

موضوع پروژه : پیاده سازی الگوریتم های رمزنگاری

دانشجو : محمدرضا مستعلی زاده

96149059

استاد مربوطه : جناب آقای دکتر شکیبا

نیم سال دوم 1400-99

**فهرست**

[**الگوریتم های رمزنگاری کلاسیک** 1](#_Toc75727462)

[1. رمز سزار (Caesar) 1](#_Toc75727463)

[2. رمز Playfair 1](#_Toc75727464)

[**الگوریتم های رمزنگاری مدرن با کلید متقارن** 2](#_Toc75727465)

[1.1. رمز ورنام (G.Vernam) 2](#_Toc75727466)

[2.1. رمز با کلید یکبار مصرف (OTP) 2](#_Toc75727467)

[1.2. استاندارد رمزنگاری داده (DES) 3](#_Toc75727468)

[2.2. استاندارد رمزنگاری داده سه تایی (3DES) 6](#_Toc75727469)

[3.2. استاندارد رمزنگاری پیشرفته AES) یاRejndael ) 7](#_Toc75727470)

[**الگوریتم های رمزنگاری مدرن با کلید نامتقارن** 10](#_Toc75727471)

[1. الگوریتم Diffie–Hellman 10](#_Toc75727472)

[2. الگوریتم RSA 11](#_Toc75727473)

[**الگوریتم های درهم سازی** 12](#_Toc75727474)

[1. الگوریتم MD5 12](#_Toc75727475)

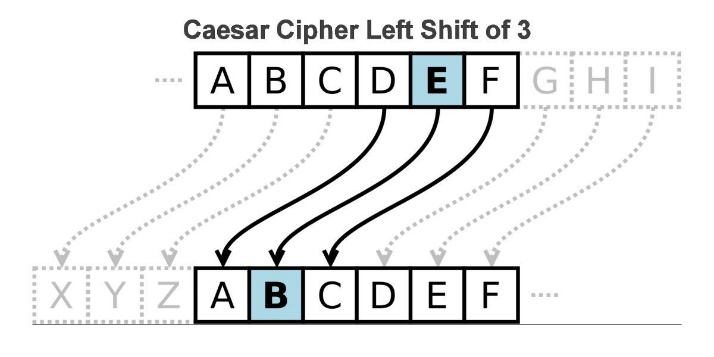
[2. خانواده SHA 13](#_Toc75727476)

[3. الگوریتم(SHA-3) \_ KECCAK 14](#_Toc75727477)

