Digital Image Encryption by AES and DES Algorithms Using Various Authentication-Encryption and Confidentiality Block Cipher Modes of Operation

Narges Mehran

Supervisor: Dr. M.R. Khayyambashi

07.2012

دانشگاه اصفهان
دانشكده فني و مهندسي
گروه مهندسي كامپيوتر

پايان‌نامه‌ي كارشناسي رشته‌ي مهندسي كامپيوتر گرايش سخت‌افزار

رمزنگاري تصوير با استانداردهاي رمزنگاري پيشرفته و رمزنگاري داده با استفاده از شيوه‌هاي مختلف رمزنگاري قطعه‌اي يا مدهاي كاري مختلف

استاد راهنما:

دكتر محمد رضا خيام‌باشي

پژوهشگر:
نرگس مهران

تيرماه 1391

چکیده

از هزاران سال پیش، رمزنگاری و ایجاد امنیّت در پیام برای مخفی نگه‌داشتن محتوای آن و جلوگیری از شناخته شدنش توسط افراد غیر مُجاز انجام‌گرفته است. گسترش استفاده از کامپیوترها و حملات خرابکاران این مقوله را با ارزش و مهم ساخته است. نیاز به امنیت در ارتباطات جهانی روز به روز بیشتر می‌شود. در واقع سنگ بنای تمامی تمهیدات و مکانیزم‌های امنیتی بر بدبینی مفرط و وسواس بی‌حد گذاشته می‌شود. بدین دلیل روش‌های مختلف رمزنگاری اعم از متقارن و نامتقارن برای محرمانه نگه‌داشتن پیام در طول انتقال آن در شبکه ابداع گشتند و بدنبال آن پروتکل‌های رمزنگاری مختلفی برای بستن راه نفوذ اخلال‌گران تولید شدند.

در پروژه‌ی پیش‌رو، شیوه‌های مختلف رمزنگاری متقارن قطعه‌ای را تشریح و در نرم‌افزار متلب پیاده‌سازی خواهد شد. سپس به شرح عملکرد هر یک از این شیوه‌ها پرداخته خواهد شد، برخی از این شیوه‌ها قابلیت موازی‌سازی و پردازش پایپ‌لاین را دارند و برخی علاوه بر این امنیت و سرعت بالا. در شیوه‌هایی که در دهه‌ی اخیر به اختراع آن‌ها منجر شده علاوه بر حفظ محرمانگی پیام، که مدهای قدیمی نیز این قابلیت را داشتند، به گونه‌ای طرح شده‌اند تا بتوان احراز سلامت پیام نیز مشخص گردد و گیرنده بتواند طرف فرستنده‌ی پیام را شناسایی کند.

در پایان‌نامه‌ی پیش‌رو ابتدا با شرح کلّی موضوع و مشکلات آن و چگونگی رفع مشکل موجود آشنا خواهید شد. در فصل دوم و سوم به تاریخچه‌ی رمزنگاری، مفاهیم آن و پیکربندی انواع تصاویر پرداخته خواهد شد. در فصل چهارم ابتدا دو الگوریتم رمز قطعه‌ای متقارن را توصیف کرده و سپس به توضیح کلّی شیوه‌ها یا مُدهای رمز قطعه‌ای پرداخته خواهد شد و در ادامه، مراحل پیاده‌سازی الگوریتم‌ها در نرم‌افزار در فصل پنجم بیان خواهدشد. در فصل ششم به ارزیابی این پیاده‌سازی‌ها و مقایسه‌ی آن‌ها با شیوه‌های احراز هویت و رمزنگاری پرداخته‌می شود. فصل هفتم حاوی نتیجه‌گیری کلّی و پیشنهاداتی است که برای ادامه این پروژه وجود دارد.

کلمات کلیدی: شیوه‌های رمزنگاری الگوریتم‌های رمز قطعه‌ای، مُدهای رمزنگاری، مُدهای کاری، رمزنگاری تصویر، الگوریتم‌های رمز بلوکی یا قطعه‌ای، رمزنگاری متقارن.

فهرست مطالب

فهرست شکل‌ها

فهرست جدول‌ها

فصل اول

۱-۱- مقدمه

در این فصل سعی در ارایه‌ی مختصری در شرح کلّی موضوع پروژه و مشکلات موجود در برخی از شیوه‌های رمزنگاری قطعه‌ای که موضوع اصلی این پژوهش است، خواهد بود. در پایان روشی که مشکلات موجود را تا حدودی برطرف کرده و به گونه‌ای احراز اصالت را به صورت فشرده در یک طرح درکنار محرمانگی پیام به ارمغان آورده است به طور مختصری شرح داده‌خواهد شد.

۱-۲- ضرورت توجه به امنیّت اطلاعات

زمانی که ژولیوس سزار پیام‌هایی را برای فرمانده‌ی ارتش خود در جنگ می‌فرستاد از بیم کشته‌شدن یا خیانت پیک، در تمام متن نامه‌ی خود هر حرف را با حرفی که سه تا بعد از آن قرار گرفته‌بود عوض‌کرد (مثلاً به جای حرف A حرف D و به جای حرف B حرف E را قرار داد) تا متن حالت اصلی خود را از دست بدهد. تنها کسی می‌توانست از مفهوم متن اطلاعاتی دریافت کند که به رمز آن (یعنی سه عدد شیفت) آگاهی داشت.

صدها و شاید هزاران سال پیش پیغام‌های مهم که از میدان جنگ به پشت جبهه ارسال می‌شدند، به صورت رمزی در می‌آمدند تا در صورتی که سرباز حامل نامه‌ها اسیر شد، اطلاعات لو نروند. در گذشته‌ی نه‌چندان دور رمزگذاری برای ایجاد امنیت در ارتباطات نظامی و دولتی استفاده می‌شد. امّا گسترش استفاده از کامپیوترها و حملات خرابکاران این مقوله را با ارزش و مهم ساخته‌است. نیاز به امنیت در ارتباطات جهانی روز به روز بیشتر می‌شود. ظهور و گسترش وسایل ارتباط که ویژگی بارز آن‌ها «سرعت» و «تنوع» روابط بود، تنها زمانی منجر به معرفی و توسعه‌ی «تجارت الکترونیک» شد که کامل‌ترین شیوه‌ی ارتباط الکترونیکی یعنی اینترنت ابداع شد. اینترنت در واقع هر دو ویژگی سرعت و تنوع را با هم ارایه می‌نمود و از سوی دیگر موجب ارزانی روابط معاملاتی نیز می‌گردید. این تحوّلات اگر چه در مدتی کمتر از یک قرن روی داد؛ با این‌حال همواره روابط الکترونیکی در معرض اختلال، تقلب، کلاهبرداری و اعمال خرابکارانه‌ی دیگر قرار داشت. امروزه رمزنگاری به همه‌ی اطلاعات الکترونیکی که از بستر

شبکه‌های کامپیوتری عبور می‌کنند، اعمال می‌شود تا در صورتی‌که یک مهاجم به شبکه نفوذ کند، نتواند از محتوای اطلاعات آگاهی پیدا کند.

اطلاعات رمزنگاری‌شده حتی در صورت انتقال از یک شبکه‌ی ناامن و حتی در صورت انتشار عمومی، امن باقی می‌مانند. در برخی سیستم‌عامل‌ها مانند یونیکس، فایلی که حاوی کلمه‌ی عبور است رمزنگاری می‌شود، به طوری که کشف آن برای مهاجمی که به صورت غیرقانونی به فایل دسترسی دارد بسیار سخت خواهد بود.

طرّاحی و پیاده‌سازی یک محیط ایمن یکی از چالش‌های اساسی در عصر حاضر محسوب می‌گردد. برای بسیاری از سازمان‌ها و موسّسات اهمیّت و ضرورت توجه جدی به مقوله امنیّت اطلاعات هنوز در هاله‌ای از ابهام قرار دارد و برخی دیگر امنیت را تا سطح یک محصول تنزل داده و فکر می‌کنند که با تهیه یک محصول نرم‌افزاری خاص و نصب آن، امنیت را برای خود به ارمغان آورده‌اند. در حالی‌که امنیت یک فرآیند است نه یک محصول!

وجود یک حفره و یا مشکل امنیتی، می‌تواند اطلاعات را به روش‌های متفاوتی تحت تاثیر قرار دهد. آشنایی با عواقب خطرناک یک حفره امنیتی و شناسایی مهمترین تهدیدات امنیتی، به طراحی و پیاده‌سازی یک مدل امنیتی کمک بسیاری می‌کند.

وقتی بافت و جوهره‌ی زندگی سنّتی در حال تبدیل به الگوهای مدرن است، بزه‌کاری‌های اجتماعی و ناهنجاری‌های مدنی نیز رنگ و بوی مدرنیته به خود می‌گیرد!

- گروه‌های اخلال‌گر برای آن‌که قدرت خود را به رخ جهانیان بکشند در سال ۱۹۹۶ به وب‌سایت-های CIA و USA DOJ (که خود از امنیتی‌ترین مراکز آمریکا به شمار می‌آیند) حمله کردند و با نفوذ در آن‌ها، چهره‌ی این وب‌سایت‌ها را تغییر دادند!

- در سال ۲۰۰۳، ویروس ها و حملات از نوع DoS (برگرفته از Denial of Service) بیشترین تبعات منفی را به ارمغان آوردند!
- در سال ۲۰۰۴، سرقت اطلاعات بالاترین جایگاه را داشته و حملات از نوع DoS با اندکی کاهش نسبت به سال ۲۰۰۳ در رتبه دوم قرار گرفته‌اند.

این‌ها حملاتی بودند که رسماً اعلام شده بودند در حالی‌که بسیاری از اخلال‌گری‌ها هرگز در جایی ثبت نمی-شود. به همین دلایل سالانه بالغ بر میلیون‌ها دلار صرف برخورد با مشکلات امنیتی در سازمان‌های دولتی و خصوصی می‌شود.

۱-۳- راهکار یا راه‌حل امنیّتی

موثرترین راهکار و یا راه‌حل امنیّتی، ایجاد یک محیط چندلایه‌ای است. در یک محیط چند لایه، مهاجمان در هر لایه شناسائی و با آنان برخورد خواهد شد. موفقیت یک مهاجم نیز به عبور موفقیت‌آمیز از هر لایه بستگی دارد. در این مدل، در هر لایه از استراتژی‌های دفاعی خاصی استفاده می‌گردد که با توجه به ماهیّت پویای امنیت اطّلاعات، می‌بایست توسط کارشناسان حرفه‌ای امنیت اطلاعات، بررسی و به‌روز گردند.

وظیفه‌ی یک شبکه‌ی کامپیوتری تنها در برقراری ارتباط دو یا چند ماشین به یکدیگر خلاصه نمی‌شود. رسالت یک شبکه‌ی کامپیوتری زمانی به تکوین می‌رسد که «برنامه‌های کاربردی» بر روی سیستم‌های نهایی[1] کاربران به اجرا درآمده و به سادگی بتوانند با همدیگر به مبادله‌ی داده بپردازند. در برقراری این ارتباط مخابراتی مطمئن بین دو پروسه‌ی کاربردی مولفه‌های سخت‌افزاری و نرم‌افزاری متعددی درگیر هستند. به دلیل پیچیدگی بسیار زیاد و گسترده‌ی این مولّفه‌ها که پیکره‌ی یک شبکه‌ی کامپیوتری را تشکیل می‌دهند، معماری یک شبکه‌ی کامپیوتری بصورت لایه‌ای طراحی می‌شود و این‌گونه است که هر لایه وظیفه‌ی مشخصی دارد و باید سرویس‌هایی خاص را به لایه‌های بالاتر ارایه دهد. برای آن‌که طراحی شبکه‌ها سلیقه‌ای و پیچیده نشود در ابتدای دهه‌ی هشتاد میلادی سازمان جهانی استانداردسازی OSI، مدلی هفت لایه‌ای را برای شبکه ارایه داد که هر لایه وظیفه‌ای دارد و در هر لایه جزییات لایه‌های زیرین نادیده گرفته می‌شود و لایه‌های بالایی باید در یک روال ساده و ماژولار از خدمات لایه‌ی زیرین خود استفاده کنند. هر چند که در شبکه‌ی اینترنت از این مدل استفاده نمی‌شود و به جای آن یک مدل چهار لایه‌ای به نام TCP/IP تعریف شده است، ولی بررسی مدل هفت لایه‌ای OSI به دلیل دقتِ ژرف، در تفکیک مسئله‌ی پیش‌رو در این پروژه با ارزش خواهد بود. در صورت نیاز به مشروح این مضامین به کتب مرجع شبکه‌های کامپیوتری مراجعه کنید.

---

[1] End System

همان‌گونه که معماری شبکه از چندین لایه‌ی مستقل با وظایف و خدمات مشخص تشکیل‌شده و در سرویس-دهی به کاربر نهایی صدها مولفه‌ی ریز و درشتِ سخت‌افزاری و نرم‌افزاری دخالت دارند، تضمین امنیّت اطلاعات نیز در یک نقطه‌ی خاص متمرکز نیست و در هر لایه و مولّفه باید تمهیدات لازم پیش‌بینی و رعایت شود. این تهمیدات از لایه‌های پایین شروع و در آخرین لایه یعنی لایه‌ی کاربرد به تکامل می‌رسد! لایه‌ی فیزیکی که پایین‌ترین سطح در معماری شبکه است و فقط بیت‌های داده در این لایه به رسمیت شناخته می‌شوند، برای ایجاد امنیت در مقابل اخلال-گران با حفاظت از رسانه‌های مادّی (مثل کابل‌های مسی یا فیبرنوری) چاره اندیشیده شده است. لایه‌ی پیوند داده با استفاده از مکانیزم‌های کشف و کنترل خطا اطّلاعات را در مقابل خطاهای اطّلاعاتی بیمه می‌کند. در این لایه برای حفظ امنیت داده‌ها در حین عبور از یک لینک می‌توان بدنه‌ی هر فریم را با استفاده از رمزنگاری (در سطح سخت-افزارِ کارت شبکه) که در این پروژه به توضیح یکی از روش‌های آن پرداخته خواهد شد، مخفی نگاه‌داشت. «رمزنگاری جریان بیت‌های ارسالی» در شبکه‌های بی‌سیم یکی از نیازهای اولیه به شمار می‌آید چرا که برای استراق سمع از روی لینک نفوذگر می‌تواند با یک کامپیوتر کیفی در حوزه‌ی پوشش رادیویی شبکه‌ی بی‌سیم حاضر شده و با ظاهری معمولی و بدون هیچ تلاش فیزیکی، داده‌ها را استراق سمع و تصرف کند. بدیهی است که رمزنگاری در سطح لایه‌ی پیوند داده فقط امنیت را بر روی یک لینک واحد تضمین می‌کند. در حالی که گاهی در شبکه‌های گسترده، برای رسیدن یک واحد اطلاعات از مبدا به مقصد، چندین لینک وجود دارد؛ پس در هر مسیریاب امکان نقب‌زدن به اطّلاعات وجود خواهد داشت و نمی‌توان اعتماد کرد. در نتیجه رمزنگاری انتها به انتها کافی نیست و برای رسیدن به امنیت بیشتر باید به رمزنگاری لینک link encryption با استفاده از سرآیندهای هر بسته که حاوی اطلاعات موجود در هر بسته‌ی یک پیام است متوسل شد. به طور خلاصه، وقتی هر دو عملِ رمزنگاری لینک و رمزنگاری انتهابه‌انتها به کار گرفته‌شدند، میزبان بخش داده‌ی کاربر را با استفاده از رمزگذاری انتها به انتها رمزگذاری می‌کند. سپس کل بسته با کلید رمزگذاری ارتباط، رمزگذاری می‌شود. با حرکت بسته در شبکه، هر سوییچ بسته را با کلید رمزگذاری ارتباط رمزگشایی می‌کند تا سرآیند را بخواند و سپس کلّ بسته را برای ارسال در لینک بعدی رمزگذاری می‌نماید. اینک تمام بسته به جز در زمانی‌که در حافظه‌ی سوییچ است، بسته امنیت دارد، و در این مدّت هم در واقع سرآیند بسته رمزدار نیست. برای ایجاد امنیت بیشتر در شبکه می‌توان در لایه‌های شبکه و انتقال از پروتکل‌هایی نظیر IPSec, SSL, TSL,... استفاده کرد.

۱-۴- امنیت در شبکه‌ی بی‌سیم WiFi (امنیت در لایه‌ی پیوند داده)

یک از چالش‌های بزرگ در شبکه‌های بی‌سیم، امنیت اطّلاعات است. در شبکه‌های سیمی مثل اترنت برای استراق سمع داده‌های جاری بر روی رسانه‌ی انتقال، دسترسی فیزیکی به کانال الزامی است (از طریق وصل کانکتور) وهمین عامل تا حدود اندکی اخلال‌گران را محدود می‌کند و مسئولین شبکه می‌توانند با تمهیدات «امنیت فیزیکی»، احتمال استراق‌سمع مستقیم داده‌ها از روی کانال را به حداقل برسانند. در شبکه‌ی بی‌سیم این اطمینانِ کم نیز از دست می‌رود و یک اخلال‌گر می‌تواند با کامپیوتر کیفی، پیرامون یک سلول شبکه پرسه زده و داده‌های در حال ارسال بر روی کانال را استراق‌سمع نماید. حتی گاهی او مجبور نخواهد بود که به حریم سازمان یا موسّسه وارد شود بلکه استقرار در نزدیکی یک سلول رادیویی برای او کافی است. برای تامین حداقل امنیت در شبکه‌ی بی‌سیم، کمیته‌ی IEEE پروتکلی به نام WEP(Privacy Wired Equivalent) را معرفی کرد که هدف آن محرمانه نگاه‌داشتن اطّلاعات در سطحی معادل با شبکه‌های مبتنی بر سیم، مثل اترنت، بوده‌است. پروتکل WEP در سال ۲۰۰۱ دردسر ساز شد، چرا که با کشف اشکالات متعدد و حفره‌های امنیتی در آن، پیرامون آن جنجال مطبوعاتی براه افتاد و این پروتکل به مضحکه تبدیل شد. مجموعه‌ی این اشکالات بود که IEEE را وادار کرد تا در استاندارد جدید از روش رمزنگاری جدید AES (که در حال حاضر قدرتمندترین روش دنیاست و در این پروژه به پیاده‌سازی آن پرداخته خواهد شد) بهره بگیرد [۱].

در واقع برای ایجاد خدمات امنیتی در لایه‌ی پیوند داده، استفاده از الگوریتم‌های رمزنگاری به معنای تبدیل پیام‌ها به دنباله‌ای نامفهوم از بیت‌ها است، به نحوی که فقط و فقط گیرنده بتواند این دنباله‌ی نامفهوم را به شکل اصلی بازیابی کند لازم است. چنین الگوریتم‌هایی، «سرویس محرمانه ماندن اطلاعات[1]» را عرضه می‌کنند و با روش‌های متقارن رمزنگاری فراهم می‌شوند. دومین گام در ارائه‌ی خدمات امنیتی، توسّل به روش‌هایی است که گیرنده و فرستنده پیام بتوانند هویت و اصالت یکدیگر را به درستی تشخیص داده و امکان جعل یا دستکاری پیام‌ها منتفی شود. این روش‌ها، ایده‌های امضاهای دیجیتال و کدها و مکانیزم‌های احراز اصالت پیام هستند.

همان‌گونه که در فصل دوم توضیح داده‌خواهد شد، الگوریتم‌های رمزنگاری به دو قسم متقارن و نامتقارن تقسیم می‌شوند. در این پروژه از برترین الگوریتم رمزنگاری متقارن یعنی راین‌دال[2]- ۱۲۸ استفاده خواهد شد. این

---

[1] confidentiality
[2] Rijndael

الگوریتم از امنیت بسیار بالایی برخوردار است و از دیدگاه سرعت و سادگی پیاده‌سازی، فضای حافظه‌ی مورد نیاز و قابلیت انعطاف، روش شگفت‌انگیزی است. این الگوریتم که توسط دو جوان بلژیکی پیاده‌سازی شده بود، در طی رقابتی چند مرحله‌ای که در ژانویه‌ی ۱۹۹۷ شروع شد و تا نوامبر۲۰۰۱ ادامه داشت، پیروز این رقابت شد و استقبال عمومی از این روش به حدی چشمگیر بود که پیاده‌سازی سخت‌افزاری و نرم‌افزاری آن در انواع تکنولوژی‌های مرتبط با امنیت اطلاعات جایگاه ویژه‌ای پیدا کرده است و این به دلیل سرعت بسیار بالایش است. این الگوریتم مبتنی-بر روش رمزنگاری «شبکه‌ی جانشینی‌وجایگشتی[۱]» است. برخلاف الگوریتم‌های قبل از خود مانند استاندارد رمزنگاری داده[۲] از شبکه‌ی فیستل[۳] استفاده نمی‌کند. پیش‌زمینه‌های ریاضی این الگوریتم بسیار مفصّل بوده و خود به چندین کتاب نیاز دارد ولی به طور مختصر می‌توان گفت آن‌چه این الگوریتم را رازآلود کرده است و باعث افزایش امنیت آن شده است، عمل تلفیق ستونی است که ضرب هر ستون چهار بایتی در یک ماتریس ثابت است و این یک ضرب معمولی نیست و بر روی میدان $GF(2^8)$ که یک «محدود گالوا» است انجام می‌گیرد.

تا سال ۲۰۰۹، تنها حمله‌های منتشرشده‌ی موفّق علیه AES حمله‌های سمتِ کانال[۴] بودند که مبتنی بر پیاده-سازی‌های فیزیکی سیستم رمز بودند، نه حملات با زور خشن[۵] و نه نقص تئوری الگوریتم، به عنوان مثال رخنه‌های الکترومغناطیسی، مصرف توان و انرژی و...که منجر به سوءاستفاده و شکستن سیستم رمز می‌شوند. سازمان امنیت ملّی[۶] همه‌ی فینالیست‌های AES را از جمله راین‌دال بررسی کرد و اعلام کرد که همگی برای حفاظت از داده‌های غیرطبقه‌بندی شده‌ی دولت آمریکا امن هستند [۱].

سرعت بالا و نیاز به حافظه‌ی دستیابی تصادفی[۷] کم، از معیارهای پردازش AES است که باعث شده‌اند، در انواع مختلف سخت‌افزار از کارت‌های هوشمند ۸ بیتی تا رایانه‌هایی با کارایی بالا به خوبی عمل کنند. همچنین پردازنده‌های Intel i5/i7 از این الگوریتم البته با مجموعه دستورالعمل‌های جدید[۸] نیز استفاده کرده‌اند [۴].

---

[1] Substitution & Permutation network
[2] DES
[3] Feistel network
[4] Side-channel attack
[5] Brute force attack
[6] National Security Agency(NSA)
[7] RAM
[8] AES-NI (AES - new instruction)

در این پروژه برای مقایسه از الگوریتم رمزنگاری سِرپِنت، که در رقابت دو مرحله‌ای NIST برای طراحی استاندارد جدید برگزار شد و رتبه‌ی دوم را کسب کرد، برای مقایسه با نتایج الگوریتم معروف AES استفاده خواهد شد. اگر چه این الگوریتم در برابر راین‌دال شکست خورد و از دریافت لقب استاندارد پیشرفته‌ی رمزنگاری AES بازماند، ولی این نکته را نباید فراموش کردکه سِرپِنت با سرعتی نزدیک به استاندارد رمزنگاری داده DES، امنیتی بسیار بالاتر از 3-DES فراهم می‌کند ولی هرگز به زیبایی و سرعت راین‌دال نیست. این الگوریتم رمز قطعه‌ای ۱۲۸ بیتی، رمزنگاری با کلیدهایی با طول‌های ۱۲۸، ۱۹۲، ۲۵۶ بیتی را نیز همانند راین‌دال حمایت می‌کند. این نوع از رمزنگاری یک شبکه‌ی جانشینی و جایگشتی است که از ۳۲ دور تشکیل شده که روی چهار عدد بلوک ۳۲ بیتی عمل می‌کند.

۳۲ دور متوالی به این معنی است که سِرپنت حاشیه‌های امنیت بیشتری نسبت به راین‌دال دارد، امّا راین‌دال با ده دور متوالی برای رمزنگاری سریع‌تر است و برای پیاده‌سازی بلوک‌های کوچک مناسب است. از این رو به عنوان برنده در رقابت AES انتخاب شده بود [۴].

روش‌های رمزنگاری متقارن مثل AES, 3-DES, Serpent و نظایر آن همگی بر روی یک قطعه داده‌ی کوچک (به طور معمول به طول ۱۲۸ بیت معادل ۱۶ بایت) عمل می‌کنند. حاصل رمزنگاری داده‌ها نیز بلوکی با همان اندازه است که باید جایگزین متن اصلی شده و ارسال گردد. بدیهی است که هرگاه بلوک‌های ورودی به سیستم رمزنگاری مشابه باشند نتیجه‌ی یکسانی به دست می‌آید. به عنوان مثال هرگاه بلوک m_i در متن اصلی صدها بار تکرار شده باشد، بلوک جایگزین آن در متن رمزنگاری شده نیز صدها بار مشاهده خواهد شد. آیا برای حل این مشکل راه‌حلی اندیشیده شده است؟! آیا می‌توان به گونه‌ای عمل کرد که اخلال‌گر با هر تغییر در فیلدهای متنِ رمزشده از آن نقطه به بعد کلّ اطلاعات آلوده و بی‌ارزش شوند و البته تغییرات قابل تشخیص باشند؟! آیا شیوه‌ای وجود دارد که در عین ارایه‌ی امنیت بتوان بصورت پردازش موازی قطعه‌ها، اطّلاعات را رمزنگاری کرد؟! آیا شیوه-ای از رمزنگاری وجود دارد که بتوان در کنار ایجاد سرویس محرمانگی اطّلاعات سرویس احراز اصالت و سلامت پیام را نیز داشته باشد؟! آیا شیوه‌ای که سرویس احراز اصالت نیز ارایه می‌دهد، سرعت و هزینه‌ی قابل قبولی دارد یا خیر؟!

در ساده‌ترین شیوه‌ی رمزنگاری که کتابچه‌ی رمز یا ECB[1] می‌باشد، با وجود سادگی پیاده‌سازی، ولی قطعات یکسان به طور مشابه رمز خواهند شد و اخلال‌گر توانایی نفوذ در آن و یا تغییر برخی بلوک‌ها را دارد و این می‌تواند باعث تغییرات عمده‌ای در سیستم اطلاعاتی شود، مثلاً فیلد حقوقی دو نفر را به صورت تصادفی تغییر دهد.

در شیوه‌ی زنجیره‌سازی بلوک‌های رمز یا CBC[2] هر قطعه‌ی رمزشده تاثیرش را بر بلوک‌های بعدی می‌گذارد و این‌گونه دیگر بلوک‌های یکسان به صورت مشابه رمز نمی‌شوند. هر تغییر در هر یک از قطعه‌ها بر روی قطعات دیگر تاثیر می‌گذارد. این شیوه دیگر قابلیت موازی‌سازی را ندارد و نمی‌توان هر بلوک را جداگانه رمزنگاری یا رمزگشایی کرد و یا همه‌ی بلوک‌ها را به صورت موازی پردازش‌کرد چون به یکدیگر وابسته‌اند. از این مد برای احراز اصالت پیام و تولید کد احراز هویت نیز استفاده می‌شود.

همان‌گونه که گفته خواهد شد می‌توان از شیوه‌های فیدبک برای رمزکردن استفاده کرد؛ مثلاً هر کاراکتر را به صورت جداگانه رمز کرد و ارسال کرد، و دیگر نیازی نیست طول پیام را به مضرب صحیحی از الگوریتم رمزنگاری تبدیل کرد. مد فیدبک یا CFB[3] نیز مانند شیوه‌ی زنجیره‌سازی بلوک رمز با ارتباط دادن هر بلوک به بلوک بعدی خود عمل می‌کند و به این دلیل است که اگر خطایی در یک بلوک ۸ بیتی ایجاد شود این خطا در یک بلوک ۶۴ بیتی منتشر خواهد شد. ولی در شیوه‌ی فیدبک خروجی این‌طور نیست زیرا بلوک رمز شده نیست که در بلوک بعد از خود تاثیر می‌گذارد و این مزیت موجب می‌شود خطا در یک بیت دیگر در بلوک مورد نظر منتشر نشود.

در شیوه‌ی فیدبک خروجی یا OFB[4]، رمزنگاری مانند رمزگشایی است و در صورت داشتن حافظه‌ی کافی می‌توان از قبل با استفاده از مقدار اولیه تصادفی دنباله‌کلیدها را بدست آورد تا هر موقع اطّلاعات آماده شدند با خروجی رمزنگاری، XOR کرده و متن رمز را ارسال کرد. تنها نقص این روش اینست که اگر برای پیام‌ها از یک مقدار تصادفی اولیه استفاده کرد و نفوذگر می‌تواند با کمک ویژگی‌های آماری متون به آن‌ها حمله و ساده‌تر آن‌ها را آشکار کند و حتی اگر یکی از پیام‌ها را داشته باشد می‌تواند دیگری را بدست آورد.

---

[1] Electronic CodeBook
[2] Cipher Block Chaining
[3] Cipher Feedback
[4] Output Feedback

شیوه‌ی رمز بلوکی شمارنده، که به اختصار آن‌را CTR[1] نامند، یکی از قدیمی‌ترین شیوه‌های موجود است و حتی قبل از الگوریتم رمز بلوکی DES نیز در دسترس بوده است. این شیوه که توسط سازمان ملّی استانداردها و تکنولوژی، استاندارد و به ثبت رسیده است سرعتی بسیار بالا دارد و در عین حال امنیتی قابل قبول نسبت به بقیه‌ی مدها دارد. همچنین قابلیت دسترسی تصادفی به بلوک‌های رمزشده‌ی داده را داراست. اگر نیاز به پیش‌پردازش خروجی الگوریتم رمزنگار باشد این شیوه این امکان را در اختیارتان قرار می‌دهد تا بعداً با قطعه‌های داده XOR شوند. همچنین مزیت پردازش موازی داده و پایپ‌لاین را این شیوه داراست.

حالت‌های تعریف‌شده در این استانداردها، تنها محرمانه‌بودن داده را تأمین می‌کنند. محافظت از تمامیت و سلامت داده، در دامنه‌ی کاربرد این استاندارد قرار نمی‌گیرد. همچنین در اکثر حالت‌ها، محرمانه بودن اطلاعات مربوط به طول پیام، محافظت نمی‌شوند.

در شیوه‌هایی که در طول دهه‌ی اخیر ابداع شده‌اند هم سرویس محرمانگی داده ارایه شده و هم سرویس احراز سلامت پیام و هویت شخص فرستنده‌ی داده. یک دلیل برای طرّاحی این شیوه‌ها اینست که این‌چنین طرح‌هایی که رمزنگاری با احراز اصالت را به طور همزمان و یک‌جا ارایه می‌دهند، کارایی و عملکرد بهتری نسبت به ادغام سرراست تکنیک‌های جداگانه‌ی رمزنگاری و احراز هویت دارند. دلیل دوم اینست که یک طرح رمزنگاری با احراز اصالت[2] کم‌تر نسبت به طرح رمزنگاری برای داشتن فقط محرمانگی، محتمل بر استفاده‌ی نادرست است.

در طرحِ این‌گونه از شیوه‌های AE[2]، دو پیشنهاد وجود دارد: یکی طرح‌های دو گذری یا ترکیبی که به طور جداگانه شیوه‌های رمزنگاری و احراز اصالت را ادغام می‌کنند، برای مثال ممکنست شیوه‌ی شمارنده برای رمزنگاری پیام و سپس شیوه‌ی CBC-MAC برای احراز اصالت به کار رود. متناوباً، در طرح‌های یک گذری یا جامعِ AE، قسمت‌های فراهم‌کننده‌ی رمزنگاری و احراز اصالت پیام به شدت به هم وابسته هستند. این قبیل مدها در حدود دهه‌ی قبل با فعالیت‌های Jutla، Katz، Yung، ... آشکار شد [۱۱].

این طرح‌های AE جامع برای افزایش کارایی نوع ترکیبی اختراع شده‌اند. در تنها مطالعه‌ی مقایسه‌ای McGrew و Viega که در سال ۲۰۰۴ به این نتیجه رسیدند که طرح ترکیبی خودشان GCM به سرعتی و حتی بیشتر از سرعت طرح جامع OCB (البته این مقایسه با OCB1 بود که تا آن موقع به ثبت رسیده بود و بعدها این مد

[1] Counter Mode
[2] AE(Authentication-Encryption)

توسعه‌داده شد). بعد از این مطالعه و تحقیق هیچ بررسی دیگری منتشر نشد و این متاسّفانه نشان‌دهنده‌ی وجود مشکلی در بررسیشان بود. پیاده‌سازیهای مربوط با نوع بهینه‌شان مقایسه شدند و هیچ‌کدام از نتایج هیچ‌وقت به خاطر استفاده از مالکیت کدهای اختصاصی تکرار نشدند. در ضمن CCM و GCM بسیار مورد توجّه واقع شدند و به عنوان مثال CCM در امنیت مدرن WiFi (802.11i) و مد GCM در IPSec و TLS مورد حمایت قرار گرفتند [۱۱].

در ادامه، ضمن بررسی تحقیقاتی که Rogaway و Krovetz در سال ۲۰۱۱ انجام دادند و در این پروژه از تحلیل‌هایشان استفاده‌شده است از لحاظ عملکرد نرم‌افزاری و سخت‌افزاری این چند مد با طول‌های پیام‌های مختلف و در بسترهای سخت‌افزاری متفاوت بررسی شدند و نتایج قابل توجهی بدست‌آمد که به ذکرشان پرداخته خواهد شد. مثلاً یکی از نتایج اینست که درپردازنده‌ی Intel i5 شیوه‌های CCM و GCM و CTR به ترتیب در 4.2cpb، 3.7cpb، 1.5cpb رمز می‌کنند، درحالی‌که روش CTR در حدود 1.3 کلاک بر بایت زمان نیاز دارد. سپس طبق نتایجی که بدست خواهد آمد روش OCB با قابلیت رمزنگاری و احراز هویت سریع‌تر از بقیه‌ی روش‌هاست.

۱-۵- ساختار پایان‌نامه

در این پایان‌نامه همانطور که در فصل اول مطالعه کردید مختصری راجع به اهداف پروژه و برخی از مشکلات موجود و دلایل انتخاب آن صحبت شد.

فصل دوم فصل توصیف کلیات و مفاهیم است. در این فصل به معرفی تاریخچه‌ی مختصری از رمزنگاری نیز پرداخته خواهدشد. . سپس مفاهیم رمزنگاری و الگوریتم‌های مختلف رمزنگاری توضیح داده‌خواهد شد. به اصول اساسی در رمزنگاری پیام‌ها نیز اشاره می‌شود.

در فصل سوم به دلیل استفاده از تصویر به عنوان پیام یا متن آشکار در ورودی سیستم رمزنگار در این پروژه، به ویژگی‌ها و ساختار تصاویر مختلف اشاره‌ای خواهد شد. همچنین به بررسی ویژگی‌های بنیادین سیستم‌های رمزنگاری متقارن پرداخته خواهد شد

سپس در فصل چهارم به توصیف الگوریتم نه چندان امن DES و بعد از آن دو الگوریتم برتر رمزنگاری متقارن قطعه‌ای یعنی استاندارد پیشرفته‌ی رمزنگاری و سِـرپِـنت پرداخته خواهدشد. سپس به ترتیب مدها و شیوه‌هایی

که برای رمزنگاری داده‌ها با طول‌های مختلف و طبیعتاً بیشتر از ۱۲۸ بیت استفاده می‌شوند توضیح داده و بررسی می‌شوند تا به روشی دست یافته‌شود که هم ویژگی محرمانگی و هم احراز هویت تصویر را به همراه هم فراهم کند.

در فصل پنجم پیاده‌سازی‌های کلّی در نرم‌افزار متلب که دارای کتاب‌خانه‌ی قدرتمند پردازش تصویر است آورده شده‌اند. تمام این پیاده‌سازی‌ها به دو صورت یکی برای تصویر خاکستری که در واقع تصویری دو بعدی است و دیگری برای تصویر رنگی که از سه مولفه‌ی قرمز و سبز و آبی تشکیل می‌شود، می‌باشند. به‌دلیل حجم زیاد کد برنامه‌ها سعی شده فقط قسمت‌های ضروری آن‌ها در این فصل آورده‌شود.

فصل ششم که فصل مقایسه‌هاست و چند مولفه از هر تصویر رمزشده توسط دو الگوریتم رمز قطعه‌ای، اندازه‌گیری و با استفاده از این نتایج، این شیوه‌ها با هم مقایسه خواهند شد.

در آخر در فصل هفتم نتیجه‌گیری کلی این پروژه و پیشنهادات آتی برای دانشجویان علاقه‌مند به فعالیت در این زمینه آورده خواهد شد.

۱-۶- نتیجه‌گیری

تحوّلات موجود در دنیای کامپیوتر و فناوری‌اطلاعات اگر چه در مدتی کمتر از یک قرن روی داد؛ با این‌حال همواره روابط الکترونیکی در معرض اختلال، تقلب، کلاهبرداری و اعمال خرابکارانه‌ی دیگر قرار داشت. امروزه رمزنگاری به همه‌ی اطلاعات الکترونیکی که از بستر شبکه‌های کامپیوتری عبور می‌کنند، اعمال می‌شود تا در صورتی‌که یک مهاجم به شبکه نفوذ کند، نتواند از محتوای اطلاعات آگاهی پیدا کند. لذا در این فصل ابتدا با دلایل و ضرورت برقراری امنیت صحبت شد و به تاریخچه‌ی برخی از حملات امنیتی اشاره شد. سپس به شرح کلّی مشکلات موجود و دلایل توجه به این موضوع پروژه اشاره شد تا در جهت رفع مشکل آن گام برداشته شود. در ادامه ضمن اشاره به تاریخچه و مفاهیم رمزنگاری با ساختار برقراری تعدادی از الگوریتم‌های رمزنگاری آشنا خواهید شد.

فصل دوم

تاریخچه‌ی رمزنگاری و مفاهیم آن

۲-۱- مقدمه

تلاش برای انتقال «احساسات درونی»، «یافته‌های فردی»، «اعلام موجودیت» و «میل به جاودانگی» در بطن روان انسان، با ابداع «نماد[1]» و در پی آن «رسم‌الخط» به نتیجه رسید. «نمادها» و «الفبا» را می‌توان گونه‌ای از رمزنگاری احساسات و دانسته‌های بشر دانست. تا وقتی که شما با الفبای یک رسم‌الخط یا نمادهای به‌کار رفته در یک متن آشنا نباشید با متنی مرموز و گنگ مواجه هستید که برای فهم آن باید پرده از رمز و راز متن بردارید! برای این کار به «کلید» رمز نیاز خواهد بود و این کلید چیزی نیست جز آن‌که معنای هر نماد و اطلاعاتی را که در خود حمل می‌کند بتوان بفهمید. به عنوان مثال شکل یک پرستو بر روی دیوار یک غار فقط نماد یک پرنده نیست بلکه می‌تواند اشاره به تغییر فصل و یا مهاجرت نیز داشته باشد. بنابراین وقتی ادعا می‌شود «رمزنگاری» و «خط» قدمتی یکسان دارند موصوف این حقیقت است که رسم‌الخط نیز گونه‌ای از رمزنگاری مکنونات درونی و یافته‌های فردی انسان‌هاست با این تفاوت که هدف آن نشر این یافته‌هاست نه پنهان نگاه داشتن آن‌ها؛ لذا ریشه و بنیان «خطّ» و «رمزنگاری» یکی است ولی با اهداف متضاد متجلی شده‌اند!

