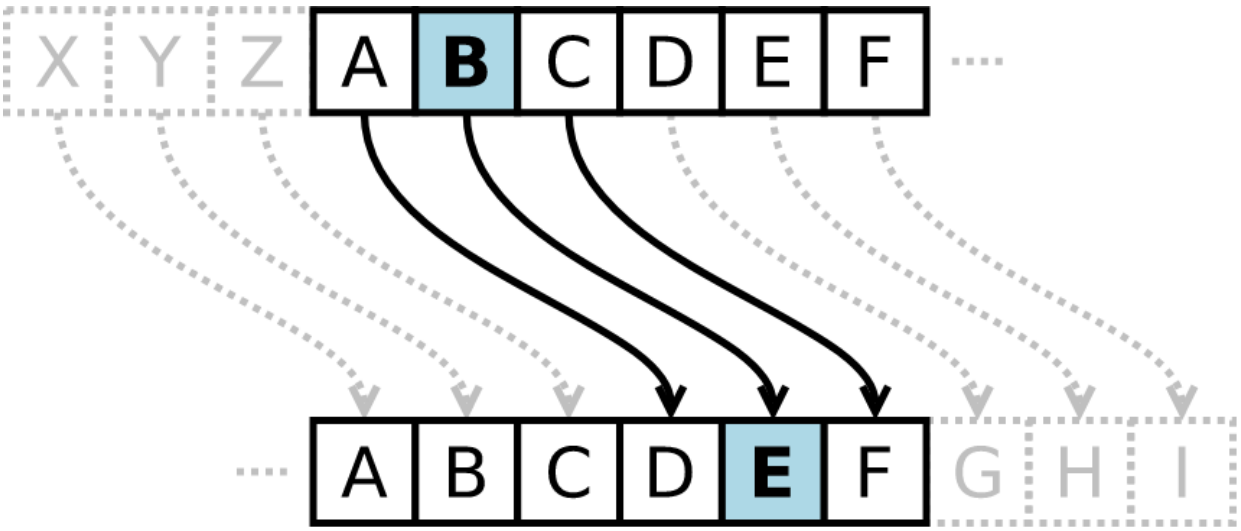
به نام خدا

محمدرسول ضیاءالدینی ۹۶۱۴۹۰۴۶ پروژه درس رایانش امن

رمز اول : سزار

این رمز یک نوع رمز جانشینی است که در آن هر حرف در متن آشکار با حرف دیگری با فاصله ثابت در الفبا جایگزین می‌شود. برای مثال با مقدار انتقال ۳، D به جای A می‌نشیند، E به جای B، و الی آخر. نام این روش از ژولیوس سزار گرفته شده‌است که از آن برای ارتباطات محرمانه خود استفاده می‌کرد.



رمز سزار مانند تمام رمزهای جانشینی تک‌الفبایی دیگر به راحتی شکسته می‌شود و با وجود تکنیک‌های مدرن، هیچ‌گونه امنیتی برای ارتباطات فراهم نمی‌کند.

تبدیل الفبای آشکار به الفبای رمز را می‌توان با هم‌ردیف کردن دو الفبا نمایش داد. الفبای رمز درواقع همان الفبای آشکار است که به میزان مشخصی به سمت راست یا چپ چرخانده شده. برای مثال، رمز سزار با چرخش به چپ میزان انتقال ۳ در جعبهٔ پایین نمایش داده شده. کلید رمز همان مقدار جابجایی است که در این مثال برابر با ۳ انتخاب شده.

آشکار: ABCDEFGHIJKLMNOPQRSTUVWXYZ

رمز: DEFGHIJKLMNOPQRSTUVWXYZABC

شکستن رمز (ارجاع لینک‌ها به ویکی پدیا)

رمز سزار حتی در شرایط حمله متن اصلی به راحتی قابل شکسته شدن است. دو موقعیت زیر می‌توانند در نظر گرفته شوند:

۱. حمله‌کننده می‌داند یا حدس می‌زند که نوعی از رمز جانشینی ساده استفاده شده‌است اما مشخصا نمی‌داند که رمز سزار است.
۲. حمله‌کننده می‌داند که رمز سزار استفاده شده‌است اما مقدار انتقال را نمی‌داند.

در حالت اول استفاده از تکنیک‌های معمول شکستن رمزهای جانشینی مانند تحلیل فراوانی به سادگی نتیجه‌بخش است. در حین استفاده از این تکنیک‌ها، حمله‌کننده به راحتی متوجه نظم موجود در سیستم جانشینی و استفاده از رمز سزار خواهد شد.

شکستن رمز در حالت دوم ساده‌تر از حالت اول است. از آنجا که تعداد ممکن انتقال‌ها محدود است (در زبان انگلیسی ۲۶ حالت ممکن) اعمال حمله جستجوی فراگیر و آزمایش تمام حالات ممکن به سرعت انجام می‌شود[۲۱]. برای مثال همان‌طور که در جدول نشان داده شده، بخشی از متن به همراه تمام انتقال‌های ممکن نوشته می‌شود و ردیف حاوی متن بامعنی به راحتی قابل تشخیص است. در این روش کافی است که زیر هر حرف از متن رمز شده، تمام حروف الفبا به ترتیب نوشته شود. در مثال جدول روبرو متن رمز شده EXXEGOEXSRGI است و به سادگی می‌توان تشخیص داد که کلید رمز استفاده شده برابر با ۴ بوده.

روش دیگر حمله جستجوی فراگیر با کمک تحلیل فراوانی است. در این روش با مقایسه فراوانی حروف در متن رمز و فراوانی حروف در متون عادی زبان مورد استفاده و جابجایی دو نمودار می‌توان میزان انتقال را پیدا کرد. برای مثال در زبان انگلیسی E و Tپراستفاده‌ترین و حروف Q و Z کم‌استفاده‌ترین حروف هستند. این روش توسط کامپیوتر هم قابل پیاده‌سازی است. برای این کار کافی است با استفاده از آزمون مربع کای، توزیع داده شده با توزیع مورد انتظار مقایسه شود.

معمولاً فقط یک متن آشکار محتمل برای یک متن رمز وجود دارد، اما برای رمزهای بسیار کوتاه ممکن است تعداد پاسخ‌های محتمل بیشتر از یکی باشد. برای مثال متن رمز MPQY می‌تواند به aden یا know برگردد. به‌طور مشابه ALIIP می‌تواند dollsیا wheel باشد و AFCCP را می‌توان به jolly یا cheer رمزگشایی کرد. به حداقل طول متن رمز شده که لازم است تا متن اصلی به صورت یکتا قابل شناسایی باشد فاصله یکتایی گفته می‌شود.

استفاده چندباره از رمز سزار بر روی یک متن به امنیت بیشتر منجر نمی‌شود. زیرا دو بار رمزگذاری با انتقال‌های A و B معادل یک بار رمزگذاری با کلید A+B است. به زبان ریاضی می‌توان گفت مجموعه رمز سزار با کلیدهای متنوع، تحت ترکیب یک گروه تشکیل می‌دهند.