# **الگوریتم های رمزنگاری کلاسیک**

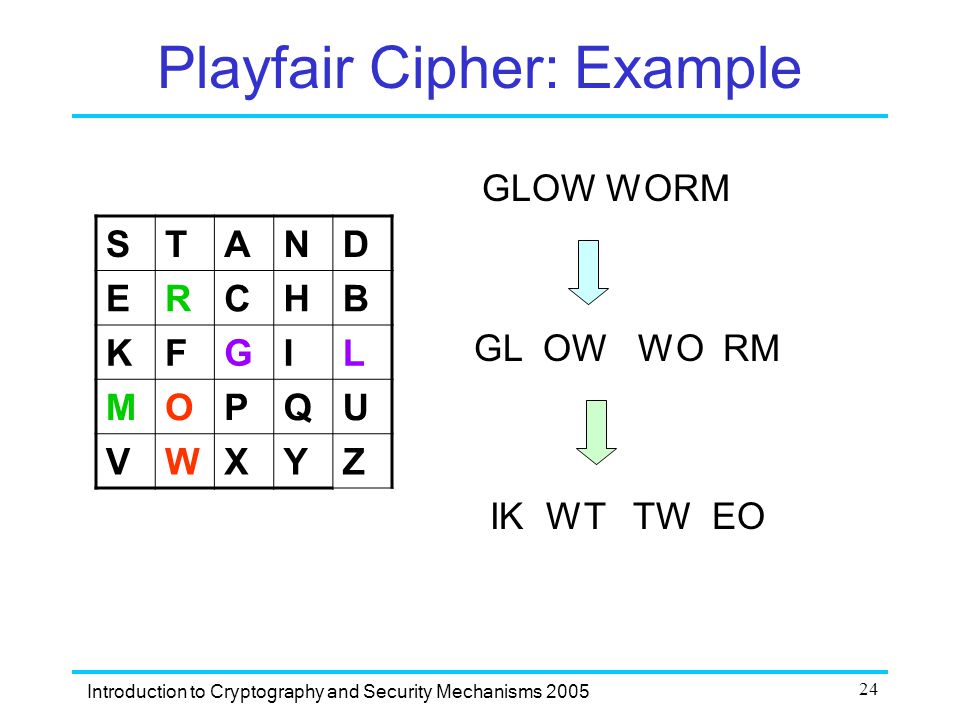
## **1. رمز سزار (Caesar)**

جایگذاری هر حرف با حرف بعدی و شامل 5 جزء : متن اصلی، متن رمزشده، کلید، الگوریتم رمزگذاری، الگوریتم رمزگشایی

* فضای متن اصلی (X) : کلیه رشته های از طول دلخواه با الفبای لاتین
* فضای متن رمزشده (Y) : کلیه رشته های از طول دلخواه با الفبای لاتین
* فضای کلید (K) : اعداد 0 تا 25

برای حمله به این رمز تمام کلیدهای ممکن چک شده و جدول ساخته می شود. سپس تک به تک رمزگشایی می شود. فضای کلید K=26<=2128 است، پس این رمز در برابر جستجوی فراگیر ایمن نیست. برای بهبود از رمز جانشانی تک الفبایی استفاده می شود.

## **2. رمز Playfair**

برای کلید یک واژه به زبان انگلیسی درنظر گرفته و کارکترهای تکراری حذف می شود. سپس در یک جدول 5×5، از بالا سمت چپ ابتدا کلمه و سپس بقیه حروف الفبا نوشته می شود (i و j در یک خانه هستند).

**رمزگذاری**

دو حرف دو حرف کلمه جدا می شود. اگر تک حرف اضافه باشد، یک حرف به دلخواه به آن اضافه می شود که تکراری نباشد. اکنون دو حرف را درنظر گرفته، یک مستطیل می شود و به جای هر حرف، حرف هم سطر آن جایگزین می شود. اگر در یک ستون قرار داشتند، حروف پایین، و اگر در یک سطر قرار داشتند، حروف سمت راست جایگزین می شود. 25! حالت برای این جدول وجود دارد.

**رمزگشایی**

برعکس قبل انجام داده، اگر هر دو در یک ستون بودند، حروف بالایی، و اگر در یک سطر بودند، حروف سمت چپ جایگزین می شود.

**امنیت Playfair**

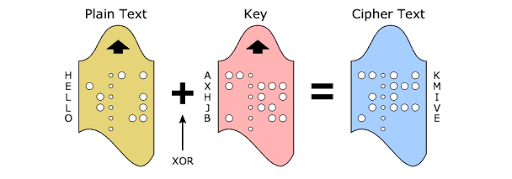
اما متاسفانه این رمز در برابر حمله تحلیل آماری آسیب پذیر است. چون فراوانی دوتایی ها داخل متن اصلی با دوتایی ها داخل متن رمزشده یکسان است. و می توان بجای فراوانی تک حرفی، فراوانی دو حرفی باشد. اگر کلید حرف تکراری داشته باشد، حروف تکراری حذف می شود.

# **الگوریتم های رمزنگاری مدرن با کلید متقارن**

1. **رمز های جریانی :** متن اصلی یک بیت یک بیت به متن رمز شده تبدیل می شود و این رمز شدن بیت ها از یکدیگر مستقل است. به عنوان مثال در تماس تلفنی و تلویزیون.
2. **رمز های قطعه ای :** چند بیت چند بیت داده ها در کنار هم قرار گرفته، تحت کلید رمزشده و ارسال می شوند. بیت های هر قطعه باهم به صورت وابسته رمزگذاری می شوند. به عنوان مثال برای فایل های حجیم و چند بخشی.

## **1.1.** **رمز ورنام \_ G.Vernam**

اگر متن اصلی و کلید این چنین باشند :

متن اصلی :X = (xn, …, x1)

: کلید K = (kl, …, k1) for l=n => yi = xi ⊕ ki

: متن رمزشده Y = (yn, …, y1)

**در رمزگذاری :** اگر کلید هم طول با متن اصلی باشد، بیت به بیت کلید با متن اصلی XOR می شود. اما اگر کلید هم طول نباشد، کارهای مختلفی می توان انجام داد. اما هرکدام ایراداتی دارند.