۲-۲- مروری بر تاریخ ۵۵۰۰ ساله‌ی رمزنگاری

در ادامه مروری خواهد شد بر تاریخچه‌ای که پیشینه‌ی ۵۵۰۰ ساله دارد:

- ۳۵۰۰ سال قبل از میلاد: **خط میخی** ابداع سومری‌ها و قدیمی‌ترین رسم‌الخط شناخته شده در تاریخ بشر است که تا قرن نوزدهم میلادی کسی از مضمون آن چیزی نمی‌دانست. نهایتاً در این قرن دانشمندان زبان‌های باستانی، موفق به رمزگشایی آن شدند. اگرچه در همین قرون کاربرد ترسیم اَشکال در انتقال مضامین (مشهور به ideograph یا ترسیم افکار) در تمدن‌های مختلف مرسوم بوده است ولی خط میخی را می‌توان نخستین «رسم‌الخط» باستانی به

[1] Symbol

حساب آورد. در مصر نیز همین حدود از پهنه‌ی تاریخ، «خطّ هیروگلیف» ابداع شده است. هیروگلیف مرموزتر و دشوارتر از خطّ میخی بود و اعتقاد بر این بود که مرموز و مبهم بودن این رسم‌الخطّ دلایل مذهبی و تاریخی دارد و عمدتاً افکار کاهنان و فراعنه را منتقل می‌کرده است ولی اکنون پژوهش‌ها نشان می‌دهد که ابداع این رسم‌الخط بیشتر جنبه‌ی اقتصادی و بازرگانی داشته است [۱].

- ۱۵۰۰ سال قبل از میلاد: در بین‌النهرین از رمزنگاری برای مخفی نگاه‌داشتن فرمول ساخت ظروف سفالی استفاده شده است. تحقیقات نشان می‌دهد که در لوح رمزنگاری شده‌ی این فرمول‌ها از حروف میخی که کمترین کاربرد و پایین‌ترین تکرار را در متون میخی رایج داشته، بهره گرفته‌است [۱].
- سال‌ها قبل از میلاد: ژولیوس سزار با ابداع روشی بسیار ساده (مبتنی‌بر جانشینی کاراکترها) به ارسال رمزنگاری‌شده‌ی پیام برای فرماندهان سپاه خود، رسمیت بخشید. این روش در عین سادگی اولین الگوی رمزنگاری ثبت‌شده در تاریخ به شمار می‌آید. در این الگو هر حرف از متن با حرفی که در جدول الفبا به فاصله‌ی k حرف فاصله داشت جانشین می‌شد. مثلاً با مقدار k برابر سه یعنی اینکه هر حرف از متن با حرفی که در جدول الفبا به فاصله‌ی سه حرف بعد آن قرار گرفته جانشین شود [۱].
- ۱۵۸۷ پس از میلاد: ویگنر: این روش رمزنگاری، روشی چندحرفی است، بدین معنی که به جای آن‌که ارتباطی یک‌به‌یک بین هر حرف و جانشین آن تعریف شود هر حرف می‌تواند با یکی از چندین حرف تعریف شده جایگزین شود. در رمز ویگنر باید کلیدی انتخاب و برای آن‌که طول کلید هم‌اندازه‌ی متن شود به تعداد مورد نیاز تکرار شود [۱].
- ۱۵۸۷ پس از میلاد: ملکه‌ی اسکاتلند وقتی متنی رمزنگاری‌شده به روش «تک‌حرفی» یا همان جایگزین یک حرف الفباء با حرفی دیگر را علیه ملکه الیزابت نوشت نمی‌دانست که «توماس فلیپس» این رمز را خواهد شکست و پرده از توطئه‌ی ملکه اسکاتلند (ملکه ماری) برخواهد افکند. شکسته شدن رمزنامه‌ی ملکه ماری که برای خیانت به ملکه الیزابت و دستگاه پادشاهی او و براندازی تاج و تختش نوشته‌بود سرش را به باد داد چرا که به دستور ملکه الیزابت فوراً گردن‌زده شد و دنیای رمزشکنی اولین قربانی خود را در تاریخ ثبت کرد [۱]!
- ۱۷۵۳ و اختراع تلگراف: با اختراع تلگراف هیچ چاره‌ای باقی نمی‌ماند مگر آن‌که حروف به نحوی با سیگنال‌های الکترواستاتیکی جایگزین شود، لذا باید نوعی از رمزگذاری بکار گرفته می‌شد تا بتوان حروف ۲۶ گانه‌ی جدول الفباء را از طریق خطوط تلگراف منتقل کرد [۱].

• ۱۸۴۵ و کد مورس: ساموئل مورس، کُد مورس را که بیش از یک قرن کاربرد داشت، ابداع کرد. کُد مورس با این هدف صورت گرفت که هر حرف الفباء با دنباله‌ای از علائم الکتریکی (شامل پنج نماد مختلف) کُد شده و بر روی یک رشته سیم واحد ارسال شود. بدیهی است که در سمت گیرنده بایستی «کُد مورس» توسط یک متخصص، بازخوانی و رمزگشایی می‌شد [۱].

• ۱۸۶۳ و شکستن رمز ویگنر توسط کاسیسکی: کاسیسکی در این سال روشی را برای شکستن رمز ویگنر ارائه کرد که مبتنی بر یافتن طول کلمه‌ی کلید بود؛ سپس متن رمزشده به قطعاتی متناسب با طول کلید تقسیم می‌شد. این قطعات در حقیقت طبق جدول ویگنر با مقادیر مشخصی جانشین شده‌بودند لذا طبق پیشنهاد کاسیسکی ابتدا این قطعات مورد مطالعه‌ی آماری قرار می‌گرفتند تا مشخص شود چه قطعاتی بیشترین تکرار را دارند. سپس این قطعات با تحلیلِ فراوانی تکرار حروف، تحلیل و رمزشکنی می‌شد تا کلید رمز و نهایت متن اصلی بدست آید.

• ۱۸۸۳ آگوست کرکهف: وی به خاطر دو مقاله‌ی بسیار مهم که در سال ۱۸۸۳ در ژُرنال «علوم نظامی» منتشر کرد در دنیای رمزنگاری مدرن شأن والایی یافته است. اصول شش‌گانه‌ای که به نام خود او در تاریخ ثبت‌شده زیربنای روش‌های رمزنگاری مدرن قرار گرفت [۱].

• تلگرام زیمرمن و جنجال ۱۹۱۷: در این سال تلگرامی توسط زیمرمن رمزنگاری و مخابره شد که به دست حکومت بریتانیا افتاد و رمزشکنی شد. در این تلگرام زیمرمن پیشنهاد داده بود که آلمان‌ها با مکزیک متحد شوند و با پیشنهاد کمک‌های مالی، سران مکزیک را به تحریک علیه آمریکا و اتحاد با آلمان ترغیب کنند. رمزگشایی این بهانه‌ای بود که ایالات متحده نیز در جنگ جهانی اول به صفّ مخالفان آلمان بپیوندد. درخلال جنگ جهانی اول ارتش آلمان از روشی موسوم به ADFGVX برای رمزنگاری پیام‌ها استفاده می‌کرد [۱].

• ماشین انیگما در ۱۹۱۸: «آرتور شریبورس» دستگاهی مکانیکی (به نام انیگما) برای رمزنگاری اسناد محرمانه عرضه کرد که براساس چرخش یک سری روتور هم‌محور و یک سری اتصالات الکتریکی ساخته شده‌بود. بر روی روتورها حروف A تا Z حکّ شده‌بودند و هر روتور به ۲۶ اتصال الکتریکی مجهز بود. به ازای هر حرف که بر روی این ماشینِ تایپ فشار داده‌می‌شد، حرفی در خروجی چاپ می‌شد که به موقعیت فعلی روتور و وضعیت کنونی اتصالات وابسته بود. نازی‌های قبل از جنگ جهانی دوم از مشتریان پروپاقرص این ماشین بودند [۱].

● ۱۹۳۷ تا ۱۹۴۵ کُد ناواجو[1]: کُد ناواجو مشهورترین روش ارسال پیام‌های سّری در قالب کُدها و کلمات نامفهوم و مبهم بود که در خلال جنگ جهانی دوم توسّط ارتش ایالات متحده بکار گرفته‌شد. این کُد از یک لهجه‌ی محلّی سرخ‌پوستی با نام Navajo الهام گرفته‌شده‌بود و طبعاً این لهجه‌ی آهنگین، پیچیده و سریع که در یک اجتماع بسیار کوچک تکلّم می‌شد در آن ایام هیچ مترجمی نداشت. در خلال جنگ جهانی دوم در کلّ دنیا شمار افراد غیربومی که این زبان را می‌فهمیدند از ۳۰ نفر فراتر نمی‌رفت و تمام این افراد نیز در آژانس امنیت ملی آمریکا خدمت می‌کردند! ایالات متحده‌ی آمریکا اعتقاد دارد که این افراد گمنام نقش و شأن بزرگی در نجات جان انسان‌ها و خاتمه‌ی جنگ داشته‌اند. در سال ۱۹۶۸ که کُد ناواجو از رده‌بندی «فوق‌سرّی» خارج و معرفی شد به پاس خدمات دست‌اندرکاران این گروه در مراسم یادبودی از آن‌ها تقدیر به عمل آمد. اکنون بسیاری ازآن‌ها در قید حیات نیستند. دلیل این قدردانی آن بود که ژاپنی‌ها هرگز نتوانستند پیام‌هایی را که بر فراز اقیانوس آرام تاکتیک‌های جنگی ایالات متّحده را دیکته می‌کردند، بفهمند و رمز آن‌را بشکنند در حالی‌که متفقین موفق به شکستن رمز پیام‌های ژاپنی‌ها شده-بودند. در رمز ناواجو ادوات جنگی با استعارات سرخ‌پوستی و نام جانوران جابه‌جا شده‌بود و سرعت تلفظ کلمات در این زبان باعث می‌شد مخابره‌ی پیام‌ها از طریق بی‌سیم بسیار سریع‌تر از روش سنتی کُد مورس انجام شود [۱].

● ۱۹۴۹ کلود شانون: آقای کلود شانون با انتشار مقاله‌ی «مفهوم اتحاد[2]»، رابطه‌ی مقدار حداقلِ اطلاعات رمزنشده‌ای را که می‌توان از معادل آن استخراج کرد (با فرض دسترسی نفوذگر به منابع بی‌پایان مثل زمان و امکانات)، پیشنهاد کرد. او با پایه‌گذاری تئوری اطلاعات، کمک بزرگی به علم سایبرنتیک کرد که دانش رمزنگاری نیز از آن بهره‌ی بی‌کران برد؛ رساله‌ی کارشناسی‌ارشد آقای شانون بهترین پروژه‌ی قرن شناخته شده‌است [۱].

● ۱۹۷۱ هارست فیستل: آقای هارست فیستل سیستم رمزنگاری متقارن خود، لوسیفر را در IBM ابداع و تکمیل کرد. این ابداع بعداً پایه‌ی روش رمزنگاری DES شدکه حدود ۲۰ سال استاندارد دولت فدرال آمریکا بود. با معرفی تحقیقات آقای فیستل در IBM دامنه‌ی تحقیقات رمزنگاری به دانشگاه‌ها و مراکز تحقیقات غیرنظامی کشیده شد و توانست جای خود را به عنوان شاخه‌ای از دانش، در اذهان باز کند چرا که تا قبل از این زمان، تحقیقات رمزنگاری با هاله‌ای از بدبینی (به دلیل اتصال آن با مراکز جاسوسی و اطلاعاتی) مواجه بود [۱].

---

[1] Navajo Code
[2] Unicity Concept

۲-۳- اصول شش‌گانه‌ی کرکهف

آگوست کرکهف شهرت خود را از پژوهشهای زبان‌شناسی و کتاب‌هایی که در این خصوص و زبان ولاپوک نوشته بود بدست آورد. او در سال ۱۸۸۳ دو مقاله با عنوان «رمزنگاری نظامی» منتشر کرد. در این دو مقاله شش اصل اساسی وجود داشت که اصل دوم آن به عنوان یکی از قوانین رمزنگاری هنوز هم مورد استفاده دانشمندان در رمز نگاری پیشرفته‌است: [۱]

- سیستم رمزنگاری اگر نه به لحاظ تئوری که در عمل غیر قابل شکست باشد.
- سیستم رمزنگاری باید هیچ نکته پنهان و محرمانه‌ای نداشته باشد. بلکه تنها چیزی که سری است کلید رمز است.
- کلید رمز باید به گونه‌ای قابل انتخاب باشد که اولاً بتوان براحتی آن را عوض کرد و ثانیاً بتوان آن‌را به خاطر سپرد و نیازی به یاداشت کردن کلید رمز نباشد.
- متون رمزنگاری باید از طریق خطوط تلگراف قابل مخابره باشند.
- دستگاه رمزنگاری یا اسناد رمز شده باید توسط یک نفر قابل حمل و نقل باشد.
- سیستم رمزنگاری باید به سهولت قابل راه‌اندازی باشد.

اگر چه تمام این قواعد به نحوی در دنیای رمزنگاری مورد استناد قرار گرفته‌اند و لیکن اصل دوم که تاکید می‌کند «جزییات الگوریتم‌های رمزنگاری بایست آشکار و در دیدِ عموم باشند و فقط کلیدهای رمزشده محرمانه هستند»، به اصل کرکهف شهرت یافته‌است. به تبعیت از همین اصل جزییات تمام الگوریتم‌های رمزنگاری کاملاً مشخص و در اختیار عموم قرار می‌گیرد. شاید یک تازه‌کار در دانش رمزنگاری را نتوان مجاب کرد که مخفی نگاه‌داشتن جزییات الگوریتم هیچ کمکی به نفوذناپذیری آن نمی‌کند ولی شاید استدلال زیر عقلانیت نهفته در اصول کرکهف را روشن‌تر کند:

- هرگاه کلید رمز در اثر خیانت یا سهل‌انگاری یا هر عامل دیگر لو برود با تغییر کلیدِ رمز جلوی ضرر گرفته می‌شود ولی افشای جزییات یک سیستم و نفوذ در آن هیچ چیزی از سیستم باقی نمی‌گذارد و تنها راه تغییر سریع سیستم رمزنگاری است که این تغییر هرگز به راحتی و در زمان کوتاه میسّر نخواهد بود.

- هرگاه روشی برای سال‌ها در معرض افکار پژوهشگران و متخصصان این فن باشد و طرق علمی و عملی به چالش کشیده شود و هیچ تلاشی در شکستن آن به ثمر نرسد می‌توان فقط احتمال داد که روش به قدر کافی محکم بوده است.

نمونه‌ی بسیار مشهوری از عدم رعایت اصل کرکهف وجود دارد: وقتی آقای «رونالد ری‌وست» روش RC4 را طراحی و جزییات الگوریتم خود را از دید عموم پنهان نگاه‌داشت، نمی‌دانست که این مخفی‌کاری فقط ۶ سال دوام می‌آورد. وقتی جزییات الگوریتم در اینترنت افشا شد پژوهشگران متوجه اشکالاتی در آن شدند ولی در آن زمان سخت‌افزار و نرم‌افزارهای زیادی مبتنی بر RC4 در دست مردم بود و تغییر یک‌شبه‌ی آن‌ها امکان نداشت.

۲-۴- رمزنگاری پیشرفته

با پدید آمدن رایانه‌ها و افزایش قدرت محاسباتی آنها، دانش رمزنگاری وارد حوزه‌ی علوم رایانه گردید و این پدیده، موجب بروز سه تغییر مهم در مسائل رمزنگاری شد [۴]:

- وجود قدرت محاسباتی بالا این امکان را پدید آورد که روش‌های پیچیده‌تر و مؤثرتری برای رمزنگاری به وجود آید.
- روش‌های رمزنگاری که تا قبل از آن برای رمزکردن پیام به‌کار می‌رفتند، کاربردهای جدید و متعدّدی پیدا کردند.
- تا قبل از آن، رمزنگاری عمدتاً روی اطلاعات متنی و با استفاده از حروف الفبا انجام می‌گرفت؛ اما ورود رایانه باعث شد که رمزنگاری روی انواع اطلاعات و بر مبنای بیت انجام شود.

۲-۵- مفاهیم رمزنگاری

۲-۵-۱. رمزنگاری چیست؟

«رمزنگاری» یا Cryptography از دو کلمه‌ی یونانی kryptos (به معنای مخفی و پوشیده) و graphia (به معنای نگارش و ترسیم) مشتق شده است. رمزنگاری عبارت‌است از یک «نظام» یا الگوی «ریاضی/ منطقی» که براساس آن اطلاعات و مفاهیم آشکار و قابل فهم برای همگان، طبق روالی برگشت‌پذیر به اطّلاعاتی نامفهوم و گنگ تبدیل می‌شود. این اطّلاعات نامفهوم و گنگ توسّط کسی که روال معکوس و پارامترهای لازم را می‌داند قابل

برگشت و بهره‌برداری است. طبق اصل کرکهف چون قرار نیست هیچ نکته‌ای در بطن الگوریتم رمزنگاری و روال معکوس آن (یعنی رمزگشایی) مخفی بماند لذا در تمام الگوریتم‌های رمزنگاری، به پارامتری به نام «کلید رمز» احتیاج است که با تغییر آن ماهیت گنگ و مبهم اطّلاعات رمزشده به نحو غیرقابل پیش‌بینی تغییر می‌کند. لذا می‌توان فرآیند رمزنگاری را تابعی به شکل زیر تصوّر کرد [۴]:

$$C = F(P, K)$$ رابطه‌ی ۲-۱ تابع رمزنگاری

پیام P که باید رمزنگاری شود؛ P را متن آشکار (Plaintext) می‌گویند. پارامتر K که متن آشکار براساس مقدار آن به نحو غیرقابل پیش‌بینی و مبهم، درهم و بی‌معنی می‌شود، به «کلید رمز» شهرت دارد. متن C حاصل فرآیند رمزنگاری متن P با کلید K و حاصل تابع F قطعه‌ای اطّلاعات بی‌معنی موسوم به «متن رمز» است.

می‌بایست تفاوت بین «رمزنگاری» و «کدگذاری[1]» توضیح داده‌شود: وقتی دو نفر با هم قرار می‌گذارند به جای کلمه‌ی «پول» از کلمه‌ی «هویج» و به جای «پرداخت» از کلمه‌ی «گاز زدن» و نظایر آن استفاده کنند، یک نوع «کدگذاری» انجام داده‌اند! در فرآیند کدگذاری، کلمات، نمادها و افعال موجود در ادبیات زبانی، با مقادیر مورد توافق تعویض می‌شوند؛ به عنوان مثال «کُد ناواجو» نوعی از کدگذاری به شمار می‌آید نه رمزنگاری.

در رمزنگاری، دنباله‌ای از بیت‌ها یا بایت‌ها بدون توجه به محتویات زبان‌شناختی آن‌ها، طبق روالی معلوم و غیرسلیقه‌ای درهم و رمز می‌شود. در روال رمزنگاری دنباله‌ی داده‌ها، کلید رمز به عنوان پارامتر در متن داده‌ها تزریق می‌شود و چگونگی درهم شدن داده‌ها به مقدارِ کلید وابسته است. (ممکن است برای رمزنگاری داده‌ها از شیفت چرخشی استفاده شود و این کلید رمز است که می‌تواند تعداد شیفت را تعیین کند). فرض بر این است که داده‌هایی که بین گیرنده و فرستنده مخابره می‌شوند براحتی توسّط بیگانگان قابل شنود است و حتّی قابل دستکاری است [۱].

۲-۵-۲. سرویس رمزنگاری

به طور کلی، سرویس رمزنگاری به قابلیت و امکانی اطلاق می‌شود که بر اساس فنون رمزنگاری حاصل می‌گردد. قبل از ورود رایانه‌ها به حوزه‌ی رمزنگاری، تقریباً کاربرد رمزنگاری محدود به رمز کردن پیام و پنهان کردن مفاد آن می‌شده‌است. اما در رمزنگاری پیشرفته سرویس‌های مختلفی از جمله موارد زیر ارائه گردیده است [۴]:

---

[1] encoding

محرمانگی یا امنیت محتوا: ارسال یا ذخیره اطلاعات به نحوی که تنها افراد مجاز بتوانند از محتوای آن مطلع شوند، که همان سرویس اصلی و اولیه‌ی پنهان کردن مفاد پیام است.

سلامت محتوا: به معنای ایجاد اطمینان از صحت اطلاعات و عدم تغییر محتوای اولیه‌ی آن در حین ارسال است. تغییر محتوای اولیه‌ی اطلاعات ممکن است به صورت اتفاقی (در اثر مشکلات مسیر ارسال) و یا به صورت عمدی باشد.

احراز هویت یا اصالت محتوا: به معنای تشخیص و ایجاد اطمینان از هویت ارسال‌کننده اطلاعات و عدم امکان جعل هویت افراد می‌باشد.

عدم انکار: به این معنی است که ارسال‌کننده‌ی اطلاعات نتواند در آینده ارسال آن را انکار یا مفاد آن را تکذیب نماید.

چهار مورد بالا، سرویس‌های اصلی رمزنگاری تلقی می‌شوند و دیگر اهداف و سرویس‌های رمزنگاری، با ترکیب چهار مورد بالا قابل حصول می‌باشند. این سرویس‌ها مفاهیم جامعی هستند و می‌توانند برای کاربردهای مختلف پیاده‌سازی و استفاده شوند. به عنوان مثال سرویس اصالت محتوا هم در معاملات تجاری اهمیت دارد و هم در مسائل نظامی و سیاسی مورد استفاده قرار می‌گیرد. برای ارائه کردن هر یک از سرویس‌های رمزنگاری، بسته به نوع کاربرد، از پروتکل‌های مختلف رمزنگاری استفاده می‌شود [۴].

۲-۵-۳. الگوریتم رمزنگاری

الگوریتم رمزنگاری، به هر الگوریتم یا تابع ریاضی گفته می‌شود که به علت دارا بودن خواص مورد نیاز در رمزنگاری، در پروتکل‌های رمزنگاری مورد استفاده قرار گیرد. اصطلاح الگوریتم رمزنگاری یک مفهوم جامع است و لازم نیست هر الگوریتم از این دسته، به طور مستقیم برای رمزگذاری اطلاعات مورد استفاده قرار گیرد، بلکه صرفاً وجود کاربرد مربوط به رمزنگاری مد نظر است. طراحی الگوریتم‌های رمزنگاری مقوله‌ای برای متخصصان ریاضی است [۱].

سیستم‌های رمزنگاری به دو رده کلّی متقارن و کلید عمومی تقسیم‌بندی می‌شوند:

۲-۵-۳-۱. رمزنگاری متقارن

در رمزنگاری متقارن، رمزنگاری و رمزگشایی اطّلاعات با کلیدی مشابه صورت می‌گیرد. این کلید باید بین طرفین توافق شده باشد. از ویژگی کلّی سیستم‌های رمزنگاری متقارن می‌توان به موارد زیر اشاره کرد [۱]:

الف) سیستم‌ها و الگوریتم‌های رمزنگاری متقارن از لحاظ عملکرد بسیار سریعند و امکان پیاده‌سازی سخت-افزاری و نرم‌افزاری آن برای رمزنگاری بی‌درنگ داده‌ها تا نرخ بالاتر از گیگابیت بر ثانیه وجود دارد.

ب) در رمزنگاری متقارن، داده‌های متن اصلی در قالب بلوک‌هایی با طول ثابت و عموماً کوتاه (۶۴، ۱۲۸ یا ۲۵۶ بیتی) پردازش و رمز می‌شوند.

ج) چون کلیدهای رمزنگاری و رمزگشایی شبیه یکدیگرند لذا طرفین ارتباط بایستی به روشی مطمئن (مثلاً از طریق ملاقات حضوری یا شخصی معتمد یا سیستمی خودکار ولی مطمئن) کلید خود را توافق کرده و از آن بهره بگیرند؛ امنیت کل داده‌ها به «امنیت کلید» گره خورده است.

د) هرگاه شخص یا سرویس‌دهنده‌ای بخواهد با تعداد زیادی از کاربران ارتباط امن و رمزنگاری‌شده داشته باشد باید با تک‌تک آن‌ها کلیدی مجزا و مستقل را توافق کند چرا که تعریف کلیدی واحد برای همه‌ی کاربران اولاً امکان استراق‌سمع کاربران از اطلاعات یکدیگر را فراهم می‌آورد، ثانیاً سهل‌انگاری یا خیانت یکی از کاربران امنیت تمام آن‌ها را به خطر خواهد انداخت.

ه) عموماً در تمام روش‌های رمزنگاری متقارن فرآیند ادغام داده‌ها و کلید و هرگونه عملیات درهم ریختن داده‌ها چندین بار تکرار می‌شود. به تکرار هر بار از این عملیات یک «دور[1]» گفته می‌شود؛ تعداد دورها بین ۸ تا ۶۴ دور متغیر است.

و) در سیستم‌های رمزنگاری متقارن عموماً فرآیند رمزنگاری و رمزگشایی تشابه کامل دارند با این تفاوت که فقط مقادیر متغیرها و ثابت‌ها عوض می‌شوند ولی در مجموع ذات ساختار الگوریتم رمزنگاری و رمزگشایی متّحدالشکل و یکسان است.

---

[1] Round

ز) در شبکه‌ای که جمعاً n کاربر وجود دارد و دوبه‌دوی آن‌ها می‌خواهند با یکدیگر ارتباطی امن و رمزنگاری شده برقرار کنند، به تعریف و توافق $n(n\text{-}1)/2$ کلید سرّی و متقارن نیاز است [۱].

از آن‌جا که در روش‌های متقارن پیام‌های بزرگ در قالب قطعات کوچک ۶۴، ۱۲۸ یا ۲۵۶ بیتی پردازش و رمز می‌شوند لذا یا باید برای هر بلوک، کلید رمز را عوض کرد که چنین کاری در عمل ممکن نیست و یا باید بلوک‌های رمز را به نحوی به یکدیگر زنجیر کرد. در غیر این‌صورت بلوک‌های مشابه از متن اصلی به بلوک‌های رمزشده‌ی مشابهی تبدیل می‌شوند. همین موضوع که یک اخلال‌گر در شنود غیرمُجاز اطلاعاتِ رمزشده، دو بلوک را مشابه ببیند ممکن است بتواند در مورد ماهیت آن دو بلوک حدس‌های درستی بزند لذا جلوگیری از نگاشته شدن بلوک‌های مشابه متن اصلی به بلوک‌های رمزشده‌ی یکسان نیاز به عملیاتی مجزا دارد که در این پروژه به آن پرداخته خواهد شد[۴].

از روش‌های رمزنگاری متقارن، گاه با نام سیستم «رمز قطعه‌ای» یاد می‌شود. هرگاه پیامی که باید رمز شود ضریبی از طول بلوک ورودی نباشد، باید به روش مناسبی به انتهای پیام داده‌های زاید چسبانده شود. به عنوان مثال هرگاه طول بلوک ورودی ۱۲۸ بیت (۱۶ بایت) و طول پیام ۳۶۰ بایت باشد باید به انتهای پیام ۸ بایت داده‌ی زاید چسبانده‌شود تا ضریبی از ۱۶ شده و بتوان آن‌را در قالب ۲۳ بلوک ۱۶ بایتی رمزنگاری کرد. گاهی نیز (ولی به ندرت) روش‌های رمزنگاری متقارن با نام‌های، تک‌کلیدی، یک‌کلیدی ، کلید سرّی یا کلید خصوصی نیز نام برده می‌شود. نوع پایه‌ی الگوریتم‌های رمزنگاری متقارن وجود دارند: رمز بلوکی[1] (قطعه‌ای) و رمز دنباله‌ای[2]. رمزهای بلوکی روی بلوک‌هایی از متن آشکار و متن رمزشده عمل می‌کنند- معمولاً بلوک‌های ۶۴ بیتی و در این پروژه ۱۲۸ بیتی- در حالی‌که رمزهای دنباله‌ای روی یک دنباله بیت یا بایت از متن آشکار یا متن رمزشده (گاهی یک کلمه‌ی 32 بیتی) در یک زمان عمل می‌کنند [۴].

۲-۵-۳-۲. رمزنگاری کلید عمومی[3]

قفلی را مجسّم کنید که دارای دو کلید سبز و قرمز است. کلید سبز فقط در جهت ساعتگرد می‌چرخد و صرفاً می‌تواند آن‌را قفل کند ولی وقتی قفل بسته شد فقط می‌توان کلید سبز را خارج کرد، چون به هیچ‌وجه در سمت

---

[1] Block Cipher
[2] Stream Cipher
[3] Public Key Cryptography

پادساعتگرد نخواهد چرخید. چنین کلیدی را می‌توان به تعداد فراوان تکثیر و در اختیار دوست و دشمن گذاشت چرا که با این کلید فقط می‌توان قفل را بست و قفل بسته را باز نخواهد کرد. در سمت مقابل کلید قرمز فقط در سمت پادساعتگرد و برای گشودن قفل می‌چرخد؛ این کلید نزد صاحب قفل می‌ماند و با وسواس از آن مراقبت می‌شود؛ تجسم چنین قفلی در دنیای مَجازی با «الگوریتم رمزنگاری عمومی» تحقق یافته است. در این الگوریتم‌ها، دو پارامتر به عنوان «کلیدعمومی[1]» و «کلید خصوصی[2]» تعریف شده که با کلید عمومی می‌توان داده‌ها را رمزنگاری کرد ولی داده‌های رمزنگاری‌شده را نمی‌توان با چنین کلیدی از رمز خارج کرد. پارامتر کلید عمومی را می‌توان براحتی در اختیار همگان قرارداد و یا آن‌را از طریق کانال ناامن به صورت فراگیر پخش کرد. پارامتر «کلید خصوصی» در نزد صاحب آن به صورت سرّی و محرمانه نگهداری می‌شود. بدین‌ترتیب فقط و فقط صاحب کلید می‌تواند داده‌های رمز شده با کلید عمومی را از رمز خارج کند [۴].

کلیدهای عمومی و خصوصی عموماً اعدادی چندصد بیتی هستند و الگوریتم‌های رمزنگاری قابل اعتماد و مستحکم نیز عموماً بر مبنای تئوری اعداد و ویژگی شگفت‌انگیز ریاضیات اعداد بنا نهاده شده‌اند.

روش‌های کلید عمومی به توافق تنها n کلید برای تبادل داده بین n نفر نیاز دارد و این باعث ساده‌تر شدن مدیریت کلیدها در مقایسه با روش‌های متقارن می‌شود (کلید خصوصی به صورت سرّی نزد افراد باقی می‌ماند). بزرگترین اشکال روش‌های کلید عمومی در سرعت بسیار پایین و بزرگی طول کلید آن‌هاست. لذا هرگز نمی‌توان این روش‌ها را جایگزین روش‌های متقارن کرد بلکه این دو روش مکمل هم هستند. برای رمزنگاری توده‌های بزرگ اطلاعات از روش‌های متقارن استفاده می‌شود در حالی‌که رمزنگاری کلید عمومی برای رمزکردن قطعات کوچک ولی بنیادی بکار می‌آیند [۴].

۴-۵-۴. حساسیت کلید در روش‌های متقارن و نامتقارن

هرگاه توده‌ای از اطلاعات محرمانه را رمزنگاری می‌کنید در حقیقت امنیت و صیانت آن‌را به کلید کوچکی گره می‌زنید! سرّی ماندن پیام‌ها در گرو مراقبت ویژه از کلید رمز است. اولین اصل بنیادین در رمزنگاری اطلاعات، کافی بودن طول کلید است. به عنوان مثالی ساده یک قفل ساده‌ی رمزدار مثل قفل برخی از کیف‌های شخصی را در

[1] Public Key
[2] Private Key

نظر بگیرید؛ روش باز کردن این قفل آن است که چند رقم را بر روی قفل به درستی تنظیم کنید. الگوریتم باز کردن قفل و حتی ساختار داخلی قفل بر همه روشن ولی کلید رمز محرمانه است. فرض کنید دستکاری مستقیم و باز کردن آن توسط روش‌های غیرمتعارف مثل سنجاق و انبرک ممکن نیست و یک سارق برای باز کردن چنین قفلی فقط باید اعداد مختلف را بر روی آن بیازماید! اگر تعداد ارقام رمز، سه رقمی باشد فضای حالت جمعاً هزار حالت مختلف دارد که به طور میانگین سارق باید نصف این فضا (یعنی حدوداً ۵۰۰ عدد) را جستجو و آزمایش کند. هر چه طول کلید بیشتر باشد فضای جستجو بیشتر شده و احتمال موفقیت جستجوگر کاهش می‌یابد. حجم عملیات لازم برای جستجوی کلید از طریق آزمون تمام کلیدهای ممکن اصطلاحاً Work Factor نامیده شده و به تمام حالات مختلفی که یک کلید می‌تواند اتخاذ کند فضای کلید[1] می‌گویند [۱].

برای روش‌های متقارن یک کلید ۱۲۸ بیتی (فضای کلید $=۲^{۱۲۸}$) کاملاً کفایت می‌کند. چرا که بزرگی این عدد از مرتبه‌ی $۳٫۴\times۱۰^{۳۸}$ (معادل تعداد مولکول‌های پانصد میلیارد تُن آب) است! هرگاه کسی بخواهد چنین فضایی را با سرعت یک میلیون کلید در ثانیه جستجو کند، زمانی به نتیجه خواهد رسید که عمر خورشید به پایان رسیده است! لذا در روش‌های متقارن یک کلید ۱۲۸ بیتی به خوبی کفایت می‌کند. امروزه بسیاری از الگوریتم‌های رمزنگاری متقارن دارای کلیدی ۱۲۸ بیتی (وگاه ۱۹۶ تا ۲۵۶ بیتی) هستند [۱].

در روش‌های رمزنگاری کلید عمومی اولاً انتخاب کلیدها تابع ضوابط خاصی است و هرگاه یک رمزشکن بخواهد با جستجو و آزمون کلیدها رمزی را بشکند لازم نیست کل این فضا را جستجو کند بلکه فضای جستجو بخش کوچکی از کلّ فضای حالت را تشکیل می‌دهد. به عنوان مثالی ابتدایی فرض کنید کلید رمزنگاری باید عددی صحیح و حاصل‌ضرب دو عدد اول باشد. بدیهی است که جستجوی چنین کلیدی در فضای بسیار کوچک‌تری از کلّ اعداد صحیح انجام می‌گیرد چون به‌راحتی اعداد زوج، اعداد بخش‌پذیر بر ۳ و ۵ و... ، اعداد مربع، مکعب و... از فضای جستجو کنار گذاشته می‌شوند. در ضمن می‌توان از روابط حاکم بر تئوری اعداد برای کاهش هر چه بیشتر فضای جستجو بهره گرفت.

حمله به رمز به هرگونه تلاش برای یافتن کلید رمز یا رمزگشایی غیرمُجاز داده‌های رمزشده (در هر یک از وضعیت‌های سه‌گانه‌ی فوق) در اصطلاح «حمله به رمز» گفته می‌شود.

---

[1] Key Space

اصل اساسی در رمزنگاری «استحکام» : هر الگوریتم رمزنگاری عامّ، بایستی علیه حملات سه‌گانه مستحکم بوده و نتیجه‌ی تلاش رمزشکن ناموفّق و مایوس‌کننده باشد. روش‌های وجود دارند که اگرچه در مقابل حمله‌ی Cipher-Text Only به‌طور کامل امن و مستحکمند ولی در مقابل حمله‌ی Known-Plaintext یا Chosen-Plaintext به راحتی درهم می‌پاشند. چنین روش‌های بطور عام بی‌ارزشند ولی در ترکیب با روش‌های دیگر یا برای موارد خاص می‌توانند مورد استفاده قرار بگیرند.

اصل اساسی در رمزنگاری «تازگی[1]»: فرض‌کنید کسی مثل آلیس (به‌عنوان مشتری) برای باب (به‌عنوان سرویس‌دهنده‌ی بانک) پیامی بسیار مهم و رمزنگاری‌شده مبنی بر انتقال پول از حسابش به حسابی دیگر ارسال می‌کند. فرض کنید شخص ثالثی این پیام را استراق‌سمع کرده و به دلیل استحکام بی‌نظیر الگوریتم رمزنگاری هرگز موفق به شکستن رمز نمی‌شود ولی یک هفته بعد همان پیام را از طرف آلیس به دروغ برای باب می‌فرستد و باب با رمزگشایی آن، عمل خواسته شده را انجام می‌دهد غافل از آن‌که این پیام تکراری و دروغین است. بنابراین رمزنگاری اطلاعات اگرچه لازم است ولی هرگز کافی نیست! باید مکانیزم‌های در کنار فرآیند رمزنگاری اتخاذ شود تا از «حمله تکرار[2]» پیشگیری شده و «تازگی پیام» به اثبات برسد و گرنه امکان هرگونه اختلال یا سوءاستفاده وجود دارد [۱].

برای اثبات تازگی پیام‌ها می‌توان برای هر پیام، تاریخ و زمان صدور و مهلت اعتبار در نظر گرفت و یا هر پیام بهمراه یک «شناسه‌ی یکتا» رمز و ارسال شود. بدین ترتیب از حمله‌ی تکرار جلوگیری خواهد شد. بعداً دیده خواهد شدکه اتکا به هر یک از این روش‌ها به تنهایی اشکالاتی را ایجاد خواهد کرد لذا در دنیای عمل ترکیبی از «مهر زمان[3]» و شناسه‌ی پیام بکار می‌رود تا گیرنده بتواند «تازگی پیام» را اثبات کند [۱].

اصل اساسی در رمزنگاری «افزونگی[4]»: فرض کنید قالب پیامی که برای سرویس‌دهنده‌ی بانک اطلاعاتی ارسال می‌شود تا نمره‌ی نهایی درس یک دانشجو در کارنامه‌ی او ثبت شود به شکل زیر باشد [۱]:

نمره‌ی درس	شماره‌ی درس	شماره‌ی دانشجویی
(یک بایت)	(دو بایت)	(پنج بایت)

[1] Freshness
[2] Replay Attack
[3] Time-Stamp
[4] Redundancy

به ازای ورود نمره‌ی هر دانشجو، پیامی به شکل مذکور تولید و پس از رمزنگاری متقارن، برای سرویس‌دهنده ارسال می‌شود. اگرچه هیچ کسی قادر به رمزگشایی پیام‌های رمزنگاری‌شده نیست ولی فرض کنید تغییر به اندازه‌ی یک بیت در مقدار هر یک از فیلدهای بالا، مقداری معتبر را به مقدار معتبر دیگری تبدیل می‌کند. (به عنوان مثال با تغییر در یک بیت از شماره‌دانشجویی یک فرد، شماره‌ی جدید متعلق به فرد دیگری باشد) با چنین فرضی، یک اخلال‌گر اگرچه قادر به رمزگشایی پیام‌هایی رمزشده با قالب بالا نیست ولی او با ایجاد یک تغییر تصادفی در پیام رمزشده مطمئن است که حاصل رمزگشایی پیام، پیام معتبر دیگری را بوجود می‌آورد؛ نتیجه آن است که نمره‌ی درسی دیگر از دانشجویی دیگر با مقداری تصادفی در کارنامه‌اش ثبت می‌شود [۱].

حتی اگر فرض شود با تغییر در پیام‌های رمزشده، حاصل رمزگشایی پیام با احتمال یک‌دهم درصد به پیامی معتبر تبدیل می‌شود و با احتمال ۹۹،۹٪ از طرف سرویس‌دهنده رد و غیرقابل قبول اعلام خواهد شد، در چنین وضعیتی باز هم اخلال‌گر این فرصت را دارد که با تولید هزاران پیام جعلی و تصادفی، آن‌را برای سرویس‌دهنده بفرستد تا اگر در موارد نادر پیامی معتبر بدست آمد در بانک اطلاعاتی نمرات درج شود؛ در این صورت اخلال‌گر در حمله‌ی خود موفق است. بدین ترتیب یک اصل اساسی در رمزنگاری بیان می‌شود:

«پیام‌ها باید دارای افزونگی باشند و این افزونگی به نحوی هوشمندانه در پیام‌ها درج شود تا هر تغییر، پیامی معتبر را با احتمال قریب به یقین به پیامی غیرمعتبر و قابل‌کشف تبدیل کند». به عنوان یک مثال ساده ممکن است قالب پیام‌ها را به همراه یک فیلد اضافی که حاصل یای انحصاری بایت به بایت کل پیام است، به شکل زیر تعریف کنید تا هر پیام دارای یک فیلد یک بایتی «توازن[1]» باشد:

توازن Parity	نمره‌ی درس (یک بایت)	شماره‌ی درس (دو بایت)	شماره‌ی دانشجویی (پنج بایت)

در این حالت هر تغییر تصادفی در پیام‌های رمزنگاری‌شده، پس از رمزگشایی به پیامی تبدیل خواهد شد که (به احتمال زیاد) اعتبار ندارد چرا که اخلال‌گر نمی‌داند چگونه پیام‌های رمزشده را تغییر بدهد تا پس از رمزگشایی پیامی معتبر بدست آید. پس بهتر است که به ازای هر پیام حساس، آن‌قدر داده‌های افزونه (زاید) طبق الگویی خاص و

[1] Parity

حساب‌شده به پیام اضافه گردد تا تغییرات احتمالی در پیام‌های رمزشده، پس از رمزگشایی، منجر به تولید پیام معتبر دیگری نشده و گیرنده را فریب ندهد.