رمز دوم: ورنام

پایه اصلی رمز ورنام عملیات XOR است که بر روی تک تک بیت های مورد استفاده برای کدگذاری کاراکترها در کد باودوت اعمال می شود. ورنام از اصطلاح "XOR" در ثبت اختراع استفاده نکرد، اما او آن عملیات را در منطق رله پیاده سازی کرد. در مثالی که ورنام داد، متن ساده A است که در باودوت به صورت «++---+» کدگذاری می شود و کاراکتر کلیدی B است که به صورت «+--++» کدگذاری می شود. متن رمز حاصل "-+++-" خواهد بود که یک G را کدگذاری می کند. ترکیب G با کاراکتر کلیدی B در انتهای دریافت، "++---+" تولید می کند که متن ساده اصلی A است. آژانس امنیت ملی این پتنت را «شاید یکی از مهم ترین ها در تاریخ رمزنگاری» نامید.

در اصطلاحات مدرن ، رمز Vernam یک رمز عبور متقارن است که در آن متن ساده با یک جریان تصادفی یا شبه داده ای (همان "جریان اصلی") با همان طول ، ترکیب می شود تا متن رمز را با استفاده از عملکرد (XOR) تولید کند. این نماد با ⊕ نشان داده می شود و با جدول زیر نشان داده می شود ، جایی که + نشان دهنده "درست" است و – نشان دهنده "نادرست" است.

INPUT		OUTPUT
A	***B***	***A*** ⊕ ***B***
–	–	–
–	+	+
+	–	+
+	+	–

رمزنگاری از این نظر متقابل است که جریان اصلی یکسان برای رمزگذاری متن ساده به متن رمزدار و رمزگشایی متن رمزنگاری برای بدست آوردن متن ساده اصلی استفاده می شود :

متن ساده ⊕ کلید = متن رمز

و

متن رمز ⊕ کلید = متن ساده

اگر جریان کلید واقعاً تصادفی است و فقط یک بار استفاده می شود ، در واقع یک پد یکبار مصرف است. جایگزینی داده های شبه تصادفی تولید شده توسط یک مولد اعداد شبه تصادفی با رمزنگاری ، ساختاری رایج و موثر برای رمزنگاری جریان است RC۴ نمونه ای از رمزنگاری ورنام است که به طور گسترده در اینترنت استفاده می شود.

اگر ، برای دو پیام از جریان اصلی یکسان استفاده شود ، که تحلیلگران به عنوان عمق آن را می شناسند ، می توان اثر جریان اصلی را از بین برد و دو متن XOR را با هم جمع کرد. نتیجه معادل رمزگذاری کلید در حال اجرا است و ممکن است دو روش متن با تکنیک های رمزنگاری زبانی از هم جدا شوند.

Ciphertext ۱ ⊕ Ciphertext ۲ = Plaintext۱ ⊕ Plaintext۲

رمز سوم: playfair

یک روش رمزنگاری متقارن است و اولین رمزنگاری جانشینی دیاگرام بوده و طرح آن اولین بار در سال ۱۸۵۴ توسط چارلز ویت‌استون اختراع شده است.

این روش جفت حروف (دیاگرام یا بیگرام) را به جای حروف در رمزنگاری جانشینی و نه سیستم‌های رمزنگاری ویژنر رمزنگاری می‌کند. شکستن رمز پلیفیر سخت‌تر است زیرا تحلیل فرکانسی که برای رمزهای جانشینی ساده به کار می‌رود، در آن کارایی ندارد. می‌توان بیگرام‌ها را به صورت فرکانسی تحلیل کرد، ولی خیلی سخت‌تر است. با ۶۰۰ بیگرام احتمالی به جای ۲۶ مونوگرام احتمالی (تک علامت‌ها، در این حوزه معمولا همان حروف الفبا است) به متن رمز بزرگتری نیاز است.

رمز پلیفیر از یک جدول ۵ در ۵ استفاده می‌کند که شامل عبارت یا واژه کلید است. به خاطرسپاری کلیدواژه و ۴ قاعده کل چیزی است که برای ایجاد یک جدول ۵ در ۵ و استفاده از رمز لازم است.

برای تولید جدول کلید، می‌توان اول فضاهای جدول را با حروف کلیدواژه پر کرد و سپس فضاهای باقیمانده را با حرف‌های دیگر الفبا به ترتیب) معمولا با حذف «J» یا «Q» برای کاهش حرف الفبا به منظور جا شدن در ۲۶ حروف الفبا در جدول (پر کرد. کلید می‌تواند در ردیف‌های بالای جدول از چپ به راست یا در الگوهای دیگر مانند شروع مارپیچی از گوشه‌ی بالا چپ و پایان در مرکز نوشته شود کلیدواژه به همراه قراردادها برای پرکردن جدول ۵ در ۵ کلید رمز را تشکیل می‌دهند.

برای رمزنگاری یک پیام، می‌توان پیام را به دیاگرام (گروه‌های دو حرفی) تقسیم کرد به طوری که مثلا «Hello World» به HE » «LL OW OR LDتبدیل می‌شود. این دیاگرام‌ها با استفاده از جدول کلید جایگزین می‌شوند. چون رمزنگاری از جفت حروف استفاده می‌کند، به پیام‌هایی با تعداد حرف فرد معمولا یک حرف غیر رایج مانند «X» اضافه میشوند تا دیاگرام نهایی را کامل کنند. دو حرف از دیاگرام در گوشه‌های مقابل هم در یک مستطیل در جدول کلید قرار می‌گیرند. برای انجام جانشینی، قاعده‌های زیر را بر حروف در یک متن ساده اعمال کنید:

اگر هر دو حرف شبیه هم بودند (یا تنها یک حرف باقی مانده)، یک X را پس از حرف اول اضافه کنید. جفت جدید را رمزگذاری کرده و ادامه دهید. بعضی از انواع پلیفیر از «Q» به جای «X» استفاده میکنند.

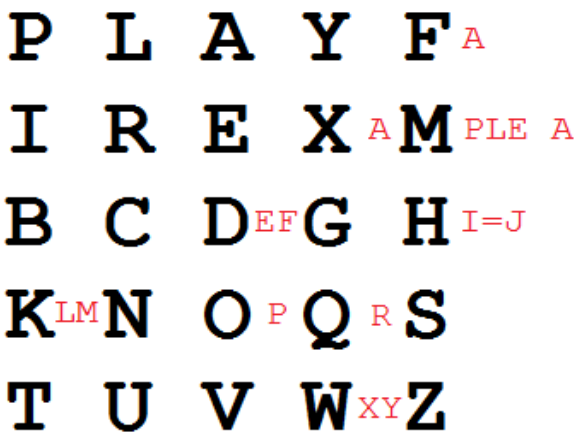
اگر حروف در همان ردیف جدول شما ظاهر می شوند ، به ترتیب آنها را با حروف سمت راست خود جایگزین کنید (اگر حروف اصلی در سمت راست ردیف قرار داشت ، از حرف سمت چپ ردیف استفاده کنید).

اگر حروف در همان ستون جدول شما ظاهر می شوند ، به ترتیب آنها را با حروف زیر خود جایگزین کنید (اگر به حروف اصلی در قسمت پایین ستون قرار داشت ،از حرف بالای ستون استفاده کنید).

اگر حروف در یک ردیف یا ستون نیستند ، حرف اول را با حرفی که در سطر حرف اول و ستون حرف دوم است جایگزین میکنیم. حرف دوم را با حرفی که در سطر حرف دوم و ستون حرف اول است جایگزین میکنیم.

برای رمزگشایی، از برعکس سه قاعده‌ی آخر استفاده میکنیم و از قاعده‌ی اول بدون تغییر آن استفاده میکنیم («X» و «Q» های اضافی حذف میشوند به دلیل اینکه وقتی پیام کامل شد هیچ معنی خاصی ندارند).