* راهکار اول، کلید تکرار شود.
* راهکار دوم، ادامه متن رمزشده به عنوان کلید در نظر گرفته شود. اما ایمن نیست.

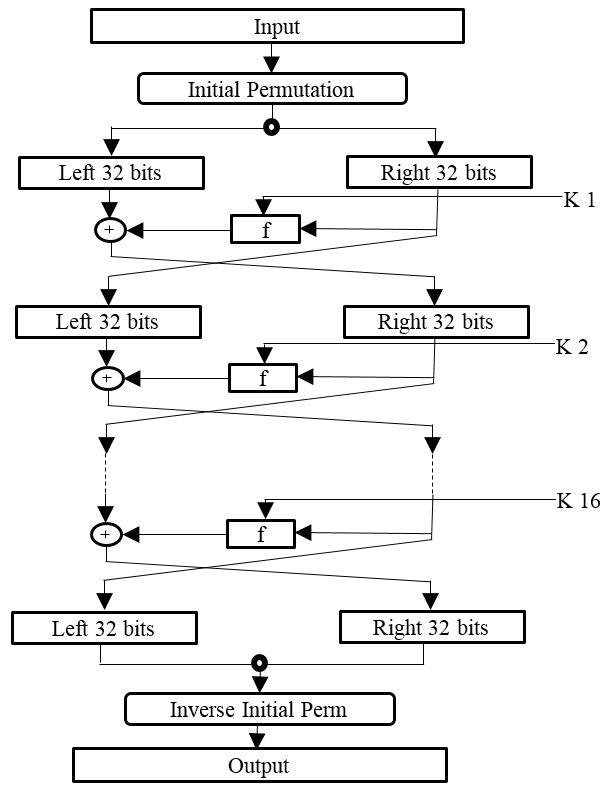
## **2.1. رمز با کلید یکبار مصرف (OTP) \_ One Time Pad**

یک رمز فوق العاده مهم از لحاظ نظری و الهام بخش طراحی خیلی از رمز های دیگر هم بوده است. این رمز دارای امنیت بدون قید و شرط است. اما در عمل قابل استفاده نیست. در واقع همان رمز ورنام است که در آن :

1. کلید به صورت تصادفی تولید می شود.
2. کلید هم طول با متن اصلی باشد.
3. از هر کلید تنها برای رمزگذاری یک متن استفاده شود.

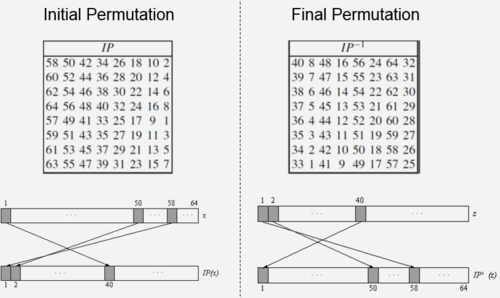
اگر یک کانال امن انتقال وجود داشته باشد، چه نیازی به انتقال کلید و سپس پیام است؟ خود پیام به جای کلید انتقال داده می شود. به همین دلیل در عمل قابل استفاده نیست.

## **1.2. استاندارد رمزگذاری داده (DES) \_ Data Encryption Standard**

یک طول بلاک (block) درنظر گرفته، با همدیگر رمزگذاری شده و یک بلاک خروجی ساخته می شود. امروزه به تنهایی کاربردی ندارد چون در برابر جستجوی فراگیر به زانو درآمد. شامل چند مرحله است که هر مرحله از مرحله قبل با یک پارامتر اضافه، با یک کلید کمکی متمایز شده است.

در این الگوریتم 16 مرحله یکسان تکرار می شود و خروجی هر مرحله می شود ورودی مرحله بعد و این روال تا آخر ادامه پیدا می کند. این نوع رمز از نوع فیستل (Feistel) است و مزیت آن این است که الگوریتم رمزنگاری و رمزگشایی یکسان است و فقط یک فرق جزئی دارند. در الگوریتم رمزگذاری، به ازای هر مرحله یک کلید وجود دارد، پس از روی کلید اصلی 16 کلید ساخته می شود. ترتیب ورود کلید ها در رمزگشایی برعکس است.

قبل از وارد شدن به این 16 مرحله، یک جایگشت اولیه و قبل از تولید خروجی، یک جایگشت پایانی انجام می شود. این دو جایگشت معکوس یکدیگر هستند (IP و IP-1).