اضافه‌کردن کُدهای کشف خطای CRC یا کُدهای تصحیح خطا به اصل پیام‌ها نیز برای جلوگیری از تبدیل یک پیام معتبر به پیام معتبر دیگر (در اثر تغییر عمدی یا سهوی) می‌تواند راه‌گشا باشد ولی این کُدها در موارد خطاهای تصادفی با توزیع یکنواخت موفق عمل می‌کنند در حالی‌که تغییرات عمدی پیام عموماً هوشمندانه و با فرکانس تکرار بسیار بالا صورت می‌گیرد؛ لذا برای افزایش امنیت، ترجیحاً بایستی از روش‌های چکیده‌ی پیام استفاده شود.

۲-۶- نتیجه‌گیری

رمزنگاری با تاریخ پنج‌هزار ساله‌ی خود عبارت‌است از یک «نظام» یا الگوی «ریاضی/ منطقی» که براساس آن اطلاعات و مفاهیم آشکار و قابل فهم برای همگان، طبق روالی برگشت‌پذیر به اطّلاعاتی نامفهوم و گنگ تبدیل می‌شود. در رمزنگاری سرّی ماندن پیام‌ها در گرو مراقبت ویژه از کلید رمز است. در این فصل با مفهوم رمزنگاری و تاریخچه‌ی آن آشنایی مختصری پیدا شد و تفاوت ساختاری روش‌های رمزنگاری متقارن و روش‌های کلید عمومی، ماهیت کلید رمز مطالعه شد.

فصل سوم

ویژگی‌های سیستم‌های مدرن رمزنگاری و مختصری در مورد تصاویر

۳-۱- مقدمه

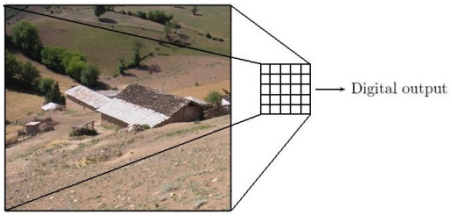
یک تصویر به‌عنوان یک تابع دوبعدی $f(x,y)$ است که در آن x و y مختصات فضایی (صفحه) هستند و دامنه‌ی f در هر یک از این مختصات وضوح عکس در آن نقطه از تصویر را نشان می‌دهد. اصطلاح سطوح خاکستری اغلب به تصاویر تک‌فام اشاره دارد. تصاویر رنگی به وضوح ترکیبی از تصاویر تکی و انفرادی هستند. به عنوان مثال در سیستم رنگی RGB یک تصویر رنگی از سه مولفه‌ی تک فام تشکیل می‌شودکه به ترتیب به آن‌ها مولفه یا تصاویر اولیه‌ی قرمز و سبز و آبی (Blue، green،Red) گویند. به همین خاطر بسیاری از تکنیک‌هایی که برای تصاویر تک‌فام ایجاد شده‌اند می‌توان برای تک تک مولفه‌های تصویر رنگی جداگانه استفاده کرد. یک تصویر می‌تواند به ترتیب در محورهای x و y و همچنین در دامنه نیز پیوسته باشد. برای تبدیل یه تصویر به نوع رقمی یا دیجیتالی، صفحات مختصات به خوبی مانند دامنه باید رقمی شوند. به دیجیتالی‌کردن مقادیر مختصات نمونه‌برداری گویند و به دیجیتالی کردن مقادیر دامنه کوانتیزه کردن گویند. بنابراین هنگامی‌که مقادیر x، y و دامنه تابع f همگی کمیت‌های متناهی و گسسته هستند به این تصاویر، دیجیتالی یا رقمی گویند [۵].

۳-۲- پیکسل[1] چیست؟

تصاویری که یک دوربین دیجیتال دریافت می‌کند به صورت شکل ۳-۱ می‌باشد. این تصاویر از نقاط مربع شکل بسیارکوچکی تشکیل شده که هر کدام از این نقاط دارای رنگی خاص است که با قرارگیری در کنار یکدیگر تشکیل یک تصویر را می‌دهند. اگر تصویری چند برابر بزرگ‌تر شود این موضوع برایمان روشن می‌شود. این نقاط پیکسل نامیده می‌شوند که کوچکترین عنصر تشکیل‌دهنده‌ی یک تصویر دیجیتال است [۲].

---

[1] Pixel



شکل ۳-۱ یک تصویر دوربین دیجیتال [۲]

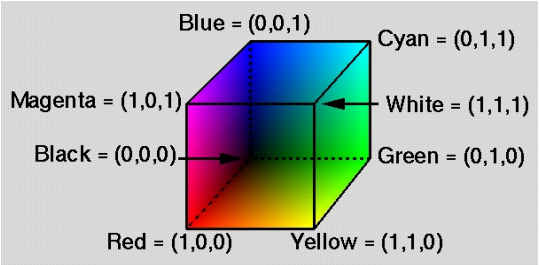
یک دوربین هر یک از پیکسل‌ها را توسط حسگرهای خود دریافت و به اعداد دیجیتال تبدیل می‌کند. تعداد این پیکسل‌ها در یک تصویر اندازه‌ی تصویر را مشخص می‌کند و برای ذکر اندازه یک تصویر از تعداد پیکسل‌های موجود در تصویر، به صورت ضرب سطر در ستون یا طول در عرض استفاده می‌شود. برای مثال اگر یک تصویر دارای ۸۰۰ پیکسل در طول و ۶۰۰ پیکسل در عرض باشد، این به معنی وجود ۴۸۰۰۰۰ عدد پیکسل (۶۰۰×۸۰۰) در آن تصویر می‌باشد، و می‌توان با استفاده از همین پیکسل‌ها و تبدیل آن‌ها به یک ماتریس، تصاویر را پردازش کرد [۲].

هر پیکسل باید به صورت یک عدد بیان شود تا قابل پردازش باشد. در صورتی که تنها از اعداد ۰ و ۱ برای نمایش پیکسل‌ها استفاده شود، تصویر تنها دارای دو رنگ سیاه و سفید خواهد بود که بدون هیچ سایه و تغییر روشنایی خواهد بود. به این نوع تصویر باینری گویند. حال در صورتی که هر پیکسل دارای تعداد بیشتری بیت باشد، می‌توان تصاویر با کیفیت و سایه‌ها و روشنی‌های بیشتری دید [۲].

۳-۳-روش شناسایی رنگ در یک پیکسل

برای شناساندن رنگ یک پیکسل به نرم‌افزار روش‌ها و مدل‌های مختلفی وجود دارد. امّا برای نرم‌افزار بهترین نوع مدل رنگ مدل RGB می‌باشدکه به دلیل سادگی بیشتر از این مدل استفاده شده است. مدل RGB که کوتاه‌شده کلمات آبی و سبز و قرمز است که این سه رنگ اصلی برای تشکیل تمامی رنگ‌های موجود در دنیا کافی می‌باشد. در دوربین‌ها و تلویزیون‌ها هم کاملاً از این نوع روش یعنی استفاده از سه رنگ اصلی برای تولید رنگ‌ها استفاده می‌شود. برای نمایش یک سیستم RGB از مکعب رنگ استفاده می‌شود. مکعب رنگی را در شکل۳-۲ مشاهده می‌کنید. برای هر یک از رنگ‌های قرمز و سبز و آبی، عددی ۸ یا ۱۶ یا ۳۲ بیتی استفاده می‌شود. برای مثال در صورت استفاده از ۸ بیت (برابر با عدد ۲ به توان ۸) برای هر یک از رنگ‌های سه‌گانه، سیستم می‌تواند تا بیش از تعداد ۱۶

میلیون رنگ را تشخیص دهد (۱۶۵۸۱۳۷۵= ۲۵۵×۲۵۵×۲۵۵). حال ببینید با ۳۲ بیت برای هر یک از رنگ‌های سه گانه، چه تعداد رنگ قابل بررسی می‌شود [۲].



شکل ۳-۲ طرح مکعب رنگی RGB. رنگ‌های اولیه و ثانویه از نقطه‌ی سیاه در مبدا تا نقطه‌ی سفید در (۱و۱و۱) [۵]

در MATLAB مقدار بیت هر پیکسل ۸ بیتی را با نام و کلاس uint8 نمایش می‌دهند و به همین صورت تصاویر ۱۶ و ۳۲ را کلاس double- uint16 به نمایش در می‌آید. در نرم‌افزار هر یک از این سه عدد ۸ تا ۳۲ بیتی به یک پیکسل اختصاص دارد که تشکیل رنگ یک تصویر را می‌دهد. این سه رنگ اصلی به نسبت‌های مختلف در صورتی که با هم ترکیب شوند، رنگ مورد نظر را تشکیل می‌دهند. این نوع از تصاویر رنگی که تشکیل‌شده از ۳ ماتریس جداگانه برای رنگ‌های سبز و آبی و قرمز، هر آرایه از سه ماتریس دو بُعدی ترکیب شده، تبدیل به رنگ کامل یک پیکسل از تصویر می‌شود. این نوع از تصاویر بیشترین زمان پردازش را به خاطر وجود سه ماتریس به خود اختصاص می‌دهند [۲].

۳-۴- تصاویر باینری

تصاویری که تنها دارای دو رنگ سیاه و سفید می‌باشند و در ماتریس آن‌ها تنها اعداد ۰ و ۱ قرار گرفته است. این نوع تصاویر کاربرد فراوانی را در زمینه‌ی پردازش تصویر دارند و اکثر برنامه‌ها تصاویر خود را در نهایت به این نوع تبدیل کرده و بر روی آن پردازش را انجام می‌دهند. در شکل ۳-۳ یک تصویر فرضی را که به نوع باینری تبدیل شده است، مشاهده می‌کنید. ساده‌ترین پردازش در مورد این تصاویر صورت می‌گیرد و این تنها به دلیل تک بیت بودن پیکسل‌های این نوع تصاویر است (اعداد ۰ به معنی رنگ سیاه و اعداد ۱ به معنی رنگ سفید می‌باشد) [۲].

شکل ۳-۳ تبدیل تصویر فرضی رنگی به باینری [۲]

۳-۵- تصاویر غیررنگی با شدت نور[1]

این نوع تصاویر غیررنگی که خاکستری[2] نامیده می‌شوند، با وضوح روشنی و تیرگی‌های تصویر به صورت تک ماتریس در ابعاد تصویر می‌باشد. آرایه‌های ماتریس این تصویر تنها نشان‌دهنده‌ی میزان روشنایی یا تیرگی و سایه‌های تصویر هستند. در قدیم آرایه‌های این نوع تصاویر عددی بین ۰ تا ۱ و تا سه رقم اعشار بود، امّا در حال حاضر این نوع تصاویر بیشتر به صورت ۸ بیتی و شبیه به اعداد ماتریس RGB می‌باشد. در حقیقت تنها تفاوت این نوع تصاویر با RGB نبود رنگ در تصاویر می‌باشد [۲].

۳-۶- ویژگی‌های سیستم‌های مدرن رمزنگاری متقارن

هر سیستم رمزنگاری متقارن بایستی در بالاترین سطح ممکن دارای دو ویژگی «پخش و پراکندگی[3]» و «گمراه‌کنندگی[4]» باشد. این دو اصطلاح در سال ۱۹۴۹ توسط «کلود شانون» (پدر تئوری اطّلاعات) معرفی و از آن پس مورد استناد قرار گرفت. این دو ویژگی عموماً معیاری غیرکمّی است که پس از مطالعه و تحلیل جمعی و به چالش طلبیدن نخبگان رمزشکن در خصوص یک الگوریتم تصدیق یا رد می‌شود [۱]:

- **پخش و پراکنده‌سازی**: از دیدگاه تئوری، سیستم رمزنگار باید ساختار و کلّ شاخص‌های آماری متن آشکار را بر روی کل متن رمزشده، توزیع و پراکنده کند. به بیان دیگر یک سیستم رمزنگار هرگز نباید ویژگی‌های آماری متن را به هر نحو در خروجی رمز شده منتقل کند. بدین ترتیب هرگاه خروجی یک سیستم رمزنگار را تحلیل آماری می‌کنید، نباید هیچ‌گونه هم‌بستگی[5] بین بیت‌های خروجی رمزنگار و

[1] Gray or Intensity
[2] Gray
[3] Diffusion
[4] Confusion
[5] Correlation

بیت‌های متن رویت شود. هر متن آشکار فارغ از آن‌که ماهیتاً از چه جنسی باشد (متن، صدا، تصویر،...) دارای الگو و شاخص‌های آماری خاص خود است و طبعاً الگوی آماری یک متن آشکار(رمزشده) را می‌توان با چندین پارامتر کمّی مدل‌سازی کرد. به عنوان مثال، میانگین، واریانس، همبستگی، سمبل‌های یک متن از شاخص‌های قابل اندازه‌گیری برای انواع متن است. به عنوان مثالی عامیانه و غیر عملی فرض کنید چند شی را که هر کدام رنگ، بو، وزن، حجم، بافت، جنس، متفاوتی دارند، چنان درهم چرخ و مخلوط کنید که معجون بدست آمده هیچ نشانی از اشیا اولیه خود نداشته باشد. چنین کاری مصداق عمل پخش و پراکنده‌سازی است چرا که خصوصیات ظاهری این معجون شامل رنگ، بو، وزن حجمی از هیچ‌کدام از اشیاء تشکیل‌دهنده‌ی آن خبر نمی‌دهد. به عنوان مثال فرض کنید پیامی متشکل از n کاراکتر $m = m_1\ m_2 \ldots m_n$ به صورت زیر باشد:

اگر رابطه‌ی زیر برای رمزنگاری این کاراکترها بکار برده‌شود، خصوصیت پخش و پراکنده‌سازی در آن وجود دارد: $Y_n = \sum (c_i \times m_i \bmod 26)$ که در آن c_i اعداد ثابت بین صفر و ۲۵ و m_i ها، کُد عددی هر کاراکتر و عمل‌گر جمع، به پیمانه‌ی ۲۶ است. بدین ترتیب در ساخته‌شدن هر کاراکتر از خروجی رمزشده، تمام کاراکترهای متن اصلی دخالت دارند. بدین ترتیب ویژگی‌های آماری متن مثل فراوانی تک‌حرفی[1]، دو حرفی[2]، سه حرفی[3] در اثر میانگین‌گیری پیمانه‌ای بر روی حجم کثیری از کاراکترهای خروجی پراکنده و محو می‌شود و هرگاه نمودار فراوانی نسبی این شاخص‌ها را ترسیم کنید تقریباً شکلی «یکنواخت[4]» خواهید داشت [۱].

- **گمراه‌کنندگی**: بدین معناست که در عمل هرگز نباید بتوان بین ورودی، خروجی و کلید هیچ رابطه‌ی سرراست و مشخصی پیدا کرد. به بیانی دیگر پیچیدگی یک سیستم رمزنگاری متقارن باید آن‌قدر زیاد ژرف باشد که استنتاج رابطه‌ای که براساس آن خروجی سیستم بر حسب کلید و ورودی بدست می‌آید در عمل از عهده‌ی هیچ کسی در جهان برنیاید؛ حتی اگر به ابررایانه‌های فوق سریع مجهز باشد [۱].

به غیر از این دو ویژگی اساسی، یک سیستم رمزنگار متقارن بایستی خواص زیر را در حد بالایی داشته باشد:

---

[1] monogram
[2] digram
[3] trigram
[4] uniform

- **اثر فروپاشی بهمنی**[1]:اثر فروپاشی بهمنی در سیستم‌های رمزنگاری متقارن بدین معناست که یک تغییر بسیار جزیی در ورودی یا کلید (حتی به اندازه‌ی یک بیت) به طرز بسیار گسترده، غیرقابل پیش‌بینی و غیرمتمرکز، خروجی را تغییر داده و متحول کند و در عین حال هیچ شاخص آماری از این تغییر و تحول به جا نگذارد. گاهی اثر فروپاشی بهمنی را با «شاخص کمّی» بیان می‌کنند، به عنوان مثال در الگوریتم DES تغییر در یک بیت از ورودی یا کلید، بطور میانگین بیش از ۳۰ بیت از خروجی را تغییر خواهد داد، ولی در عین حال ماهیت این تغییر تصادفی و غیرقابل پیش‌بینی است.
- **تولید شبه‌نویز:** هرگاه خروجی رمزنگار را به ازای هزاران ورودی یا کلید مختلف مشاهده می‌کنید باید آن‌را شبیه به یک دنباله‌ی کاملاً تصادفی، مستقل از کلید و بدون هیچ شاخص آماری مرتبط با متن، ارزیابی کنید. چنین سیستمی به اصطلاح «دنباله‌ی شبه‌نویز» تولید می‌کند.
- **رعایت اصل دوم کرکهف:** آگاهی از جزییات الگوریتم رمزنگاری هرگز نباید سبب تضعیف آن شود و امنیت الگوریتم صرفاً باید در گرو مخفی نگاه‌داشتن کلید سرّی باشد. مخفی نگاه داشتن کلیات روش (شامل الگوریتم و مراحل رمزنگاری) و جزییات روش (شامل جداول جانشینی، ترتیب جایگشت، ماهیت ثابت‌ها و متغییرها و استدلال هر عمل) هرگز در بلندمدّت کمکی به امنیت و استحکام روش نمی‌کند.
- **وجود مولفه‌های غیرخطّی:** در هر الگوریتم رمزنگاری متقارن بایستی از مولفه‌هایی بهره گرفت که غیرخطّی عمل کنند. به عنوان مثال یک S-Box با جدول جانشینی مناسب می‌تواند به شدّت غیرخطی عمل کند در حالی‌که جایگشت بیتی یا جمع معمولی فرآیندی خطی است. عدم وجود عناصر غیرخطّی در یک سیستم رمزنگار متقارن در تضاد با ویژگی گمراه‌کنندگی است. زیرا بیت‌های ورودی و کلید با رابطه‌ای خطی خروجی را توصیف کرده و می‌توان خروجی را برحسب ورودی و کلید بدست آورد. حال منظور از غیرخطی چیست: تابعی مثل f را غیرخطی می‌گویند هر گاه $f(a + b) \neq f(a) + f(b)$.

تابعی مثل f را به شدت غیر خطی گویند مشروط بر این‌که تابع نه تنها غیرخطی باشد بلکه تخمین این رابطه به صورت $f(a + b) = g(f(a)) + g(f(b))$ نیز غیرممکن باشد. به‌عنوان مثال تابع x^2 اگر چه غیرخطّی است ولیکن: $f(a + b) = f(a) + f(b) + 2a.b$ بنابراین این رابطه‌ی غیرخطی، شرط «به شدت غیرخطی بودن» را ندارد. رابطه‌ای مثل $f(x) = \sqrt{(Lnx + x^2)}$ می‌تواند به شدت غیرخطی تلقی شود. اگر یای انحصاری

[1] Avalanche Effect

XOR عمل جمع به پیمانه‌ی ۲ فرض شود عمل جایگشت نسبت به این جمع خطّی و عمل جانشینی نسبت به این جمع به شدت غیرخطی است مشروط بر آن‌که جداول جانشینی به درستی انتخاب شده باشند.

- **برابری طول خروجی با ورودی:** سیستم رمزنگار متقارن حق ندارد طول داده‌ها را افزایش بدهد یا از آن بکاهد. (طول ورودی و خروجی ضمن هم‌اندازه بودن عموماً ثابت است).
- **عدم امکان مدل‌سازی رمزنگار با روابط جبری:** الگوریتم باید چنان پیچیده و گمراه‌کننده باشد که هرگز نتوان آنرا با رابطه‌ای جبری (حتی رابطه‌ای تقریبی) مدل کرد.
- **قدرت تمام کلیدها:** الگوریتم رمزنگاری متقارن بایستی به‌گونه‌ای طراحی شده باشد که انتخاب هر کلید از فضای حالات ممکن تفاوتی نداشته باشد وچیزی به نام کلیدهای ضعیف و قوی مطرح نباشد. به عنوان نمونه هرگاه کسی بیت‌های کلید را تماماً صفر یا یک فرض کرد خروجی نباید از خود الگویی منظم و قابل پیش-بینی نشان دهد.

۳-۷- نتیجه‌گیری

در علم پردازش تصویر، رنگ و تصاویر اساس کار با سیستم‌ها و ماشین‌های بینایی را تشکیل می‌دهند. در سیستم‌های پردازش تصویر ورودی رنگ و تصویر است که بعد از تبدیل آن‌ها به سیگنال‌های دیجیتال با الگوریتم‌ها و فیلترهای متنوع آن‌ها را قابل کنترل می‌نماید. از آن‌جایی که نرم‌افزار متلب تمامی تصاویر را به صورت ماتریس شناسایی می‌کند و جعبه‌ابزار قدرتمندی برای پردازش تصویر دارد، در این پروژه از آن استفاده‌شده است. در این فصل در مورد مفاهیم رنگ و تصویر به صورت مختصری توضیح داده‌شد. سپس در مورد برخی اصول الگوریتم‌های رمزنگاری متقارن از جمله گمراه‌کنندگی و پخش و پراکنده‌سازی و... صحبت شد. این ویژگی‌ها و مضامین آن‌ها زیربنای الگوریتم‌های ذکر شده در این پروژه هستند.

فصل چهارم

الگوریتم‌های رمز قطعه‌ای و شیوه‌های رمز

۴-۱- مقدمه

در ابتدای دهه‌ی هفتاد دولت فدرال آمریکا و شرکت آی‌بی‌ام مشترکاً روشی را برای رمزنگاری داده‌ها ایجاد کردند تا به عنوان استانداردی برای محرمانه نگه‌داشتن اسناد دولتی مورد استفاده قرار بگیرد؛ این روش استاندارد رمزنگاری داده یا DES نام‌گرفت. این الگوریتم در این فصل توضیح داده‌خواهد شد. به دلایلی که ذکر خواهد شد، الگوریتم DES قربانی حمایت دولت فدرال آمریکا شده‌بود ولی حتّی اگر پشتیبانِ بنیان برافکن را نمی‌داشت باز هم نمی‌توانست در هزاره‌ی سوّم موقعیتی مناسب در دنیای فناوری اطلاعات برای خود دست‌وپا کند. سازمان NIST[1] نخواست که تجربه ناموفّق DES را تکرار کند، لذا با برگزاری یک رقابت چندمرحله‌ای که از ژانویه‌ی ۱۹۹۷ شروع شد و تا نوامبر ۲۰۰۱ ادامه داشت، تلاش کرد استاندارد جدید رمزنگاری را از بین طرح‌های پیشنهادی شخصیت‌های حقیقی یا حقوقی برگزیند تا هرگونه شائبه‌ی دخالت در فرآیند طراحی و وجود رخنه Backdoor در آن‌را از میان بردارد. پس از طی مراحل چندگانه‌ی این رقابت، دو جوان بلژیکی به نام‌های Rijmen و Daemen پشت حریفان سنگین وزنی مثل RSA و IBM را به خاک فشردند و پیروز رقابت شدند. الگوریتم این دو پژوهشگر جوان توسط NIST استانداردسازی و در سند FIPS 197 تحت نام استاندارد پیشرفته‌ی رمزنگاری عرضه شد. به نظر می‌آید استقبال عمومی از این روش بسیار چشمگیر بوده و پیاده‌سازی سخت‌افزاری و نرم‌افزاری آن در طی این زمان کوتاه در انواع تکنولوژی‌های مرتبط با امنیت اطلاعات جایگاه ویژه‌ای پیدا کرده‌است. این روش نه‌تنها DES را که الگوریتم‌های دیگری مثل RC4 را نیز به حاشیه رانده است.

در این فصل با الگوریتم استاندارد رمزنگاری داده و راین‌دال و نیز الگوریتم سِرپِنت آشنا و سپس به شیوه‌های متفاوت رمزنگاری قطعه‌ای اشاره خواهد شد.

---

[1] National Institute of Standards and Technology

۴-۲- استاندارد رمزنگاری داده DES

۴-۲-۱. خصوصیات الگوریتم

در سال ۱۹۷۲ موسسه بین‌المللی استاندارد و فناوری آمریکا اعلام کرد که به یک الگوریتم برای حفاظت از اطلاعات غیر رده‌بندی خود نیاز دارد. این الگوریتم می‌بایست ارزان قابل دسترس و بسیار مطمئن می‌بود. سپس فراخوانی برای چنین الگوریتمی اعلام نمود ولی هیچ‌یک از الگوریتم‌هایی که در پاسخ ارائه شدند شرایط لازم را نداشتند. در سال ۱۹۹۶ الگوریتم DES به عنوان یک استاندارد به ثبت رسید. این الگوریتم رشته‌ای از متن اصلی با طول ثابت را به عنوان ورودی دریافت و توسط کلیدی، متن رمز را تولید می‌کند. در این الگوریتم طول قطعات پیام و کلید ۶۴ بیتی است که در عمل تنها از ۵۶ بیت آن استفاده می‌شود و بقیه ۸ بیت به عنوان بیت‌های پرچم استفاده می‌شوند (در ابتدای تابع توسیع کلید این بیت حذف خواهند شد). الگوریتم شامل ۱۶ مرحله‌ی مشابه است. متنی که قرار است رمزگذاری شود، ابتدا در معرض یک جایگشت اولیه[1]، سپس یک سری اعمال پیچیده وابسته به کلید و در نهایت در معرض یک جایگشت نهایی[2] قرار می‌گیرد. عملیات IP و FP معکوس هم هستند و در حقیقت FP عملی که توسط IP انجام می‌شود را خنثی می‌کند. بنابراین از جنبه‌ی رمزنگاری اهمیت چندانی ندارند و برای تسهیل نمودن بار کردن قطعات داده در سخت‌افزارهای دهه ۱۹۷۰ استفاده شدند ولی اجرای DES در نرم‌افزار را کُند می‌کردند. قبل از دور اول، داده به دو بخش ۳۲ بیتی تقسیم می‌شود که این دو نیمه به طور متناوب مورد پردازش قرار می‌گیرند این تقاطع به عنوان شکل فیستل شناخته می‌شود. ساختار فیستل تضمین می‌کند که رمزگذاری و رمزگشایی دو رویه کاملاً مشابه هستند و تنها تفاوت آن‌ها این است که زیرکلیدها در زمان رمزگشایی در جهت معکوس رمزگذاری به کاربرده می‌شوند و بقیه‌ی الگوریتم در هر دو یکسان است که این امر پیاده‌سازی را به خصوص در سخت‌افزار بسیار آسان می‌کند و دیگر نیازی به الگوریتم‌های متفاوت برای رمزگذاری و رمزگشایی نیست. تابع $F(R_{i-1}, K_n)$ یک تابع غیرخطی مشتمل بر عملیات توسیع، جانشینی، XOR و جایگشت است؛ پیچیدگی و استحکام DES از همین تابع منشا گرفته است. این تابع شامل مراحل زیر است [۴]:

- بسط: با استفاده از یک جایگشت انبساطی ۳۲ بیت به ۴۸ بیت گسترش داده می‌شود.

---

[1] Initial Permutation (IP)
[2] Final Permutation (FP)

- ترکیب کلید: در این مرحله حاصل مرحله‌ی قبل با یک زیرکلید XOR می‌شود. ۱۶ کلید ۴۸ بیتی با استفاده از الگوریتم توسیع کلید، از کلید اصلی تولید می‌شود.
- جایگزینی: بعد از ترکیب کلید هر قطعه داده به هشت بخش ۶ بیتی تقسیم می‌شود (قبل از پردازش توسط جعبه‌های جایگزینی) هر کدام از S-Boxها ورودی ۶ بیتی خود را با استفاده از یک تبدیل غیرخطی به یک خروجی ۴ بیتی تبدیل می‌کند S-Boxها قلب هستند و بدون آن‌ها رمز، خطی خواهد بود و در نتیجه قابل شکستن است.
- جایگشت: در نهایت ۳۲ بیت خروجی S_Boxها با استفاده از یک جایگشت ثابت دوباره سازماندهی می‌شود. برای مشاهده‌ی دقیق‌تر می‌توانید به کُد اصلی برنامه واقع در فصل پنجم مراجعه کنید.

۴-۲-۲. امنیت DES

اساسی‌ترین حمله برای هر رمزی، امتحان‌کردن کلیه‌ی مقادیر ممکن است. طول کلید، تعداد مقادیر ممکن برای کلید و همچنین عملی‌بودن این روش را مشخص می‌کند. تردیدی که از ابتدا و حتی قبل از اینکه DES به عنوان استاندارد شناخته‌شود در مورد آن وجود داشت کافی بودن طول کلید بود. NSA، شرکت IBM را به کاهش طول کلید از ۱۲۸ بیت به ۶۴ بیت و سپس به ۵۶ بیت وادار نمود. طرح‌های متنوعی برای یک ماشین که قادر به شکستن کلیدهای DES باشد مطرح گردیده است. ماشینی که در سال ۱۹۹۳ طراحی شد، یک میلیون دلار قیمت داشت و کلید را در هفت ساعت می‌یافت. همچنین پروژه‌های دیگری طرح شدند ولی هیچ یک به صورت عملی پیاده نشدند. در سال ۱۹۹۷ پروژه‌ی DESCHALL موفق شد با استفاده از زمان بیکاری هزاران کامپیوتر در اینترنت پیغام رمزشده توسط این الگوریتم رمز را در انظار عمومی بشکند. عملی بودن شکستن DES با اختراع یک DES-Cracker توسط تیم EFF درسال ۱۹۹۸، در فاصله‌ی زمانی ۵۶ ساعت، بر همگان روشن شد و انگیزه‌ی این تیم نشان دادن عملی شدن تئوری شکستن DES بود. ماشین COPOCOBANA که در یکی از دانشگاه‌های آلمان ساخته‌شد، شامل صد و بیست عدد FPGA (مدار مجتمع با قابلیت برنامه‌ریزی دوباره) است که به طور موازی با هم کار می‌کنند. یکی از جنبه‌های جالب این ماشین فاکتور هزینه‌ی آن است که با کاهش هزینه‌ای با ضریب ۲۵ نسبت به EFF نشان‌دهنده‌ی پیشرفت‌های متوالی در زمینه‌ی سخت‌افزارهای دیجیتالی است. این ماشین در نُه روز رمز را می‌شکند. پس از یک سال پیشرفت در زمینه‌ی نرم‌افزار، این زمان را به شش روز کاهش داد [۴].

پس از این روش 3-DES مطرح شد که روش جدیدی نبود و حاصل تلاش شرکت IBM برای افزایش موثر طول کلید و ایجاد اطمینان بیشتر در الگوریتم DES بود؛ در این الگوریتم داده‌ها به کمک دو عدد کلید ۵۶ بیتی سه بار رمزنگاری می‌شوند. فضای کلید از ۵۶ بیت به ۱۱۲ بیت افزایش‌یافته و دست هیچ ماشین رمزشکنی به آن نمی‌رسد. در این روش ابتدا بلوک ۶۴ بیتی ورودی با کلید K_1 رمزشده سپس حاصل این مرحله با کلید K_2 رمزگشایی می‌شود (این رمزگشایی هیچ فرقی با رمزنگاری ندارد) و در آخر بار دیگر حاصل با کلید K_1 رمزنگاری می‌شود. تا نتیجه‌ی رمزشده بدست آید. دلیل آن‌که در مرحله‌ی دوم به جای رمزنگاری از رمزگشا استفاده شده است آنست که هر گاه کلید K_1 و K_2 مثل هم باشند دو بلوک اول تاثیر یکدیگر را خنثی کرده و این روش به همان DES معمولی تک‌کلیدی تبدیل می‌شود.

۴-۳- استاندارد پیشرفته‌ی رمزنگاری[1]

۴-۳-۱. خصوصیات الگوریتم

روش رمزنگاری راین‌دال[2] به چند دلیل در دنیای «رمزنگاری متقارن» یک نقطه‌ی عطف یا بهتر است گفته شود یک شگفتی به حساب می‌آید [۶]:

الف) این روش به هیچ عنوان از الگوی سنتی روش‌های فیستلی پیروی نکرده و مبتنی بر حالت خاصی از «میدان گالوا[3]» است. این سنت‌شکنی بسیار جالب، حوزه‌ی جدیدی را در دنیای رمزنگاری کلید متقارن گشود. دو ابداع‌کننده‌ی جوان این الگوریتم رویکردهایی را که در برخی از روش‌های موجود زمان (مثل RC4) بسیار جالب و رازآلود به نظر می‌آمدند و به نحوی در روش‌های دیگر تقلید شده بودند، به مسخره گرفتند و آن‌ها را یک لطیفه خواندند تا بُتِ روش‌هایی را که ذاتی فیستلی هستند و فقط در جزییات اختلافات مضحک دارند، بشکنند[۱۳].

ب) انتخاب این روش بعنوان استاندارد دولت فدرال آمریکا در فضایی آزاد و بدون اعمال نفوذ عوامل جاسوسی یا امنیتی ایالات متحده صورت گرفته و تحت قوانین غیرانحصاری به ثبت رسیده‌است لذا بهره‌برداری از آن در محصولات مختلف آزاد است [۱].

---

[1] AES(Advanced Encyption Standard)
[2] Rijndael
[3] Galios Field

ج) AES گذشته از امنیّت بسیار بالا، از دیدگاه «سرعت»، «سادگی پیاده‌سازی»، «فضای حافظه‌ی موردنیاز» و «قابلیت انعطاف» روش شگفت‌انگیزی است [۱].

نسخه‌ی اصلی الگوریتم راین‌دال (که پس از انتخاب به عنوان استاندارد دولت فدرال و ثبت در سند FIPS-197 به نام AES معرفی شد) دارای یک اختلاف بسیار جزیی با AES است:

- در راین‌دال طول کلید و طول بلوک داده می‌تواند ۱۲۸، ۱۹۲، ۲۵۶ بیت باشند و طول کلید و طول قطعات داده مستقل از هم قابل انتخابند لذا می‌توان گفت که راین‌دال دارای ۹ انتخاب متفاوت برای رمزنگاری اطلاعات است.
- در AES طول بلوک داده صرفاً باید ۱۲۸ بیتی (معادل ۴ کلمه‌ی ۳۲ بیتی) باشد لذا در انتخاب طول بلوک داده هیچ انتخابی و جود ندارد ولی کماکان می‌توان طول کلید را از بین مقادیر ۱۲۸، ۱۹۲، ۲۵۶ بیتی انتخاب کرد. بدین ترتیب AES کلاً دارای ۳ انتخاب است.

آنچه در این پروژه پیاده‌سازی خواهد شد ابتدا الگوریتم رمز بلوکی AES است که بسیار ساده و سرراست می‌باشد و تشریح آن تقریباً از همه‌ی روش‌های موجود ساده‌تر به نظر می‌رسد. این الگوریتم به گونه‌ای در کُد آن مشاهده خواهید کرد دارای چهار عملکرد اصلی زیر است:

- جانشینی بایت Byte Substitution
- شیفت چرخشی کلمات به اندازه‌ی یک بایت Shift Rows
- تلفیق و درهم سازی ستونی Mix Columns
- جمع (Xor) کلید با کلمات در هر دور

در این تحقیق برای سادگی پیاده‌سازی و نیز کم‌کردن حجم محاسبات پردازنده و بدست‌آوردن سریع‌تر جواب، پیاده‌سازی برای داده و کلیدهای ۱۲۸ بیتی، که تعداد دور اجرای الگوریتم در این حالت ده دور می‌شود، انجام شده است. در کُد الگوریتم AES ابتدا با تعریف چهار ثابت ابتدایی، تعداد بایت‌های داده معادل «۱۶ بایت/ ۱۲۸ بیت» (در ماتریسی ۴×۴) و تعداد دورها ۱۰ دور در نظر گرفته شده‌است. از آن‌جا که در AES داده‌ها در متغیر ماتریسی با چهار سطر پردازش می‌شود و با توجه به ۱۶ بایتی بودن داده‌ها، ماتریس حاوی داده، دارای چهار سطر و چهار ستون خواهد

بود و طبق استاندارد تعداد دفعات پردازش ده‌مرتبه است که همه‌ی این‌ها در تعریف ابتدایی کُد آورده‌شده‌است. در همین‌جا اولین اختلاف AES و DES بروز می‌کند؛ AES بر روی کلمات کار می‌کند و DES بر روی بیت‌ها!

تابع رمزنگار AES دارای سه آرگومان است:

PlainText: آرایه‌ای (۴×۴) به طول ۱۶ بایت حاوی داده‌های رمزنشده‌اصلی

CipherText: آرایه‌ای (۴×۴) به طول ۱۶ بایت برای برگرداندن معادل رمزنگاری شده‌ی داده‌های ورودی

Key: آرایه‌ای (۴×۴) به طول ۱۶ بایت حاوی کلید رمزنگاری

متغیر دو بعدی(ماتریس) state یک آرایه‌ی (۴×۴) است که داده‌های ورودی آن منتقل و کلیه‌ی پردازش‌ها بر روی آن انجام می‌گیرد. این متغیر «متغیر حالت» نام‌گذاری می‌شود.

متغیر RoundKey که دارای ۱۱ عدد ماتریس ۴×۴ است و هر یک از ماتریس‌ها باید در یک دور از اجرای الگوریتم استفاده و با متغیر حالت XOR شوند. از آن‌جا که فقط یک شاه‌کلید ۱۶ بایتی وجود دارد، مابقی ده-کلیدفرعی، طبق الگوریتم نسبتاً پیچیده‌ای از روی شاه‌کلید اصلی تولید می‌شود. (فرآیند تولید کلیدهای فرعی، مستقل از روش رمزنگاری است و در بخشی مستقل بدان پرداخته خواهد شد).

حال برنامه چگونه عمل می‌کند:

- ابتدا با تابعی اولیه، جدول‌های S-BOX و Inv-S-BOX آن‌را تولید کرده و سپس با تابع KeyExpansion شاه‌کلید (یعنی متغیر Key) به یازده کلید فرعی توسعه داده‌خواهد شد و درون متغیر RoundKey قرار می‌گیرد تا در بطن برنامه از آن‌ها استفاده شود.
- در مرحله‌ی بعدی در تابع رمزنگاری AES، متن اصلی به درون متغیر حالت منتقل می‌شود تا درخلال ده دور متوالی پردازش شود.
- قبل از شروع حلقه‌ی تکرار RoundKey[0] با متغیر حالت، بایت به بایت یای انحصاری شود.
- در این لحظه زمان شروع محاسبات اصلی فرا می‌رسد. حلقه‌ی تکرار ده دور محاسبات رمزنگاری را انجام می-دهد. هر دور شامل چهار عمل است:

الف) جانشینی متغیر حالت: این تابع یکایک بایت‌های ماتریس متغیر حالت را براساس یک جدول جانشینی و مشخص با مقادیر جدید جایگزین می‌کند. «جدول جانشینی بایت» در این الگوریتم دارای ۲۵۶ درایه[1] است که در ماتریس ۱۶× ۱۶ سازماندهی شده‌اند. برای جایگزین کردن بایت با مقدار معادل، چهر بیت پرارزش آن بایت به عنوان شماره‌ی سطر و چهار بیت کم‌ارزش به‌عنوان شماره‌ی ستون به این جدول اعمال شده و درایه‌ی متناظر با آن بجای مقدار اصلی قرار می‌گیرد. برخلاف روش استاندارد رمزنگاری داده، که دارای ۸ جدول جانشینی برای هر «دور» است، این الگوریتم در این مرحله فقط دارای یک جدول جانشینی است که یکی از نقاط قوّت این الگوریتم به شمار می‌آید؛ زیرا نیاز به حافظه‌ی نوع فقط خواندنی را در پیاده‌سازی سخت-افزاری آن کاهش خواهد داد (کلّ این جدول به ۲۵۶ بایت فضای حافظه نیاز دارد و با حداقل مدار منطقی قابل پیاده‌سازی است). تمام بایت‌های متغیر حالت طبق همین الگو با مقدار معادل، جایگزین می‌شوند تا ماتریس جدید متغیر حالت به دست‌آید. از آن‌جا که جانشینی هر بایت مستقل از دیگری انجام می‌گیرد لذا برای رمزنگاری فوق‌سریع، می‌توان برای انجام کلّ این جانشینی در یک سیکل ماشین، به تکنیک‌های «موازی‌سازی سخت‌افزار» متوسل شد.