با استفاده از "playfair exmaple" به عنوان کلید (با فرض اینکه I و J قابل تعویض هستند) ، جدول به صورت زیر در می‌آید:(حروف قرمز رنگ حذف میشوند):



اگر متن کافی وجود داشته باشد رمز پلیفیر مانند اغلب رمزهای کلاسیک می‌تواند به راحتی کرک شود. اگر متن ساده و متن رمز معلوم باشند، دستیابی به رمز بسیار ساده است. وقتی تنها متن رمز معلوم باشد، تحلیل رمز شامل جستجو در فضای کلید برای یافتن تعداد تطبیق بین تعداد وقوع حرف در دیاگرام و تعداد وقوع حرف در پیام اصلی است.

تحلیل رمز در پلیفیر شبیه تحلیل رمز رمزهای ۴ مربعی و دو مربعی است، هرچند سادگی نسبی سیستم پلیفیر باعث ساده‌تر شدن شناسایی رشته‌های متن ساده می‌شود. یک دیاگراف پلیفیر و معکوس آن) مانند AB و (BA به الگوی حروفی مشابه در متن ساده رمزگشایی می‌شوند) مانند RE و .(ER در زبان انگلیسی، کلمات زیادی وجود دارند که شامل این دیاگراف‌های معکوس هستند مانند REceivER و .DEpartED شناسایی دیاگراف‌های معکوس نزدیک در متن رمز و تطبیق دادن آن به یک فهرست از واژه‌های شناخته شده که شامل اینطور کلمات است،راهی ساده برای تولید متن اصلی برای شروع ساخت کلید است.

یک رویکرد متفاوت برای مقابله با رمز پلیفیر استفاده از روش shotgun hill climbing است. این با یک مربع تصادفی از حروف شروع می‌شود. تغییرات جزیی ایجاد میشوند (یعنی تغییر حروف، ردیف یا منعکس‌کردن کل مربع) تا بررسی شود که متن به وجود امده از مربع، شباهت بیشتری به متن اصلی استاندارد دارد یا خیر. اگر مربع جدید یک بهبود در نظر گرفته شود، آنگاه پذیرفته شده و سپس جهش می‌یابد(همان تغییرات جزئی ایجاد میشوند) تا یک نامزد بهتر پیدا شود. در نهایت، متن اصلی چیزی بسیار شبیه یافت می‌شود. این فراتر از شکیبایی انسان معمولی است، ولی کامپیوترها می‌توانند از این الگوریتم برای رمزگشایی رمزهای پلیفیر با یک متن نسبتا کوچک استفاده کنند.

رمز چهارم: AES

استاندارد رمزنگاری پیشرفته (Advanced Encryption Standard) یا به اختصار AES مشخصه‌ای برای رمزنگاری داده‌های دیجیتال است که در سال ۲۰۰۱ توسط مؤسسه ملی فناوری و استانداردهای ایالات متحده ایجاد گردید. این رمز که در ابتدا ریندال (Rijndael) نامیده می‌شد و توسط دو رمزنگار بلژیکی به نام‌های ژوآن دیمن (Joan Daemen) و وینسنت رینمن (Vincent Rijmen) توسعه داده شد.

این الگوریتم رمزنگاری به جای استاندارد رمزنگاری داده‌ها (DES) که در سال ۱۹۷۷ منتشر شده،[۳] جایگزین گردیده‌است. الگوریتم AES یک الگوریتم کلید متقارن است، بدین معنی که از یک کلید یکسان برای رمزنگاری و رمزگشایی استفاده می‌شود.

استاندارد رمزنگاری پیشرفته بر اساس یک قاعده طراحی به نام substitution-permutation network است و به هر دو صورت سخت‌افزاری و نرم‌افزاری سریع است. برخلاف DES، استاندارد رمزنگاری پیشرفته از رمزنگاری فیستل استفاده نمی‌کند و همچنین گونه‌ای از Rijndael است که اندازه بلاک ثابت ۱۲۸ بیتی و اندازه کلید ۱۲۸، ۱۹۲ و ۲۵۶ بیتی دارد.

اندازه کلید استفاده شده در رمزAES ، تعداد تکرارهای چرخه‌های تبدیل را تعیین می‌کند که ورودی، با نام متن عادی را به خروجی نهایی با نام متن رمز شده تبدیل می‌نماید.

تعداد چرخه‌های تکرار به صورت زیر است:

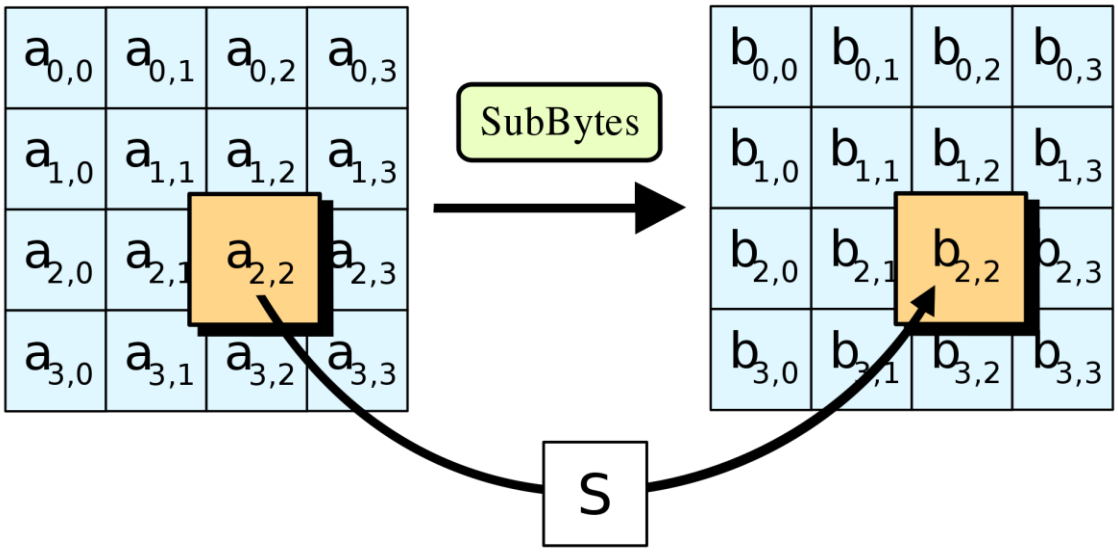
۱. ۱۰ چرخه تکرار برای کلیدهای ۱۲۸ بیتی.
۲. ۱۲ چرخه تکرار برای کلیدهای ۱۹۲ بیتی.
۳. ۱۴ چرخه تکرار برای کلیدهای ۲۵۶ بیتی.

هر تکرار شامل چندین مرحله پردازشی است، که یک مرحله بستگی به کلید رمزنگاری دارد. مجموعه‌ای از چرخه‌های معکوس برای تبدیل متن رمز شده به متن اصلی با استفاده از همان کلید رمزنگاری بکار گرفته می‌شود.