**اعمال جایگشت اولیه IP**

یک جایگشت اولیه از متن اصلی و به صورت سطری است. عدد تکراری وجود ندارد و هیچ عددی نیست که داخل متن اصلی باشد و داخل جدول نباشد. داده ای از بین نمی رود. بعد از جایگشت اولیه، 64 بیت خروجی به دو نیمه 32 بیتی سمت راست (R) و سمت چپ (L) تقسیم می شود. نیمه های ابتدایی چون مربوط به ورودی مرحله اول هستند، با R0 و L0 و خروجی مرحله اول با R1 و L1 نشان داده می شود. همینطور ادامه پیدا می کند تا 16 مرحله.

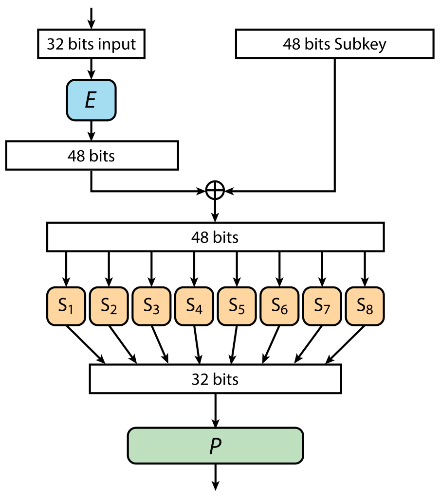
**محاسبه Ri و Li ها**

* برای R1 : به عنوان ورودی R0 را به تابع f() داده و سپس با L0 ، XOR می شود و به عنوان خروجی داخل R1 قرار می گیرد.
* برای L1 : R0 بدون تغییر داخل L1 قرار می گیرد.

در فرمول K1 همان کلید مرحله اول است. سپس ورودی مرحله دوم R1 و L1 می شود. پس به ازای هر 16 مرحله این رابطه برقرار است :

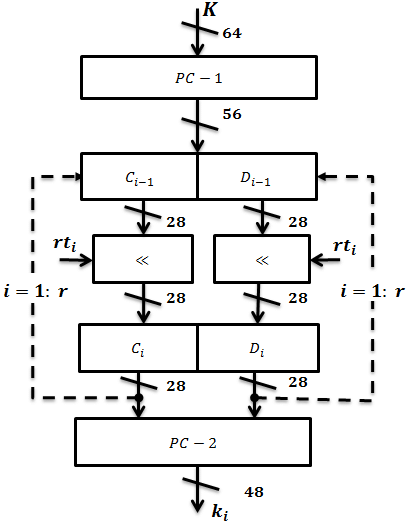
توی هر مرحله از ورودی نصفه (R0) ثابت می ماند و نصفه (L0) رمز می شود. 16 مرحله دارد پس 8 بار کل متن رمزگذاری می شود.

**عملکرد تابع f()**

به صورت کلی 32 بیت را به 48 بیت و مجدد به 32 بیت تبدیل می کند، بدون اینکه چیزی از بین رفته باشد و ترتیب را بهم می ریزد. شاید دلیل استفاده از این جایگشت ها، کند کردن پیاده سازی نرم افزاری در برابر سخت افزاری باشد.

1. برای اینکه بتوان 32 بیت را با 48 بیت کلید XOR کرد، از جایگشت extension استفاده می شود. برای تولید بیت، طبق یک الگو از بیت های اصلی به صورت تکراری استفاده می شود و به ازای 32 بیت ورودی 48 بیت خروجی تحویل می دهد.
2. با کلید مرحله XOR می شود.
3. به 8 قسمت 6 بیتی تقسیم شده و به 8 جعبه جانشانی متمایز (S-box) برای تبدیل 6 بیت به 4 بیت فرستاده می شود. در مجموع می شود 32 بیت.
4. در نهایت جایگشت P اعمال شده و نتیجه می شود خروجی تابع f().

پس تابع f() نیمه سمت راست را از مرحله قبل و کلید را می گیرد، خروجی ای تولید می کند که این خروجی برای رمزگذاری نیمه سمت چپ بکار می رود.