ب) چرخش سطرها: این تابع هر چهار سطر از آرایه‌ی متغیر حالت را به سمت چپ می‌چرخاند؛ سطر شماره‌ی یک تغییری نمی‌کند و چرخشی ندارد، سطر شماره‌ی دو، یک درایه بصورت چرخشی به سمت چپ می-چرخد. سطر شماره‌ی سه و چهار به ترتیب دو و سه درایه به سمت چپ می‌چرخند. این الگو برای چرخش بایت‌ها بدان دلیل انتخاب شده که چون محاسبه‌ی تلفیقی و درهم سازی داده‌ها بصورت ستونی انجام می‌گیرد لذا اگر این «چرخش سطری» وجود نداشت کلّ عملیات بر روی ستون‌های ۳۲ بیتی متمرکز می‌شد؛ چنین چیزی معادل است با رمزنگاری بلوک‌های ۳۲ بیتی اطلاعات!

ج) تلفیق ستون‌ها: این تابع هر ستون از آرایه‌ی حالت را بطور مستقل از دیگر ستون‌ها تلفیق و درهم‌سازی می‌کند. در فرآیند «تلفیق» هر ستون از متغیر حالت، در یک ماتریس ثابت ضرب می‌شود تا ستون جدید به دست آید؛ عمل ضرب ماتریسی، بر روی $GF(2^8)$ که میدان «محدود گالوا» است صورت می‌گیرد. (عملیات در این میدان دارای مبانی ریاضیات بسیار قوی هستند). خوشبختانه بدلیل کوچک بودن محدوده‌ی اعداد یک بایتی (در میدان 2^8) می‌توان عمل تلفیق ستونی را با انجام مجموعه‌ای از پیش‌محاسبات، درون جداولی ذخیره

[1] Entry

کرد. در این صورت، عمل ضرب در این میدان با دو عمل «جستجو» lookup و یک عمل قابل محاسبه است.

آن‌چه که AES را از دیگر روش‌های موجود متمایز ساخته است زیربنای ریاضی «عملیات تلفیق و درهم‌سازی» است وگرنه مابقی مراحل پردازش شامل جانشینی ، شیفت چرخشی و XOR در تمام روش‌ها به صور مختلف وجود دارند.

عمل تلفیق و درهم‌سازی ستونی، بطور یکتا وارون‌پذیر است. وارون‌پذیری این عمل پس از آشنایی با میدان‌های گالوا و چند جمله‌ای‌هایی که بر روی این میدان تعریف می‌شوند. به سادگی قابل اثبات است.

در آخرین دور از رمزنگاری عمل تلفیق و درهم‌سازی ستونی انجام نمی‌شود. این قرارداد پیاده‌سازی الگوریتم رمزگشایی را ساده‌تر می‌کند. به عبارتی با تغییر در ثابت‌های تابع رمزنگار می‌توان از آن برای رمزگشایی داده‌ها نیز بهره گرفت.

د) Xor_Roundkey_inTo_State: در چهارمین مرحله و آخرین مرحله‌ی هر دور «کلید فرعی متناسب با آن دور» با داده‌ها XOR خواهد شد. این تابع محتویات متغیر حالت را با محتویات کلید متناظر با دور بایت به بایت XOR می‌کند. در دورهای بعدی همین چهار عمل متوالیاً بر روی محتویات ماتریس حالت انجام می‌گیرد. پس از اتمام دورها، کلّ عملیات رمزنگاری بلوک داده‌ی ۱۲۸ بیتی به پایان می‌رسد و حاصل رمزنگاری داده‌ها که درون متغیر حالت قرار دارد به متغیر ciphertext منتقل می‌شود.

وارون‌پذیری عملیات جانشینی، شیفت‌چرخشی و XOR مثل روز روشن است. پس از اثبات آن‌که عملیات تلفیق و درهم‌سازی ستونی نیز وارون‌پذیر است. وارون‌پذیری روش رمزنگاری AES دارای توجیه خواهد بود.

روش AES ضمن تضمین امنیت بسیار بالا، سرعت بسیار عالی را نیز تضمین می‌کند. پیاده‌سازی این روش بر روی ماشین‌های ۸ بیتی، ۱۶ بیتی، ۳۲ بیتی بصورت کارآمد و موثر امکان‌پذیر است. سخت‌افزار بی‌درنگ AES می‌تواند رمزنگاری بیش از صد کانال رمزنگاری ویدئویی MPEG-2 را بصورت همزمان انجام بدهد.

۴-۳-۲. توصیف ریاضی مختصری از الگوریتم AES

کوچکترین واحد اطلاعات در الگوریتم AES بایت است. هرگاه یک بایت به شکل دودویی در قالب $\{b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\}$ نشان داده‌شود که مقادیر b_i صفر یا یک هستند نمایش ریاضی آن در میدان GF(2) به صورت زیر خواهد بود [۶]:

$$b_0 + b_1 X + b_2 X^2 + b_3 X^3 + b_4 X^4 + b_5 X^5 + b_6 X^6 + b_7 X^7 = \sum_{i=0}^{7} b_i \ X^i$$

رابطه‌ی ۴-۱ نمایش ریاضی یک بایت در میدان گالوا

به عنوان مثال بایت نمایش چندجمله‌ای: $\{01100011\} \equiv X^6+X^5+X+1$

جمع بایت‌ها: در نمایش چندجمله‌ای برای بایت‌ها، عمل جمع دو بایت، با XOR ضرایب جمله‌های «هم‌توان» انجام می‌گیرد[۶]: $\{X^6+X^5+X^2+X+1\} + \{X^7+X+1\} = X^7+X^6+X^5+X^2$

یا به شکل ساده‌تر : $\{01010111\}$ XOR $\{10000011\} = \{11010100\}$ که در مبنای شانزده به شکل: $\{57\}$ XOR $\{83\} = \{d4\}$ است.

هر سه عبارت بالا هم‌ارزند. براحتی می‌توانید نتیجه بگیریدکه پیاده‌سازی عمل جمع دو بایت که در میدان GF(2) مدل شده‌اند. همان عمل XOR است که در تمام پردازنده‌ها در قالب یک دستور ماشین انجام می‌گیرد و هیچ پیچیدگی خاصی ندارد؛ عمل ضرب اندکی پیچیده‌تر است [۶].

ضرب بایت‌ها: در نمایش چندجمله‌ای بایت‌ها عمل ضرب دو بایت (که با نماد • نمایش داده می‌شود). معادل است با ضرب چندجمله‌ای‌های متناظر با آن‌هاست. از آن‌جا که حاصل‌ضرب دو چندجمله‌ای که بزرگ‌ترین توان هر یک از آن‌ها می‌تواند از مرتبه‌ی X^7 باشد، حداکثر دارای درجه‌ی ۱۴ است لذا عمل ضرب چندجمله‌ای در میدان $GF(2^8)$ به صورت «ضرب چند جمله‌ای متناظر با بایت‌ها به پیمانه‌ی یک چندجمله‌ای تحول‌ناپذیر از درجه‌ی ۸» تعبیر می‌شود؛ یعنی [۶]:

$$C(x) = a(x) \bullet b(x) \mod m(x)$$

رابطه‌ی۴-۲ عمل ضرب در میدان گالوا

در رمزنگاری AES عمل ضرب دو بایت در میدان $GF(2^8)$ به پیمانه‌ی $m(x)=X^8+X^4+X^3+X+1$ انجام می‌گیرد. از آن‌جا که باقی‌مانده تقسیم هر چندجمله‌ای بر $m(x)$ دارای درجه‌ای کمتر از ۸ خواهد بود لذا ضرب دو بایت ضمن وارون‌پذیری محض، نتیجه‌ای خواهد داشت که به صورت هشت بیت قابل نمایش است [۶].

ضرب یک چند جمله‌ای در x، در میدان گالوا $GF(2^8)$ به پیمانه‌ی $m(x)=x^8+x^4+x^3+x+1$، عبارتست از یک بیت شیفت بایت به سمت چپ و سپس یک عمل XOR مشروط به یک بودن بیت نقلی حاصل از شیفت [۶].

الگوریتم رمزگشایی با رمزنگاری تفاوت بنیانی ندارد مگر تغییر در ثابت‌ها و «کلیدهای دور» ! وارون عمل AddRoundKey یا همان XOR تکرار آن است زیرا: (a XOR k) XOR k =a. برای وارون‌پذیری عمل جانشینی بایت‌ها، کافی است جدول جانشینی با مقادیر مناسب پر شود و از لحاظ ماهیت پیاده‌سازی (چه سخت‌افزاری و چه نرم‌افزاری) تفاوتی ندارد؛ (یعنی در رمزنگاری و رمزگشایی فقط مقادیر جدول جانشینی فرق دارد). وارون عمل شیفت چرخشی، باز هم شیفت چرخشی است! برای وارون‌سازی Mix-Columns، کافی است همین عمل با مقادیر «ماتریس وارون» یا a^{-1} از نو تکرار شود.

۴-۳-۳. توسیع کلید در AES

در این الگوریتم مثل بسیاری از الگوریتم‌های دیگر، تنها یک «شاه‌کلید» وجود دارد که کلیه‌ی «کلیدهای دور» طبق یک الگوریتم پیچیده‌ی وارون‌ناپذیر از روی آن ساخته می‌شود. کُد این الگوریتم توسیع‌کلید را توصیف‌کرده‌است؛ دقت‌کنید که تعداد کلیدهای فرعی موردنیاز، به تعداد دورها و تعداد دورها به طول کلید Nk وابسته است: در شرح الگوریتم باید بدین نکته اشاره کرد که در AES-128 طول کلید ۱۲۸ بیت و تعداد دورها ده‌تاست لذا باید ده-کلید فرعی دیگر از روی شاه‌کلید ساخته‌شود. برای تولید اولین کلید فرعی، شاه‌کلید در ستون یک تا چهارم، از یک ماتریس به ابعاد ۴×۴۴ قرار می‌گیرد [۱].

ستون چهارم از کلید را به درون یک آرایه‌ی ۴×۱ منتقل کرده و در اولین گام، به آن یک شیفت چرخشی از پایین به بالا داده‌می‌شود. اگر در تفسیری واقع‌گرایانه‌تر فرض شود ستون چهارم، در یک کلمه‌ی چهاربایتی کپی‌شده، این عمل معادل است با شیفت چرخشی به سمت چپ به اندازه‌ی یک بایت. پس از این چرخش یکایک بایت‌های ستون طبق یک جدول جانشینی ثابت با مقادیر جدید جایگزین می‌شوند. روش اعمال یک بایت به عنوان اندیس جدول جانشینی مثل قبل است: چهار بیت پرارزش به عنوان شماره‌ی سطر و چهار بیت کم‌ارزش به عنوان شماره‌ی ستون به جدول اعمال‌شده و درایه‌ی متناظر با آن جایگزین مقدار قبلی می‌شود [۱].

پس از عمل جانشینی بایت‌ها ستون جدید با ستون اوّل از کلید اصلی XOR می‌شود.

پس از عمل XOR، نوبت به XOR مجدد آن با ستون متناظر از جدولی ثابت به نام R_Con Round Constant است. ستون متناظر از این جدول عبارت است از شماره‌ی ستون فعلی (یعنی ستون شماره‌ی ۴ در حال ساخت) تقسیم‌بر NK، که در این مرحله برابر Rcon[1] می‌شود.

پس از این XOR ستون اول از کلید فرعی حاضر است. سه ستون بعدی راحت‌تر بدست می‌آیند؛ ستون اول از کلید اول و ستون دوم از کلید قبلی، XOR شده و ستون سوم از کلید جدید بدست می‌آید. ستون بدست‌آمده با ستون چهارم از کلید قدیم باز هم XOR شده و ستون سوم از کلید جدید را می‌سازند. ستون به دست‌آمده با ستون چهارم از کلید قبل، پس از XOR شدن، ستون چهارم از کلید جدید را می‌سازد. بدین ترتیب چهار ستون از کلید فرعی اوّل محاسبه می‌شود.

برای محاسبه‌ی دومین کلید فرعی کافی است کلید محاسبه‌شده‌ی قبلی را کلید اصلی فرض کرده و همین روال را از ابتدا تکرار کنید تا کلید دوم بدست آید.

۴-۴- مار بزرگ: رتبه‌ی دوّم!

۴-۴-۱. خصوصیات الگوریتم سِـرپِـنت

در مسابقه‌ی گزینش استاندارد AES که به پیروزی الگوریتم راین‌دال انجامید، پیشنهاد دیگری به نام Serpent (اژدها) در رده‌ی دوم قرار گرفت و اگر چه نتوانست استاندارد دولت فدرال آمریکا شود ولی استحکام و قدرت آن تحسین بسیاری از بزرگان رمزنگاری را برانگیخت. این الگوریتم پس از ختم رقابت AES نظرها را به خود جلب کرد. این الگوریتم توسط «راس اندرسون»، «الی بیهام» و «لارس نودسن» ابداع شد تا در رقابت AES شرکت کند. لذا طبق ضوابط این رقابت بایستی از کلیدهای ۱۲۸، ۱۹۲، ۲۵۶ بیتی پشتیبانی می‌کرد و طول بلوک‌های ورودی آن نیز صرفاً ۱۲۸ بیت بود. از آن‌جا که پدیدآورندگان آن طبق گفته‌ی خودشان در ارائه‌ی این الگوریتم بسیار محافظه‌کارانه عمل کرده‌اند لذا برخلاف روش راین‌دال به سراغ عمگرها و عملیات جدید در رمزنگاری نرفتند و سعی کردند از بلوک‌ها و عملگرهایی استفاده کنند که حداقل برای ۲۵ سال مورد آزمایش و حمله قرار گرفته‌بودند؛ از این رو علی‌رغم قدرت استحکام بسیار بالا، در ظاهر بدیع به نظر می‌رسد و کُد اجرایی آن چهار برابر حجیم‌تر از راین‌دال است که امتیاز منفی برای آن به بار آورد و آن را در رتبه‌ی دوّم نشانید! روش سِرپِـنت داده‌ها را در ۳۲ دور متوالی به

کمک شبکه‌ای از بلوک‌های «جانشینی[1]» و «جایگشت[2]»، تلفیق و با کلیدی ۱۲۸ بیتی رمز می‌کند. (تعداد دورها در مقایسه با رانداِل سه برابر بیشتر است!) پردازش داده‌های ورودی در قالب کلمات ۳۲ بیتی انجام می‌گیرد که پیاده-سازی نرم‌افزاری و سخت‌افزاری آن بسیار سریع و بهینه باشد. سِرپِنت در خلال دورهای پردازش جمعاً به ۳۳ کلید فرعی ۱۲۸ بیتی نیازمند است که همگی طبق روالی مجزا، از شاه‌کلید استخراج می‌شوند. این کلیدهای فرعی K_1 تا K_{33} نام‌گذاری می‌شوند. طول کلید اگرچه می‌تواند متغیر باشد ولی برای برآورده شدن شرایط رقابت AES، در سه حالت ۱۲۸، ۱۹۲، ۲۵۶ بیتی تنظیم شده‌است. برای ساده‌تر شدن الگوریتم، کلیدهای کوچک‌تر از ۲۵۶ بیت با افزودن یک بیت ۱ در انتهای سمت چپ کلید و سپس چسباندن بیت صفر به تعداد لازم، طول آن به ۲۵۶ بیت توسعه می‌یابد تا الگوی رمزنگاری همیشه براساس کلیدهای ۲۵۶ بیتی بنا نهاده شود. رمزنگار سِرپِنت از عملیات زیر تشکیل شده است [۱].

۱. جایگشت مقدماتی مشهور به IP[3]

۲. سی‌ودو دور متوالی شامل عملیات تلفیق[4]، جانشینی و تبدیل خطّی[5]

۳. جایگشت نهایی مشهور به FP[6]

- **جایگشت مقدماتی و نهایی:** جایگشت مقدماتی و نهایی کمکی به استحکام روش نمی‌کنند و برای بهینه-سازی و تقارن سیستم رمزنگار تعریف شده‌اند. فرض کنید بلوک ۱۲۸ بیتی داده‌های ورودی، به ترتیب از چپ به راست از یک تا ۱۲۸ شماره‌گذاری شده باشد. هرگاه بلوک داده مشمول جایگشت مقدماتی شود ترتیب بیت‌های خروجی تغییر خواهد کرد. یعنی بیت شماره‌ی دو خروجی همان بیت پنجم از ورودی است و بیت شماره‌ی سی‌و دو از خروجی همان بیت ۱۲۵اُم است. به گونه‌ای که مشهود است جایگشت مقدماتی در سِرپِنت روالی کاملاً منظم و غیرتصادفی دارد. پس از ۳۲ دور متوالی پردازش، نوبت به «جایگشت نهایی» می‌رسد که دقیقاً معکوس جایگشت مقدماتی است. در جایگشت مقدماتی بیت شماره‌ی یک تغییر موقعیت نداشته است لذا در جایگشت معکوس نیز در

---

[1] Substitution
[2] Permutation
[3] Initial Permutation
[4] Mixing
[5] Linear Transformation
[6] Final Permutation

جای خود ظاهر می‌شود ولی چون بیت شماره‌ی دو طبق جدول قبلی در موقعیت سی‌وسوم ظاهر شده، اکنون باید به سرِجای خود برگردد بنابراین بیت شماره‌ی سی‌و سوم در موقعیت دو ظاهر می‌شود [۷].

• **دورهای پردازش:** فرض کنید خروجی بلوک جایگشت مقدماتی را B_1 نامیده و دورهای پردازش از یک تا سی‌و دو شماره‌گذاری شده است. خروجی دور شماره‌ی یک، B_2 و... قرارداد خواهد شد [۷].

الف) در آغاز هر دور، ابتدا ورودی ۱۲۸ بیتی با کلید فرعی آن دور، یک XOR ساده می‌شود.

ب) در ادامه، نتیجه‌ی ۱۲۸ بیتی مرحله‌ی قبل به ۳۲ قسمت ۴ بیتی تقسیم و هر یک از دسته‌های چهاربیتی به یک جدول جانشینی S-Box اعمال می‌شوند تا هر یک از دسته‌های چهاربایتی براساس مقدارشان به مقدار جدیدی نگاشته شوند. هر دور به ۳۲ جدول جانشینی نیاز دارد که تمام آن‌ها برای هر دور دقیقاً مثل هم هستند. به عبارت دیگر در هر دور، از ۳۲ نسخه‌ی یکسان از یک جدول جانشینی استفاده شده است و بدیهی است که مقادیر مشابه ورودی، خروجی‌های معادلی را ایجاد خواهد کرد. ممکنست فوراً به این نتیجه برسید که در کل فرآیند رمزنگاری به ۳۲ جدول جانشینی نیاز است ولیکن برای صرفه‌جویی در فضای حافظه، فقط هشت جدول جانشینی تعریف شده که با چهار بار تکرار، جمعاً ۳۲ جدول برای تمام دورها فراهم شده است. به عبارت دیگر اگر برای دور اول جدول S_1 برای دور دوم جدول S_2 و... برای دور هشتم مجدداً از جدول S_1 استفاده می‌شود و روال به همین ترتیب ادامه می‌یابد. لذا در کلّ الگوریتم به هشت جدول جانشینی S_1 تا S_8 احتیاج است. پس از فرایند جانشینی نوبت به تبدیلی می‌رسد که «تبدیل خطی» نام گرفته است. به دلیل این‌که جداول جانشینی به شدت غیرخطّی عمل می‌کنند ولی در بلوک «تبدیل خطّی» هیچ اثری از عمل جانشینی نیست پس می‌توان آن‌را خطی تصور کرد.

• **تبدیل خطّی:** خروجی ۱۲۸ بیتی مرحله‌ی جانشینی به بلوک «تبدیل خطّی» وارد می‌شود؛ این بلوک مجموعه‌ای از عملیات شیفت چرخشی و XOR است. ورودی ۱۲۸ بیتی به چهار دسته‌ی ۳۲ بیتی تقسیم و پردازش می‌شوند؛ اگر این بلوک‌ها به ترتیب X_1،X_2، X_3، X_4 نامیده شوند. فرآیند تلفیق‌داده‌ها که در طی «تبدیل خطی» جمعاً در ۱۰ دستورالعمل انجام می‌گیرد و هرگاه پردازنده قادر به انجام عملیات موازی باشد، هر جفت دستورالعمل متوالی را می‌توان به طور همزمان انجام داد [۷].

در آخرین دور از پردازش، عمل «تبدیل خطّی» انجام نمی‌شود؛ به همین دلیل در کُد برنامه همان‌گونه که مشاهده خواهید کرد، شرطی گذاشته شده که در دور سی و دوّم، بدون انجام عمل تبدیل خطّی، داده‌ها را جهت انجام آخرین عمل XOR و جایگشت نهایی به بلوک بعدی هدایت کند؛ دلیل این عمل مشخص است: چشمپوشی از آخرین تبدیل، سیستم رمزنگار را نسبت به وسط، متقارن و براحتی می‌توان با تغییر در کلیدها از همین معماری برای رمزگشایی نیز بهره گرفت [۷].

برخلاف آن‌چه که در ظاهر به نظر می‌رسد به نظر می‌رسد الگوریتم رمزنگاری سِرپِنت شباهت زیادی به DES دارد و طراحان روش به این شباهت اذعان دارند ولی در عین حال تصریح می‌کنند که فقط شانزده دور از سی‌و دو دور از سِرپِنت امنیتی برابر با 3-DES دارد؛ بنابراین کلّ سی‌و دو دور آن، سِرپِنت را به قوی‌ترین روش دنیا تبدیل خواهد کرد. در ضمن طبق ادعای طراحان روش و تصدیق برخی از بررسی‌کنندگان، کلّ عملیاتی که در هر دور از سِرپِنت بر روی داده‌ها انجام می‌گیرد، دو برابر سریع‌تر از DES است لذا سرعت سِرپِنت (با قدرتی که با افزایش دورها و پیچیده‌تر شدن عملیات تلفیق به آن بخشیده است) هرگز از DES کم‌تر نیست [۱].

گرچه سِرپِنت از راین‌دال شکست خورد و از دریافت لقب استاندارد پیشرفته‌ی رمزنگاری AES بازماند ولی نباید این نکته را فراموش کرد که سِرپِنت، سرعتی نزدیک به DES ولی امنیتی بسیار بالاتر از 3-DES فراهم می‌کند ولی هرگز به زیبایی و سرعت راین‌دال نیست [۷].

۴-۴-۲. نگاهی به معیارهای انتخاب S-BOX ها

جداول جانشینی موسوم به S-BOX در سِرپِنت چهاربیتی هستند و طبعاً یک مقدار چهاربیتی (با ۱۶ حالت مختلف) را با مقداری جدید جایگزین می‌کنند. معیارهای انتخاب این جداول به شرح ذیل بوده‌است:

الف) هرگاه دوالگوی چهاربیتی فقط در یک بیت باهم تفاوت داشته باشند خروجی آن‌ها از S-BOX قطعاً بیش از یک بیت تفاوت خواهد داشت [۷].

ب) احتمال وجود رابطه‌ی خطّی بین یک بیت از ورودی با یک بیت از خروجی چیزی معادل $1/2 \pm 1/8$ است. لذا تشخیص یک رابطه‌ی خطی بین بیت‌های ورودی و خروجی در عمل ممکن نیست [۷].

ج) هرگاه شانزده حالت مختلف ورودی و معادل خروجی جانشین‌شده‌ی آن‌ها به صورت جفت $(X_i، Y_i)$ در نظر گرفته شوند رابطه‌ی Y بر حسب X کاملاً غیرخطی و تقریباً از مرتبه‌ی سه است؛ یعنی به سختی می‌توان y را به شکل $ay^3 = bx^2 + cx + d$ مدل کرد (گذشته از آن‌که حل این معادله آسان نیست پس از سی‌و دو دور متوالی هیچ ردّپایی از تاثیر ورودی‌ها و کلید بر روی خروجی باقی نخواهد ماند) [۱ و ۷].

طراحان روش اذعان داشته‌اند که در انتخاب مقادیر S-BOXها از شیوه‌ی رمزنگاری RC-4 (متعلق به آقای «رونالد ری‌وست» که خود نیز در این رقابت حضور داشت و سوم شد!) الهام گرفته‌اند. گذشته از آن‌که مقادیر جداول جانشینی DES و معیارهای انتخاب آقای «هارست فیستل » را نیز در طراحی خود دخالت داده‌اند.

۴-۴-۳. الگوی تولید کلیدهای فرعی از کلید اصلی

روش سِرپِنت به ۳۳ عدد کلید فرعی نیازمند است که همه‌ی آن‌ها از شاه‌کلید ۱۲۸ بیتی استخراج می‌شوند. این ۳۳ کلید فرعی ۱۲۸ بیتی هستند و هرگاه آن‌ها در قالب کلمات ۳۲ بیتی در نظر گرفته شوند، جمعاً به ۱۳۲ کلمه‌ی چهار بایتی نیاز خواهد بود. هر چهار کلمه‌ی متوالی از این ۱۳۲ کلمه، یکی از کلیدهای فرعی مورد نیاز در هر دور را تشکیل می‌دهند: مهم‌ترین نکته‌ای که در مورد روش سِرپِنت اهمیت دارد آن است که طول کلید خواه ۱۲۸ بیت و خواه ۱۹۲ بیت، ابتدا به طول ۲۵۶ بیت توسعه داده می‌شود تا طول کلید به ۲۵۶ بیت برسد. روش توسعه‌ی کلید با افزودن یک بیت ۱ و سپس چسباندن تعدادی صفر در سمت چپ کلید انجام می‌گیرد. لذا الگوریتم تولید کلیدهای فرعی، مقدار شاه‌کلید اصلی را همیشه به صورت ۲۵۶ بیتی تحویل می‌گیرد [۷].

الگوریتم تولید کلید در فصل پنجم توضیح داده خواهندشد.

۴-۴-۴. رمزگشایی سِرپِنت

طبق سنّت حاکم بر روش‌های رمزنگاری متقارن، فرآیند رمزگشایی به الگویی جدا و مستقل نیاز ندارد بلکه با تغییر در ترتیب و مقدار پارامترهای الگوریتم رمزنگاری، عمل رمزگشایی صورت می‌گیرد. روش سِرپِنت نیز از این مقوله مستثنی نیست. برای بررسی روش رمزگشایی باید الگوی رمزنگاری را پیش رو داشته باشید و فرض نمایید که بخواهید بلوکی رمزشده را از رمز خارج کنید:

الف) با یک نگاه ساده متوجّه خواهید شد که تاثیر جایگشت نهایی در رمزنگاری در همان گام اول از رمزگشایی (یعنی جایگشت مقدماتی) خنثی می‌شود و به هیچ کاری نیاز نیست.

ب) بلوک رمز شده‌ی قبل از جایگشت نهایی با کلید K_{33} یک XOR ساده شده‌بود، بنابراین در اولین مرحله از رمزگشایی باید یک بار دیگر این XOR تکرار شود؛ لذا نتیجه‌ی جالب به دست خواهد آمد: در فرآیند رمزگشایی باید ترتیب کلیدهای K_1 تا K_{33} دقیقاً برعکس شود. حال نتیجه‌ی جالب بعدی این‌که در فرآیند رمزنگاری، در آخرین دور تبدیل خطّی وجود نداشته است و فقط یک جانشینی انجام گرفته و حال باید عکس این عمل صورت بگیرد بنابراین جداول جانشینی در فرآیند رمزگشایی، معکوس جداول جانشینی در رمزنگاری با ترتیب وارونه هستند. یعنی چون در اولی دور از رمزگشایی باید تاثیر آخرین دور از رمزنگاری خنثی شود لذا باید به جای استفاده از S_1 از معکوس S_8 (یعنی $Inv\text{-}S_8$) بهره گرفته شود. بدین ترتیب تاثیر آخرین دور از رمزنگاری خنثی شده است.

ج) حال اولین تبدیل خطی در فرآیند رمزگشایی، باید تاثیر آخرین تبدیل خطّی دردور سی‌و دوم را خنثی کند. پس بلوک‌های سی‌و دو بیتی D_{j+1}، C_{j+1}، B_{j+1}، A_{j+1} یک‌بار دیگر از بالا به پایین به بلوک تبدیل خطّی وارد می‌شوند: چه تغییری در پارامترها باید اتفاق بیفتد که بدون تغییر در ساختار، داده‌ها به شکل اصلی خود برگردند؟ معکوس عمل شیفت چرخشی به سمت چپ باز هم شیفت چرخشی به سمت چپ است ولی باید مقدار شیفت تغییر کند. معکوس عمل XOR نیز یک XOR دیگر با مقدار قبلی است. برای بازگشت A_{j+1}، C_{j+1} به مقدار اولیه‌ی دو عمل شیفت و یک XOR (که خودشان معکوس خودشان هستند) کفایت می‌کند مشروط بر آن‌که مقدار شیفت و مقدار XOR به درستی انتخاب شود فلذا هیچ نیازی به تغییر در ساختار نیست. به همین ترتیب B_{j+1}، D_{j+1} برای بازگشت به حالت قبل به XOR و شیفت نیازمندند.

پس از معکوس شدن فرآیند تبدیل خطّی، دور جدید رمزگشایی باز هم از مرحله‌ی بعد با اعمال جداول جانشینی معکوس از شروع شده و این فرآیند دقیقاً عین روش رمزنگاری تا سی‌و دو دور ادامه می‌یابد.

روش سِرپِنت پس از برنده نشدن در رقابت AES تقریباً به حاشیه رفته است و به ندرت در جایی از آن استفاده می‌شود ولی رتبه‌ی دوّم شدن آن در این رقابت و حضور شخص بیهام (به عنوان یکی از بزرگان رمزشکنی

جهان) در تیم طراح به سِرپِنت اهمیت آکادمیک ویژه‌ای می‌بخشد. این الگوریتم در جایی به ثبت نرسیده و در استفاده از آن برای عموم آزاد است [۱].

با یک نگاه کلّی به جداول «جایگشت مقدماتی» و «جایگشت نهایی» متوجه وجود نظم کامل در آن‌ها خواهید شد؛ به تصریح طراحان روش، این عمل هیچ کمکی به استحکام الگوریتم نمی‌کند بلکه وظیفه‌ی «سفیدسازی[1]» بلوک‌های داده را برعهده دارد، کما اینکه در بسیاری از روش‌های دیگر و از جمله راین‌دال از این جایگشت صرف‌نظر شده‌است، بدون اینکه چیزی از امنیت روش کاسته شود.

۴-۵- شیوه‌های رمز: الگوهای زنجیره‌سازی بلوک‌های رمز

روش‌های رمزنگاری متقارن مثل AES، 3-DES، Serpentو نظایر آن همگی بر روی یک قطعه داده‌ی کوچک (به طور معمول به طول ۱۲۸ بیت معادل ۱۶ بایت) عمل می‌کنند. حاصل رمزنگاری داده‌ها نیز بلوکی با همان اندازه است که باید جایگزین متن اصلی شده و ارسال گردد. بدیهی است که هرگاه بلوک‌های ورودی به سیستم رمزنگاری مشابه باشند نتیجه‌ی یکسانی به دست می‌آید. به عنوان مثال هرگاه بلوک m_i در متن اصلی صدها بار تکرار شده باشد، بلوک جایگزین آن در متن رمزنگاری شده نیز صدها بار مشاهده خواهد شد. دلیل منطقی این اتفاق بسیار روشن است: تابع رمزنگار$E_k(m_i)$ یک تابع یک به یک و وارون‌پذیر است لذا هرگاه $m_i=m_j$، الزاماً داریم: $E_k(m_i)=E_k(m_j)$ و برعکس، هرگاه $E_k(m_i)=E_k(m_j)$ بدون تردید داریم: $m_i=m_j$. به بیان ریاضی یک رابطه‌ی دوشرطی است.

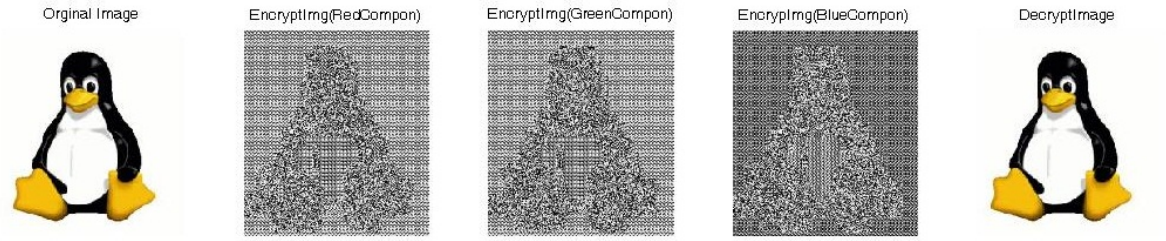
۴-۵-۱. شیوه‌ی کتابچه‌ی رمز

هرگاه در رمزنگاری یک پیام، کل پیام را به قطعات کوچک‌تر با طول ثابت (متناسب با طول ورودی سیستم رمزنگار قطعه‌ای) تقسیم کرده و هر بلوک مستقل از دیگری با کلید k رمز و جانشین متن اصلی شود اصطلاحاً مبتنی بر «شیوه‌ی کتابچه‌ی رمز» ECB[2] عمل می‌شود. به‌عنوان نمونه یک فایل تصویر صدکیلو بایتی را رمز می‌شود. برای این کار بایستی طبق الگویی، آن را به بلوک‌های هم‌اندازه تبدیل‌کرده و پس از رمزنگاری هر بلوک، به جای بلوک-های اصلی جایگزین شود. از آن‌جا که هرگاه بلوک‌های اصلی دقیقاً مثل هم باشند نتیجه‌ی رمز یکسان خواهد بود لذا

[1] Whitening
[2] Electronic Code Book

در تصویری که بسیاری از بلوک‌های آن مثل هم هستند، آن‌چه که به جای یکی از بلوک‌ها جایگزین می‌شود در مناطق دیگر تصویر نیز تکرار خواهد شد.

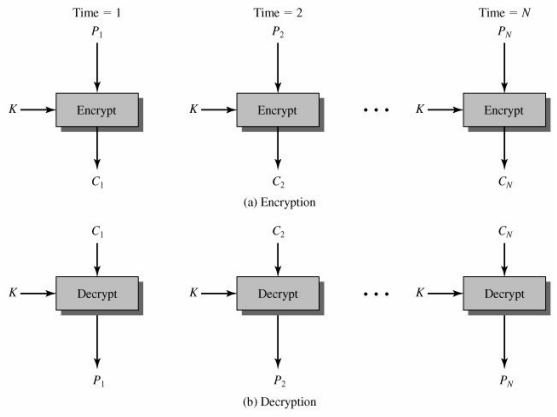
درشکل ۴-۱ بسیاری از بلوک‌های تصویر (چه با رنگ سفید و چه با رنگ خاکستری) یکسان هستند و پس از رمزنگاری با هرچه که جایگزین شوند مثل هم خواهند بود. متاسّفانه این موضوع باعث می‌شود که زمینه‌های بدون تغییر (مثل پس‌زمینه‌ی تصویر) پس از رمزنگاری نیز بدون تغییر بمانند. در این حالت علی‌رغم آنکه تصویر را با قدرتمندترین روش ممکن رمز شده است ولی با یک نگاه کلیات تصویر قابل رویت خواهد بود! این تصویر از نتایج بدست‌آمده در این پروژه است.



شکل ۴-۱ تصویر رنگی رمزشده در نرم‌افزار متلب به شیوه **ECB** با الگوریتم **AES**

به عنوان مثال تصویر بالا که پس‌زمینه‌ی یکنواخت دارد پس از رمزنگاری، بلوک‌های جایگزین پس‌زمینه، مشابه‌هم خواهندبود. ولی آن‌چه که نیاز است، نابود و محو شدن کلّ اطلاعات موجود در تصویر است! موارد خطرناک دیگری هم شیوه‌ی ECB را تهدید می‌کند. به عنوان مثالی دیگر فرض کنید یک بانک اطلاعاتی با الگوریتم 3DES و طبق شیوه‌ی ECB رمزنگاری و جایگزین شده باشد. نفوذگر توانسته در حین انتقال این پایگاه اطلاعاتی، آن‌را قاپیده و در اختیار بگیرد. او اگرچه نمی‌تواند رمز این اطلاعات را بشکند ولی می‌تواند جای برخی از فیلدها را با هم عوض کند. بدین ترتیب بدون آن‌که کسی متوجه این تغییر بشود توانسته در روند عملیات اختلال ایجاد کند و یک حمله‌ی غیرفعال علیه سیستم رمز به وقوع پیوسته است. به عنوان مثال اخلال‌گر می‌تواند بی‌سر و صدا مقدار فیلد حقوق دو نفر را تغییر بدهد! در چنین مواردی باید شیوه‌ای اتخاذ کرد که با تغییر در هر نقطه از فایل یا پایگاه داده، از آن نقطه به بعد کلّ اطلاعات آلوده و بی‌ارزش شود و رمزگشایی آن‌ها، مقادیر رکوردها را از نقطه دستکاری شده، غیرقابل استفاده و نابود کند؛ بدین‌ترتیب تغییر ایجادشده پنهان نخواهد ماند. اینجاست که باید شیوه-های جدیدی برای زنجیره‌سازی بلوک‌های رمز معرفی شود. در شکل ۴-۲ شمای رمزنگاری و رمزگشایی اطّلاعات

در شیوه‌ی «کتابچه رمز» را نشان داده‌شده است. به نحوی که مشاهده می‌کنید بلوک‌های متن مستقل از یکدیگر رمزنگاری و رمزگشایی می‌شوند و تغییر در هر بلوک رمز، تاثیری در رمزگشایی بلوک‌های قبلی و بعدی آن نخواهد داشت. در این شکل سیستم رمزنگار می‌تواند هر یک از روش‌های رمزنگاری متقارن مثل DES، AES ، 3DES، Serpent و امثال آن‌ها باشد. در ضمن شاه‌کلید رمزنگاری برای تمام بلوک‌ها ثابت است [۱].



شکل ۴-۲ شمای روش کتابچه رمز یا ECB [۶]

این شیوه از رمزنگاری برای رمز کردن داده‌های کوچک مثلاً یک کلید رمزنگاری مناسب است. بنابراین اگر قصد ارسال امن کلید الگوریتم استاندارد رمزنگاری داده[1] را دارید، این مُد برای این کار مناسب خواهد بود [۶].

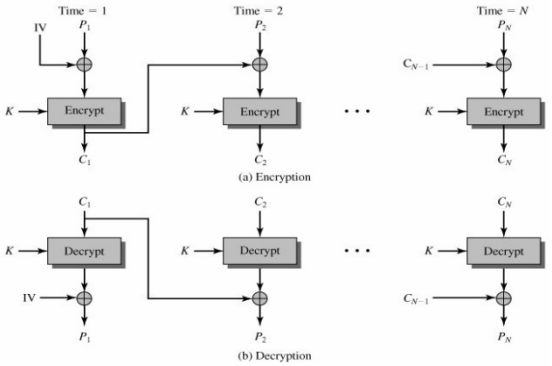
۴-۵-۲. شیوه‌ی ساده زنجیره‌سازی بلوک‌های رمز[2]

ساده‌ترین شیوه‌ی زنجیره‌سازی بلوک‌های رمز آن است که نتیجه‌ی رمزنگاری هر بلوک کوچک، در رمزنگاری بلوک‌های بعدی نیز دخالت داده شود. طبق این شیوه هر گونه تغییر یا جابجایی در یک بلوک باعث خواهد شد که متن رمزنگاری‌شده از محل دستکاری، به بلوک‌های آشغال و بی‌معنی تبدیل گردد. این روش به اختصار شیوه‌ی CBC یا شیوه‌ی زنجیره‌سازی بلوک‌های رمز نامیده می‌شود. در شکل ۴-۳ این شیوه‌ی رمزنگاری نشان داده شده است [۶].