چهار چرخه در هر تکرار وجود دارد که به شرح هر کدام میپردازیم:

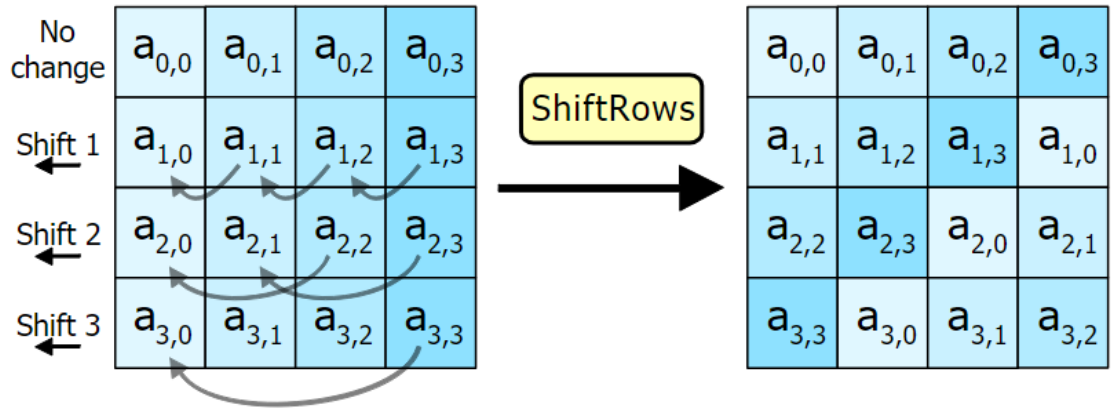
۱) مرحله SubBytes

در مرحله SubBytes، هر بایت در ماتریس state با استفاده از substitution box ۸ بیتی Rijndael S-box با یک SubByte جایگزین می‌گردد.S-box استفاده شده از معکوس فزاینده (multiplicative inverse) روی ()۲۸GF مشتق شده‌است که به داشتن خصوصیات غیرخطی خوب مشهور است. برای اجتناب از حملات مبتنی بر خصوصیات جبری ساده، S-box به وسیله ترکیب تابع معکوس با یک affine transformation معکوس پذیر ایجاد می‌گردد. affine transformation S-box همچنین به گونه‌ای انتخاب می‌شود که از هرگونه نقاط ثابت (و همچنین آشفتگی) و هرگونه نقاط ثابت معکوس اجتناب شود.



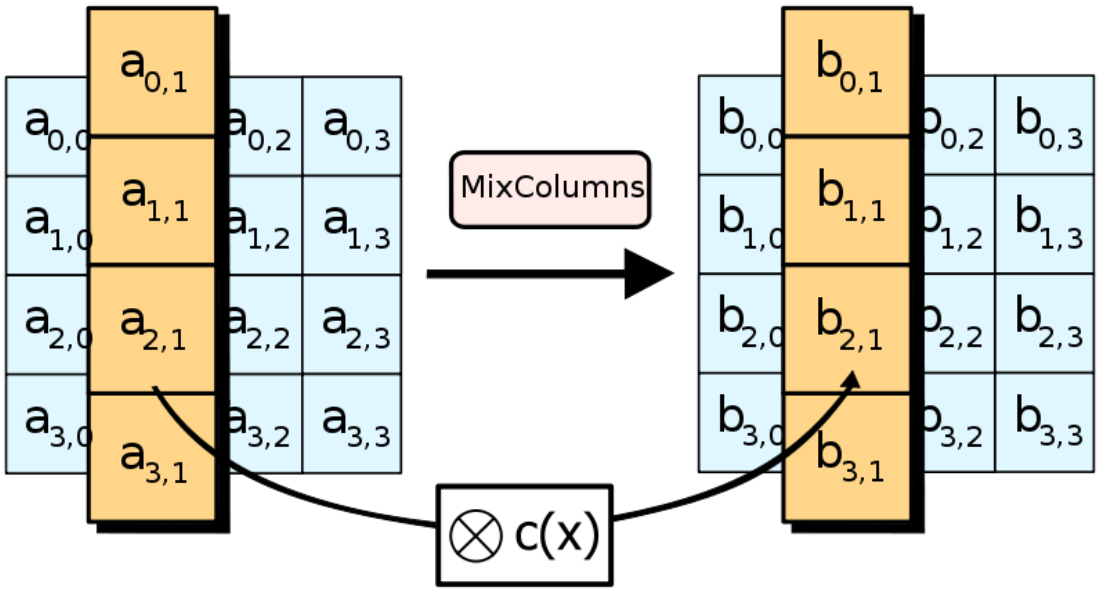
۲) مرحله ShiftRows

مرحله ShiftRows روی سطرهای state عمل می‌کند. در این مرحله بایت‌های هر سطر به وسیله یک آفست (Offset) معین به صورت چرخشی شیفت می‌یابد. برای AES، نخستین سطر بدون تغییر باقی می‌ماند. هر بایت از سطر دوم یکی به سمت چپ شیفت می‌یابد. به صورت مشابه، سطرهای سوم و چهارم به ترتیب با آفست‌های دو و سه شیفت می‌یابند. برای بلاک‌های با اندازه ۱۲۸ و ۱۹۲ بیتی، الگوی شیفت دادن یکسان است. سطر n به تعداد n- بایت به صورت چرخشی به چپ شیفت می‌یابد. بدین صورت، هر ستون از state خروجی در این مرحله ترکیب شده بایت‌های هر ستون از state ورودی است. (انواع Rijndael با اندازه بلاک بزرگتر، آفست‌هایی اندکی متفاوت دارند) برای یک بلاک ۲۵۶ بیتی، نخستین سطر بدون تغییر باقی می‌ماند و سطرهای دوم و سوم و چهارم به ترتیب یک، دو و سه بایت شیفت می‌یابد. این تغییر تنها برای رمز Rijndael با بلاک ۲۵۶ بیتی اعمال می‌شود چون AES بلاک‌های ۲۵۶ بیتی استفاده نمی‌کند.



۳) مرحله MixColumns

در مرحله MixColumns، چهار بایت از هر ستون state با استفاده از تبدیل خطی معکوس ترکیب می‌شوند. تابع MixColumns چهار بایت را به عنوان ورودی در نظر می‌گیرد و چهار بایت را به خروجی می‌دهد، که هر بایت ورودی بر هر چهار بایت خروجی تأثیر می‌گذارد. به همراه ShiftRows، مرحله MixColumns آشفتگی و پخش را در رمزنگاری فراهم می‌نماید.