**زمان بندی کلید**

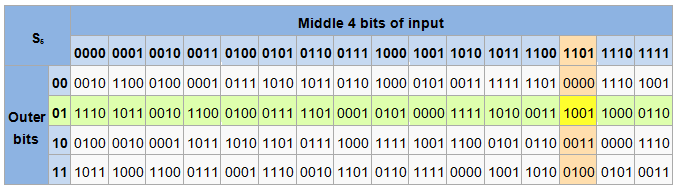
1. کلید ورودی دریافت شده و یک جایگشت اولیه کلید (PC-1) روی آن اعمال می شود. با از بین بردن بیت های پریتی، 64 بیت ورودی تبدیل می شود به 56 بیت در خروجی.
2. کلید به دو نیمه 28 بیتی C0 و D0 تقسیم می شود.
3. مراحل تبدیل و عملیات شیفت، نیمه ها به صورت مستقل به سمت چپ شیفت می خورند. در مراحل 1و2و9و16 یک بیت و در بقیه مراحل دو بیت شیفت به سمت چپ داده می شود.

چرا مهم است؟ چون 4 تا یک شیفتی و 12 تا دو شیفت داریم که مجموعشان می شود 28 تا، هم طول نیمه ها.

نکته : C0 با C16 و D0 با D16 برابر است. چون عملکرد الگوریتم رمزگشایی دقیقا مثل رمزگذاری است و فقط ترتیب کلید ها عوض می شود. پس در رمزگشایی شیفت به جای سمت چپ، به سمت راست داده می شود. به این طریق می توان کلید های مرحله را بدست آورد.

1. بعد جایگشت ثانویه کلید (PC-2) روی آن اعمال می شود و 56 بیت می شود 48 بیت. پس تعدادی از بیت ها از بین می رود.

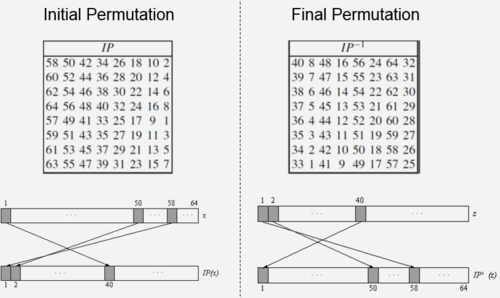
**نحوه عملکرد S-box ها**

به صورت استاندارد 8 عدد وجود دارد. ورودی آن یک 6 بیتی و خروجی یک 4 بیتی است. بر مبنای این 6 بیت تعیین می شود خروجی چه خواهد بود. طبق قاعده :

1. 2 بیت در طرفین و 4 بیت در وسط درنظر گرفته می شود.
2. با 2 بیت طرفین شماره سطر و با 4 بیت وسط شماره ستون مشخص می شود. در 2 بیت 4 سطر و 4 حالت و توی 4 بیت 24 حالت یعنی 16 ستون خواهد بود.

**جابجایی دو نیمه نهایی**

بعد از اینکه 16 مرحله تمام شد، جای خروجی های مرحله شانزدهم (R16 و L16) عوض می شود.

**اعمال جایگشت نهایی IP-1**

سپس جایگشت نهایی اعمال می شود (IP-1).

به عنوان مثال بیت 1 به خانه 40 رفته است، پس اگر بخواهیم سر جای خود برگردد، باید مقدار با اندیس 40 سر جای آن و در خانه 1 قرار گیرد.

**نحوه رمزگشایی DES**

الگوریتم رمزگشایی با DES-1 نمایش داده می شود. برای رمزگشایی باید 16 مرحله برعکس شده و کلید ها نیز برعکس استفاده شوند. اما اگر طبق الگوریتم تولید کلید به ترتیب، کلید مرحله یک، بعد کلید مرحله دو و الی آخر در حافظه ذخیره شده و به ترتیب عکس استفاده شود.انجام این کار نیازمند رجیستر است که هزینه بر است و هزینه پیاده سازی زیاد می شود.

اگر C0 و D0 با C16 و D16 یکسان باشند، به غیر از مرحله اول، در بقیه حالت ها شیفت به سمت راست انجام می شود و ابتدا K16 و درنهایت K1 را خواهیم داشت. یعنی به صورت خودکار ترتیب کلید ها برعکس شد.

* برای رمزگذاری : شیفت ها را به سمت چپ اعمال می کند و برای مرحله اول هم یک واحد شیفت اعمال می شود.
* برای رمزگشایی : مرحله اول شیفتی انجام نمی شود و بقیه مراحل، مثل مرحله رمزگذاری ولی شیفت به سمت راست انجام می شود. اینجوری کلید ها به ترتیب عکس تولید می شوند.

