلخواه و ممکن است)؛ فلذا با تغییر نام هر گونه فایل متنی، مالتی‌مدیا و غیره به نام‌های مربوطه، انجام عملیات رمزنگاری و بازگرداندن پسوند آن‌ها پس از رمزگشایی، فایل‌ها بایستی مشابهت کامل و هویت خود را حفظ نمایند. برای این کار لازم است تا خواندن و نوشتن مقادیر فایل‌های ورودی و خروجی را به صورت باینری (و نه متنی) انجام دهید.

* برنامه‌های نوشته شده می‌بایست Robustness لازم را دارا باشد و در هر مرحله با پیغام خطای مناسب علت عدم اجرای صحیح را اطلاع رسانی کند. تحت هیچ شرایطی بروز Segmentation Fault و یا Exception ها و Error های پیش فرض در روند اجرای برنامه قابل قبول نیست.

* تمامی الگوریتم‌های لازم می‌بایست به صورت کامل پیاده‌سازی شوند. تنها استفاده از توابع ریاضی توان، لگاریتم و اپراتورهای ریاضی پیش‌فرض قابل قبول است.

* برنامه‌های شما می‌بایست با زبان Python نوشته شده باشد.

* استفاده از هرگونه کد قبلی و نمونه کدهای آنلاین غیر مجاز است. تمامی مراحل و توابع می‌بایست توسط خود شما پیاده‌سازی گردد.

* در این تمرین، تأکیدی روی Performance نیست. تمرکز خود را روی پیاده‌سازی دقیق الگوریتم قرار دهید.

* ملاک جزئیات الگوریتم‌ها، مطالب تدریس شده در کلاس درس می‌باشد.

* با توجه به تصحیح خودکار تمرین‌ها، لطفاً توجه لازم را به نام‌گذاری فایل‌ها، ورودی‌ها و خروجی‌ها مبذول دارید.

* به شماتیک ضمیمه این تمرین توجه لازم را داشته باشید.

* انجام این تمرین به صورت گروهی، مشورتی و یا استفاده از کدهای از پیش آماده غیر مجاز است. موارد تحویلی می‌بایست توسط خود دانشجو انجام شده باشد.

* فایل‌های تحویلی شما می‌بایست **یک فایل فشرده با فرمت zip با نام شماره دانشجویی شما و فقط شامل ۳ فایل سورس کد keystream_gen.py، encrypt.py و decrypt.py** باشد.