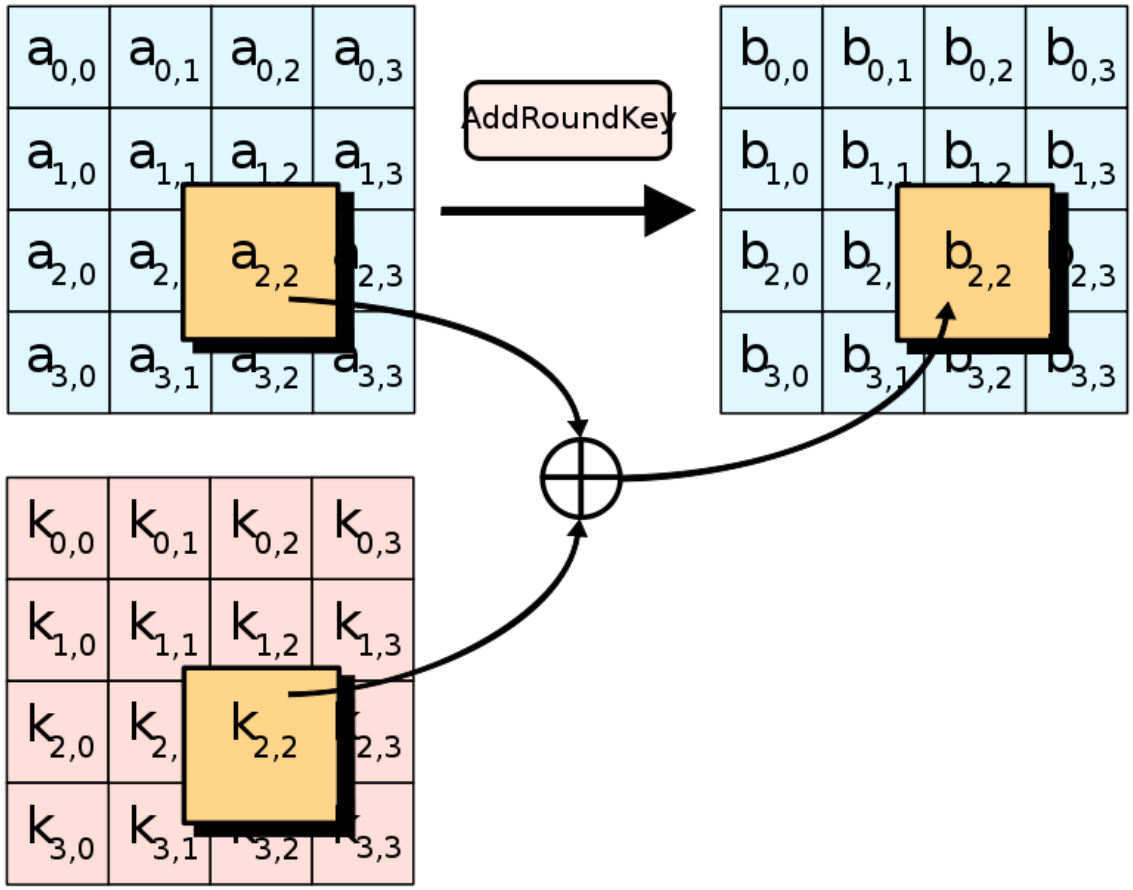
در طول این عمل، هر ستون توسط ماتریس شناخته شده‌ای که برای کلید ۱۲۸ بیتی است ضرب می‌گردد.

عمل ضرب بدین صورت تعریف می‌شود: ضرب در ۱ به معنی بدون تغییر، ضرب در ۲ به معنای جابجایی به سمت چپ و ضرب در ۳ به معنای جابجایی به سمت چپ و سپس انجام XOR را با مقدار اولیه جابجانشده. پس از جابجایی، اگر مقدار جابجاشده بیشتر از ۰xFF باشد، XOR شرطی با ۰x11B باید انجام شود.

به صورت کلی تر، هر ستون به عنوان یک چند جمله‌ای روی GF(۲^۸) تلقی می‌شود و پس از آن پیمانه x^4+1 با یک چند جمله‌ای ثابت $c(x) = 0x03 \cdot x^3 + x^2 + x + 0x02$ ضرب شده است. ضرایب با معادل مبنای ۱۶ از نمایش دودویی بیت‌های چندجمله‌ای $GF(2)[x]$ نمایش داده می‌شود. مرحله MixColumns همچنین می‌تواند به صورت ضرب یک ماتریس خاص MDS در یک finite field دیده شود. این فرایند در مقاله ستون‌های ترکیبی Rijndael به صورت مفصل تر شرح داده شده است.

۴) مرحله AddRoundKey

در مرحله AddRoundKey، subkey با state ترکیب می‌شود. در هر دور، یک subkey از کلید اصلی توسط زمانبند کلید Rijndael مشتق می‌شود. هر subkey به همان اندازه state است. subkey با ترکیب کردن هر بایت از state با بایت متناظر از subkey با استفاده از XOR بیتی جمع بسته می‌شود.



تا ماه مه ۲۰۰۹، تنها حملات منتشر شده موفق علیه AES کامل، حملات Side-Channel در برخی از پیاده‌سازی‌های خاص بود. طراحی و قدرت تمام طول کلیدهای الگوریتم AES (به عنوان مثال ۱۲۸، ۱۹۲ و ۲۵۶) برای محافظت از اطلاعات طبقه‌بندی شده تا سطح محرمانه کافی است. اطلاعات خیلی محرمانه نیاز به استفاده کلیدهای با طول ۱۹۲ یا ۲۵۶ دارد. پیاده‌سازی AES در محصولات در نظر گرفته شده برای حفاظت از سیستم‌های امنیت ملی و / یا اطلاعات باید توسط NSA، پیش از استفاده، بازبینی و مجوز داده شود.

AES دارای ۱۰ چرخه برای کلیدهای ۱۲۸ بیتی، ۱۲ چرخه برای کلیدهای ۱۹۲ بیتی و ۱۴ چرخه برای کلیدهای ۲۵۶ بیتی می‌باشد. در سال ۲۰۰۶، بهترین حملات شناخته شده در ۷ چرخه برای کلیدهای ۱۲۸ بیتی، ۸ چرخه برای کلیدهای ۱۹۲ بیتی، و ۹ چرخه برای کلیدهای ۲۵۶ بیتی بودند.

رمز پنجم: DES

الگوریتم استاندارد رمزگذاری داده Data Encryption Standard (DES) در دهه ۷۰ میلادی در آمریکا به‌عنوان یک استاندارد کدگذاری مطرح شد. این الگوریتم این‌گونه عمل می‌کند که رشته‌ای از متن اصلی با طول ثابت را به عنوان ورودی می‌گیرد و پس از انجام یک سری اعمال پیچیده روی آن خروجی را که طولی برابر طول ورودی دارد تولید می‌کند. DES هم چنین از یک کلید برای ایجاد رمز استفاده می‌کند و تنها کسانی قادر به رمزگشایی خواهند بود که مقدار کلید را می‌دانند. اگرچه تحلیل‌هایی که دربارهٔ DES انجام شده‌است از هر روش رمز قطعه‌ای دیگری بیشتر است ولی عملی‌ترین حمله علیه این الگوریتم جستجوی جامع فضای کلید است.

در DES طول قطعات ۶۴ بیت است. کلید نیز شامل ۶۴ بیت است ولی در عمل تنها از ۵۶ بیت آن استفاده می‌شود و از ۸ بیت دیگر فقط برای چک کردن parity استفاده می‌شود. الگوریتم شامل ۱۶ مرحله مشابه‌است که هر مرحله یک دور ۴نامیده می‌شود. متنی که قرار است رمزگذاری شود ابتدا در معرض یک جایگشت اولیه (IP) قرار می‌گیرد. سپس یک سری اعمال پیچیده وابسته به کلید روی آن انجام می‌شود و در نهایت در معرض یک جایگشت نهایی (FP) قرار می‌گیرد.

IP,FP معکوس هم هستند FP عملی که توسط IP انجام شده‌است را خنثی می‌کند؛ بنابراین از جنبه رمزنگاری اهمیت چندانی ندارند و برای تسهیل نمودن بار کردن قطعات داده در سخت‌افزارهای دهه ۱۹۷۰ استفاده شدند ولی اجرای DES در نرم‌افزار را کند کردند. قبل از دور اصلی، داده به دو بخش ۳۲ بیتی تقسیم می‌شودکه 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 هم هستند و تنها تفاوت آنها این است که زیر کلیدها در زمان رمزگشایی در جهت معکوس رمزگذاری به کار برده می‌شوند؛ و بقیه الگوریتم درهر دو یکسان است که این امر پیاده‌سازی رابه خصوص در سخت‌افزاربسیار آسان می‌کند و دیگر نیازی به الگوریتم‌های متفاوت برای رمزگذاری و رمزگشایی نیست.