پس، الگوریتم رمزگشا دقیقا مانند الگوریتم رمزگذار است، و به همین دلیل جای دو نیمه در آخرین مرحله عوض می شود. در طی اینکه نیمه ها رمزگشایی و رمزگذاری می شود، برای ترتیب برعکس آماده سازی می شوند.

دلیل اینکه جایگشت و عکس آن اعمال می شود، متنی که به ورودی داده و خروجی شده است، اگر دوباره به عنوان ورودی داده شود، این دو جایگشت یکدیگر را نقض (خنثی) می کنند. جایگشت در معکوس خودش ضرب شده، اعمال شده و انگاری که این جایگشت اصلا اعمال نشده باشد.

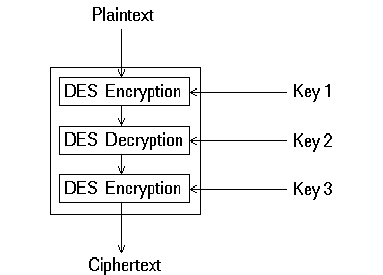
**امنیت رمز DES**

امنیت این رمز از امنیت 264 جستجوی فراگیر 264 نیست بلکه معادل با 256 است و از یک کلید 56 بیتی استفاده می کند. از این 8 بایت کلید، بیت با کمترین ارزش داخل هر بایت به عنوان بیت پریتی (بیت زوجیت) درنظر گرفته می شود.

2 مورد نگرانی برای امنیت این رمز وجود دارد :

1. این محرمانه ماندن منجر به این فکر شده که ممکن است یک حمله تحلیلی وجود داشته باشد که فقط سازندگان و طراحان رمز از آن باخبر هستند و دیگر افراد از آن اطلاعی ندارند.
2. طول کلید کوتاه است. این طول کلید کوتاه منجر می شود که حملات جستجوی فراگیر بتوانند علیه این رمز اعمال بشوند. الان کوتاه است ولی و در زمان خودش مقاومت مناسبی داشته بود. و تا کنون هیچ حمله تحلیلی علیه آن تا به امروز عمومی کشف نشده است. دلیل این موفقیت ها، پیشرفت قدرت محاسباتی بوده است. اما نسخه های تصحیح شده از اون و نسخه های قدرتمندتری از اون نظیر DESX ، Triple-DES و مانند اون استاندارد هستند و مورد استفاده قرار میگیرند.

## **2.2. استاندارد رمزگذاری داده سه تایی (3DES)\_ Triple Data Encryption Standard**

ایرادات رمز DES :

1. یک پیاده سازی سخت افزاری است همراه با هزینه تجهیزات فیزیکی.
2. طول بلاک کوچک.
3. طول کلید کوچک که منجر می شود حملات جستجوی فراگیر راحت علیه آن انجام شود.

برای رفع آن، متن ورودی چند بار رمز می شود. ابتدا با کلید k1، بعد با کلیدk2 و سپس با کلید k3 رمز می شود. این الگوریتم 3DES نام دارد و با سه کلید 64 بیتی که جمعا می شود 192 بیت. طول کلید افزایش یافت، پس امنیت بهتری دارد (با هزینه سه بار رمزگذاری). برای رمزگشایی باید عکس این عملیات را انجام داد. ابتدا DES-1 برای k3 بعد برای k2 و بعد برای k1 حساب شود.

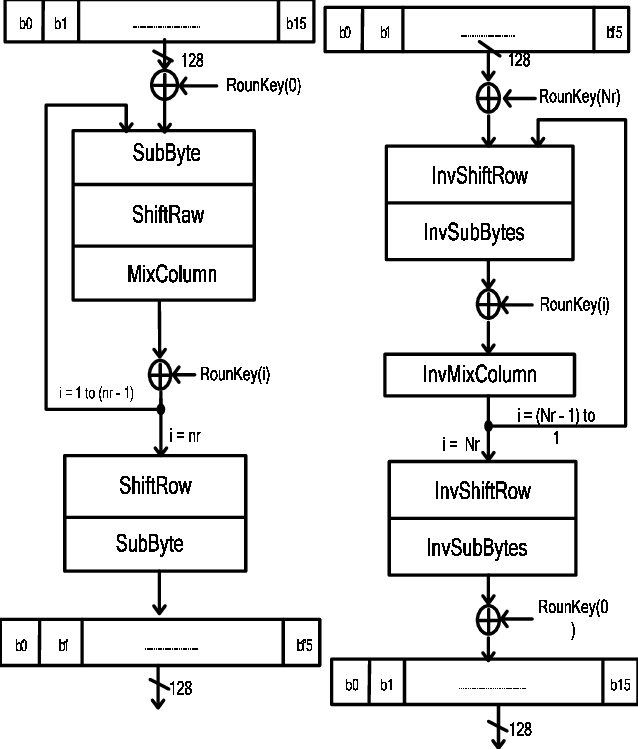
در سیستم های قدیمی به دلیل استفاده از DES ، برای استفاده از 3DES به این شیوه طراحی شده است :