در شیوه‌ی CBC، قبل از آن‌که هر بلوک از متن اصلی رمز شود، ابتدا با بلوک رمزشده‌ی قبلی یای‌انحصاری می‌شود و سپس نتیجه، به ورودی رمزنگار اعمال می‌شود تا خروجی بدست آید. بدین ترتیب بلوک‌های یکسان متن

[1] DES
[2] Cipher Block Chaining

اصلی، هرگز دارای خروجی مشابه نخواهند بود و شیوه‌ی رمزنگاری از حالت جانشینی ساده‌ی بلوک‌ها خارج خواهد شد. برای رمزنگاری اولین بلوک متن (که هیچ بلوک رمزشده‌ی قبلی ندارد)، یک مقدار اولیه مشهور به IV (Initialization Vector) در نظر گرفته می‌شود که از لحاظ طول هم‌اندازه با بلوک متن است. به عبارت دیگر اولین بلوک متن با یک مقدار اولیه، پای انحصاری می‌شود. مقدار IV اهمیتی ندارد و به طور تصادفی انتخاب می شود و به همراه داده‌ها (به صورت آشکار) ارسال می‌شود شود و به همراه داده‌ها (به صورت آشکار) ارسال می‌شود چرا که شاه‌کلیدِ رمز، مخفی است و بدون کلید، هیچ چیزی از رمز خارج نخواهد شد [۸].



شکل ۴-۳ شمای شیوه‌ی زنجیره‌سازی بلوک‌های رمز یا CBC [۶]

با بررسی شکل ۴-۳ به سادگی می‌توان شیوه‌ی زنجیره‌سازی بلوک‌های رمز را مطالعه کرد. اگر بلوک‌های متن اصلی را به ترتیب $P_2\ P_1\ P_0$... در نظر گرفته و عملکرد سیستم رمزنگار به صورت $E_k(P_i)$ مدل شود و بلوک‌های خروجی به صورت $C_2\ C_1\ C_0$ ... در نظر گرفته می‌شوند.

حال برای بدست‌آوردن متن اصلی (P_i) باید بلوک رمزشده‌ی قبلی(C_{i-1}) را با مقدار بدست آمده پای انحصاری کرد. بدین ترتیب برای رمزگشایی هر بلوک باید بلوک رمزشده قبلی آن نیز در اختیار باشد (به‌عبارت دیگر در هر لحظه سیستم باید مقادیر C_i و C_{i-1} و کلید را در اختیار داشته باشد).

اگر تغییر در یکی از بیت‌های بلوک‌های رمز بوجود آمده باشد، از آن‌جا که رمزگشایی هر بلوک به دو بلوک مجاور C_i و C_{i-1} وابسته است لذا فقط دو بلوک مجاور پس از رمزگشایی غیرقابل استفاده خواهند بود [۳].

حال شیوه‌ی CBC از بُعد دیگری بررسی خواهد شد: هر بیت از بلوک P_0 و IV در خروجی C_0 تاثیر عمیقی دارد.C_0 در رمزنگاری بلوک P_1 دخالت دارد و این روال بطور پی‌درپی ادامه می‌یابد یعنی تغییر در یکی از بیت‌های

بلوکِ متن اصلی یا مقدار IV در خروجی تمام بلوک‌های رمز تاثیر خواهد داشت. بدین‌ترتیب اطلاعات موجود در متن اصلی که می‌تواند از لحاظ اندازه بسیار بزرگ باشد، محو و نابود می‌شود [۱].

تفکیک این دو موضوع قابل شرح است که تغییر در مقدار اولیه (IV) یا هر یک از بیت‌های متن اصلی، کل خروجی را تا انتهای متن تحت تاثیر قرار می‌دهد و این تاثیر (حتی به اندازه‌ی یک بیت) بسیار ژرف خواهد بود. در طرف مقابل تغییر در یک بیت از متن رمزشده در خلال عبور از کانال انتقال فقط رمزگشایی دو بلوک مجاور را با اختلال مواجه می‌کند. بدین ترتیب بلوک‌های مشابه در متن اصلی هرگز به بلوک‌های یکسان در خروجی تبدیل نخواهند شد. یکی از اشکالات شیوه‌ی زنجیره‌سازی بلوک‌ها آن است که رمزنگاری اطّلاعات باید به صورت «ترتیبی» انجام شود و امکان موازی‌سازی عملیات جهت بالارفتن سرعت، وجود ندارد. هرگز نمی‌توان بلوک P_i را رمز کرد مگر آن‌که تمام بلوک‌های P_{i-2}، P_{i-1} تا P_0 را از قبل رمز شده باشند.

به ویژگی تاثیرپذیری کلّ خروجی رمزشده از بیت‌های قبلی متن اصلی با مقدار IV، «ویژگی انتشار رو به جلو» گفته می‌شود. این ویژگی در خصوص شیوه‌ی CBC بسیار عالی است. به ویژگی تاثیرپذیری خروجی رمزگشا از تغییر در یکی از بلوک‌های رمزشده (درحین انتقال) «ویژگی انتشار خطا[1]» گویند. این ویژگی در خصوص شیوه‌ی CBC، به ازای هر بیت، دو بلوک است. به عنوان مثال در روش AES-128 با این مُد، اختلال در یک بیت از هر بلوک رمز، منجر به انتشار آن در ۲۵۶ بیت خواهد شد. این موضوع برای سیستم‌های صدا یا تصویر که احتمال خطا در انتقال بیت‌ها قابل توجّه است ایجاد اشکال می‌کند لذا در ایجاد کانال‌های صدا یا تصویر کُدشده[2] شیوه‌ی CBC مورد توجه نیست و بیشتر برای کانال‌های داده به کار برده می‌آید [۱].

مقدار عدد تصادفی باید هم برای فرستنده و هم برای گیرنده مشخص باشد، امّا باید طوری باشد که شخص نفوذگر نتواند آن‌را تغییر دهد و برایش قابل پیش‌بینی نباشد. برای امنیت بالاتر می‌توان این مقدار را توسط شیوه‌ی کتابچه‌ی رمز یا همان ECB رمز کرد. به این دلیل است که هر تغییری در مقدار IV مقدار بلوک رمزگشایی شده‌ی اول را نیز تغییر خواهد داد.

علاوه بر آن‌که این مُد برای حفظ محرمانگی استفاده می‌شود، برای تهیه‌ی کُد احراز هویت و سلامت پیام نیز در شیوه‌ی CCM نیز کاربرد دارد.

---

[1] Error Propagation
[2] Scrambled Channel

۴-۵-۳. شیوه‌ی فیدبک[1]

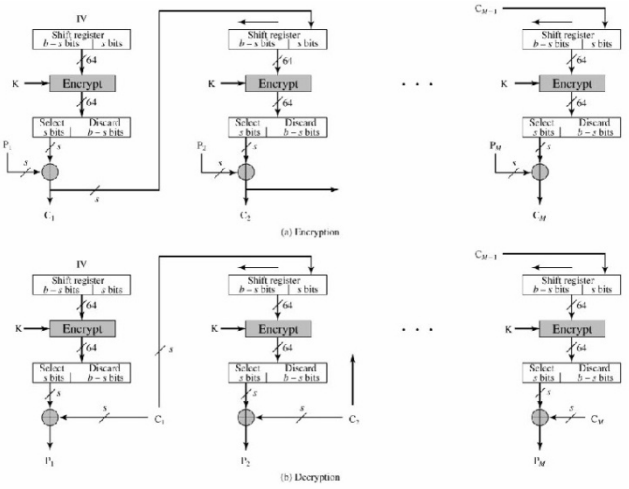
در رمزنگاری به شیوه‌ی فیدبک، بلوک‌های متن هرگز به سیستم رمزنگار متقارن اعمال نمی‌شوند. بلکه عمل رمزنگاری فقط براساس یک XOR ساده انجام می‌گیرد! با توجه به شکل ۴-۴ رمزنگاری بلوک‌های متن اصلی به شیوه‌ی فیدبک، بدین نحو است که هر بلوک متن (P_i) فقط با یک مقدار هم‌اندازه با خودش XOR می‌شود تا نتیجه بدست آید. مسئله این است که هر بلوک با چه مقداری XOR می‌شود؟ در شیوه‌ی فیدبک ابتدا یک مقدار اولیه‌ی کاملاً تصادفی و آشکار (به نام IV) توسط یک روش متقارن مثل AES رمز می‌شود. رمزنگاری یک مقدار تصادفی با کلیدی مخفی، حاصلی غیرقابل پیش‌بینی و تصادفی خواهد داشت. این مقدار با اولین بلوک از متن اصلی XOR می‌شود، تا اولین بلوک رمزبدست آید. حال برای رمزنگاری بلوک بعدی باید دنباله‌ی[2] عوض شود. برای این کار خروجی رمزشده‌ی مرحله‌ی قبل یک‌بار دیگر توسط رمزنگار رمز می‌شود تا خروجی غیرقابل پیش-بینی جدید با بلوک بعدی از متن اصلی XOR شود.

پس برای رمزگشایی هر بلوک، به غیر از خود بلوک، به بلوک ماقبل آن هم احتیاج است. مقدار رمزشده‌ی $E_k(C_{i-1})$ همان مقداری است که در مبدا با بلوک P_i، XOR شده بود و حال باید با C_i، XOR شود تا مقدار بلوک اصلی بدست آید. در حقیقت سیستم رمزنگار مستقیماً بر روی داده‌ها عمل نمی‌کند بلکه با رمز کردن خروجی مرحله‌ی قبل، یک دنباله‌ی وابسته به متن و غیر قابل محاسبه تولید می‌کند.

همانند شیوه‌ی زنجیره‌سازی بلوک‌های رمز یا CBC، شیوه‌ی فیدبک، نیز تغییر در یک بیت از متن یا IV را تا پایان داده‌ها در خروجی منتشر می‌کند و هرگز بلوک‌های یکسان متن، بلوک‌های مساوی در خروجی ایجاد نخواهند کرد. از آن‌جا که برای رمزگشایی بلوک C_i به مقدار $E_k(C_{i-1})$ نیاز است و C_{i-1} نیز قبلاً دریافت شده لذا به محض دریافت C_{i-1} می‌توان همزمان با شروع رمزگشایی آن عملیات محاسبه‌ی $E_k(C_{i-1})$ را نیز شروع کرد تا به محض دریافت C_i فقط یک عمل XOR بلوک داده را از رمز خارج کند. این کار نوعی از فرآیند Pipelining به حساب می‌آید که با شیوه‌ی فیدبک امکان‌پذیر است. و می‌توان تاخیر بین دریافت و رمزگشایی هر بلوک را کاهش چشمگیری بدهد.

---

[1] Cipher Feedback
[2] Pad



شکل ۴-۴ طرح شیوه فیدبک یا CFB [۶]

شیوه‌ی فیدبک را می‌توان با یک اصلاحیه‌ی کوچک تغییر داد تا به جای آن‌که سیستم رمزنگار در حالت بلوکی عمل کند به صورت بایت به بایت اطلاعات را رمزنگاری و رمزگشایی نماید. این کمک می‌کند که رمزنگاری قطعه‌ای DES را با این شیوه به رمزنگاری دنباله‌ای[1] تبدیل کرد. در این صورت تاخیر بین ورودی و خروجی فقط یک بایت خواهد بود و هر کاراکتر می‌تواند جداگانه رمزنگاری و ارسال شود و پیام باید دیگر مضربی از طول روش رمزنگاری نباشد بلکه مضربی از ۸ بیت باشد و درغیر اینصورت داده‌های زاید به آن اضافه شود [۴].

نقش کلید، IV و شیفت‌رجیستر را تولید یک دنباله‌ی هم‌اندازه با طول داده و کاملاً تصادفی، فرض نمایید. در سیستم رمزنگاری، به شیوه‌ی فیدبک تنها از یک ماژول رمزنگار استفاده می‌شود و حتی برای رمزگشایی داده‌ها، به ماژول رمزگشا، نیازی نیست و باز هم از ماژول رمزنگار استفاده می‌شود چرا که باید به همان نحوی که دنباله در مبدا تولید شده بود یک‌بار دیگر برای XOR شدن در مقصد نیز تولید شود. در این سیستم رمزنگاری هرگاه یک یا چند بیت از داده‌های رمزشده در حین عبور از کانال انتقال تغییر کنند، مادامی‌که بیت‌های خطا در درون شیفت‌رجیستر حضور دارند خروجی اشتباه خواهد بود. بنابراین یک بیت خطا حداقل منجر به خرابی یک بلوک از داده‌ها خواهد شد. بنابراین خطای یک بیت در مجموعه‌ای از بیت‌ها منتشر می‌شود [۴].

[1] Stream cipher

۴-۵-۴. رمزنگاری و رمزگشایی به شیوه‌ی استریم

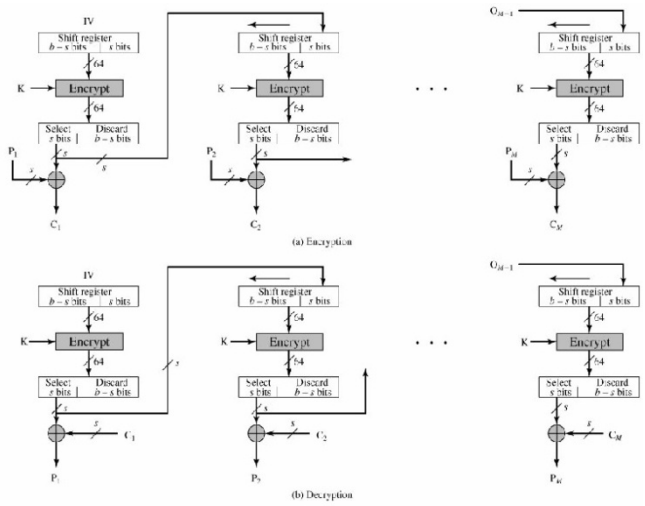
در این شیوه اساسِ کار، بر XOR کردن دنباله‌ی متن ورودی با دنباله‌ای از اعداد تصادفی، استوار است. روش تولید این دنباله‌ی تصادفی آن است که یک مقدار اوّلیه‌ی IV به کمک یک سیستم رمزنگار متقارن رمزشده و خروجی آن با ورودی XOR می‌شود. سپس حاصل رمزنگاری‌شده‌ی IV، بار دیگر رمز می‌شود تا عدد تصادفی دیگری برای XOR شدن با بلوک بعدی داده، بدست آید. حال خروجی رمزنگار باز هم رمز می‌شود و این فرآیند تا وقتی که داده‌ای برای ارسال وجود دارد تکرار می‌شود. شکل ۴-۵ رمزنگاری و رمزگشایی به شیوه‌ی استریم را نشان می‌دهد. اگر با دقّت به شکل نگاه کنید خروجی رمزنگار متقارن در اولین مرحله، حاصل رمزنگاری IV با کلید k است. در مرحله‌ی بعد یک‌بار دیگر این خروجی رمز می‌شود تا مقدار تصادفی جدیدی برای XOR کردن با داده‌ی ورودی بدست آید. بنابراین نه متن اصلی و نه متن رمزشده هیچ‌کدام در تولید دنباله‌ی تصادفی دخالتی ندارند؛ بدین ترتیب گیرنده‌ی اطّلاعات با داشتن کلید و مقدار IV می‌تواند تمام اعداد لازم برای رمزگشایی را پیشاپیش تولید و در جایی ذخیره کند. به خروجی هر مرحله از سیستم رمزنگار متقارن که باید با داده‌ها XOR شوند، به اصطلاح «دنباله‌ی کلید[1]» گفته می‌شود. از آن‌جا که در هر مرحله خروجی رمزنگار متقارن جهت تولید عدد تصادفی جدید در ورودی خودش اعمال می‌شود به شیوه‌ی استریم در اصطلاح «شیوه‌ی فیدبک خروجی[2]» نیز گفته می‌شود.

مهم‌ترین مزیت رمزنگاری به شیوه‌ی استریم آن است که خرابی یک بیت در حین انتقال داده‌ها بر روی کانال، صرفاً منجر به خرابی یک بیت در خروجی خواهد شد، لذا در خروجی منتشر نخواهد شد. این موضوع می‌تواند در انتقالات آنالوگِ دیجیتالی شده مثل انتقال صدا و ویدئوی دیجیتالی شده مفید شود [۶]. یک خصوصیت بسیار زیبای این شیوه اینست که بیشتر این عملیات می‌تواند به صورت برون خطی[3]، قبل از این‌که متن آشکار دریافت شود، انجام شوند؛ سپس به هنگام دریافت پیام، با مقادیر به دست آمده از خروجی سیستم رمزنگاری، XOR شده تا متن رمز شده بدست آید [۱].

---

[1] KeyStream
[2] OFB(Output Feedback)
[3] offline

هر گاه خواسته‌شود که رمزنگاری به شیوه‌ی استریم بر روی بایت‌ها عمل کند نه بر روی بلوک‌های ۸، ۱۶، ۳۲ بایتی براحتی می‌توان به ازای هر بایت ورودی یک دنباله‌ی داده تولید کرد و بایت سمت چپ آن‌را با بایت ورودی XOR و از بقیه صرف‌نظر کرد.



شکل ۴-۵ طرح فیدبک خروجی [۶]

برای رمزگشایی داده‌ها کافی است با IV و کلید، شروع به تولید دنباله‌ها کرده و آن‌ها به ترتیب با بلوک‌های رمزشده‌ی ورودی، XOR شوند. از آن‌جا که دنباله‌ها صرفاً به کلید و IV وابسته‌اند لذا هیچ‌گاه از خطاهایی که بر روی کانال انتقال، داده‌ها را تهدید می‌کند تاثیر نمی‌پذیرد و می‌توان برای سرعت بخشیدن به عملیات رمزگشایی به عملیات محدود Pipelining متوسل شد.

۴-۵-۵. حمله‌ی KeyStream Reuse علیه شیوه‌ی استریم

یکی از بزرگ‌ترین تهدیدهایی که رمزنگاری به شیوه‌ی استریم را تهدید می‌کند آن است که در این شیوه نبایست از زوج یکسان (کلید، IV) برای دنباله‌های متفاوت داده استفاده شود. عدم دقت به این مورد بود که سال ۲۰۰۱ به یک سرشکستگی علمی برای کارت‌های شبکه‌ی بی‌سیم WiFi منجر شد. حال موضوع از چه قرار است: فرض کنید دو فایل بزرگ به نام‌های P و Q را جداگانه ولی با زوج (کلید و IV) مشابه رمزنگاری کرده و آن‌ها از طریق خطوط ناامن انتقال داده‌شوند. از آن‌جا که کلید و IV برای هر دو دنباله یکی است بنابراین KeyStreamهایی که برای آن‌ها تولید می‌شود کاملاً مشابهند. پس رمزنگاری هر دو فایل به‌صورت $(P_0 \oplus K_0)$

$(Q_0 \oplus K_0)\ (Q_1 \oplus K_1)\ (Q_2 \oplus K_2) \ldots (Q_n \oplus K_N)$ و $(P_1 \oplus K_1)\ (P_2 \oplus K_2) \ldots (P_n \oplus K_N)$ خواهد بود که در آن Q_i و P_i بلوک kام از فایل‌های P و Q هستند و K_i کلید هر مرحله (یا همان KeyStream) به حساب می‌آید. حال فرض کنید یک نفوذگر بلوک‌های رمزشده هر دو فایل را استراق‌سمع کند. او می‌تواند هر بلوک رمزشده از فایل اول را با بلوک متناظر از فایل دوم XOR کند تا کلید رمز از بین برود: $(P_i \oplus K_i) \oplus (Q_i \oplus K_i) = P_i \oplus Q_i$. بنابراین با حذف کلید برای هر بلوک، حاصل XOR دو بلوک متن اصلی باقی می‌ماند که با توجّه به ویژگی‌های آماری متون می‌توان ساده‌تر به آن حمله کرد و آن‌ها را آشکار ساخت. بهر حال حذف کلید تصادفی، کار رمزشکنی را بسیار ساده‌تر خواهد کرد.

حل این مشکل بسیار ساده است. کافی است به ازای هر دنباله‌ی جدید از داده‌ها، مقدار IV عوض شود. بخاطر بیاورید که IV مقداری تصادفی و دلخواه دارد و حتی می‌توان آن‌را آشکارا بر روی خط فرستاد؛ لذا تغییر آن به ازای هر دنباله‌ی جدید هیچ مشکلی را ایجاد نخواهد کرد و پیاده‌سازی نرم‌افزاری یا سخت‌افزاری آن بسیار ساده خواهد بود.

اگرچه می‌توان KeyStream ها را پیشاپیش تولید و ذخیره کرد ولی برای دنباله‌های طولانی، این کار چندان مقرون‌به‌صرفه نیست (چون به حافظه نیاز دارد). بنابراین هرگاه از این شیوه (یا شیوه‌های قبلی) برای رمزنگاری و ذخیره‌ی فایل بر روی کامپیوترها استفاده شود آن‌گاه امکان دسترسی مستقیم به بلوک‌های فایل ممکن نخواهد بود زیرا برای دسترسی به یک بلوک دلخواه از فایل، به KeyStream همان بلوک نیاز است و برای محاسبه‌ی آن باید تمام Keystreamهای قبل از آن محاسبه شده‌باشند [۱].

۴-۵-۶. رمزنگاری به شیوه‌ی شمارنده[1]

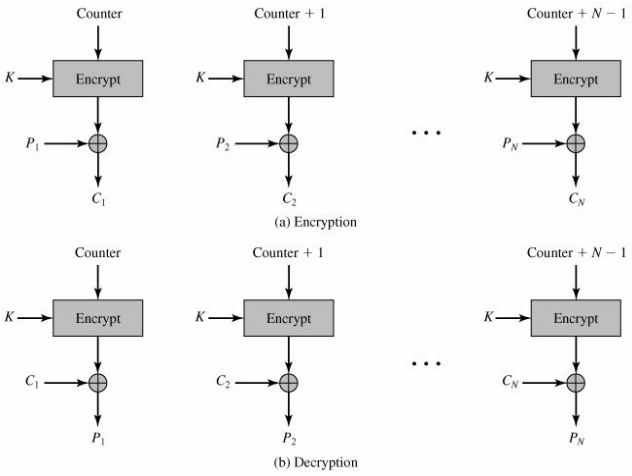
هیچ یک از شیوه‌های رمزنگاری، برای رمزکردن فایل‌های طولانی (که دسترسی مستقیم به هر بلوک آن‌ها جزء ضروریات است)، مناسب نیستند. لذا برای سیستم فایل باید شیوه‌ای اتخاذ شود تا ضمن زنجیره‌سازی بلوک‌های رمز، بتوان به طور مستقیم هر بلوک دلخواه از آن را، ذخیره یا بازیابی کرد. بهترین شیوه برای امکان دسترسی مستقیم به هر بلوک رمزشده، **شیوه شمارنده** است. طرح کلّی این روش در شکل ۴-۶ نشان داده‌شده است.

[1] Counter Mode

در این شیوه نیز متن اصلی به صورت مستقیم رمز نخواهد شد بلکه عملیات رمزنگاری با یک XOR ساده انجام می‌شود. بلوک‌های متن اصلی را P_0،P_1 تا P_n و بلوک‌های رمزشده را C_0،C_1 تا C_n نامیده می‌شوند. بلوک-ها می‌توانند یک بایتی یا بیشتر باشند. در ابتدای کار به ازای هر فایل (یا دنباله‌ی طولانی داده‌ها) یک مقدار اولیه‌ی تصادفی مشهور به Nonce یا IV در نظر گرفته‌شده است که هیچ لزومی به مخفی نگه‌داشتن آن نیست. یک شاه-کلید سرّی نیز وجود دارد که کلمه‌ی عبور کاربر تلقی می‌شود. برای رمزنگاری اولین بلوک متن یعنی P_0، مقدار IV (یا همان Nonce) به کمک کلید رمز و یک رمزنگار متقارن، رمزشده و خروجی آن با P_0، XOR می‌شود تا اولین بلوک رمز یعنی C_0 بدست آید. برای رمزنگاری بلوک بعدی، یک واحد به IV اضافه شده و همین روال تکرار می-شود؛ برای رمزنگاری بلوک P_n، با داشتن IV به اندازه‌ی n واحد به آن اضافه کرده و پس از رمزنگاری این مقدار با شاه‌کلید سرّی، خروجی رمزنگاری با بلوک nاُم، XOR می‌شود. برای رمزگشایی بلوک نیز کافی است همین کار را یک‌بار دیگر تکرار کنید چرا که معکوس عمل XOR باز همان XOR است. بدین‌ترتیب دسترسی مستقیم به هر کدام از بلوک‌های رمز، رمزگشایی، تغییر و ذخیره‌ی مجدد آن بلوک بر روی دیسک، با دانستن IV، شماره‌ی بلوک و کلید رمز به سادگی میسّر خواهد بود. این شیوه نیز مورد تهدید حمله‌ی Key stream Reuse است و اگر برای فایل‌های متفاوت از مقدار IV مختلف استفاده نشود یک نفوذگر می‌تواند با استخراج یا استراق‌سمع فایل‌های رمزشده، بلوک‌های هم‌طراز آن‌ها را دوبه‌دو با هم XOR کرده و کلیدِ رمز را از میان بردارد. سپس به کمک تکنیک‌های آماری رمز آن‌ها را استخراج کرده و Key stream را برای رمزگشایی مابقی فایل‌ها محاسبه کند. لذا باید IV برای هر فایل جدید تغییر کند. مقدار IV به ازای هر بلوک جدید یک واحد افزایش می‌یابد. در برخی از پیاده‌سازی‌های سخت‌افزاری مقدار IV در قالب یک رجیستر به دو بخش تقسیم می‌شود: بخش اول که مقداری کاملا تصادفی دارد و Nonce نامیده می‌شود و بخش دوم به صورت شمارنده طراحی شده است. در این پروژه نیز از این شیوه استفاده خواهد شد. طول بخش Nonce و بخش شمارنده باید به گونه‌ای تنظیم شود تا اطمینان حاصل شود که هیچگاه مقدار شمارنده برای بلوک‌های متوالی یک فایل به صفر برنمی‌گردد. چرا که آن‌گاه برای یک فایل واحد بلوک‌هایی وجود خواهد داشت که مقدار کلید و IV آن‌ها یکسان است و امکان حمله‌ی Key stream وجود خواهد داشت. رمزنگاری به شیوه‌ی شمارنده (همانند شیوه‌ی استریم) این حُسن بزرگ را خواهد داشت که خطا در یک بیت از داده‌های رمزشده، تنها یک بیت از داده‌های رمزگشایی‌شده را با خطا مواجه می‌کند. بدین‌ترتیب می‌توان از آن برای انتقال دنباله‌های صدا و تصویر رمزشده بهره گرفت و پیاده‌سازی ساده‌ای خواهد داشت. در ضمن امکان موازی‌سازی عملیات تا هر سطح ممکن میسّر است [۸].

شیوه‌ی شمارنده این مزایا را دارد: برخلاف سه شیوه‌ی زنجیره‌سازی، رمزنگاری و رمزگشایی می‌تواند به صورت موازی روی چند بلوک رمز یا بلوک متن آشکار انجام شود. در شیوه‌های زنجیره‌ای محاسبات روی یک بلوک باید قبل از شروع محاسبات روی بلوک بعدی تمام شود. و این موضوع بیشترین ظرفیت پذیرش الگوریتم را کم می‌کند. در این شیوه ظرفیت پذیرش بوسیله‌ی میزان موازی‌سازی که بدست می‌آید محدود می‌شود. به‌طور مشابه به خاطر فرصت‌های اجرای موازی در این شیوه، پردازنده‌هایی که مشخصه‌های همزمانی را مانند خطِ لوله، ارسال چند دستورالعمل در هر سیکلِ کلاک، تعداد زیادی ثبات، دستورالعمل‌های یک دستورالعملی و چند داده‌ای[1] و... را حمایت می‌کنند، می‌توانند به طور موثری مورد استفاده قرار گیرند [۸].

برخلاف شیوه‌های ECB و CBC، این شیوه فقط به اجرای الگوریتم رمزنگاری رمز قطعه‌ای و نه رمزگشایی رمز قطعه‌ای نیاز دارد [۸].



شکل۴-۶ طرح شیوه‌ی شمارنده CTR [۶]

۴-۵-۷. انتشار خطا و تضمین صحت و دست‌نخوردگی داده‌ها

همان‌گونه که اشاره شد رمزنگاری به شیوه‌ی کتابچه‌ی رمز یا ECB صحت و دست‌نخوردگی دنباله‌ی طولانی داده‌ها (مثل فایل یا پایگاه اطّلاعاتی) را تضمین نخواهد کرد. برای زنجیره‌سازی بلوک‌های رمز روش‌های متعددی مثل شیوه‌ی فیدبک، شیوه‌ی شمارنده، شیوه‌ی استریم و نظایر آن وجود داشتند که در ادامه بیشترتحلیل خواهند شد.

[1] SIMD

ولی آیا این شیوه‌ها صحت داده‌ها و دست‌نخوردگی آن‌ها را تضمین خواهند کرد؟ پاسخ واقعی این سوال این‌است که این روش‌ها لازمند ولی کافی نیستند. برای تضمین صددرصد صحّت داده‌ها باید به سراغ الگوریتم‌های امضای دیجیتال رفت. روش‌هایی مثل شیوه‌ی شمارنده یا فیدبک اگرچه به نحوی طرّاحی شده‌اند که نتوان جای دو بلوک را عوض کرد ولی گاهی یک اخلال‌گر تمایل دارد یک بیت دلخواه از یک بلوک خاص را تغییر دهد، بدون آن‌که در خروجی تغییرات بنیادی رخ دهد. در رمزنگاری به شیوه‌ی استریم و شمارنده این کار به سادگی امکان‌پذیر است زیرا رمزنگاری در این دو شیوه عبارتست از یک XOR ساده بین متن اصلی و دنباله‌ی کلیدها KeyStream ! بنابراین هر تغییری که به صورت عمد در حین انتقال داده‌های رمزشده ایجاد شود صرفاً یک بیت از داده‌های رمزگشایی‌شده را تحت تاثیر قرار خواهد داد. حتی در شیوه‌ای مثل CBC این مسئله یک تهدید به شمار می‌آید. اگر یک اخلال‌گر تمایل داشته بیت دلخواهی را وارونه کند. از آن‌جا که در شیوه‌ی CBC هر بلوک رمز از دو بلوک مجاور خود تاثیر می‌پذیرد، اخلال‌گر با ایجاد تغییر عمدی بر روی بلوک ماقبل، می‌تواند بیتی را طبق نقشه‌ی خود به نحوی وارونه کند تا تاثیر (پس از عمل رمزگشایی) در بیت مورد نظر منعکس شود. در این وضعیت اگر چه بلوک رمزگشایی‌شده‌ی قبلی بطور کامل نابود می‌گردد ولی بیت دلخواه اخلال‌گر نیز وارونه خواهد شد [۱].

در نتیجه روش‌های زنجیره‌سازی بلوک‌های رمز قطعاً لازمند زیرا نباید اجازه داد که بلوک‌های یکسان در متن اصلی به بلوک‌های مشابه در متن رمز، نگاشته شود ولیکن به مکانیزم‌های اضافی برای جلوگیری از دستکاری عمدی در پیام‌های مهم، نیاز خواهد بود.

شیوه‌هایی برای تلفیق دو عمل زنجیره‌سازی و حفظ سلامت و دست‌نخوردگی پیام‌ها ابداع شده‌اند که در ادامه برخی از آن‌ها بررسی خواهند شد. از این شیوه‌ها می‌توان به OCB، CCM، IAPM، IACBC، ... نام برد.

از آن‌جا که روش‌های رمزنگاری متقارن بر روی داده‌هایی به طول ثابت و مشخص ۶۴، ۱۲۸ یا ۲۵۶ بیتی کار می‌کنند لذا نمی‌توان انتظار داشت که تمام پیام‌ها، در این پروژه منظور تصاویر هستند، ضریبی از این مقادیر باشند. به عنوان مثال در AES-128 که طول بلوک‌های ورودی و خروجی رمزنگار ۱۲۸ بیتی(معادل ۱۶ بایت) است. چگونه می‌توان پیامی ۵۰ بایتی را رمزنگاری کرد. اینجاست که باید به انتهای پیام مقادیری زائد چسباند تا طول پیام ضریبی از طول بلوک ورودی شود. به این عمل در اصطلاح افزونگی بی‌ارزش[1] گفته می‌شود و در برخی از شیوه‌های

[1] Padding

رمزنگاری (مثل CBC) به آن احتیاج خواهد بود. برای آن‌که گیرنده پیام بتواند پس از رمزگشایی بخش حقیقی و معتبر داده را از بخش بی‌ارزش پیام تشخیص بدهد. بایستی طول دقیق داده‌های معتبر به نحوی در شناسنامه پیام درج شده‌باشد. در برخی از شیوه‌های رمزنگاری به جای مشخص‌کردن طول داده‌های معتبر در شناسنامه‌ی پیام، قسمت افزونه‌ی بی‌ارزش را با کُد "0\" مشخص می‌کنند. این رویکرد در مورد داده‌های دودویی (مانند تصاویر باینری) جوابگو نیست. اگر چه شیوه‌های زنجیره‌سازی بلوک‌های رمز بسیار متنوعند ولی مهم‌ترین آن‌ها، شیوه‌هایی است که در این پروژه پیاده‌سازی خواهند شد. به عنوان مثال در استاندارد IEEE 802.11 (استاندارد شبکه‌های بی‌سیم WiFi) از شیوه‌ی استریم Stream Mode بهره‌گرفته‌شده و سیستم رمزنگار متقارن در آن RC4 انتخاب شده-است (بعداً RC4 با AES جایگزین شد و طول IV از ۲۴ بیت به ۱۲۸ بیت افزایش یافت) [۱].

۴-۶- کدهای احراز هویت و سلامت پیام

حملاتی مانند تغییر محتوایی پیام فرستنده (مانند پاک کردن، اضافه کردن، ایجاد تغییر و تبدیل محتوای آن و...)، یا تایید صحت دریافت پیام توسط شخص ثالثی به غیر از گیرنده‌ی پیام، حمله‌ی تکرار[1] با ارسال یه یک پیام قدیمی برای شروع تبادل داده و نیز ایجاد تاخیر در ارسال پیام توسط شخص ثالث از جمله حملاتی هستند که توسط کدهای احراز هویت و صحت پیام می‌توان جلوگیری کرد [۶].

هر مکانیزم احراز هویت و امضای دیجیتالی دو سطح عملکرد دارد: در سطح اول یک مقداری که اثبات کننده‌ی هویت است از روی پیام ساخته شده و به آن اضافه می‌شود تا گیرنده بتواند فرستنده را تصدیق کند [۴]. در این‌جا مختصری راجع به این تصدیق‌کننده‌های پیام صحبت می‌شود: سطح دوم که مکانیزم عدم انکار[2] را نیز حمایت می‌کند، موضوع این پروژه نیست و به آن پرداخته نشده است، برای مطالعات بیشتر می‌توانید از مراجع مربوطه استفاده کنید.

۱. کد احراز هویت پیام: این کُدها (که زین پس MAC[3] گفته خواهند شد) قطعه‌ی کوچکی از اطلاعات در هم فشرده‌ی کلّ پیام هستند که قادرند هویت فرستنده و همچنین سلامت پیام را به صورت همزمان تایید یا رد کنند. الگوریتم‌های تولید کُدهای MAC، یک پیام به طول دلخواه به همراه یک کلید سرّی و توافق‌شده بین طرفین را

[1] replay
[2] Non-repudiation
[3] Message Athentication Code

گرفته و براساس آن یک رشته‌ی بیتی موسوم به «MAC» یا «Tag» یعنی برچسب تولید می‌کند. فرستنده، اصل پیام را به همراه کُد MAC آن برای طرف مقابل خود می‌فرستد. در سمت گیرنده، همین کار دقیقاً تکرار می‌شود: اصل پیام دریافتی به همراه کلید سرّی تحویل الگوریتم شده و کُد MAC متناظر با آن تولید و با کد همراه پیام مقایسه می‌شود. اگر نتیجه‌ی این مقایسه مثبت بود اولاً مطمئن خواهد شد که پیام در طول مسیر هیچ‌گونه تغییری نداشته و سالم است. ثانیاً از آن‌جا که هیچ‌کس در جهان کلید توافق‌شده‌ی طرفین را نمی‌داند لذا کسی نمی‌توانسته چنین پیامی را جعل و کُد MAC آن‌را به درستی تنظیم کند. بدین‌ترتیب از هویت طرف مقابل اطمینان حاصل می‌شود. لذا:

الف) کُدهای MAC مبتنی بر الگوریتم‌های تولید چکیده‌ی پیام هستند با این تفاوت که برای تولید کُد MAC بایستی طرفین بر روی یک کلید سرّی توافق کرده‌باشند.

ب) الگوریتم تولید کُد MAC باید در مقابل انواع حملات رمزشکنی مقاوم باشد و هیچ‌کس در جهان نتواند پیامی مثل M' را به‌گونه‌ای پیدا کند که کُد MAC آن با کُد MAC پیامی دیگر مثل M یکسان باشد.

ج) الگوریتم‌های تولید MAC از آن‌جا که مبتنی بر الگوریتم‌های محاسبه‌ی چکیده‌ی پیام هستند بسیار سریع عمل می‌کنند و پیاده‌سازی آن‌ها در سطح سخت‌افزار و نرم‌افزار امکان‌پذیر است. از این الگوریتم‌ها در تولید بی‌درنگ (Real Time) امضاهای دیجیتالی برای فرم‌های داده در سطح سخت‌افزار کارت‌شبکه استفاده شده است.

د) کُدهای MAC را می‌توان گونه‌ای از امضاهای دیجیتال قلمداد کرد که در آن هیچ روش رمزنگاری (اعم از متقارن یا نامتقارن) دخالت ندارد.

ه) کُدهای MAC بسیار کوتاه (بین ۱۶ تا ۶۴ بایت) و دارای طول ثابتی هستند.

و) از آن‌جا که کلید سری بین طرفین توافق می‌شود لذا ویژگی «انکار ناپذیری[1]» در این مکانیزم وجود ندارد و هر یک از طرفین می‌توانند پیام‌های ارسالی خود را منکر شوند. و تنها این خصوصیت با استفاده از امضای دیجیتال تامین می‌شود.

---

[1] Non-Reputation

۲. به الگوریتم‌هایی که از درون یک پیام با طول متغیر، یک چکیده با طول کوتاه و ثابت محاسبه و استخراج می‌کنند، «توابع درهم‌ساز[1]» و به چکیده‌ی پیام «کد درهم‌شده[2]» گفته می‌شود. برخلاف MAC تابع درهم‌ساز از کلیدی استفاده نمی‌کند ولی تابع پیام ورودی است. این کد تابعی از تمام بیت‌هاست و هر تغییری در هر بیت پیام، موجب تغییر در این چکیده می‌شود [۴].

۴-۶-۱. روش OCB[3]

این مد یکی از مدهای رمزنگاری بلوکی است که همزمان هم محرمانگی و هم اصالت پیام را برای داده‌ی آشکارِ تامین‌شده توسط کاربر، حفظ می‌کند. به این قبیل مدها، طرحِ رمزنگاری با احراز اصالت گویند. یک الگوریتم رمزنگاری و احراز هویت مبتنی بر اعداد تصادفی با داده‌ی همراه دارای ورودی‌های (K, M, A) می‌باشد. فضای کلید K یک مجموعه‌ی متناهی و غیرتهی است. ورودی A داده‌ی همراه است که از $\{1،0\}^*$ تشکیل‌شده است. الگوریتم یک عدد تصادفی N را تولید خواهد کرد. به همراه این الگوریتم یک برچسب که برگرفته از تک تک بیت‌های پیام است به همراه پیام ضمیمه می‌شود. آن‌چه که OCB را قابل‌توجه کرده است، اینست که رمزنگاری احرازاصالت شده را با سرعتی برابر با مدهای مرسوم مانند CTR، که تنها موجب حفظ محرمانگی پیام می‌شوند، فراهم می‌کند. اجرای این مد چه با سخت‌افزار چه با نرم‌افزار راحت است. شیوه‌ی OCB کارش را بدون استفاده از تابع درهم‌ساز عمومی به‌انجام می‌رساند، تکنیکی که جای خود را به پیاده‌سازی‌هایی که هم در سخت‌افزار و هم در نرم‌افزار سریع هستند نمی‌دهد. با مقداری دقت بیشتر، متوجه می‌شوید که OCB مشکل رمزنگاری با احراز اصالت با داده همراه[4] مبنی بر نانس (عدد تصادفی) را حل کرده است. نام قسمت داده‌ی همراه بدین معنی‌است: هنگامی‌که OCB یک متن آشکار را رمز می‌کند، می‌تواند آن‌را به یک رشته‌ی دیگری، به نام داده‌ی همراه، که تصدیق می‌شود ولی رمز نمی‌شود، متصل کند. نام قسمت مقدارِ اولیه‌ی تصادفی یا نانس به این معنی است که OCB یک نانس برای رمز کردن هر پیامی نیاز دارد. این مُد نیازی به نانس رَندم و تصادفی ندارد؛ یک شمارنده نیز این کار را خوب انجام می‌دهد. برخلاف برخی مُدهای دیگر، متن آشکار فراهم‌شده می‌تواند هر طولی داشته باشد، درست مثل داده‌ی همراه [۹].