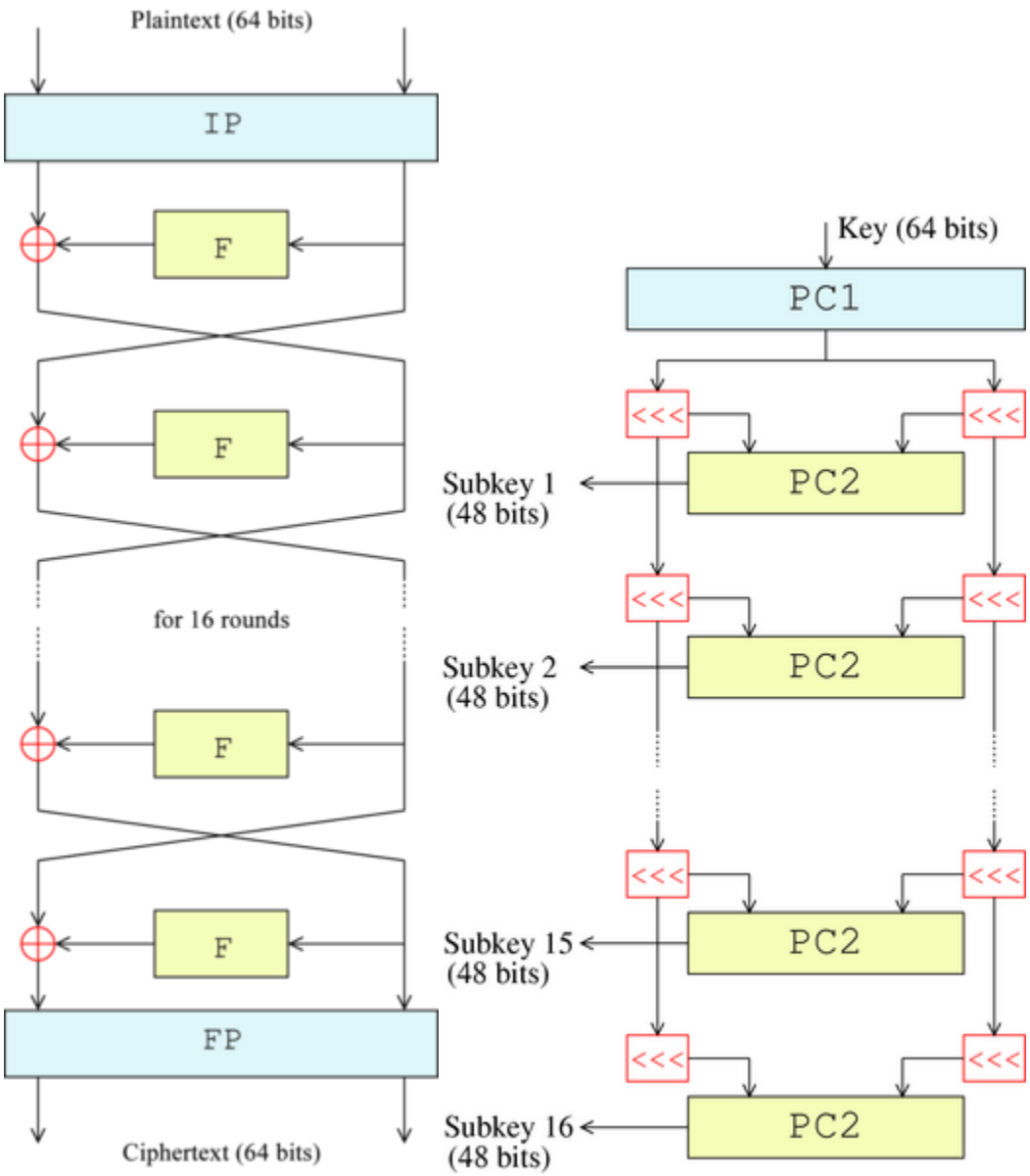
تابعی که خروجی IP را می‌گیرد وپس از شانزده مرحله ورودی FP را فراهم می‌کند تابع F نامیده می‌شود. این تابع یک ورودی ۳۲ بیتی و یک ورودی ۴۸ بیتی دارد و یک خروجی ۳۲ بیتی تولید می‌کند. بلاک ورودی شامل ۳۲ بیت که نیمه سمت چپ را تشکیل می‌دهد و با L نشان داده می‌شود و به دنبال آن ۳۲ بیت دیگر که نیمه راست را تشکیل می‌دهد و با R نمایش داده می‌شود است. پس کل بلاک را می‌توان به صورت LR نمایش داد.

اگر K یک بلاک ۴۸ بیتی باشد که از کلید اصلی ۶۴ بیتی مشتق شده‌است و خروجی یک دور با ورودی LR و خروجیLR۱ به صورت زیر تعریف می‌شود $L_1=R$ $R_1=L \text{ XOR } F(R,K)$ اگر KS تابعی باشد که کلید ۶۴ بیتی KEY و یک عدد صحیح در محدوده ۱ تا ۱۶ را به عنوان ورود ی می‌گیرد و کلید ۴۸ بیتی Kn را به عنوان خروجی تولید می‌کند به طوری که بیتهای Kn از تغییر محل بیتهای KEYحاصل شده‌اند داریم $Kn= KS (n.KEY)$

KS را تابع key schedule می‌نامند؛ بنابراین در حالت کلی داریم: Ln=Rn-۱ Rn=Ln-۱ XOR f(Rn-۱,Kn) برای رمزگشایی نیز داریم: R=L۱ L=R۱ XOR f(L۱,K)

در نتیجه رمزگشایی با همان الگوریتمی که برای رمزگذاری استفاده شد انجام می‌شودو در هر مرحله همان K بیتی که به عنوان کلید برای رمزگذاری استفاده شده بود مورد استفاده قرار می‌گیرد بنابراین می‌توان نوشت Rn-۱=Ln Ln-۱=Rn XOR f(Ln,Kn)

برای محاسبات رمزگشایی ۱۶L۱۶R ورودی IP و ۰L۰R ورودی FP است. کلید شانزدهم در مرحله اول، کلید پانزدهم در مرحله دوم و به همین ترتیب کلید اول در مرحله شانزدهم مورد استفاده قرار می‌گیرد.