* اگر k1 = k2باشد، آنوقت 3DES همان DES خواهد بود. انگاری که فقط یکبار با k3 رمز شده است.
* اگر k2 = k3 باشد، انگار هیچ کاری انجام نشده است.

**حمله ملاقات در وسط برای 3DES**

در حمله 3DES طول کلید به 3×56 افزایش پیدا می کند، اما حمله ملاقات در وسط را داریم. برای حرکت از سمت y ها، باید رمزگذاری با k2 انجام شود و عملیات ها برعکس هستند. در نتیجه در رمز 3DES طول کلید می شود 3×56 و طول کلید موثر 2×56.

## **3.2. استاندارد رمزگذاری پیشرفته AES) یاRejndael ) \_ Advanced Encryption Standard**

طول بلاک و خروجی آن 128 بیت است و از کلید های 128، 192 و 256 بیتی پشتیبانی می کند که به ترتیب برای امنیت کوتاه مدت، میان مدت و بلند مدت از آن ها استفاده می شود. این ساختار مبتنی بر ساختارهای فایستل نیست و از ریاضیات قدرتمندی به اسم میدان های گالوا استفاده می کند. از نظر ماهیت با رمز DES متفاوت است و براساس تکرار یک تعداد گام متن اصلی را رمزگذاری و به متن رمزشده معادل تبدیل می کند. در هر تکرار همه متن رمزگذاری می شود.

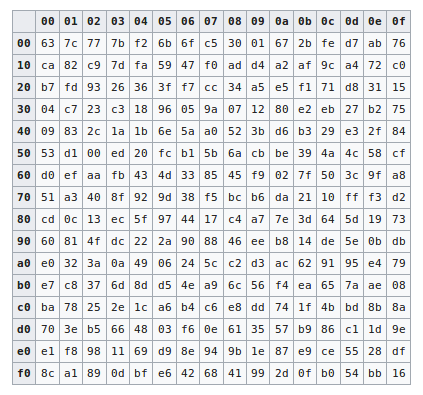
**روند کلی عملکرد رمز AES**

مانند DES یک الگوریتم تکراری است. تعداد تکرار های آن براساس طول کلید مشخص می شود.تعداد مراحل تکرار با پارامتر nr نشان داده می شود و بسته به اندازه و طول کلید، تعداد مراحل مورد استفاده در الگوریتم AES متفاوت است و تغییر می کند. در همه مراحل یک ساختار یکسان اعمال می شود غیر از آخرین مرحله که در آن از لایه ترکیب ستونی استفاده نمی شود.

* برای کلید 128 بیتی = 10 مرحله تکرار
* برای کلید 192 بیتی = 12 مرحله تکرار
* برای کلید 256 بیتی = 14 مرحله تکرار

در AES چهار عملیات پایه داریم :