---

[1] Hash function
[2] Hash code
[3] Offset codebook
[4] authenticated-encryption with associated-data (AEAD)

طرح OCB به شدت متاثّر از طرح Charanjit Jutla به نام IAPM است. در گذشته معمول‌ترین کار برای پیاده‌سازی مکانیزمی که هم محرمانگی و هم احراز اصالت را حفظ کند، استفاده از دو کلید جداگانه بود، یکی برای رمزکردن و دیگری برای محاسبه‌ی MAC . با رمزنگاری، محرمانگی پیام و با MAC، احراز اصالت آن تامین می-شود. هزینه‌ای که با آن هم اختفاء و هم هویت پیام احراز شود برابر با هزینه‌ی رمزنگاری برای پنهان‌کردن اطّلاعات به همراه هزینه‌ی محاسبه‌ی MAC برای تصدیق اصالت فرستنده‌ی پیام است. به دنبال این مشکل رمزنگاران در جستجوی روشی که هم ساده باشد و هم ارزان، بودند که در ادامه این روش توضیح داده خواهد شد. با فرض داشتن یک الگوریتم بلوکی با کلید ۱۲۸ بیتی و به همین اندازه طول بلوک و داشتن یک عدد تصادفی (نانس) به طول ۹۶ بیت به شرح الگوریتم پرداخته خواهد شد. اگر M پیامی باشد که قرار است رمزشود و A داده همراه باشد (اگر وجود نداشت، طول آن صفر در نظر گرفته می‌شود). پس از شکستن M به بلوک‌های ۱۲۸ بیتی، مقدار۱۲۸ بیتی $\text{Checksum} = M_1 \oplus \ldots \oplus M_m$ بدست می‌آید. مقدار Auth یا همان MAC با مقادیر داده‌ی همراه و کلید بدست می‌آید. برچسب τ یک مقدار با اندازه‌ی $\tau = 128$ دارد. متن ارسالی برای مقصد شامل $128m+\tau$ بیت می‌باشد، که m همان تعداد بلوک‌های متن می‌باشد و τ برچسبی برای اطمینان از صحت پیام است [۹].

۴-۶-۲. شیوه‌ی CBC-MAC[1]

این مُد تکنیکی است برای ساخت کُد احراز اصالت پیام از یک الگوریتم رمزِ بلوکی. پیام با یکی از الگوریتم‌های رمزنگاری متقارن به شیوه‌ی زنجیره‌سازی بلوک‌های رمز، که در آن هر بلوک به بلوک رمزشده‌ی قبل از خود وابسته است، رمز می‌شود. این وابستگی در این مُد رمزنگاری باعث می‌شود که هر تغییری که در هر یک از بیت‌های متن آشکار ایجاد شود، نتوان آن‌را بدون داشتن کلید پیش‌بینی کرده و اثرش را خنثی کرد. با استفاده از یک الگوریتم رمز قطعه‌ای امن، این مُد برای رمزکردن پیام‌هایی با طول ثابت مناسب هستند و برای پیام‌هایی با طول متغیر مناسب نیستند. فرد اخلال‌گری که جفتِ پیام و برچسب -که با مُد CBC-MAC بدست آمده‌است- (m,t) (m',t') را می‌داند، می‌تواند پیام سومی مانند m'' را که برچسبش نیز t' باشد، تولید کند. این به راحتی با XOR کردن اولین بلوک m' با t و سپس الحاق m به این پیام تغییر یافته m' به دست می‌آید [۴]. یعنی:

$$M'' = m \,||\, [(m_1' \oplus t) \,||\, m_2' \,||\, \ldots \,||\, m_x']$$ رابطه‌ی ۴-۳ چگونگی نفوذ اخلال‌گر در پیام

[1] Cipher block chaining message authentication code

این مشکل با اضافه کردن بلوکی به سایز پیام حل نمی‌شود، و توصیه شده از مُدهای دیگر رمزنگاری با کُدهای چکیده‌ی پیام دیگری استفاده شود [۴].

۴-۶-۳. شیوه‌ی CCM[1]

این مد نیز هم محرمانگی و هم اصالت پیام را تضمین می‌کند. این مُد مبتنی بر یک رمز قطعه‌ای متقارن اثبات شده است که سایز بلوک ۱۲۸ بیتی است، مانند الگوریتم استاندارد رمزنگاری پیشرفته[2] که در (FIPS) Pub.197 مشخص شده است. بنابراین CCM با الگوریتم 3-DES که سایز بلوک‌هایش ۶۴ بیتی است، نمی‌تواند استفاده شود. مانند دیگر مُدهای رمزنگاری یک کلید تکی برای رمز قطعه‌ای باید از قبل بین طرفین ارتباط مبادله شود. مُد CCM در محیط‌هایی با بسته‌های کوچک[3] استفاده می‌شود یعنی هنگامی‌که همه‌ی داده‌ها در ذخیره‌گاه[4] از قبل در دسترس باشند. ورودی به این مُد شامل سه عنصر است: داده‌ای که هم رمز می‌شود و هم احراز اصالت که بار مفید[5] نامیده می‌شود. دوم داده‌ی همراه است، مثلاً سرآیند[6] که احراز هویت می‌شود ولی رمز نمی‌شود و سوم یک مقدار تصادفی است که نانس نامیده شده و به پیام اصلی و داده‌ی همراه اضافه می‌شود. مُد CCM شامل دو نوع پروسه است: پروسه تولید و رمزنگاری[7] و سپس پروسه‌ی رمزگشایی و تایید[8] که دو عملیات پایه‌ی رمزنگاری یعنی رمزگشایی شیوه‌ی شمارنده و شیوه‌ی زنجیره‌سازی بلوک‌های رمز با احراز هویت را با هم ترکیب می‌کند. در مرحله‌ی تولید و رمزنگاری، زنجیره‌سازی بلوک رمز به بار مفید و داده‌ی همراه و نانس برای تولید کُد احراز اصالت پیام MAC اعمال شده، آن‌گاه مُدِ شمارنده به بار مفید و MAC برای تبدیل آن‌ها به یک فرم غیرقابل دسترس که همان متن رمز[9] نامیده‌می‌شود، اعمال می‌شود. در نتیجه، این مرحله سایز بارِمفید را به اندازه‌ی MAC توسعه می‌دهد. در مرحله‌ی رمزگشایی و تایید، مُد شمارنده به متنِ رمز مقصود برای بازیابی چکیده‌ی پیام و بار مفید معادل اعمال شده، سپس زنجیره‌سازی بلوک رمز به بارِمفید و داده‌ی همراه و عددتصادفی نانس دریافت‌شده اعمال شده، تا صحت آن‌ها

---

[1] Counter with Cipher Block Chaining Message Authentication Code
[2] AES
[3] packet environment
[4] Storage
[5] Payload
[6] header
[7] generation-encryption
[8] decryption-verification
[9] ciphertext

را با مقایسه‌ی با MAC دریافتی بررسی کند. تصدیق و تایید موفقیت‌آمیز، این موضوع که داده‌ی همراه و بارِ مفید از طرف منبعی که به کلید اصلی دستیابی داشته، فرستاده شده را تضمین می‌کند [۱۰].

۴-۶-۴. روش GCM[1]

این شیوه نیز به دلیل کارایی و عملکردش، بسیار مورد استفاده قرار گرفته است. کانال‌هایی ارتباطی با منابع سخت‌افزاری قابل قبول، می‌توانند برای دستیابی به سرعت بالا از این مد بهره بگیرند. به‌هنگام استفاده از یک نوع الگوریتم رمز قطعه‌ای، شیوه‌های رمزنگاری مختلف می‌توانند به طور قابل توجهی عملکرد و مشخصه‌هایی با کارایی مختلف داشته باشند. این شیوه می‌تواند مزیت پردازش موازی کامل را داشته باشد. همان‌طور که نامش بیان می‌کند این شیوه، مُد شمارنده برای رمزکردن را با شیوه‌ی جدید گالوا برای احراز اصالت ترکیب می‌کند. نکته کلیدی این است که ضرب در میدان گالوا می‌تواند به راحتی به صورت موازی نسبت به الگوریتم زنجیره‌سازی بلوک رمز برای احراز اصالت پیام محاسبه شود [۴].

۴-۷- نتیجه‌گیری

برای رمزنگاری توده‌های بزرگ اطلاعات از روش‌های متقارن استفاده می‌شود، در حالی‌که روش‌های رمزنگاری کلید عمومی برای رمز کردن قطعات کوچک و بنیادی بکار می‌آیند. بنابراین باید از شیوه‌های عملیاتی رمزهای قطعه‌ای استفاده کرد که علاوه بر داشتن امنیت کافی بتوانند سرعت مناسبی نیز داشته باشند. روش رمزنگاری DES به علت داشتن طول کوتاه کلید روش امنی به شمار نمی‌رفت. لذا سازمان به همین دلیل در جهت یافتن الگوریتم‌های قطعه‌ای امن مسابقه‌یی را برگزار کرد که در آن الگوریتم‌هایی از جمله راین‌دال و سِرپِنت و توفیش و... موفق عمل کردند. شیوه‌های رمزنگاری قطعه‌ای مختلف در این فصل پیاده‌سازی شدند. برخی از این شیوه‌ها می‌توانند هم رمزنگاری را برای محرمانه‌ماندن اطلاعات انجام دهند و هم احراز هویت طرف فرستنده‌ی پیام‌را. شیوه-هایی مثل OCB، که در این فصل معرفی شدند، با روش‌های فقط رمزنگاری مانند شمارنده در برخی از پلت‌فرم‌ها سرعتی یکسان بود. امّا مزیت محاسبه‌ی چکیده‌ی پیام با کمک تابع درهم‌ساز طرح‌شده توسط طراحان شیوه را نباید نادیده گفت.

---

[1] Galios/Counter Mode

فصل پنجم

پیاده‌سازی‌های الگوریتم‌ها در نرم‌افزار مـَتلَب

۵-۱- مقدمه

نتایج بدست آمده در این پروژه، تماماً بر روی یک سیستم Intel® Core™2 Duo @1.80 GHz تحت سیستم‌عامل ویندوز هفت با حافظه‌ی RAM 2.00 GB بوده است.

نرم‌افزار MATLAB برای انجام کارهای ریاضی، آماری، مهندسی و... می‌باشد. در اوایل دهه ۱۹۷۰ توسط Cleve Moler به وجود آمد و در ۱۹۸۴ شرکت Mathwork[18] برای پشتیبانی از این محصول تأسیس شد. در این نرم‌افزار به جای نوشتن دستورات در پنجره command و اجرای تک تک آن‌ها، می‌توان مجموعه‌ای از دستورات را در یک فایل قرار داد؛ به این فایل script یا M-file گفته می‌شود. مجموعه دستورات مورد نظر را در یک ویرایشگر نوشته سپس با فراخوانی نام فایل برنامه اجرا می‌شود. در این فصل سعی خواهدشد که پیاده‌سازی‌های دو الگوریتم برتر رمزنگاری متقارن یعنی استاندارد پیشرفته‌ی رمزنگاری و سِرپنت را و نیز الگوریتم استاندارد رمزنگاری داده، که در نرم‌افزار متلب اجرا شده‌اند، ارایه شوند. سپس شیوه‌های مختلف رمزنگاری که به پیاده‌سازی پنج عدد از آن‌ها و نیز نسخه‌ی سوم شیوه‌ی جدید OCB که شیوه‌ی رمزنگاری به همراه احراز اصالت است ارایه خواهد شد.

برنامه‌ی رمزنگاری DES در پیوست آورده شده است. همانطور که خواهید دید، ابتدا بیت‌های داده طبق جدولی ۶۴ بیتی جابه‌جا می‌شوند، مثلا بیت پنجاه و هشتم به موقعیت یکم منتقل می‌شود. آن‌گاه در ۱۶ دورِ حلقه بر روی قسمت‌های سمت راستِ ۳۲ بیتی، تابع f اعمال شده و حاصل آن با سمت چپ XOR می‌شود. و پس از آن عکس عمل جایگشتی که اعمال شده بود صورت می‌گیرد تا بیت‌ها سرجای اصلی‌شان برگردند. تابع F به صورت زیر پیاده‌سازی شده است:

```
function out=F(R,RoundKey)
```

```
% The 'f' function as used in the encryption rounds for each round, Rn = Ln-1 + f(Rn-1,Kn)
% f(Rn-1,Kn) is to be calculated like this: Kn + E(Rn-1) => B1..B8  f = P( S1(B1)..S8(B8) )
[S1 S2 S3 S4 S5 S6 S7 S8]=S_Box_DES; % Producing eight 4*16 S-Boxes
%   Expansion = 6*8 =48
Expansion =[32, 1, 2, 3, 4, 5,...
            4, 5, 6, 7, 8, 9,...
            8, 9,10,11,12,13,...
            12,13,14,15,16,17,...
            16,17,18,19,20,21,...
            20,21,22,23,24,25,...
            24,25,26,27,28,29,...
            28,29,30,31,32,1];
Exp_R=R(Expansion);     %   Expand 32-bit string to 48-bit
R_Ki=num2str(mod(Exp_R+RoundKey,2));    % XOR RoundKey with Expanded 48-bit string
% Using bit_number 1 and 6 for choosing row and bit_number 2-5 for choosing column
val1=S1(bin2dec([R_Ki(1) ,R_Ki(6) ])+1,bin2dec(R_Ki(2:5))  +1);
val2=S2(bin2dec([R_Ki(7) ,R_Ki(12)])+1,bin2dec(R_Ki(8:11)) +1);
val3=S3(bin2dec([R_Ki(13),R_Ki(18)])+1,bin2dec(R_Ki(14:17))+1);
val4=S4(bin2dec([R_Ki(19),R_Ki(24)])+1,bin2dec(R_Ki(20:23))+1);
val5=S5(bin2dec([R_Ki(25),R_Ki(30)])+1,bin2dec(R_Ki(26:29))+1);
val6=S6(bin2dec([R_Ki(31),R_Ki(36)])+1,bin2dec(R_Ki(32:35))+1);
val7=S7(bin2dec([R_Ki(37),R_Ki(42)])+1,bin2dec(R_Ki(38:41))+1);
val8=S8(bin2dec([R_Ki(43),R_Ki(48)])+1,bin2dec(R_Ki(44:47))+1);

% converting decimal values to 32 bit binary one
R_Subs=[dec2bin(val1,4),dec2bin(val2,4),dec2bin(val3,4),dec2bin(val4,4),dec2bin(val5,4),dec2
bin(val6,4),dec2bin(val7,4),dec2bin(val8,4)];

% FinalPermut = 1*32 permutation of bits in f function
FinalPermut= [16,7,20,21,29,12,28,17,...
              1,15,23,26,5,18,31,10,...
              2,8,24,14,32,27,3,9,...
              19,13,30,6,22,11,4,25];
out = R_Subs(FinalPermut);%  decreasing 48-bit string to 32-bit one
```

الگوریتم توسیع کلید در پیوست آورده شده است. برای نوشتن برنامه‌ی الگوریتم AES بصورت زیر عمل شد:

```
function [S_Box Inv_S_Box RoundKey]=AES_Initializing(key)
%Key is 128 Bits
[S_Box Inv_S_Box]=S_Box_gen_AES;
RoundKey=KeyExpansion_AES(key،S_Box); %Producing the Round Keys from the Master Key
```

به این صورت که ابتدا با ارسال کلید به این تابع، مقدارگذاری اولیه الگوریتمAES با تولید جدول جانشینی و نیز معکوسش و سپس توسیع کلید ۱۲۸ بیتی پرداخته می‌شود. مقادیر این جداول نیز به صورت آماده در حافظه ذخیره نشدند بلکه به ساخت آن‌ها با استفاده از ضرب در میدان گالوا اقدام‌شده است. به این شکل که ابتدا مقادیر {0F}...{02},{01}و سپس در ستون بعد ...{11} ,{10} و به این شکل تا سطر آخر در ماتریسی ذخیره شده، آن‌گاه هر مقداری در این جدول به مقدار معکوس خود در میدان محدود گالوا ($GF(2^8)$ نگاشته می‌شود، مثلاً $\{95\}^{-1} = \{8A\}$. سپس این معکوس در بیت‌های ($b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$) ذخیره‌شده و به فرمول زیر اعمال می‌شوند تا این مقدار از جدول جانشینی بدست‌آید [۶].

$$b'_i = b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus c_i$$

رابطه‌ی ۵-۱ چگونگی جانشینی در جدول جانشینی

که در آن {01100011}={63 hex} = c. مثلاً برای مقدار ذکرشده در بالا این مقدار {2A} بدست می-آید. آن‌گاه این مقادیر به عنوان ورودی به توابع رمزنگاری و رمزگشایی ارسال می‌شوند. الگوریتم این رمز بلوکی در پیوست آورده‌شده است [۶].

طبق الگوریتم هر ستون از پیام با جدول جانشینی جایگزین می‌شود. سپس هر ماتریس حالت به‌صورت سطری شیفت داده می‌شود. همانطور که در فصل ۴ توضیح داده‌شد، چرخش سطرها به ترتیب به اندازه‌ی صفر، یک، دو و سه است. سپس عمل مهم تلفیق ستون‌هاست که طبق قسمت **توصیف ریاضی** در میدان گالوا انجام می‌شود [۶].

```
%The Constant Matrix which its Cells are produced according To the
%Polynomial a(x) = {03}x^3 + {01}x^2 + {01}x + {02}
Mix_hex = {'02' '03' '01' '01';
           '01' '02' '03' '01';
           '01' '01' '02' '03';
           '03' '01' '01' '02'};
%multiplication of a value by x (i.e. by {02}) can be implemented as a 1-bit left-shift
%followed by a conditional bitwise XOR with (0001 1011) if the leftmost bit of the original
%value (prior to the shift) is one. Multiplication of a value by x+1 (i.e. by {03} = ({01}
%xor {02})) can be implemented just as a conditional bitwise XOR then like above a 1-bit
%left shift followed by a conditional bitwise XOR
```

بدین معنی که اگر مقدار متناظر ضرب، در ماتریس ثابت {02} باشد، یک بیت شیفت داده‌شده و اگر بیت سمت چپ قبل از شیفت یک بود باید با مقدار "1b" هگزا دسیمال یای انحصاری شود و برای عنصر {03} نیز علاوه بر عملیات حالت قبل به یک یای انحصاری دیگر نیاز است. این عملیات در قطعه کد زیر نشان داده شده‌اند[۶]:

```
for i=1:4
     for j=1:4
         for k=1:4
            Mixed_State=State(k,j); % Copy the cell of state to a local variable
            if Mix(i,k) == 3 % if cell in constant matrix is 3
                State_out(i,j)=bitxor(uint8(State_out(i,j)),uint8(Mixed_State));
                  % Shift one bit to Left all the 8 bit of Cell
                Mixed_State=bitshift(Mixed_State,1,8);
                if  bitget(uint8(State(k,j)),8)== 1
                    Mixed_State=bitxor(Mixed_State,uint8(hex2dec('1b')));
                end
            elseif  Mix(i,k) == 2 % if cell in constant matrix is 2
                Mixed_State=bitshift(Mixed_State,1,8);
                  % if the last bit is not zero
                if  bitget(uint8(State(k,j)),8)== 1
                    Mixed_State=bitxor(Mixed_State,uint8(hex2dec('1b')));
                end
            end
            State_out(i,j)=bitxor(uint8(State_out(i,j)),uint8(Mixed_State));
         end
     end
 end
```

در آخر نیز این ماتریس حالت با کلید دور یای انحصاری می‌شود.

قطعه برنامه‌ی رمزگشایی نیز مانند رمزنگاری است ولی در جهت معکوس عملیات رمزنگاری. در پیوست این پایان‌نامه این قطعه برنامه آورده‌شده است. الگوریتم روش سِرپِنت نیز همانند روش AES، یک تابع برای مقداردهی اولیه داراست. برنامه‌ی رمزنگاری آن در قسمت پیوست آورده شده است.

در الگوریتم Serpent ابتدا با جدول جایگشت ابتدایی مقدار جایگزین‌شده و پس از عمل XOR با کلید دور، که روش ساختن آن‌ها در زیر توضیح داده‌خواهد شد، بنابر شماره‌ی دور مذکور یکی از هشت جدول S_Box انتخاب خواهد شد تا هر چهار بیت از ۱۲۸ بیت متغیر حالت جایگزین شوند. سپس در انتهای الگوریتم برای استحکام بیشتر جایگشت نهایی اعمال می‌شود. بقیه‌ی برنامه با توضیحات روشن خواهند شد.

با فرض این‌که شاه‌کلید اصلی و ۲۶۵ بیتی با هم به صورت کلمات ۳۲ بیتی در نظر گرفته شوند. این کلید طبق نمادهای پیشنهادی طراحان سِرپِنت به صورت زیر نامگذاری شده است: Extended_Master_Key= {W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8}

در شبه‌کد اصلی، این اندیس‌ها منفی انتخاب شده‌اند ولی بدلیل این‌که نرم‌افزار مَتلب اندیس منفی قبول نمی‌کند، مجبور به تغییر این اندیس‌ها شده، لذا شمارنده‌ی حلقه‌ی for زیر به جای 131~0 در بازه‌ی 140~9 قرار داده، سپس از متغیر مجموعه آرایه‌های کمکی Word برای ساختن ۳۳ عددکلید فرعی کمک گرفته شد. شبه‌کد این قسمت به صورت زیر است [۷]:

```
for i=9:140
     %   W(i) := (W(i-8) XOR W(i-5) XOR W(i-3) XOR W(i-1) XOR Ta XOR i)<<< 11
     %   Ta is double & should be converted to Integer
     W{i}=bitxor(bitxor(bitxor(bitxor(W{i-8},W{i-5}),W{i-3}),W{i-1}),floor(Ta));
     Word{k}=bitRotate(bitxor(W{i},k-1),11);
     W{i}=Word{k};
     Word{k}=dec2bin(Word{k},32);
     k=k+1;
end
```

در این شبه‌کد Wها کلمات ۳۲ بیتی شاه‌کلید هستند و W_9 تا W_{140} براساس هشت عنصر قبلی خود پر می‌شوند. عدد Ta عددی ثابت است و مقدار اعشاری عدد ۲ ÷ (۱+√۵) (پس از ضرب در $۲^{۳۲}$ معادل هگزادسیمال 9e3779b9) برای آن انتخاب و به «کسر طلایی» شهرت یافته است؛ این ثابت همانی است که توسط «ریجمن» و «دمن» (ابداع‌کنندگان راین‌دال) به مسخره گرفته‌شد! در ساخته‌شدن کلمات اندیس i هم دخالت داده‌شده است. بنابراین هر کلمه از آرایه Word[i] عبارتست از تلفیقی از چهار کلمه‌ی ماقبل خود به همراه عدد ثابت و طلایی Ta و اندیس i که پس از عمل یای‌انحصاری باهم، به اندازه‌ی یازده بیت شیفت چرخشی به سمت چپ داده می‌شوند [۷].

سپس با اعمال جداول جانشینی S-Box که بیت‌های هر کلید فرعی را طبق جداول جانشینی S-Box(1) تا S-Box(8) (در گروه‌های چهار بیتی) با مقادیر چهار بیتی جدید جانشین می‌کند. جداول جانشینی همان‌هایی هستند که در خود الگوریتم رمزنگاری به کار گرفته شدند. ولی ترتیب به کارگیری آن‌ها بصورت زیر عوض شده است[۷]:

$$S\_Box_4, S\_Box_3, S\_Box_2, S\_Box_1, S\_Box_8, S\_Box_7, S\_Box_6, S\_Box_5$$

گروه‌های ۱۲۸ بیتی که به صورت زیر یکی از کلیدهای فرعی مورد نیاز در خلال ۳۳ دور از رمزنگاری را فراهم می‌کنند Word(4j+1) تا Word (4j+4) هستند یعنی RoundKey{j} از چهار کلید فرعی ساخته می‌شود. قطعه برنامه‌ی آن در پیوست شماره‌ی دو آورده شده است.

حال به سراغ شیوه‌های مختلف رفته و شش شیوه‌ی پیاده‌سازی شده تشریح می‌شوند. در همه‌ی این‌ها به دلیل داشتن دو نوع تصویر یکی رنگی و دیگری خاکستری طبق شرط عملیات متفاوت خواهند بود. زیرا تصاویر رنگی از سه مولفه تشکیل شده‌اند که باید هر مولفه جداگانه رمز شود ولی تصویر خاکستری یک مولفه‌ای است و فقط همان یک مولفه رمزمی‌شود.

```
tStart = tic;
tEnd = toc(tStart);
fprintf('%d minutes and %f seconds\n',floor(tEnd/60),rem(tEnd,60));
```

این چند خط از برنامه برای نشان‌دادن زمان اجرای شیوه‌های مختلف است و به این صورت عمل می‌کند که خط اول ساعت سیستم را در ابتدای برنامه نگه‌داشته و خط بعد از آن که در انتهای برنامه نوشته می‌شود، زمان کلّ اجرای برنامه را با استفاده از دستور موجود در خط اول محاسبه و با دستور پرینت برحسب دقیقه و ثانیه چاپ می‌کند.

در ابتدای تمام این کُدها این اختیار با کاربر است که نوع الگوریتم رمز قطعه‌ای را انتخاب کند، AES یا Serpent یا DES یا 3-DES.

```
if reply == 'A' || reply == 'a'
    [S_Box   Inv_S_Box   RoundKey]=AES_Initializing(key);
    fun1=@AES_Encryption;
    fun2=@AES_Decryption;
elseif reply == 'S' || reply == 's'
    [S_Box  Inv_S_Box    RoundKey]=Serpent_Initializing(key);
    fun1=@Serpent_Encryption;
    fun2=@Serpent_Decryption;
elseif reply == 'D' || reply == 'd'
    RoundKey=KeyExpansion_DES(key(1:2,:));
    fun1=@DES_Encryption;
    fun2=@DES_Decryption;
elseif reply == '3'
    RoundKey1=KeyExpansion_DES(key(1:2,:));
    RoundKey2=KeyExpansion_DES(key(3:4,:));
```

```
    fun1=@TripleDES;
    fun2=@TripleDES_Decryption;
else
    [S_Box   Inv_S_Box   RoundKey]=AES_Initializing(key);
    fun1=@AES_Encryption;
    fun2=@AES_Decryption;
end
```

در این برنامه‌ها همه‌ی قطعات ۱۲۸ بیتی به صورت ماتریسی ۴×۴ از اعداد صحیح و قطعات ۶۴ بیتی به صورت ماتریس ۲×۴ نشان داده‌می‌شوند. توضیح اینکه در قطعه کُدها fun1 و fun2 هَندل تابع رمزنگاری و رمزگشایی است که ورودی‌های آن مقادیر ذکرشده در مقابل آن است.

در پیوست به طور فشرده برنامه‌ی مربوط به تصاویر خاکستری آورده‌شده است و برای تصاویر رنگی نیز به همین صورت اقدام می‌شود با این تفاوت که برای هر مولفه به صورت جداگانه عمل می‌شود و در مقصد گیرنده‌ی پیام تصاویر رمزگشایی‌شده را به یکدیگر متصل می‌کند تا تصویر رنگی اصلی به دست آید.

```
I=resize(Im); % If Row & Column are not factor of 4(128 bit),they should be resized
%   The Image should be converted to double
I=double(I);
%   Image is Broken to Blocks & each block is encrypted
%   B = BLKPROC(A,[M N],FUN) => Distinct block processing for image.
if reply=='D' || reply=='d'
        Img_Encrypted = blkproc(I,[2 4],fun1,RoundKey);
elseif reply=='3'
        Img_Encrypted = blkproc(I,[2 4],fun1,RoundKey1,RoundKey2);
else
        Img_Encrypted = blkproc(I,[4 4],fun1,RoundKey,S_Box);
end
```

در شیوه‌ی کتابچه‌ی رمز خط اول برای افزایش سایز تصویر به مضربی از ۱۲۸ بیت است. برای انجام تمام عملیات باید تصویر از کلاس uint8 به کلاس double تبدیل شود. سپس با استفاده از تابع کتاب‌خانه‌ای blkproc تصویر به بلوک‌های ۴×۴ شکسته‌شده است تا تابع fun1 که بنا به درخواست کاربر یکی از توابع رمز قطعه‌ای در آن قرار دارد، بر روی تک‌تک بلوک‌ها اعمال شود و جواب به صورت بلوکی در متغیر Img_Encrypted قرار بگیرد.

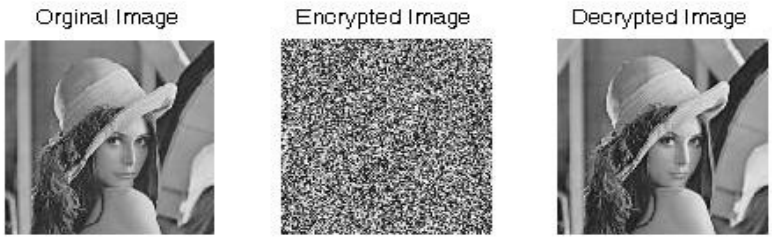
توسط دستور روبرو سایز اصلی تصویر برای گیرنده نگه داشته می‌شود: `SizeOfOriginalPic=size(Im);`

آن‌گاه در سمت گیرنده تابع رمزگشایی به بلوک‌های تصویر رمزشده اعمال می‌شود، تا تصویر ارسالی بدست آید، ولی سایز تصویر اصلی این‌چنین نیست و باید بنابر سایز ارسالی قسمتی از آن حذف شود. توضیحات این تابع کتابخانه‌ای در بالا آورده‌شده است.

```
if reply=='D' || reply=='d'
        Img_Decrypted = blkproc(Img_Encrypted,[2 4],fun2,RoundKey);
elseif reply=='3'
        Img_Decrypted = blkproc(Img_Encrypted,[2 4],fun2,RoundKey1,RoundKey2);
else
```

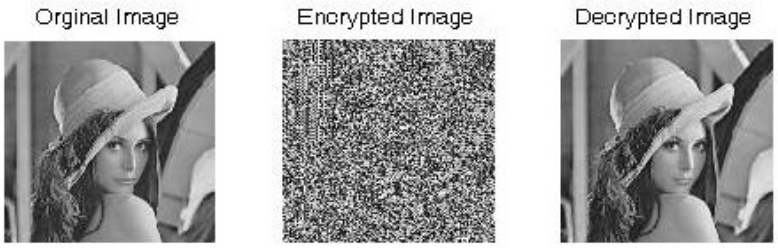
```
        Img_Decrypted = blkproc(Img_Encrypted,[4 4],fun2,RoundKey,Inv_S_Box);
end
Img_Decrypted = imcrop(Img_Decrypted,[0 0 SizeOfOriginalPic(1,2) SizeOfOriginalPic(1,1)]);
```

با استفاده از یک نمونه کلید ۱۲۸ بیتی نتیجه‌ی حاصل به صورت زیر است:



شکل ۵-۱ تصویر با شیوه کتابچه رمز و رمز قطعه‌ای AES

در شکل ۵-۱ همانطور که مشاهده می‌شود قسمت‌های یکسان تصویر و پیش‌زمینه‌ی آن به صورت یکسان رمزشده‌اند. در زیر تصویر رمزشده با الگوریتم DES را نشان داده است، در این الگوریتم بلوک‌های تصویر به سایز ۲×۴ شکسته شده‌اند و ۶۴ بیت اول کلید استفاده‌شده است و هر گاه کاربر نیاز به رمزگذاری با DES یا 3-DES داشته باشد باید تصویر به جای بلوک‌های ۴×۴ در روش‌های AES و Serpent به سایز ۲×۴ شکسته می‌شود زیرا که این الگوریتم‌ها برای بلوک ۶۴ بیتی طراحی‌شده است و همچنین در DES از کلیدهای ۶۴ بیتی استفاده می‌شود ولی در 3-DES یک کلید ۱۲۸ بیتی به دو کلید ۶۴ بیتی شکسته شده و در طول الگوریتم استفاده خواهد شد.



شکل ۵-۲ تصویر با شیوه‌ی کتابچه‌ی رمز و رمز قطعه‌ای DES

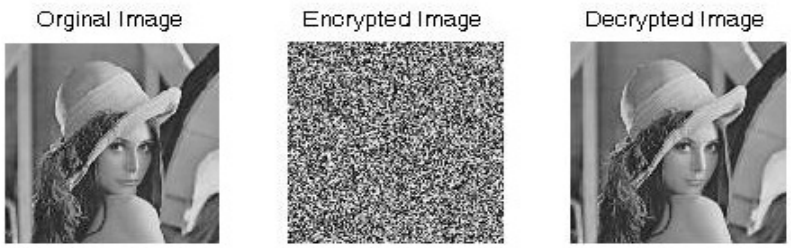
شیوه‌هایی که در ادامه استفاده می‌شوند، برای رمزنگاری از یک مقدار تصادفی استفاده می‌کنند، و این مقدار رمزشده برای گیرنده‌ی پیام ارسال می‌شود:

```
n = 255; % it will use for mapping
IV(1:4,1:4)= floor(n.*rand(4,4)); %Produce 16 random number and map them to 0~255
```

```
IV=double(IV);
```

در شیوه‌ی زنجیره‌سازی بلوک‌های رمز دیگر از تابع کتاب‌خانه‌ای نمی‌توان استفاده‌کرد چون هر بلوک رمزشده باید تاثیر خود را بر بلوک بعد از خود بگذارد به همین خاطر توسط mat2cell تصویر به بلوک‌های ۴×۴ یا ۲×۴ شکسته می‌شود:

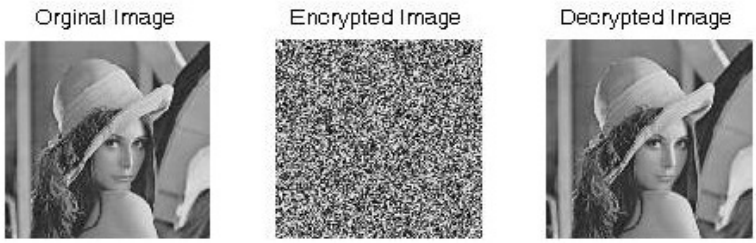
```
% Vector1 = Vector2 = 4
Cell=mat2cell(I,Vector1,Vector2); % breaking the image to 4*4 cells
Cel2=bitxor(IV,Cell{1,1});
Cell{1,1}=fun1(Cel2,RoundKey,S_Box);
for i=1:Rows
   for j=1:Cols
          if j~=1 % i~=1///i==1   %if it's not the first-column cell
                Cel2=bitxor(Cell{i,j},Cell{i,j-1});
                Cell{i,j}=fun1(Cel2,RoundKey,S_Box);
          elseif i~=1 && j==1     %if it's the first-column cell
                Cel2=bitxor(Cell{i,j},Cell{i-1,Cols});
                Cell{i,j}=fun1(Cel2,RoundKey,S_Box);
          end
   end
end
Img_Encrypted=cell2mat(Cell); % Joining the cells to one matrix
```



شکل ۵-۳ تصویر با شیوه CBC و رمز قطعه‌ای سِرپنت

در شیوه‌ی فیدبک نه‌تنها تصویر به بلوک‌های ۴×۴ (به معنی۱۲۸بیتی) شکسته می‌شود بلکه باز هم تابع رمزنگاری جداگانه بر روی هر یک از سلول‌های تصویر نیز اعمال می‌شود. قطعه کد برنامه به شکل زیر است:

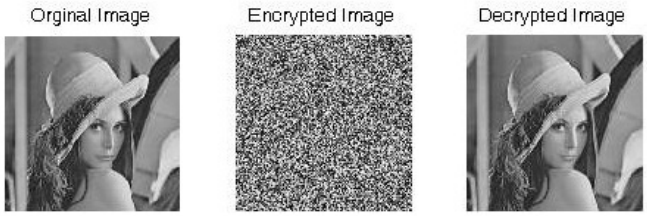
```
%   C = MAT2CELL(X,M,N) breaks up the 2-D array X into a cell array of
%   adjacent submatrices of X. X is an array of size [ROW COL], M is the
%   vector of row sizes (must sum to ROW) and N is the vector of column
%   sizes (must sum to COL).
Cell=mat2cell(I,Vector1,Vector2); % Vector1 = Vector2 = 4
    for i=1:Rows
        for j=1:Cols
            for k=1:4
                for s=1:4
                    IV_Encrypt=fun1(IV,RoundKey,S_Box);
                    Cell_encr{i,j}(k,s)=bitxor(IV_Encrypt(1,1),Cell{i,j}(k,s));
                    IV=ShiftLeft(IV,Cell_encr{i,j}(k,s)); % Shift IV to left
                end
            end
        end
    end
Img_Encrypted=cell2mat(Cell_encr);
```



شکل ۵-۴ تصویر با شیوه CFB و رمز قطعه‌ای سرپنت

در شیوه‌ی فیدبک خروجی دیگر خروجی رمزنگار که با مقدار داده یای‌انحصاری‌شده، فیدبک نمی‌شود بلکه این بار خروجی رمزنگار است که برای مرحله‌ی بعد برگردانده می‌شود:

```
Cell=mat2cell(I,Vector1,Vector2);    % Vector1 = Vector2 = 4
IV2=IV;
for i=1:Rows
    for j=1:Cols
        for k=1:4
            for s=1:4
               IV_Encrypt=functn(IV,RoundKey,S_Box); %Encrypting with the BlockCipher
               Algorithm the 128-bit IV
               IV=ShiftLeft(IV2,IV_Encrypt(1,1));  % Shift IV to left for feedBacking The 8-
               bit encrypted value
               Cell_Encryption{i,j}(k,s)=bitxor(IV_Encrypt(1,1),Cell{i,j}(k,s));
            end
        end
    end
end
Img_Encrypted=cell2mat(Cell_Encryption);
```

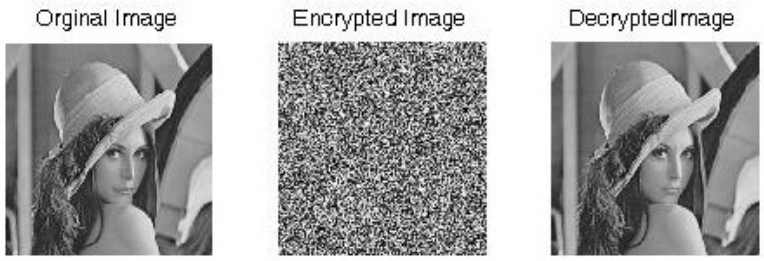


شکل ۵-۵ تصویر با شیوه OFB و رمزقطعه‌ای سرپنت

در شیوه‌ی شمارنده نیز از یک مقدار تصادفی به عنوان شمارنده استفاده می‌شود به طوری‌که باید به تعداد بلوک‌ها قابلیت شمارش داشته باشد و این‌گونه نباشد که برخی قطعه‌ها با مقادیر یکسان رمز شوند.

```
Cell=mat2cell(I,Vector1,Vector2);
for i=1:Rows
    for j=1:Cols
            IV=double(IV);
```

```
            Cel2=fun1(IV,RoundKey,S_Box);
            Cell{i,j}=bitxor(Cel2,Cell{i,j});
            IV=uint8(IV);
            %   incremnting the counter one number
            IV_new=hex2dec(strcat(dec2hex(IV(4,1),2),dec2hex(IV (4,2),2),…
                   dec2hex(IV (4,3),2),dec2hex(IV(4,4),2))) +1 ;
            IV_new =dec2hex(IV_new,8);%   converting decimal number to hexadecimal
            % concating 2 hexadecimal number then each 8-bit number is converted to
            decimal one
            IV(4,1)=hex2dec(strcat(IV_new(1,1),IV_new(1,2)));
            IV(4,2)=hex2dec(strcat(IV_new(1,3),IV_new(1,4)));
            IV(4,3)=hex2dec(strcat(IV_new(1,5),IV_new(1,6)));
            IV(4,4)=hex2dec(strcat(IV_new(1,7),IV_new(1,8)));
    end
end
Img_Encrypted=cell2mat(Cell);
```



شکل ۵-۶ تصویر با شیوه CTR و رمزقطعه‌ای AES

در همه‌ی الگوریتمها و شیوه‌های مختلف برای تصویر رنگی ابتدا با شرط `if isrgb(Im)` رنگی‌بودن تصویر بررسی‌شده و در صورت رنگی بودن تصویر به صورت زیر تک‌تک مولفه‌هایش تفکیک می‌شوند:

```
% Extract the individual red, green and blue color channels.
redChannel = Im(:, :, 1);
greenChannel = Im(:, :, 2);
blueChannel = Im(:, :, 3);
```

در مقصد نیز پس از رمزگشایی هر مولفه به کمک دستور زیر آن‌ها با یکدیگر ادغام می‌شوند:

```
Img_Decrypted = cat(3,Img_Decrypted_red,Img_Decrypted_green,Img_Decrypted_blue);
```

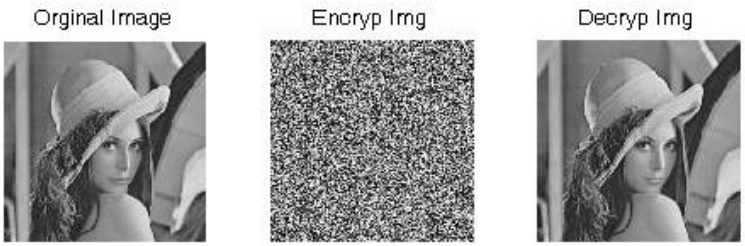
البته قبل از این مرحله به دلیل تبدیل هر مولفه تصویر به کلاس double، باید تک تک مولفه‌ها به کلاس uint8 تبدیل شوند:

```
Img_Decrypted_red=uint8(Img_Decrypted_red);
Img_Decrypted_green=uint8(Img_Decrypted_green);
Img_Decrypted_blue =uint8(Img_Decrypted_blue);
```

امّا شیوه‌ی مطرح این پروژه، شیوه‌ی OCB می‌باشد که به طور کامل در قسمت پیوست آورده‌شده است. به ابتدای نانس ۹۶ بیتی، مقدار ۳۲ بیتی «۰۰۰۰۰۰۰۱» هگزادسیمال را اضافه کرده، سپس آخرین ۶ بیت را در متغیر Bottom ذخیره‌کرده (حاصل مقداری است بین ۰ تا ۶۳) و به جای این شش بیت مقدار صفر قرار داده می‌شود.