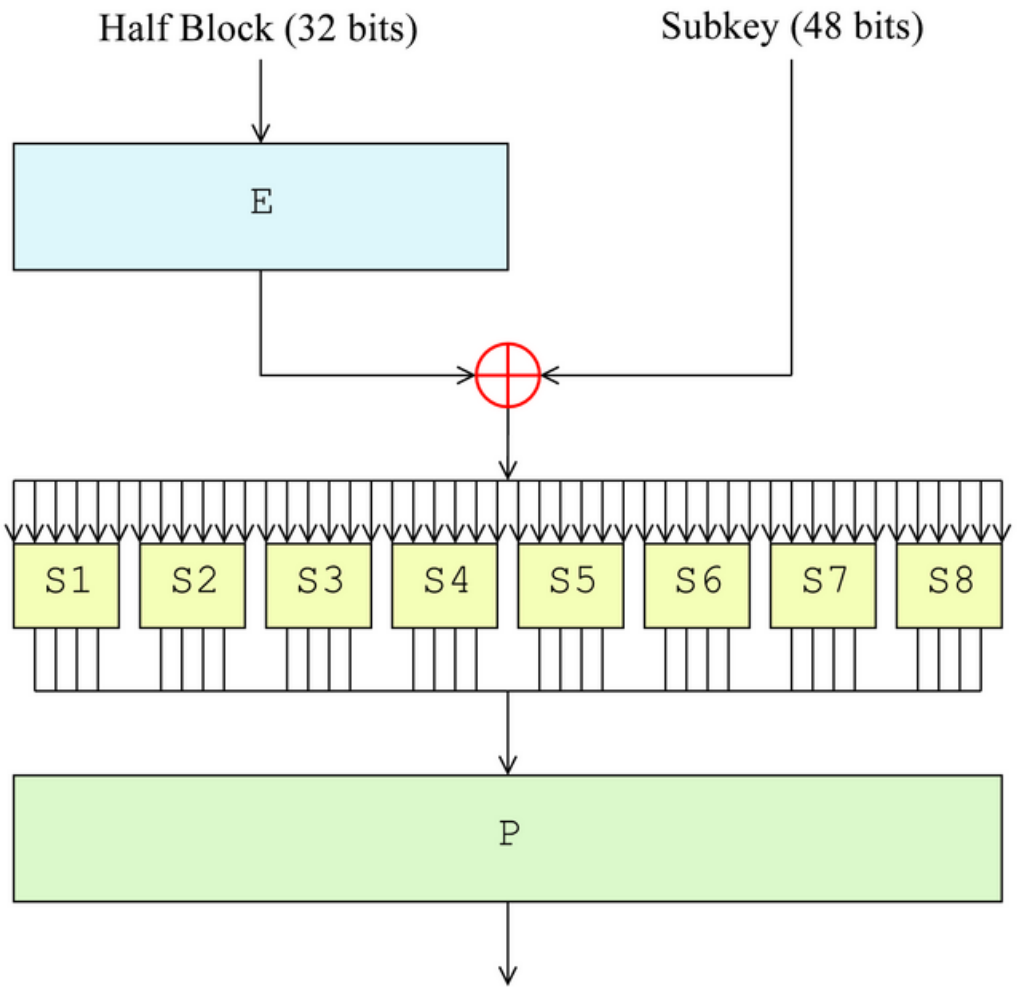
شرح تابع f:

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

ترکیب کلید: در این مرحله حاصل مرحله قبل با یک زیر کلید XOR می‌شود. شش کلید ۴۸ بیتی با استفاده از الگوریتم key schedule 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-box ها ورودی ۶ بیتی خود را با استفاده از یک تبدیل غیر خطی که به شکل یک جدول look up است به یک خروجی ۴ بیتی تبدیل می‌کند S-box ها قلب DES هستند و بدون آنها رمز خطی خواهد بود و در نتیجه قابل شکستن خواهد شد.

جایگشت: در نهایت ۳۲ بیت خروجی S-box ها با استفاده از یک جایگشت ثابت مجدداً سازماندهی می‌شود (P-box).



رمز ششم: RSA

از اولین شیوه های رمزنگاری به روش کلید عمومی است که به صورت گسترده برای تامین امنیت انتقال داده استفاده می شود. در این چنین سیستم های رمزنگاری، کلید رمزگذاری عمومی است و از کلید رمزگشایی که مخفی است، جداست. در RSA، این عدم تقارن مبتنی بر این است که تجزیه از عددی که ضرب دو عدد اول بزرگ است در عمل بسیار دشوار است.

یک کاربر RSA، یک کلید عمومی را بر اساس دو عدد اول بزرگ را همراه با یک مقدار تصادفی ساخته و به صورت عمومی منتشر می کند. هر کسی می تواند از این کلید عمومی برای رمزگذاری یک پیام استفاده کند، اما تنها کسی که آن دو عدد اولی که کلید بر اساس آن ها ساخته شده را می داند، قادر به رمزگشایی پیام است. شکستن رمزگذاری RSA به مسئله ی RSA معروف است. تاکنون هیچ روشی برای شکست دادن این سیستم (در صورت استفاده ی کلید به اندازه ی کافی بزرگ) منتشر نشده است.

RSA شامل ۴ مرحله است: ساخت کلید، توزیع کلید، رمزنگاری و رمزگشایی.

یک اصل اساسی در RSA این است که یافتن سه عدد صحیح مثبت بسیار بزرگ مانند e, n و d که رابطه ی زیر برایشان بر قرار باشد، عملی است.

$$(m^d)^e \equiv m(\mod n)$$

و با دانستن این که e و n و یا حتی m، یافتن d می تواند بسیار مشکل باشد.

RSA به‌طور کلی از دو کلید تشکیل می‌شود. کلید عمومی و کلید خصوصی. کلید، عددی ثابت است که در محاسبات رمزنگاری استفاده می‌شود. کلید عمومی برای همه معلوم بوده و برای رمزنگاری پیام استفاده می‌شود. این پیام فقط توسط کلید خصوصی باز می‌شود. به بیان دیگر همه می‌توانند یک پیام را رمز کنند اما فقط صاحب کلید خصوصی می‌تواند پیام را باز کند و بخواند. کلید عمومی توسط اعداد صحیح n و e نمایش داده میشود و کلید خصوصی، توسط عدد صحیح d.

m نمایان گر پیام است که از قبل توسط یک تکنیک خاص آماده شده است و در ادامه این تکنیک شرح داده شده است.

هر چند از لحاظ ریاضی کلیدهای عمومی و خصوصی با یکدیگر ارتباط دارند اما تقریباً محال است که کسی بتواند حتی با تجهیزات پیشرفته و صرف وقت زیاد با داشتن یکی از کلیدها، دیگری را تشخیص دهد. در واقع می‌توان گفت که با توجه به سطح دانش کنونی و سامانه‌های رایانه‌ای موجود، الگوریتم رمزنگاری و ارتباط میان کلیدها تقریباً غیرقابل شکستن است.

کلید عمومی تشکیل می‌شود از:

عدد n که عدد مشترک است و عدد e که عدد عمومی است.

کلید خصوصی تشکیل می‌شود از:

عدد n که عدد مشترک است و عدد d که عدد خصوصی است.

مراحل ساخت کلید:

مراحل زیر برای ساخت کلید طی می‌شود:

۱. دو عدد اول بزرگ p و q را به صورت تصادفی بیابید به‌طوری‌که $p \neq q$.

- برای اهداف امنیتی، p و q باید به صورت تصادفی انتخاب شوند، و در اندازه مشابه باشند اما طول آن ها در حد چند رقم متفاوت باشد تا تجزیه را کمی دشوار تر کند. اعداد صحیح اول می توانند به صورت کارآمد توسط یک تست اول بودن یافت شوند.
- p و q پنهان باقی می مانند.

2.عدد n را محاسبه کنید به‌طوری‌که $n = pq$.

- n به عنوان پیمانه برای هر دو کلید خصوصی و عمومی استفاده می شود. طول کلید، تعداد بیت های n، طول کلید را مشخص می کند.

3.تابع $\lambda(n)$ را محاسبه کنید که λ Carmichael function است . از آن جایی که n=pq، $\lambda(n) = lcm(lambda(p), lambda(q))$ و از آن جایی که p و q اول هستند، $\lambda(p) = \phi(p) = p - 1$ و به همین روال $\lambda(q) = q - 1$. بنابراین $\lambda(n) = lcm(p - 1, q - 1)$

- این مقدار مخفی باقی می ماند.
- مقدار lcm ممکن است از طریق الگوریتم اقلیدسی محاسبه شود، از آن جایی که $lcm(a, b) = |ab| / \gcd(a, b)$

4.عدد e را انتخاب کنید به‌طوری‌که $1 < e < \lambda(n)$ و نسبت به $\lambda(n)$ اول باشد.

- عدد e به عنوان توان کلید عمومی منتشر می‌شود.
- e با داشتن تعداد کم بیت و وزن وزن همینگ کم، منجر به رمزگذاری کارآمد تری می شود. معمول ترین مقدار انتخاب شده برای e، حدود عدد 2^{16} یک که برابر با 65536 است می باشد. کوچک ترین و سریع ترین مقدار ممکن برای e ، عدد 3 است اما چنین مقدار کمی نشان داده است که در بعضی ساختار ها امنیت کم تری را ایجاد می کند.

5.عدد d که $d \equiv e^{-}1(\mod \lambda(n))$ را به دست بیاورید.

- عدد d به عنوان توان کلید خصوصی محافظت می‌شود.
- در واقع $de \equiv 1 \pmod{\lambda(n)}$. این مقدار می تواند به صورت کارآمدی توسط الگوریتم تعمیم‌یافته اقلیدس پیدا شود از آن جایی که d و $\lambda(n)$ اول هستند،این معادله یک فرمی از قضیه بزو است که در آن d یکی از ضریب ها است.

دو عدد اول می‌توانند توسط روش **پیدا کردن اعداد اول احتمالی** پیدا شوند.

:RNG

یک random number generator (RNG) دستگاه یا تابع یا تولید کننده‌ای است که یک یا یک سری اعداد را به صورت تصادفی تولید می‌کند.

در کامپیوتر ها همه محاسبات بر مبنای منطق و صفر و یک می باشد بنابراین یک کامپیوتر توانایی ساخت یک عدد تصادفی حقیقی را ندارد؛ از این رو، کامپیوتر ها از توابع شبیه ساز اعداد تصادفی استفاده می کنند. به این مدل از توابع تصادفی pseudo-random number generators (PRNG) گفته میشود. این تولید کننده ها الگوی مشخصی دارند و با استفاده از ترکیب مواردی مختلف اعداد تصادفی را تولید و در کنار همدیگر قرار می دهند.

برخی از پدیده‌های طبیعی الگوهای مناسبی برای تولید این اعداد هستند به عنوان مثال برخی پدیده‌های فیزیکی از جمله اختلالات حرارتی در دیودهای زنر دارای رفتاری کاملاً تصادفی هستند و می‌توانند پایه‌ای برای تولیدRNG های فیزیکی و سخت‌افزاری باشند.

همانطور که اشاره شد، الگوهای طبیعی جالبی برای تولید اعداد تصادفی وجود دارد؛ یک روش متداول استفاده از یک تابع درهم ساز است که ورودی اش جریانی از فریم‌های ویدئویی یک منبع غیرقابل پیش‌بینی می‌باشد. به عنوان مثال لاواراند از تصاویر تعدادی لامپ لاوا استفاده کرد. Lithium Technologies از تصاویر آسمان و Random.org از صداهای آشفته جوی استفاده می‌کند.

تولیدکننده‌های اعداد شبه تصادفی الگوریتم‌هایی با قابلیت تولید اعداد تصادفی هستند هرچند اعداد تولید شده توسط آن‌ها به‌طور تناوبی تکرار می‌شود یا آنکه حافظه زیادی را اشغال می‌کنند. یکی از متداولترین تولیدکننده‌های اعداد تصادفی، LCG Linear Congruential Generator است که رابطه‌ای بازگشتی دارد:

$$R_{n+1} = (aR_n + b) \bmod m$$

بیشترین تعداد عددی که این رابطه می‌تواند تولید کند ، m عدد شبه تصادفی است.

بسیاری از زبان‌های برنامه‌نویسی رایانه شامل توابع کتابخانه‌ای هستند که برای تولید اعداد تصادفی (یک بایت، کلمه ویا اعداداعشاری تصادفی با توزیع یکنواخت بین ۰ و ۱)طراحی شده‌اند. این توابع کتابخانه‌ای اغلب از لحاظ خصوصیات آماری ضعیف هستند و الگوهایشان پس از تنها ۱۰۰۰ رشته دوباره تکرار می‌شود، آن‌ها اغلب با زمان واقعی رایانه به عنوان seed راه‌اندازی می‌شوند. در واقع این توابع در بعضی موارد به تعداد کافی رویداد تصادفی تولید می‌کنند (مثلاً در بازی‌های ویدئویی) ولی وقتی رویدادهای تصادفی با کیفیت بالا مورد نظر است، ناکارآمد هستند (مثلاً در رمزنگاری).

از آن‌جایی‌که رایانه‌ها ماشین‌هایی از نوع معیّن (Deterministic) هستند، با دریافت ورودی یکسان، همیشه یک خروجی بیرون می‌دهند. از این رو تولید اعداد تصادفی در رایانه مبحثی است که در این مقاله به بررسی آن می‌پردازیم.

در زبان‌های برنامه‌نویسی گوناگون، تابعی وجود دارد که عددی تصادفی و معمولاً در بازهٔ صفر و یک تولید می‌کند. این تابع باید به گونه‌ای باشد که با چند بار تولید عدد تصادفی کاربر قادر به حدس زدن و پیدا کردن قاعده و الگویی در ایجاد این اعداد نشود.

هر بار که این تابع صدا زده می‌شود، رایانه عدد تولید شدهٔ پیشین را به عنوان ورودی جدید تابع تولید عدد تصادفی می‌فرستد. منشاء مشکل نیز در همین مرحله است.

هر بار که این تابع صدا زده شود، بر اساس ماهیت جبری ماشین و با توجه به مقدار اولیهٔ فرستاده شده به تابع تولید عدد تصادفی (seed) باید با یک دنباله از اعداد مشابه یکدیگر مواجه شویم.

راحت‌ترین راه حل این مسئله در دنیای کامپیوتر استفاده از زمان فعلی دستگاه است. کامپیوترهای امروزی زمان را با دقت میلی‌ثانیه در دسترس دارند. برنامه‌ها می‌توانند زمان اولین اجرای خود را به عنوان seed به اولین باری که تابع تولید اعداد تصادفی صدا زده می‌شود، بفرستند. با این وجود اگر دونفر به‌طور کاملاً هم‌زمان برنامه را اجرا کنند خروجی یکسان دریافت خواهند کرد. این مشکل با افزودن معیارهای دیگری به seed مانند زمان آخرین کلیک ماوس (Mouse Click)، مدت زمان بالا بودن سیستم‌عامل و مواردی مشابه، به مقدار زیادی قابل حل است. با افزودن این معیارها و معیارهای مشابه دیگر به برنامه احتمال ایجاد تشابه را به سمت صفر کاهش می‌دهیم.

اعداد تصادفی تولید شده توسط رایانه و محاسبات ریاضی اعداد کاملاً تصادفی نبوده و از اینرو این اعداد را اعداد شبه تصادفی می‌نامند.