مقدار ۱۲۸ بیتی محاسبه‌شده را در متغیری به نام Top ذخیره‌کرده و با رمز قطعه‌ای $Ktop = E_K (Top)$ رمزکرده و مقداری برابر ۲۵۶بیت به‌شکل $Stretch = Ktop \| (Ktop \oplus (Ktop<<8))$ در متغیر Stretch قرار داده می‌شود؛ $Ktop<<8$ یعنی این‌که متغیر را به اندازه‌ی ۸ بیت به سمت چپ شیفت می‌دهد. متغیری آفست اولین ۱۲۸ بیت متغیر $Stretch<<Bottom$ است که آفست اولیه نامیده خواهد شد [۹].

اما چرا به این محاسبات عجیب و غریب نیاز است؟ اگر نانس یک شمارنده باشد، Ktop هر ۶۴ مرتبه یک‌بار عوض می‌شود. در نتیجه اگر Ktop به اندازه‌ی کافی بزرگ باشد، هر آن‌چه که برای محاسبه‌ی مقدارگذاری اولیه نانس نیاز است، ۶۳ مرتبه از ۶۴ مرتبه، یک عمل منطقی برای استخراج شش بیت کم‌ارزش نانس N و سپس یک شیفت منطقی در رشته‌ی Stretch با این تعداد بیت است. علاوه بر این‌ها، در سیکل ۶۴ اُم دیگر به یک فراخوانی الگوریتم رمز بلوکی نیازی نیست. هزینه‌ی میانگین در فراخوانی‌های متوالی پایین است و به اجرای چیزی شبیه ضرب در میدان گالوا $GF(2^{128})$ دیگر نیازی نیست. برای مقدار S با اندازه‌ی ۱۲۸ بیت اگر بیت سمت راست آن صفر باشدکه $Doubl(S) = S << 1$ و در غیر این‌صورت $Doubl(S) = (S << 1) \oplus 135$ اجرا می‌شود. یعنی این تابع، ضرب S در میدان محدود گالواست که از ۱۲۸ نقطه تشکیل شده است و حاصل ضرب نیز در همین محدوده‌ی نقاط قرار می‌گیرد. بنابراین اگر شرط اول برقرار باشد که فقط متغیر S یک بیت شیفت داده شده در غیر اینصورت با مقدار ۱۳۵ نیز یای‌انحصاری می‌شود [۹].



شکل ۵-۷ تصویر با شیوه‌ی OCB و رمزقطعه‌ای AES

۵-۲- نتیجه‌گیری

در بین الگوریتم‌های پیاده‌سازی‌شده الگوریتم رمز قطعه‌ای DES با وجود سادگی پیاده‌سازی به دلیل طول کلید کوتاه‌تر و محاسبات کمتر، نسبت به بقیه الگوریتم‌ها سریع‌تر اجرا می‌شد. به دنبال آن 3-DES نیز به‌علّت اجرای سه مرتبه از الگوریتم DES کندتر اجرا می‌شد ولی به علت داشتن فضای کلید بیشتر، امن‌تر به حساب می‌آید. الگوریتم راین‌دال-۱۲۸ به‌دلیل تعداد دورهای کمتر و درعین حال محاسبات پیچیده‌تر الگوریتم سریعی به شمار می-رفت. در این فصل در بین شیوه‌های مختلف پیاده‌سازی شده، روش شمارنده در عین سادگی از امنیت بیشتری برخوردار بود ولی روش‌های فیدبک با رمزکردن جداگانه‌ی هر هشت بیت از بلوک داده، نیاز به زمان اجرای بیشتری داشتند. در انتها روش OCB، با محاسبه‌ی تابع چکیده‌ی پیام طراحی‌شده به روشی متفاوت، ارائه شد. در این روش شناسه‌ی فرستنده‌ی پیام را، به عنوان مثال نام کامپیوتر رمزگذار تصویر، در شناسنامه‌ی تصویر رمزشده قرار داده تا گیرنده پس از رمزگشایی هویت فرستنده برایش مسجّل شود. در ضمن مهر زمان هم در شناسنامه‌ی پیام قرار داده شده است تا از حمله‌ی تکرار جلوگیری شود.

فصل ششم

بررسی نتایج پیاده‌سازی‌های انجام‌شده

۶-۱- مقدمه

در این فصل ضمن ارایه‌ی تعاریفی در مورد روش‌های مختلف مقایسه‌ی تصاویر دیجیتال و رمزشده‌ی آن‌ها، سعی در بررسی نتایج حاصل از کار خواهد بود تا بتوان تفاوت بین شیوه‌های مختلف رمزنگاری متقارن را بیشتر درک کرد.

۶-۲- NPCR[1] and UACI[2]

درصد تعداد پیکسل‌های متفاوت بین دو تصویر رمزشده و رمزنشده را نشان می‌دهد. نرخ تغییرات تعداد پیکسل‌ها و نیز وضوح تغییرات متوسط به ترتیب به صورت زیر تعریف می‌شوند [۱۳]:

$$\mathrm{NPCR} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} D(i,j)}{m \times n} \times 100\%$$

رابطه‌ی ۶-۱ محاسبه‌ی درصد پیکسل‌های متفاوت

شدّت یکنواختی تغییرات را نشان می‌دهد به این معنا که مقادیر به چه اندازه به صورت یکنواخت دستخوش تغییر شده‌اند.

$$\mathrm{UACI} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} |A(i,j) - B(i,j)|}{255 \times m \times n} \times 100\%$$

رابطه‌ی ۶-۲ محاسبه‌ی درصد شدت یکنواختی دو ماتریس

[1] Number of Pixel Change Rate
[2] Unified Average Changing Intensity

که A و B به ترتیب تصاویر اصلی و تصویر رمز شده هستند:

$$D(i,j) = \begin{cases} 0, A(i,j) = B(i,j) \\ 1, A(i,j) \neq B(i,j) \end{cases}$$

رابطه‌ی ۶-۳ محاسبه‌ی تعداد مقادیر یکسان دو ماتریس

قطعه برنامه‌ی متلب این روش‌ها در زیر آورده شده است:

```
function OUT=NPCR(Im,Im_Encrypted)
OUT=0;
[ROW,COL]=size(Im);
for i=1:ROW
    for j=1:COL
        if Im(i,j) == Im_Encrypted(i,j)
            OUT=OUT+0;
        elseif Im(i,j) ~= Im_Encrypted(i,j)
            OUT=OUT+1;
        end
    end
end
OUT=(OUT/(ROW*COL))*100;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function OUT=UACI(Im,Im_Encrypted)
%%%%% Unified Average Changing Intensity %%%%%
OUT = ((mean (mean ( Im(:)-Im_Encrypted(:) )))/(255))*100;
```

۶-۳- ضریب همبستگی[1]

ابزاری آماری برای تعیین نوع و درجه رابطه‌ی یک متغیر کمی با متغیر کمی دیگر است. ضریب همبستگی، یکی از معیارهای مورد استفاده در تعیین همبستگی دو متغیر است. ضریب همبستگی شدت رابطه و همچنین نوع رابطه (مستقیم یا معکوس) را نشان می‌دهد. این ضریب بین ۱ تا ۱- است و در صورت عدم وجود رابطه بین دو متغیر، برابر صفر است [۴].

حال برای محاسبه‌ی این ضریب برای تصاویر اصلی و رمزشده ابتدا باید مجموعه‌ای از جفت پیکسل‌ها را از تصویر مورد نظر انتخاب کرد، به عنوان مثال همه‌ی جفت پیکسل‌های افقی، یا همه‌ی جفت پیکسل‌های عمودی یا جفت پیکسل‌های قطری و اریب. سپس فرمول آماری CorrCoef به مجموعه‌های بالا اعمال خواهد شد. ابتدا با جفت پیکسل‌های افقی یکتا شروع می‌شود، اگر پیکسل‌ها از اولین ستون تصویر انتخاب و در مجموعه‌ی x قرار داده‌شوند آن‌گاه تمام پیکسل‌های افقی مجاور پیکسل‌های ستون دوم خواهند بود که در مجموعه‌ی y قرار می‌گیرند، و به همین ترتیب اگر تا ستون آخر ادامه شود، مشاهده خواهد شد که پیکسل‌های ستون یک تا یکی مانده به آخرین ستون، مجاور پیکسل‌های ستون دو تا آخرین ستون در تصویر خواهند بود [۱۷].

[1] Correlation Coefficient

```
x = Im(:,1:end-1);  %# All rows and columns 1 through 127
y = Im(:,2:end);    %# All rows and columns 2 through 128
```

با منطقی مشابه بالا می‌توان مجموعه‌ای از جفت پیکسل‌های مجاور منحصر به فرد عمودی را نیز این چنین تعریف کرد:

```
x = Im(1:end-1,:);  %# Rows 1 through 127 and all columns
y = Im(2:end,:);    %# Rows 2 through 128 and all columns
```

به علاوه مجموعه‌ای از جفت پیکسل‌های مجاور منحصر به فرد قطری این‌چنین تعریف می‌شود:

```
x =    Im(1:end-1,1:end-1);  %# All but the last row and column
y = Im(2:end,2:end);    %# All but the first row and column
```

سپس با استفاده از تابع کتاب‌خانه‌ای CorrCoef ضریب همبستگی در متلب محاسبه می‌شود:

```
r_xy = corrcoef(x(:),y(:));
```

هنگام فراخوانی این تابع ورودی‌های x و y ، به بردارهای تک‌ستونی تغییر شکل می‌دهند. و این ضریب بین جفت پیکسل‌های مورد نظر حساب می‌شوند [۱۷].

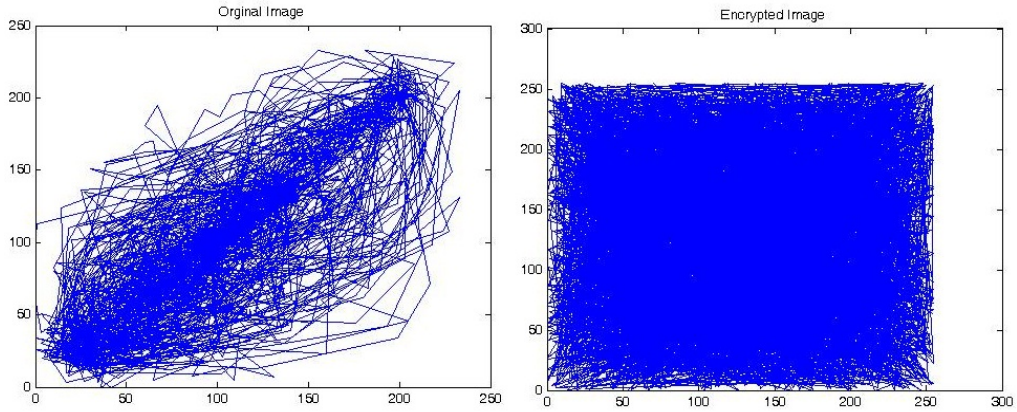
در این پروژه از تصویر موجود در کتاب‌خانه‌ی متلب، تصویر لنا استفاده‌شده است که ضریب همبستگی بین پیکسل‌های افقی آن ۰،۸۰۶۷۵۶ و بین پیکسل‌های عمودی آن ۰،۹۱۸۸۳۷ می‌باشد.

حالتهای مختلف برای r_xy:

- $r\_xy = 1$ همبستگی کامل و مستقیم است. با افزایش مقدار x مقدار y به طور قطعی زیاد می‌شود.
- $r\_xy = -1$ همبستگی را کامل و معکوس است. با افزایش مقدار x مقدار y کاهش می‌یابد.
- $-1 < r\_xy < 0$ همبستگی ناقص و معکوس است. با افزایش مقدار x مقدار y به طور نسبی کاهش می‌یابد.
- $0 < r\_xy < 1$ همبستگی ناقص و مستقیم است. با افزایش مقدار x مقدار y به طور نسبی افزایش می‌یابد.

شکل ۶-۱ نمودار همبستگی بین پیکسل‌ها را در تصویر اصلی و رمز شده نشان می‌دهد. برای محاسبه‌ی ضریب همبستگی از قطعه برنامه‌ی زیر استفاده می‌شود:

```
disp('The Correlation Coefficient of the Original Image:');
x = Im2(:,1:end-1);  %# All rows and columns 1 through end-1
y = Im2(:,2:end);    %# All rows and columns 2 through end
h_xy_Im = corrcoef(x(:),y(:));     fprintf('Horizental is: %f\n',h_xy_Im(1,2));
figure,plot(x(:),y(:))   %# ploting the coefficent each pixel to its neighbor
```



شکل ۶-۱ ضریب همبستگی افقی برای تصویر اصلی و تصویر رمزشده با مد کاری **OCB** و رمز بلوکی **AES**

۶-۴- ماکزیمم نسبت سیگنال به نویز یا PSNR[1]

یک اصطلاح فنی است که نسبت بین بالاترین مقدار قدرت سیگنال تصویر اصلی و قدرت سیگنال تصویر رمزشده را نشان می‌دهد. از آن‌جا که سیگنال‌ها بازه‌ی دینامیکی وسیعی دارند، این مقدار در مقیاس دسیبلِ لگاریتمی نشان داده می‌شود. این مقدار MSE با استفاده از رابطه‌ی زیر قابل محاسبه است. در این رابطه A و B تصاویر تک-فام مورد نظر هستند که ابعاد m*n را دارند. تصویر B رمز شده‌ی تصویر A است [۴].

$$MSE = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n}\left[A(i,j)-B(i,j)\right]^2}{m \times n}$$

رابطه‌ی ۶-۴ محاسبه‌ی میانگین مجذور خطا

مقدار PSNR با رابطه‌ی زیر تعریف می‌شود که در آن MAX_I ماکزیمم مقدار ممکن پیکسل تصویر اصلی می‌باشد. در حالتی‌که پیکسل‌های تصویر با استفاده از هشت بیت نشان داده‌می‌شوند، این مقدار برابر با ۲۵۵ می‌باشد. در حالت کلّی هنگامی‌که نمونه‌های تصویر با استفاده از مدولاسیون کُد پالس[2] خطی با b بیت در هر نمونه نشان داده می‌شوند، این مقدار برابر با 2^b-1 می‌شود [۴].

$$PSNR = 10.\log_{10}\frac{MAX_I^2}{MSE}$$

رابطه‌ی ۶-۵ محاسبه‌ی حداکثر مقدار سیگنال به نویز

برای بدست آوردن این مقادیر از قطعه برنامه‌ی زیر استفاده شده است. تابع PSNR به شرح زیر است:

[1] peak signal-to-noise ratio
[2] Pulse Code Modulation (PCM)

```
function p = psnr(x,y)
% psnr - compute the Peak Signal to Noise Ratio
disp('The Mean Square Error between the Original Image & Encrypted Image :');
mse =MSE(x,y)     %  computing the mean square error
p = 10*log10 (double((255^2)/mse)); %  the logarithm is is in base 10
```

که در آن تابع MSE به صورت مقابل بدست می‌آید:

```
function M_S_E=MSE(Im,Im_Encrypted)
M S E = mean( mean( (double(Im(:))-double(Im Encrypted(:))).^2 ) );
```

طبق نتایج بدست‌آمده در جدول ۶-۱ الگوریتم Serpent در کل دارای سرعت پایین‌تری نسبت به روش AES است، به‌خصوص در شیوه‌های فیدبک‌دار به علّت رمزکردن هر بلوک ۸ بیتی به صورت جداگانه زمان بیشتری می‌گیرند. البته این نتایج در مورد الگوریتم سرپنت و نیز در مورد شیوه‌های فیدبک، دور از انتظار نبود، زیرا این رمز قطعه‌ای دارای دورهای پردازش بیشتری نسبت به راین‌دال است. الگوریتم شیوه‌ی OCB با وجود محاسبه‌ی مقدار درهم‌ساز، زمان مناسبی را صرف رمز کردن می‌کند.

جدول ۶-۱ نتایج بررسی‌ها با الگوریتمAES و Serpent روی تصویر خاکستری ۶۴×۶۴

Gray 64*64	AES						Serpent					
	NPCR%	UACI%	MSE	PSNR	Corr-Coef Encrypted	Time	NPCR%	UACI%	MSE	PSNR	Corr-Coef Encrypted	Time
ECB	99.68	4.84	$8.835*10^3$	8.6688	0.012503	0 min 58.626450	99.76	0.42	$8.6997*10^3$	8.7358	0.022068	1 min 27.113531
CBC	99.68	0.80	$8.985*10^3$	8.5956	0.009400	0 min 58.459779	99.58	2.3729	$8.8004*10^3$	8.6858	0.009201	1 min 31.196993
CFB	99.51	5.70	$8.734*10^3$	8.7188	-0.007993	9 min 8.098195 s	99.61	0.0012	$8.7638*10^3$	8.7039	0.014448	23 min 16.630156
OFB	99.63	0.08	$8.698*10^3$	8.7367	-0.005332	8 min 55.322738	99.66	14.4712	$8.8132*10^3$	8.6794	-0.000664	23 min 34.059977
CTR	99.54	9.06	$8.816*10^3$	8.6776	-0.003889	0 min 43.603469	99.68	16.0994	$8.8318*10^3$	8.6703	-0.033927	1 min 25.935126
OCB	99.59	22.90	$8.723*10^3$	8.7244	0.009844	1 min 1.395948 s	99.51	7.8878	$8.7260*10^3$	8.7226	0.007053	2 min 21.038356

نتایج این جدول نشان می‌دهد که مقدار NPCR در تمام روش‌ها نرخ تغییر تعداد پیکسل‌های تقریباً یکسان و در حد عالی دارند. مقادیر UACI نشان می‌دهند که در روش شمارنده در الگوریتم سِرپنت و نیز OCB با الگوریتم AES، پیکسل‌های تصاویر را به صورت یکنواخت دستخوش تغییر کرده‌اند و این مقدار هر چه‌قدر زیاد باشد بهتر است. میانگین مجذور خطا برای روش CBC با رمزقطعه‌ای AES بیشترین مقدار و برای شیوه‌ی فیدبک کمترین مقدار را دارد. در مقایسه‌ی PSNR که افت کیفیت تصویر پس از رمزنگاری را نشان می‌دهد، در شیوه‌ی زنجیره-سازی بلوک رمز کمترین مقدار را دارد. البته این اختلاف با شیوه‌های دیگر بسیار ناچیز است و می‌توان گفت افتی که تمام روش‌ها ایجاد می‌کنند، در یک بازه قرار دارد. همان‌طور که در شکل ۶-۱ نیز نشان داده شد، پس از اعمال شیوه‌ی OCB نیز توزیع همبستگی بین پیکسل‌ها از خاصیت گمراه‌کنندگی حمایت می‌کند. این نتایج نیز در جدول هم روشن هستند. در جدول ۶-۲ نتایج مربوط به بررسی همین شیوه‌ها ولی با الگوریتم‌های DES و 3-DES نشان-

داده شده است. این الگوریتم اگرچه نتایج یکسانی با رمزقطعه‌ای AES و Serpent دارد ولی به دلیل طول کلید کوتاه‌تر و نیز بلوک‌های داده ۶۴ بیتی، دیگر مورد استفاده قرار نمی‌گیرد.

جدول ۶-۲ نتایج بررسی‌ها با الگوریتم DES و 3-DES روی تصویر خاکستری ۶۴×۶۴

Gray 64*64	3-DES						DES					
	NPCR%	UACI%	MSE	PSNR	Corr-Coef Encrypted	Time	NPCR%	UACI%	MSE	PSNR	Corr-Coef Encrypted	Time
ECB	99.56	25.95	$8.484*10^3$	8.8450	-0.030633	2 min 46.940670	99.46	15.81	$8.9461*10^3$	8.6147	0.010538	0 min 55.235669
CBC	99.43	11.78	$8.861*10^3$	8.6559	0.020005	2 min 33.314102	99.60	0.02	$9.0443*10^3$	8.5671	-0.018393	0 min 56.716301
CFB	99.68	11.48	$8.944*10^3$	8.6154	0.017089	21 min 8.294339	99.56	0.014	$8.7394*10^3$	8.7160	0.024632	6 min 55.620923
OFB	99.68	0.18	$8.816*10^3$	8.6779	-0.013185	22 min 3.577893	99.71	19.99	$8.5676*10^3$	8.8022	0.016507	7 min 7.123191
CTR	99.95	26.65	$9.291*10^3$	8.4501	0.068445	2 min 43.121379	99.58	11.26	$9.3966*10^3$	8.4011	0.004777	0 min 55.480588
OCB	99.39	0.1519	$8.757*10^3$	8.7070	0.006003	3 min 30.395948	99.59	1.08	$8.7157*10^3$	8.7278	-0.001625	1 min 1.038356

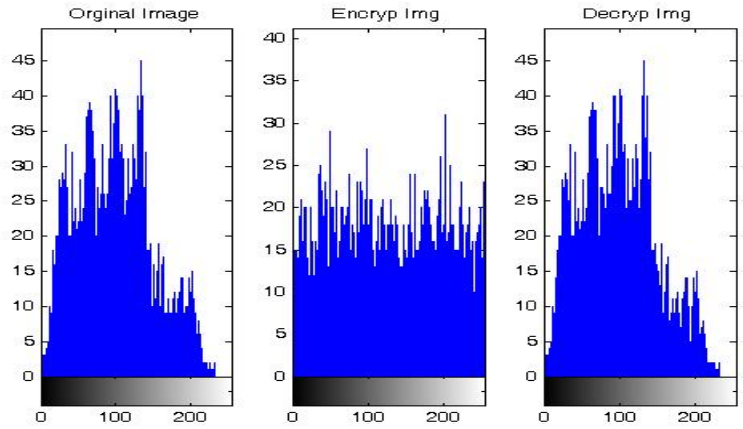
در این جدول نیز مقادیر NPCR شیوه‌های مختلف یکسان و در حد ایده‌آلی است. ولی مقدار UACI در روش ECB از همه بیشتر و در شیوه‌ی OCB از همه کمتر است. مقدار ماکزیمم سیگنال به نویز در روش شمارنده از همه کمتر است، پس افت کیفیت تصویر از همه کمتر است. همبستگی بین پیکسل‌ها پس از اعمال روش OCB از همه کمتر است و به همین دلیل این روش مناسب به نظر می‌آید.

زمان‌ها در دو جدول بالا نشان‌دهنده‌ی این هستند که روش‌های فیدبک سرعت‌های بسیار پایینی دارند، در ضمن شیوه‌ی OCB با وجود محاسبه‌ی چکیده‌ی پیام، سرعتش نسبت به روش شمارنده با رمز قطعه‌ای AES بسیار مناسب است. در بین الگوریتم‌های رمز قطعه‌ای همان‌گونه که انتظار داشتیم روش سِرپِنت نیز از بقیه‌ی الگوریتم‌ها به خاطر دورهای پردازش بیشتر، زمان طولانی‌تری صرف رمزکردن می‌کند.

۶-۵- هیستوگرام تصاویر اصلی و رمزشده

نمودار هیستوگرام تنها در تصاویری از نوع خاکستری، برای نمایش میزان کنتراست تصویر (محور افقی) نسبت به شدّت نور تابیده‌شده به تصویر (محور عمودی) می‌باشد. این نمودار برای تصاویر دو بعدی میزان دامنه‌ی شدّت نور را در مقادیر مختلف کنتراست، که هم به صورت دیجیتالی و هم به صورت رنگی در محور افقی نمایش داده‌شده است، بیان می‌کند. این نمودار بهترین روش برای مقایسه‌ی دو تصویر از یک نما خواهد بود. برای نمایش این تابع از دستور imhist(IM) استفاده می‌شود [۲].

می‌توان مشاهده‌کرد که هیستوگرام تصویر رمزشده به طور مساعدی یکنواخت و به طور قابل توجهی با هیستوگرام تصویر اصلی متفاوت است. همچنین از هیستوگرام نیز می‌توان متوجّه شد که تفاوت فاحشی بین درصد تعداد پیکسل‌ها در مقادیر مختلف عکس خاکستری در بازه‌ی ۰ تا ۵۰ وجود دارد. این تفاوت برای تصویر رمزشده بسیار یکنواخت است. در زیر هیستوگرام شیوه‌ی OCB با رمز قطعه‌ای AES آورده‌شده است.



شکل ۶-۲ هیستوگرام تصویر رمزشده به شیوه OCB با رمز قطعه‌ای AES

۶-۶- مقایسه‌ی عملکرد نرم‌افزاری مُدهای احراز هویت و رمزنگاری[1]

در مقاله‌ی [۱۱] عملکرد سه مُد احراز هویت و رمزنگاری[1] که به اختصار AE نامیده می‌شود، CCM و GCM و OCB بررسی‌شده است. در چند سال اخیر تلاش‌های فراوانی در جهت ایجاد طرح‌های احراز اصالت و رمزنگاری شده است. یک دلیل اینست که طرحی با ارائه‌ی سرویس محرمانگی و اصالت پیام درکنار هم، از تکنیک‌هایی که جداگانه این سرویس‌ها را فراهم می‌کنند، کاراتر است.

برای این طرح‌ها دو پیشنهاد وجود دارد: یکی طرح دوگذری[2] که محرمانگی و احراز اصالت را جداگانه با هم ادغام می‌کند، مثلاً ابتدا با مُد شمارنده پیام رمزشده و سپس با مُد CBC-MAC چکیده‌ی پیام محاسبه می‌شود و یکی طرح یک گذری[3] یا جامع[4] است که در آن مکانیزم بخش‌هایی که مسئول رمز و اعتبار سنجی پیام هستند به هم وابسته‌اند. این قبیل روش‌ها در دهه قبل با فعالیت‌های Jutla آشکار شدند [۱۱]. در واقع Jutla دو طرح ارائه داد:

[1] AE (Authentication and Encryption)
[2] two-pass
[3] one-pass
[4] Integrated

یکی CBC-like (IACBC) و دیگری ECB-like (IAPM) که طرح IAPM مبنای کار OCB قرار گرفت و ایده‌های بسیاری از این طرح را در خود جا داد والبته حیله‌های بسیاری نیز به آن اضافه شده است [۹].

بعد از مقاله‌ی McGrew و Viega در سال ۲۰۰۴ ، هیچ مطالعه‌ی دیگری در مورد مقایسه‌ی کارایی این الگوریتم‌ها صورت نگرفت. در ضمن CCM و GCM به طور کامل مورد اهمیت واقع شدند؛ مثلاً CCM برای امنیت WiFi (802.11i) مدرن و GCM در پروتکل‌های IPsec و TLS [۱۱].

جدول ۶-۳طرحهای احراز اصالت و رمزنگاری برای الگو [۱۱]

نام طرح	تاریخ	توصیف سطح بالا	استاندارد
IAPM	۲۰۰۱	اولین مُد اصلی طرح جامع	---
OCB1	۲۰۰۱	طرح بهینه‌شده‌ی شبیه IAPM	---
CCM(RFC 3610)	۲۰۰۲	CTR Encryption + CBC MAC	NIST 800-38C
GCM(RFC 4106)	۲۰۰۴	CTR Encryption + GF(2^{128})-based MAC	NIST 800-38D
EAX	۲۰۰۴	CTR Encryption + CMAC یک نوع CCM اصلاح‌شده	ISO 19772
OCB2	۲۰۰۴	OCB1 با داده‌ی همراه و افزایش سرعت	ISO 19772
OCB3	۲۰۱۰	بهبود نسخه‌های قبلی OCB	---

در بررسی‌های انجام‌شده McGrew و Viega (طراحان شیوه‌ی GCM) دو موضوع مورد بحث را در طراحی OCB1 تشخیص دادند: ابتدا این مُد m+2 فراخوانی رمز قطعه‌ای را برای رمز کردن یک پیام به تعداد m=[|M| /128] بلوک نیاز دارد. در حالی که GCM با m+1 فراخوانی این کار را انجام می‌دهد. مورد دوم این- که OCB1 قبل از فراخوانی الگوریتم AES دیگری، از یک نتیجه‌ی الگوریتم AES دوبار استفاده می‌کند. چه در سخت‌افزار و چه در نرم‌افزار این موضوع موجب افت کارایی می‌شود. علاوه بر این ، مُدهای جامعِ حاضر نمی‌توانند از موقعیت شمارنده‌ها در مُد CTR استفاده کنند–در شمارنده‌ها بیت‌های پرارزش بلوک‌های ورودی معمولاً بدون تغییر باقی می‌مانند و بیت‌های کم‌ارزش به ازای هر قطعه‌ی جدید افزایش می‌یابند. با همه‌ی این مسائل، ممکنست که GCM سرعتش از OCB بالاتر باشد و به شکل عمومی‌تری بتوان گفت، ممکنست که طرح‌های ترکیبی سرعتشان از طرح‌های جامع بیشتر باشد. این امکان توسط MAC های قابل محاسبه با سرعتِ بسیار بالاتر فراهم می‌شود [۱۱].

همان‌گونه که در فصل چهارم ذکر شد در شیوه‌ی OCB، وقتی‌که عدد تصادفی یک شمارنده است، مُدی که نامش OCB3 است، برای رمزکردن هر پیام، یک عدد رمزگذار AES را کمتر محاسبه می‌کند و تنها این فراخوانی در ۹۸٪ زمان انجام می‌شود. منظور از این‌که عدد تصادفی یک شمارنده است، این است که در یک نشست، بخشِ باارزش نانس ثابت است در حالی‌که به ازای هر پیام، قسمت کم‌ارزش این عدد یکی اضافه می‌شود. این پیشنهادی است که در RFC 5116 توصیه شده است، و راه مرسوم برای استفاده از یک طرح رمزنگاری و احراز هویت AE می‌باشد. اگر نانس یک شمارنده نباشد، چیزی شبیه ضرب در میدان گالوا ($GF(2^{128})$) برای جبران فراخوانی رمز بلوکی حذف شده، معرفی نخواهد شد و البته در مقایسه با OCB1 نیز جریمه‌ی قابل توجّهی پرداخت نمی‌شود [۱۱].

در ادامه کارایی نرم‌افزاری CCM و GCM و نسخه‌های مختلفی از OCB مطالعه می‌شود. سریع‌ترین کُد در دسترسِ عموم برای اینتل x86، با و بدون دستورالعمل‌های جدید اینتل برای سرعت بخشیدن به AES به کار برده‌شده است. برای سکوهای سخت‌افزاری دیگر مانند: ARM، PowerPC، SPARC از کتاب‌خانه‌ی اصلاح‌شده و پرطرفدار OpenSSL استفاده شده است. سرعت رمزنگاری روی پیام‌ها با طول‌های مختلف از یک بایت تا یک کیلوبایت تست‌شده است. در این بررسی کُدهای OCB به زبان C نوشته شده‌اند. نتیجه این بود که با طول‌های مختلف پیام و سکوهای سخت‌افزاری متفاوت OCB سریع‌تر از CCM و GCM است. برخلاف یافته‌های McGrew و Viega، اختلافات مشاهده‌شده در سرعت، بین GCM و OCB1 زیاد است. به‌عنوان مثال یافته‌ها نشان دادند که پیام‌های 4KB روی یک پردازنده‌ی اینتل i5 (Clarkdale) برای CCM در 4.17cpb[1]، GCM در 3.73cpb، OCB1 در 1.48cpb، OCB2 در 1.80cpb، OCB3 در 1.48cpbرمزنگاری شده‌اند و در مقام قیاس، شیوه‌ی CTR در 1.27cpb اجرا شد. در این پیاده‌سازی‌ها، از دستورالعمل‌های جدیدِ AES[2] پردازنده، شامل ضرب بدون رقم نقلی، برای اجرای GCM استفاده شده‌است. سربار احراز هویت OCB3 (زمانی‌که این مُد اضافه بر مُد CTR صرف رمزنگاری می‌کند) حدود 0.21cpb و تفاوت بین سربار OCB و GCM یک ضریبی از ده است. ابتدا با تنظیم اعداد تصادفی و این‌که آخرین چند بیت انتهایی با فراخوانی رمز بلوکی یکسانی استفاده شوند نیاز بود. برای کاهش زمان اجرا و هزینه‌ی تولید کلید، یک روش درهم‌سازی جدید با نام Stretch-then-shift-xor-universalHash معرفی می‌گردد. کاهش نهفتگی بوسیله‌ی تغییرات در تعریف مُدها و عملکردشان روی CheckSum بدست می‌آیند. یک یافته‌ی تعجب‌آور این است که در همه‌ی سکوها، OCB2 کندتر از

[1] Clock per Byte
[2] AES-NI(AES-New Instructions)

OCB1 است. برای توضیح بیشتر یادآوری می‌شود که بیشتر طرح‌های جامع شامل محاسبه‌ی یک آفست برای هر بار فراخوانی رمز قطعه‌ای هستند و در OCB1 هر آفست بوسیله‌ی XOR یک مقدار وابسته به کلید (طبق پیشنهاد Jutla) محاسبه می‌شود. در OCB2 هر آفست بوسیله‌ی تابع Doubl در میدان محدود گالوا ($GF(2^{128})$ محاسبه می‌شوند. این پیشنهاد سریع‌تر، برای OCB3 انتخاب شد [۱۱].

حذف یک فراخوانی رمز قطعه‌ای: همان‌گونه که ذکر شد، نسخه‌های OCB1 و OCB2، به m+2 (m تعداد بلوک‌های پیام) مرتبه الگوریتم رمز قطعه‌ای را فراخوانی می‌کنند: یکی برای نگاشت نانس N به یک آفست اولیه، m تا به تعداد بلوک‌های۱۲۸ بیتی پیام M و در آخر یک فراخوانی نیز برای رمزکردن CheckSum نهایی. آفست اولیه با به‌کار بردن تابع درهم‌ساز جهانی و XOR محاسبه می‌شود: $offset[1] = H_k(N) = (Stretch \ll$ Bottom)[1 … 128]، که متغیر Bottom آخرین شش بیت N را در خود جای داده است و بقیه‌ی بیت‌هایش صفر شده است. متغیر Stretch مقداری (۱۲۸+۶۴) بیتی است که با رمزگذاری متغیری که آخرین شش بیت نانس را صفر کرده است، بدست می‌آید. استفاده از این تابع درهم‌ساز Stretch-then-shift، که Xor-universal نیز نامیده می‌شود، تضمین می‌کند هنگامی‌که نانس یک شمارنده است، آفست اولیه بدون محاسبه‌ی یک فراخوانی رمز قطعه‌ای جدید، در ۶۳/۶۴= ۹۸٪ زمان محاسبه شود. با این روش، هزینه‌ی محاسبه از m+2 فراخوانی تابع رمز قطعه‌ای به مقدار m+1.016 فراخوانی رمز قطعه‌ای باضافه‌ی زمان اندکی برای محاسبه‌ی تابع درهم‌ساز کاهش می‌یابد [۱۱].

کاهش نهفتگی: فرض کنید پیامی که رمز می‌شود ضریبی از ۱۲۸ بیت نباشد یعنی بلوک نهایی پیام تعداد بیت کم‌تری نسبت به بقیه‌ی بلوک‌ها داشته باشد. در نسخه‌های قدیمی‌تر OCB، برای محاسبه‌ی CheckSum، فراخوانی رمز قطعه‌ای باید روی بلوک یکی مانده به آخر متوقف و پس از اجرای آن دوباره رمز قطعه‌ای در آخرین مرتبه فراخوانی می‌شد. این موضوع موجب افت عملکرد خط لوله می‌شود و فقدان موازی‌سازی باعث افزایش میزان کار می‌گردد. برای مثال با روش قطعه بیتیِ AES پیاده‌شده توسط Schwabe و Kasper با استفاده از مُد ECB، هشت بلوک، به یک‌باره رمز می‌شوند ولی با استفاده از مُدهای OCB1 و OCB2 برای رمزکردن یک رشته‌ی ۱۰۰ بایتی به رمزگذاری تعداد بلوک‌های به مراتب بیشتر نیاز است، و از مرتبه‌ای به بعد باید روی خروجی AES منتظر بماند تا ورودی تابع AES مرتبه‌ی بعد تکمیل شود. در OCB3 ساختار الگوریتم تغییر یافته، بنابراین CheckSum به هیچ متنِ رمزی نیاز ندارد. به طور مشخص $CheckSum= M_1 \oplus M_2 \ldots \oplus M_m10^*$ برای بلوک آخر با سایز کوتاه‌تر و $CheckSum= M_1 \oplus M_2 \ldots \oplus M_m$ برای بلوک آخر با سایز کامل [۱۱].

افزایش آفست: در OCB1 هر آفست غیر اولیه با XOR مقداری مشتق‌شده از کلید با آفست قبلی بدست می‌آید. در OCB2 هر آفست غیراولیه با ضرب آفست قبلی ولی در میدان گالوا ($GF(2^{128})$) در یک مقدار ثابت محاسبه می‌شود. هر آفست $\Delta=(\Delta<<1) \oplus (MSB(\Delta) \| 135)$ در تابع Doubl محاسبه می‌شود. به نظر می‌رسد که Doubl به دلیل نیاز نداشتن به مراجعه‌ی حافظه یا پرداختن به اندیس i نسبت به روش اوّل سریع‌تر است. امّا جواب آزمایشات این‌گونه نبودند! زیرا که وقتی این تابع به پنج دستور اسمبلیِ Intel x86 – 64 bit کُد می‌شود، هنوز خیلی کند اجرا می‌شود. در برخی تنظیمات حتی Doubl کندتر هم هست، مثلاً در ماشین‌های ۳۲ بیتی و برخی کامپایلرها که در تبدیل کُد برنامه به زبان ماشین ضعیف عمل می‌کنند [۱۱].

نتایج آزمایشی: پیاده‌سازی بهینه‌ی CTR و GCM برای x86 بصورت عموم در دسترس است. Kasper و Schwabe سرعت را برای کُدهای ۶۴ بیت، بدون AES-NI، رکوردگیری کردند: نرخ حداکثر گزارش‌شده ۷،۶ و ۱۰،۷ سیکل بر بایت برای مُد CTR و GCM بود. با AES-NI، نسخه‌های پیشرفته‌ی OpenSSL نرخ 1.3cpb برای مُد CTR و 3.3cpb برای GCM را بدست آوردند. برای بدست آوردن این نتایج از تراشه‌ها و مکانیزم‌های زمان‌بندی مختلفِ x86 استفاده‌شده است. تنها پیاده‌سازی‌های غیر x86، با معماری‌های مشخص و نیز بدون مالکیت، برای مُد GCM و رمز قطعه‌ای AES که یافت شد، همه در OpenSSL هستند. پیاده‌سازی‌های OpenSSL برای مقایسه‌های غیرِ x86 انتخاب شدند [۱۱].

محیط‌های سخت‌افزار و نرم‌افزار: پنج معماری با مجموعه دستورالعمل قابل نمایش انتخاب شد: 32-bit x86, 64-bit x86, 32-bit ARM, 64-bit PowerPC, 64-bit SPARC . مجموعاً این معماری‌ها، ایستگاه‌های کاری، سِرورها و بازارهای محاسباتی قابل حمل را تصرف می‌کنند. پردازنده‌ی x86 که برای هر دو تست ۳۲ و ۶۴ بیتی استفاده‌شده است، یک "Intel Core i5-650 "Clarkdale است که از دستورالعمل‌های AES-NI حمایت می‌کند. پردازنده‌ی ARM یک Cortex-A8 است، PowerPC یک 970fx است، SPARC یک UltraSPARC IIIcu است. هر کدام سیستم‌عامل Debian Linux 6.0 با کرنل ۲،۶،۳۵ را با کامپایلر GCC 4.5.1 اجرا می‌کرد [۱۱].

روش‌های امتحان کردن: تعداد سیکل‌هایی از واحد پردازنده‌ی مرکزی که برای رمز کردن یک پیام نیاز است تقسیم بر طول پیام، هزینه‌ی بر بایت را برای رمزنگاری پیام‌ها در اختیار قرار می‌دهد. برای این‌که نتایج عملکرد

بوسیله‌ی زیرسیستم حافظه‌ای یک کامپیوتر میزبان بیش از اندازه تحت تاثیر قرار نگیرد، در تدابیری می‌بایست قبل از شروع زمان‌بندی همه کُدها و داده‌ها در سطحِ یکِ حافظه‌ی پنهان[1] قرار گیرند [۱۱].

نتایج: یافته‌های خلاصه شده، در مجموعه جداول ۶-۴ نشان داده شده‌اند. در همه‌ی معماری‌ها و نیز پیام‌های با طول متفاوت، OCB3 بطور قابل توجهی سریع‌تر از GCM و CCM است. به جز برای پیام‌های کوتاه که تقریباً به سرعت مُد CTR است. در x86، قابل رقابت‌ترین سکوی GCM، سربار اصالت پیام (هزینه‌ی اضافه‌بر رمزنگاری CTR) در حدود ۴-۱۶٪ است (این مقادیر چه با و چه بدون کُد AES-NI و نیز در هر دو مبنای طول پیامِ 4KB و اندیس کارایی اینترنت[2] بدست آمده‌اند). در همه‌ی تست‌ها CCM هرگز نرخ‌هایی با طول‌های IPI یا 4KB بهتر از GCM ندارد. بررسی دقیق‌تر این‌که فقط رجیسترهای کوچک ضرب‌های GCM را پُرهزینه می‌کنند یا دستورالعمل‌های AES-NI، رمزگذاری قطعه‌ای CCM را سرعت می‌بخشند. نتایج در بقیه‌ی معماری‌ها یکسان هستند [۱۱].

جدول ۶-۴عملکرد شیوه‌های AE [۱۱]

x86-64 Kasper-Schwab		
Mode	4K	IPI
CCM	22.4	26.7
GCM	10.9	15.2
OCB1	8.28	13.4
OCB2	8.55	13.6
OCB3	8.05	9.24
CTR	7.74	8.98

x86-64 AES-NI		
Mode	4K	IPI
CCM	4.17	4.57
GCM	3.73	4.53
OCB1	1.48	2.08
OCB2	1.80	2.41
OCB3	1.48	1.87
CTR	1.27	1.37

x86-32 AES-NI		
Mode	4K	IPI
CCM	4.18	4.70
GCM	3.88	4.79
OCB1	1.60	2.22
OCB2	1.79	2.42
OCB3	1.59	1.59
CTR	1.39	1.52

PowerPC 970		
Mode	4K	IPI
CCM	75.7	77.8
GCM	53.5	56.2
OCB1	38.2	41.0
OCB2	38.1	41.1
OCB3	37.5	39.6
CTR	37.5	37.8

Ultra-SPARC III		
Mode	4K	IPI
CCM	49.4	51.7
GCM	39.3	41.5
OCB1	25.5	27.7
OCB2	24.8	27.0
OCB3	25.0	26.5
CTR	24.1	24.4

ARM Cortex-A8		
Mode	4K	IPI
CCM	51.3	53.7
GCM	50.8	53.9
OCB1	29.3	31.5
OCB2	28.5	31.8
OCB3	28.9	30.9
CTR	25.4	25.9

هنگامی‌که به نرخ رمزنگاری پیام‌های 4KB یا IPI توجه می‌شود، سربار OCB3 از سربار GCM یا CCM از ۱۲ درصد بر روی PowerPC یا SPARC، یا ۱۸ درصد بر روی ARM تجاوز نمی‌کند. برای درک عملکرد بهتر OCB چهار فاز را برای رمزنگاری OCB3 در نظر بگیرید: تولید آفست اولیه، رمزنگاری بلوک‌های کامل یا

[1] level-1 cache
[2] IPI

رمزنگاری بلوک‌های غیرکامل (در صورت وجود تنها یک بلوک غیرکامل است) و تولید برچسب. در همه به غیر از کوتاه‌ترین پیام‌ها، پردازش بلوک‌های کامل بر هزینه کلّی بر بایت غلبه می‌کند. در اینجا OCB1،OCB3 به‌طور خاص کارآمد هستند. در یک پیاده‌سازی با ۴ بلوک در هر تکرار، سرباری روی چهار مرتبه‌ی خواندن‌ها، نوشتن‌ها و فراخوانی‌های رمز قطعه‌ای خواهد داشت: ۱۶ عمل XOR (هرکدام روی کلمه‌های ۱۶ بایتی)، یک محاسبه‌ی ntz و یک جدول جستجو با مقادیر ۱۶ بایتی. در x86 به ۲۳ سیکل یا 0.36cpb، هزینه نیاز دارند. در واقعیت، روی معماری x86 – 64-bit با استفاده از AES-NI مشاهده می‌شود که مُد شمارنده یا CTR روی پیام‌های 4KB، 1.27cpb زمان می‌گیرد، وقتی‌که OCB3، ۱،۴۸ سیکل بر بایت نیاز دارد (با یک سربار ۰،۲۱). هنگامی‌که پردازش بلوک کامل نیست، سه فاز دیگر رمزنگاری است که کارایی را مشخص می‌کند. این موضوع با هزینه‌ی رمزنگاری یک بایت در مراحل تولید آفست اولیه، رمزنگاری یک بلوک ناقص و تولید برچسب توجیه خواهد شد. مُد CCM در پیام‌های کوتاه فقط با OCB3 در رقابت است. بر x86 – 64-bit، بدون AES-NI و با استفاده از الگوریتم Kasper-Schwabe AES که هر هشت بلوک را در یک مرتبه پردازش می‌کند، کارایی OCB3 بسیار بیشتر است زیرا دو فراخوانی رمز بلوکی برخلاف CCM و GCM می‌تواند بصورت همروند اجرا شوند. با حمایت‌های سخت‌افزاری که AES ارزان‌تر شده است، سربار احراز اصالت چشم‌گیر است. دستورالعمل‌های AES-NI، ظرفیت پذیرش[1] AES-128 را ۲۰ سیکل بر بلوک ممکن می‌سازد. این سرعت‌ها می‌توانند سربار احراز اصالت را بسیار پرهزینه‌تر از خود رمزنگاری کنند. مثلاً با Kasper-Swchabe AES (بدون AES-NI) بر مبنای IPI، سربار OCB3 حدود ۳ درصد هزینه‌ی رمزنگار است و با AES-NI این هزینه به ۲۷ درصد افزایش می‌یابد. علاوه بر این، سربار GCM از ۴۱٪ به ۷۰٪ می‌رسد. از آن‌جایی که سربار بیشتر ناشی از فراخوانی‌های رمزقطعه‌ای است، ممکنست تصور شود CCM با استفاده از AES-NI بهتر عمل می‌کند، ولی استفاده از شیوه‌ی زنجیره‌سازی بلوک‌های رمز یا همان CBC (یکی از مُدهای سریال) در بخش احراز اصالت این مُد، ظرفیت پذیرش AES را به حدوداً ۶۰ سیکل بر بلوک کاهش می‌دهد که این موجب سربار احراز اصالت ۷۰ درصد خواهد شد. یک پردازنده با AES-NI یک محیط تقریباً ایده‌آلی برای OCB3 فراهم می‌کند: برای ذخیره‌ی مقادیر استفاده‌شده‌ی اخیر در حافظه‌ی پنهان، شانزده عدد رجیستر ۱۶ بایتی وجود دارند که عملیات یای‌انحصاری را انجام می‌دهند و همچنین واسطی برای فراخوانی‌های AES فراهم می‌کنند. هنگامی‌که از AES-NI روی x86 استفاده نشود، سربار بنابر فراخوانی تابع و استفاده از یک واسط مبتنی‌بر حافظه برای AES، به مقدار کمی افزایش می‌یابد. در x86 – 64-bit با استفاده از پیاده‌سازی

[1] Throughput

Kasper-Schwabe AES و مُد OCB3، ۰،۲۶ سیکل بر بایت بیشتر از مُد CTR هزینه‌بر است. برای معماری‌های PowerPC و SPARC این‌چنین نتایج یکسانی نیز بدست می‌آید. روش OCB3 سربار بیشتری روی ARM بنابر مجموعه رجیسترهای کوچکش دارد، امّا هنوز سربارش یک هفتمِ روش GCM است. همانطور که انتظار می‌رفت عملکرد پیام‌های طولانی در OCB3،OCB1 به خاطر پردازش قطعه‌های کامل (۱۲۸بیتی) و مشابه، شبیه‌به‌هم هستند. نسخه‌ی OCB2 در همه‌ی سکوهای امتحان‌شده به جُز SPARC (به دلیل محاسبه‌ی کند ntz در این معماری)، پیام‌های طولانی را کندتر پردازش می‌کند. شیوه‌ی OCB3 با یک عدد تصادفی مبتنی بر شمارنده، با استفاده از چند عمل شیفتِ مقدار ذخیره‌شده در حافظه‌ی پنهان، آفست اولیه رمزنگاری را تولید می‌کند و مانند OCB1، OCB2 دیگر از رمز قطعه‌ای برای محاسبه‌ی این آفست استفاده نمی‌کند. این موضوع کارایی میانگین را برای رمزنگاری پیام‌های کوتاه بالا می‌برد. تاثیر کلّی اینست که بر مبنای IPI و در سکوی ۶۴ بیتی x86، با استفاده از AES-NI سربار OCB3، ۶۵ درصدِ OCB1 و ۴۰ درصدِ OCB2 است. امّا وقتی‌که نانس تولیدی یک شمارنده نیست عملکرد OCB3 در اکثر محیط‌های آزمایشی از OCB1 غیرقابل تشخیص است [۱۱].

۶-۷- نتیجه‌گیری

ضمن ارایه‌ی روش‌های مختلف در جهت مقایسه‌ی کارایی و عملکرد الگوریتم‌ها رمزقطعه‌ای و شیوه‌های ارایه شده در این پروژه، به کمک این معیارها به بررسی آن‌ها پرداخته شد. مطلبی که قابل ذکر است نرم‌افزار متلب با وجود داشتن کتاب‌خانه‌های بسیار قوی و کامل، سرعت پایینی دارد و برخی برنامه‌ها در آن بسیار کند اجرا می‌شوند. از این نرم‌افزار صرفاً به دلیل پردازش تصویر و رمزنگاری تصویر استفاده شده است. روش OCB ،که یکی از شیوه-های پیاده‌سازی‌شده بود، هم برای حفظ محرمانگی و هم برای احراز اصالت فرستنده، با وجود سربار احراز اصالت در معماری‌های مختلف سرعت قابل قبولی و حتی طبق آزمایشات در برخی معماری‌ها نظیر x86_32-bit با داشتن سربار احراز اصالت کم سرعتی در حد شیوه‌ی شمارنده داشت. نتایجی که شخصاً در این پروژه بدست آوردم، در مورد سرعت الگوریتم‌های رمز قطعه‌ای بود که راین‌دال-۱۲۸ بیتی سرعتی به مراتب بیشتر و در عین حال گمراه-کنندگی و به‌هم ریختگی در حد الگوریتم‌های SERPENT و DES و 3-DES داشت. امّا روشی مثل DES با داشتن طول بلوک داده‌ی کوتاه‌تر و فضای کلید کوچک‌تر($۲^{۵۶}$) قابل‌نفوذ بوده است.

فصل هفتم

نتیجه‌گیری و پیشنهادات آتی

۷-۱- نتیجه‌گیری

هدف از انجام این پروژه پیاده‌سازی الگوریتم رمزقطعه‌ای متقارن با استفاده از شیوه‌های امن و سریع رمز قطعه-ای برای پیام مانند تصویر بود؛ لذا در فصل اول مختصری راجع به این پروژه و دلایل و ضرورت ایجاد امنیت در شبکه و راه‌های پیاده‌سازی امنیت صحبت شد. از طرفی به موضوع این پروژه و کمکی که می‌تواند در جهت ایجاد امنیت در یکی از لایه‌های شبکه انجام شود، صحبت شد.

در فصل دوم به تاریخچه‌ی چند هزار ساله‌ی رمزنگاری اشاره‌ای شد و مفاهیم رمزنگاری و انواع مختلف آن و راه‌های بالا بردن اطمینان در پیاده‌سازی این روش‌ها نیز توضیح داده شد. این روش‌ها انواع مختلفی دارند که فقط نمی‌توان با استفاده از یکی به فراهم کردن امنیت مطمئن شد بلکه باید اصول مختلفی فراهم شوند تا در جهت بستن راه نفوذ بر اخلال‌گر و شخص ثالث غیرمُجاز در طول ارسال پیام‌آشکار، گام برداشته شود.

تصاویر ساختار خاصی دارند و انواع تصاویر در متلب همگی با استفاده از اعداد صحیح ۸ بیتی یا به اصطلاح uint8 نمایش داده می‌شوند. لذا در فصل سوم مختصری در مورد آن‌ها صحبت شد. مثلاً تصاویر رنگی به‌صورت سه‌بعدی و با سه ماتریس جداگانه نمایش داده می‌شوند و هر پیکسل با ترکیبی از سه عدد از این ماتریس‌ها ساخته می‌شود ولی تصاویر خاکستری فقط از یک ماتریس اعداد در بازه‌ی ۲۵۵-۰ تشکیل می‌شوند. تصاویر باینری فقط از دو مقدار صفرو یک تشکیل می‌شود. این‌ها مطالب مختصری بودند که در فصل سوم به آن‌ها اشاره شد. همچنین اصول بنیادین رمزنگاری متقارن نظیر گمراه‌کنندگی و... در این فصل مطرح شدند.

همان‌طور که در فصل چهارم ذکر گردید برای داشتن سرعت بیشتر در رمزکردن پیام و نیز حفظ امنیت کافی، چنانچه به رمزنگاری متقارن نیاز بود، می‌توان از الگوریتم استاندارد رمزنگاری پیشرفته با هر طول کلیدی از بین ۱۲۸،

۱۹۲، ۲۶۵ بیتی استفاده کرد. این نتیجه نیز با استفاده از نتایج حاصل از رمزنگاری و رمزگشایی تصویر و نیز مقادیر بدست آمده از راه‌های مختلف مقایسه در این پروژه بدست آمد. البته الگوریتم سرپنت نیز به دلیل داشتن دورهای بسیار زیاد در رمزنگاری پیام روش بسیار امنی به شمار می‌رود، ولی همین تعداد دورهای پردازش باعث کندشدن عملکرد الگوریتم نیز می‌شوند. الگوریتم رمز قطعه‌ای ۶۴ بیتی با کلید ۵۶ بیتی DES نیز در این پروژه بررسی و پیاده-سازی شد. این الگوریتم‌ها برای ۱۲۸ بیت یا ۶۴ بیت طراحی شده بودند و برای تصویری با تعداد بیت‌های بیشتر باید از شیوه‌های مختلف رمزنگاری استفاده شود که البته این شیوه‌ها نیز هر یک در جای خود مناسب هستند و نمی‌توان از مزایای آن‌ها غافل شد. در این پروژه با برخی از این شیوه‌های معروف و پیاده‌سازی‌های آن‌ها آشنا و سپس شیوه‌هایی جدید که در آن‌ها علاوه بر رمزنگاری پیام می‌توان از صحت پیام اطمینان حاصل کرد معرفی شدند. در این شیوه ها نیز باز هم سرعت مطرح بود و در انتهای فصل نیز سعی در ارایه‌ی شیوه‌ای شد که در سه نسخه طراحان آن سعی در بهبود روش خود داشتند و در آخر طبق بررسی‌ها و نتایج حاصل از مقایسه‌ای که با معماری‌ها و پلت‌فرم‌های متفاوت و نیز با پیام‌هایی با طول مختلف انجام شد، OCB3 نسبت به CCM و GCM سریع‌تر بود.

۷-۲- پیشنهادات آتی

برای ادامه‌ی این پروژه پیشنهادات موجود اینست که دانشجویان به دنبال روش‌هایی باشند که بتوان علاوه بر فراهم‌کردن سرویس‌های محرمانگی و احراز اصالت، سرویس عدم انکار را نیز فراهم کند، تا دیگر فرستنده نتواند پیام ارسالی خود را انکار کند، زیرا این‌گونه شیوه‌ها در کنار یکدیگر می‌توانند امنیت را به صورت نسبی فراهم آورند. همچنان رمزنگاران و طراحان مختلفی به دنبال ارایه‌ی شیوه‌های گوناگون رمز قطعه‌ای و ثبت آن‌ها در NIST هستند و این می‌تواند زمینه‌ای برای تحقیقات آتی باشد. پیشنهاد دیگر این‌که، چنان‌چه دانشجویان علاقه‌مند به فعالیت در زمینه‌ی رمزنگاری تصویر باشند بهتر است در مورد رمزنگاری بصری که در دهه ۱۹۹۰ معرفی شد و حتی می‌توانند از استادانی که در دانشگاه‌های ایران در این زمینه مشغول فعالیت هستند، اطلاعاتی کسب کنند. حتی می-توان با استفاده از رمزنگاری بصری و سپس روش‌های رمزنگاری متقارن ذکر شده در این پروژه، با شیوه‌ای امن یک امنیت دولایه‌ای را بر روی تصویر ایجاد کرد.

مراجع

مراجع فارسی

[۱] ذاکر الحسینی، علی؛ ملکیان، احسان. «امنیت داده‌ها». انتشارات نص، چاپ دوم: زمستان ۸۷.

[۲] حیدری، عبدالرحمن. «پردازش تصویر در MATLAB». انتشارات به‌آوران-کلک زرین، پاییز۸۸.

[۳] فن‌آوری اطلاعات. روشهای امنیتی. حالتهای عملیاتی یک الگوریتم رمزنگاری قطعه‌ای n بیتی. مؤسسه استاندارد و تحقیقات صنعتی ایران. کمیسیون فنی تدوین استاندارد. 9600 چاپ اول. Last visited @June 2012: http://std.isiri.org/std/9600.PDF

مراجع انگلیسی

[4] http://www.wikipedia.org/ last visited @June 2012

[5] R.C. Gonzalez, R.E. Woods, S.L. Eddins, "Digital Image Processing Using MATLAB", New Jersey, Prentice Hall, 2009.

[6] W. Stallings, "Cryptography and Network Security", Principles and Practices, Fourth Edition, Prentice Hall, 2005.

[7] R. Anderson, E. Biham, L. Knudsen, "Serpent: A Proposal for the Advanced Encryption Standard" .Available: http://www.cl.cam.ac.uk/~rja14/serpent.html

[8] B. Schneier, "Applied Cryptography)", John Wiley & Sons, Inc., 1996.

[9] P. Rogaway, University of California, UC Davis, 2011. Available: http://www.cs.ucdavis.edu/~rogaway/ocb/ocb-faq.htm

[10] M. Dworkin, "Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality", NIST Special Publication 800-38C, 2004(updated-July20_2007).

[11] T. Krovetz and P. Rogaway."The Software Performance of Authenticated-Encryption Modes", Fast Software Encryption 2011 (FSE 2011). Availabe: http://www.cs.ucdavis.edu/~rogaway/papers/ae.pdf

[12] D. Whiting and R. Housley and N. Ferguson, "AES Encryption & Authentication Using CTR Mode & CBC-MAC", IEEE P802.11 doc 02/001r2، May 2002.

[13] L. YAN and R. YE. "Image Encryption Using Novel Mappings over GF (2^n)", Studies in Mathematical Sciences, Vol. 2, No. 1, 2011.

[14] C. Jutla, "Encryption modes with almost free message integrity", EUROCRYPT 2001, LNCS, vol. 2045, Springer, pp. 529-544, 2001.

[15] J. Daemen, P. Rijmen, "AES Proposal: Rijndael", 2001

[16] T. Krovetz, P. Rogaway. The OCB Authenticated-Encryption Algorithm, draft-krovetz-ocb-03(Internet Engineering Task Force) January, 2012.

[17] http://stackoverflow.com/ last visited @ June 2012

[18] http://www.mathworks.com/ last visited @ June 2012

[19] http://www.intel.com last visited @ June 2012

پیوست ۱

- در الگوریتم رمزنگاری AES-NI، که در فصل ششم از آن به عنوان یکی از الگوریتم‌ها برای سرعت بخشیدن به اجرای دستورالعمل‌ها استفاده شده‌بود، برای افزایش سرعت بخش‌های مختلف الگوریتم استاندارد پیشرفته‌ی رمزنگاری شامل ۷ دستورالعمل است. چهار دستورالعمل برای اجرای اولین دور و آخرین دور از ۱۰ دور رمزنگاری پیام‌های ۱۲۸ بیتی را از متن آشکار به متن رمزشده تبدیل می‌کند و برعکس. یک دستورالعمل برای عمل تلفیق ستون و یکی برای تولید کلید دور مرحله‌ی بعد. هفتمین دستورالعمل، CLMUL، عمل ضرب بدون رقم نقلی را در سخت‌افزار انجام می‌دهد. مزیت این دستورات اینست که حمله‌های نرم‌افزاری سمت کانال و سربار عملیاتی را کاهش می‌دهد [۱۹].
- شاخص عملکرد اینترنت یا IPI که در بررسی عملکرد شیوه‌های مختلف رمزنگاری قطعه‌ای استفاده شد، مقدار متوسطی از پیام‌هایی با طول مختلف است که هر کدام توزیعی به صورت زیر دارند و تابعی به عنوان f(s) میزان کسر مورد انتظار که این سایز از بسته‌ها را حمل می‌کنند، مشخص می‌کند. طبق تعریف ارایه‌شده در مقاله‌ی Claffy, Miller Thompson (1998) این توزیع به صورت زیر است:

$$f(1500)=0.6,\ f(576)=0.2,\ f(552)=0.15,\ f(44)=0.05$$

این مقادیر طبق مطالعاتی از ستون‌فقرات اینترنت بدست‌آمده است و به‌عنوان معیاری از بسته‌های که طولشان ضریبی از ۱۶ بایت (۱۲۸ بیت) نیست، استفاده شده‌اند. پس IPI مقدار بیت‌های مورد انتظار است که در هر سیکل ساعت برای این نوع از توزیع بسته‌ها پردازش می‌شوند [۱۱].

پیوست ۲

الگوریتم رمزنگاری DES

```
function CipherText=DES_Encryption(P,RoundKey)
% Converting P to 128-bit Binary
P_binry=reshape(reshape(dec2bin(reshape( reshape(P,2,4)',1,8),8),8,8)',1,64);

% InitialPermutation & FinalPermutation is for
% Optimization & symmetry of the network
% By changing the places of 128 bits
P_IP=InitialPermutation_DES(P_binry);
L=P_IP(1:32);% Left part of the PlainText after Permutaion
R=P_IP(33:64);% Right part of the PlainText after Permutaion
for i=1:16  % Run the encryption rounds
    T=num2str(mod(L+F(R,RoundKey(i,:)),2)); % R(n) <=L(n-1) + f(R(n-1),K(n))
    k=1;
    for j=1:size(T,2)
        if T(1,j)~=' '
            T1(1,k)=T(1,j);
            k=k+1;
        end
    end
    Mi=[R,L];   % Swap the parts
    L=Mi(1:32); %  L(n) <= R(n-1)
    R=T1;
end
M=[R,L];    % Swap and concatenate the two sides into R16L16
CipherText_bin=FinalPermutation_DES(M);% Final Permutaion
CipherText=reshape(bin2dec(reshape(CipherText_bin,8,8)'),4,2)'; %#Converting 64-bit binary
to 2*4 Matrix
```

الگوریتم توسیع کلید در DES

```
function K=KeyExpansion_DES(key)
%   producing 48-bit roundkeys from the 56-bit master key
%   permutation choise 1 = PC1 = 1*56
key_binry=reshape(reshape(dec2bin(reshape( reshape(key,2,4)',1,8),8),8,8)',1,64);
key_PC1=key_binry(1,[57,49,41,33,25,17,9,1,...
    58,50,42,34,26,18,10,2,...
    59,51,43,35,27,19,11,3,...
    60,52,44,36,63,55,47,39,...
    31,23,15,7,62,54,46,38,...
    30,22,14,6,61,53,45,37,...
    29,21,13,5,28,20,12,4]);
%   permutation choise 2 = PC2 = 1*48
PC2=[14,17,11,24,1,5,3,28,...
    15,6,21,10,23,19,12,4,...
    26,8,16,7,27,20,13,2,...
    41,52,31,37,47,55,30,40,...
    51,45,33,48,44,49,39,56,...
    34,53,46,42,50,36,29,32];
L=key_PC1(1:28); % select left 28 bits
R=key_PC1(29:56); % select right 28 bits
% This is the schedule of shifts.
% Each LnRn is produced by shifting the previous by 1 or 2
% The Round Number [1   2   3   4   5   6   7   8   9  10  11   12  13 14  15  16]
% Number Of shifts [1 , 1 , 2 , 2 , 2 , 2 , 2 , 2 , 1 , 2 , 2 , 2 , 2 , 2 , 2 , 1]
for i=1:16
    if i==1||1==2||i==9||i==16
```

```
        L=[L(2:28),L(1)];
        R=[R(2:28),R(1)];
    else
        L=[L(3:28),L(1:2)];
        R=[R(3:28),R(1:2)];
    end
    Ki=[L,R];
    K(i,:)=Ki(PC2);
end
```

الگوریتم رمزنگاری AES

```
function CipherText=AES_Encryption(PlainText,RoundKey,S_Box)

%Copy PlainText into the State Variable
State=PlainText;

%First XOR with the State Variable
State=Xor_Roundkey_inTo_State(State,RoundKey,0);
for r=1:10
    for j=1:4 %Substitute each Byte with the Substitution Table
        State(j,:)=Sub_Word(State(j,:),S_Box);
    end
    State=Rotate_Rows(State);    %Rotating the Rows according to thier row numbers
    if r<10              % Mix the Cells (((the most important part of the Algorithm)))
        State=Mix_Columns(State);
    end
    State=Xor_Roundkey_inTo_State(State,RoundKey,r); %Xor the State Variable with the
round key
end
CipherText=State; % CipherText is Produced
```

الگوریتم رمزنگاری سرپنت

```
function CipherText=Serpent_Encryption(P,RoundKey,S_Box)
%   32-round substitution permutation network (SPN)
% Converting P to 128-bit Binary
P_binry=reshape(reshape(dec2bin(reshape( reshape(P,4,4)',1,16),8),16,8)',1,128);
%InitialPermutation & FinalPermutation is for Optimization & symmetry of the network
By %changing the places of 128 bits
P_IP=InitialPermutation(P_binry);

% Reshaping to 4*4 decimal matrix
PLinTrans=reshape(bin2dec(reshape(P_IP,8,16)'),4,4)';
vector(1:32)=4;
for r=1:1:32
    P_AfterXor = bitxor(PLinTrans,RoundKey{r}); %   XORing data with Current RoundKey
    % Reshaping  4*4 decimal matrix to 128 bit binary one
    P_AfterXor_Binry=reshape(reshape(dec2bin(reshape(reshape(P_AfterXor,4,4)',1,16),8)
,16,8)' ,1,128);
    P_Block = mat2cell(P_AfterXor_Binry,1,vector);  % Breaking into thirty-two 4-bit
Blocks
    if r < 8
        K=r;
        A = S_Box(K,:);   % Choosing one of the 8 S_Boxes
    else
        K = mod(r,8);
        A = S_Box(K+1,:);   % Choosing one of the 8 S_Boxes
    end

    for j=1:32
        Y=A(bin2dec(P_Block{j})+1); % Substituting values of Desired S_BOX for every 4
bits
```

```
            P_Cell{j}=dec2bin(Y,4);
        end
        P_Mat=cell2mat(P_Cell);
        % Convert the contents which is extracted from S_BOX into a single matrix
        if  r == 32 %   The last round doesn't execute the LinearTransformation
            break;
        else
            %   Every 31 round executes the LinearTransformation Function
            PLinTrans = reshape(bin2dec(reshape(LinearTransformation(P_Mat),8,16)'),4,4)';
            %   The Output OF the Function should be converted
            %   to 4*4 decimal matrix
        end
    end
    P_AfterXor=bitxor(reshape(bin2dec(reshape(P_Mat,8,16)'),4,4)',RoundKey{33});

    % Every changing in InitialPermutation has to return
    % To its primary place By FinalPermutation function
    P_Decrypted=FinalPermutation(reshape(reshape(dec2bin(reshape( reshape(P_AfterXor,4,4)',
1, 16) ,8) ,16,8)' ,1,128));
    CipherText=reshape(bin2dec(reshape(P_Decrypted,8,16)'),4,4)';  % Converting to 4*4
Matrix
```

ادامه‌ی توسیع کلید الگوریتم سِرپِنت

```
    while 1
        %   The Sequence of using S_Boxs is:  S_Box(4,:), S_Box(3,:),
        %   S_Box(2,:), S_Box(1,:), S_Box(8,:), S_Box(7,:), S_Box(6,:), S_Box(5,:)
        %   Then Concating every four 32-bit cell in a 128-bit single matrix
        %   & then breaking that matrix to thirty two 4-bit cells in order
        %   To substitute values in one of eight S_Boxes for every four-bit cell
        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
            komakii=S_Box( 4 , bin2dec( WW{s} )+1 );    %S_Box(4,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        %   Producing one of thirty three 128-bit RoundKeys
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell),8,16)'),4,4)';

        if j == 33  %   we just want to produce 33 RoundKeys
            break;
        end
        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
            komakii=S_Box( 3 , bin2dec( WW{s} )+1 );  %   S_Box(3,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell),8,16)'),4,4)';


        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
            komakii=S_Box( 2 , bin2dec( WW{s} )+1 );  %   S_Box(2,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell),8,16)'),4,4)';

        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
            komakii=S_Box( 1 , bin2dec( WW{s} )+1 );  %   S_Box(1,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell),8,16)'),4,4)';

        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
```

```
            komakii=S_Box( 8 , bin2dec( WW{s} )+1 );  %   S_Box(8,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell),8,16)'),4,4)';

        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
            komakii=S_Box( 7 , bin2dec( WW{s} )+1 );  %   S_Box(7,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell)،8،16)')،4،4)';

        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
            komakii=S_Box( 6 , bin2dec( WW{s} )+1 );  %   S_Box(6,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell),8,16)'),4,4)';

        WW=mat2cell(cell2mat(Word(4j+1:4j+4)),1,vector2);
        for s=1:32
            komakii=S_Box( 5 , bin2dec( WW{s} )+1 );  %   S_Box(5,:)
            key_cell{s}=dec2bin(komakii,4);
        end
        j=j+1;RoundKey{j}= reshape(bin2dec(reshape(cell2mat(key_cell),8,16)'),4,4)';
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

شیوه‌ی رمزنگاری OCB برای تصاویر خاکستری

```
function CipherText=OCB_ENCRYPT(key,Im,A)
% this program is for 128-bit blocks
tStart = tic;
disp('Which BlockCipher Algorithm do u want? AES/Serpent');
reply = input(' if AES press A else press S? A/S [A]: ', 's');
if reply == 'A' || reply == 'a'
    [S_Box   Inv_S_Box   RoundKey]=AES_Initializing(key);
    fun1=@AES_Encryption;
elseif reply == 'S' || reply == 's'
    [S_Box  Inv_S_Box    RoundKey]=Serpent_Initializing(key);
    fun1=@Serpent_Encryption;
end
n = 255;
IV(1:3,1:4)= ceil(n.*rand(3,4));
SizeOfOriginalPic=size(Im);
% Storing size of I,age in IV
IV(1,1)=SizeOfOriginalPic(1,1);
IV(1,2)=SizeOfOriginalPic(1,2);
N=double(IV);
%
% Nonce-dependent and per-encryption variables
%
Top(1,1:4)= [0 0 0 1];
Top(2:4,1:4) =N;
Noc=dec2bin(Top(4,4),8);
bottom = bin2dec(Noc(3:8));    % A 6-bit value Between 0~63
Top(4,4)=double(bin2dec(strcat(Noc(1:2),'000000'))); % an 8-bit value
Ktop = fun1( Top ,RoundKey,S_Box);  % Encrypting
Ktop_bin=reshape(reshape(dec2bin(reshape( reshape(Ktop,4,4)', 1, 16) ,8) ,16,8)'
,1,128);% Converting Ktop to 128-bit Binary
%   Stretch = Ktop || (Ktop?(Ktop<<8)): this is what is done in 4 below lines
%   Shifting every cell of the 4*4 matrix to left
%   0 is inserted into the last cell
Ktop_Shifted=ShiftLeft(Ktop,0);
Stretch = Ktop_bin ;
Stretch2=bitxor(Ktop(1:2,1:4) , Ktop_Shifted(1:2,1:4));    % 256-bit String
```

```
     Stretch(129:192)=reshape(reshape(dec2bin(reshape( reshape(Stretch2,2,4)', 1, 8) ,8)
,8,8)' ,1,64);
     % First Offset
     Offset = reshape(bin2dec(reshape(Stretch(1+bottom:128+bottom),8,16)'),4,4)';
     % The first 128 bits of Stretch << Bottom{1,1}
     Offset_Copy=Offset;
     L0=Doubl(fun1(zeros(4,4),RoundKey,S_Box));
     L{1,1}=Doubl(L0);
     I=resize(Im);
     SizeOfOriginalPic2=size(I);
     Rows=SizeOfOriginalPic2(1,1);    Rows=Rows/4; Vector1(1:Rows)=4;
     Cols=SizeOfOriginalPic2(1,2);    Cols=Cols/4; Vector2(1:Cols)=4;
     I= double(I);    %   The Image should be converted to double
     Cell=mat2cell(I,Vector1,Vector2);   %   Image is breaked to 4*4 Cells
     Checksum = zeros(4,4);
     %%%% keySetup %%%
            for i=1:Rows
                for j=1:Cols
                    if j~=1 % i~=1///i==1
                        L{i,j}=Doubl(L{i,j-1}); % Doubl is a shift and XOR if needed
                    elseif i~=1 && j==1
                        L{i,j}=Doubl(L{i-1,Cols});
                    end
                end
             end
             %
             % Process blocks
             %
             for i=1:Rows
                  for j=1:Cols
                      if j~=1 % i~=1///i==1
                          Offset = bitxor(Offset , L{ntz(i)+1,ntz(j)+1});
                         Checksum = bitxor(Checksum, Cell{i,j});
                     elseif i~=1 && j==1
                          Offset   = bitxor(Offset , L{ntz(i)+1,ntz(j)+1});
                         Checksum = bitxor(Checksum, Cell{i,j});
                      end
                     C{i,j} =
double(bitxor(Offset,fun1(bitxor(Cell{i,j},Offset),RoundKey,S_Box) ));
                  end
             end
     % a 128 bit tag
     Tag = double(bitxor(fun1( bitxor(bitxor(Checksum,Offset),L0),RoundKey,S_Box),
Hash(double(A),key,reply)));
     Img_Encrypted=cell2mat(C);
     %
     % Assemble ciphertext
     %
CipherText=C;
Cipher_Size=size(CipherText);
CipherText{Cipher_Size(1,1)+1,Cipher_Size(1,2)} =Tag;
name = getenv('COMPUTERNAME');
CipherText{Cipher_Size(1,1),Cipher_Size(1,2)+1} =name;
clk=fix(clock);
clk_str=strcat(int2str(clk(1,4)),': ',int2str(clk(1,5)),'min, ',int2str(clk(1,6)),'sec');
CipherText{Cipher_Size(1,1)+1,Cipher_Size(1,2)+1} =clk_str;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function Auth=Hash(A,key,reply)
if reply == 'A' || reply == 'a'
         [S_Box   Inv_S_Box   RoundKey]=AES_Initializing(key);
         fun1=@AES_Encryption;
elseif reply == 'S' || reply == 's'
         [S_Box  Inv_S_Box    RoundKey]=Serpent_Initializing(key);
         fun1=@Serpent_Encryption;
end
A=resize(A);
%
% Consider A as a sequence of 128-bit blocks
```

```
%
SizeOfOriginalPic2=size(A);
Rows=SizeOfOriginalPic2(1,1);    Rows=Rows/4;        Vector1(1:Rows)=4;
Cols=SizeOfOriginalPic2(1,2);    Cols=Cols/4;        Vector2(1:Cols)=4;
A=double(A);    %    The Image should be converted to double
A=mat2cell(A,Vector1,Vector2);
%
% Process blocks
%
Sum = zeros(4,4);
Offset = zeros(4,4);   %   the initial value is  Init = 0
%
% Key-dependent variables
%
L=Doubl(fun1(zeros(4,4),RoundKey,S_Box));
L_Key_dependent{1,1}=Doubl(L);
for i=1:Rows
   for j=1:Cols
      if j~=1 % i~=1///i==1
        L_Key_dependent{i,j}=Doubl(L_Key_dependent{i,j-1});
      elseif i~=1 && j==1
        L_Key_dependent{i,j}=Doubl(L_Key_dependent{i-1,Cols});
      end
   end
end
for i=1:Rows
    for j=1:Cols
        Offset = bitxor(Offset, L_Key_dependent{ntz(i)+1,ntz(j)+1});
       Sum = bitxor(Sum,fun1(bitxor(A{i,j} ,Offset),RoundKey,S_Box));
    end
end
Auth=Sum;
```

تابع ntz به صورت زیر محاسبه می‌شود:

```
function n = ntz (x)
%   % number of trailing zeros (ntz), which counts the number of zero bits
%   % following the least significant one bit i.e ntz(4)=2, ntz(1)=ntz(3)=0
    if x == 0
        n = 8;
    elseif bitand(x,hex2dec('7f')) == 0
        n = 7;
    elseif bitand(x,hex2dec('3f')) == 0
        n = 6;
    elseif bitand(x,hex2dec('1f')) == 0
        n = 5;
    elseif bitand(x,hex2dec('0f')) == 0
        n = 4;
    elseif bitand(x,hex2dec('07')) == 0
        n = 3;
    elseif bitand(x,hex2dec('03')) == 0
        n = 2;
    elseif bitand(x,hex2dec('01')) == 0
        n = 1;
    end
```