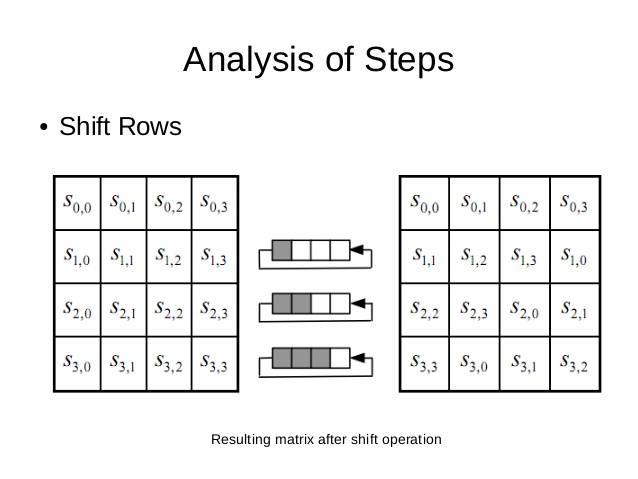
1. جانشینی بایتی (Byte Substitution) : با استفاده از ساختار های S-box انجام می شود ( از یک S-box واحد استفاده می شود).
2. شیفت سطری (Row shift) : سطر ها را به صورت بایت به بایت شیفت می دهد.
3. ترکیب ستونی (Column Mix) : ستون ها را ترکیب می کند و برای آن از میدان های گالوا استفاده می شود.
4. اضافه کردن کلید (Key Addition) : به تعداد مراحل، طبق زمان بند کلید، کلید مرحله تولید می شود.

**1. جانشینی بایتی (Substitution Byte)**

در رمز AES کار با بایت ها انجام می شود، پس هرکدام از مربع ها یک بایت است. 16 بایت هست که می شود 128 بیت.از ساختاری به اسم S-box استفاده می شود. برخلاف S-box های DES، در اینجا فقط یک دانه وجود دارد که 8 بیت ورودی می گیرد و 8 بیت خروجی می دهد.

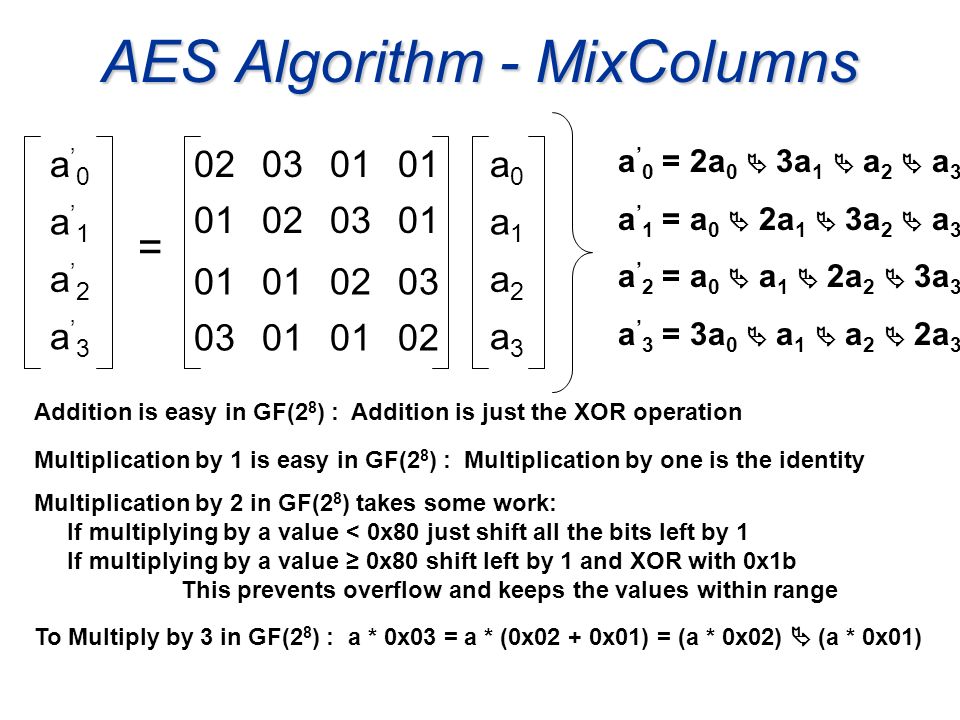
در S-box بایت به دو نصفه 4 بیتی x و y تقسیم و به صورت یک جدول نشان داده می شود. در این جدول مقدار تکراری وجود ندارد، یعنی یک تابع دوسویی است (هم یک به یک و هم پوشاست).

**2. شیفت سطری (Shift Row)**

16 بایت را به صورت ستونی پشت سرهم نوشته و به صورت یک ماتریس 4×4 در نظر گرفته، سپس به صورت چرخشی به سمت چپ شیفت داده و به صورت ستونی در C ها قرار داده می شوند.

* سطر اول صفر بایت شیفت به سمت چپ.
* سطر دوم یک بایت شیفت به سمت چپ.
* سطر سوم دو بایت شیفت به سمت چپ.
* سطر چهارم سه بایت شیفت به سمت چپ.

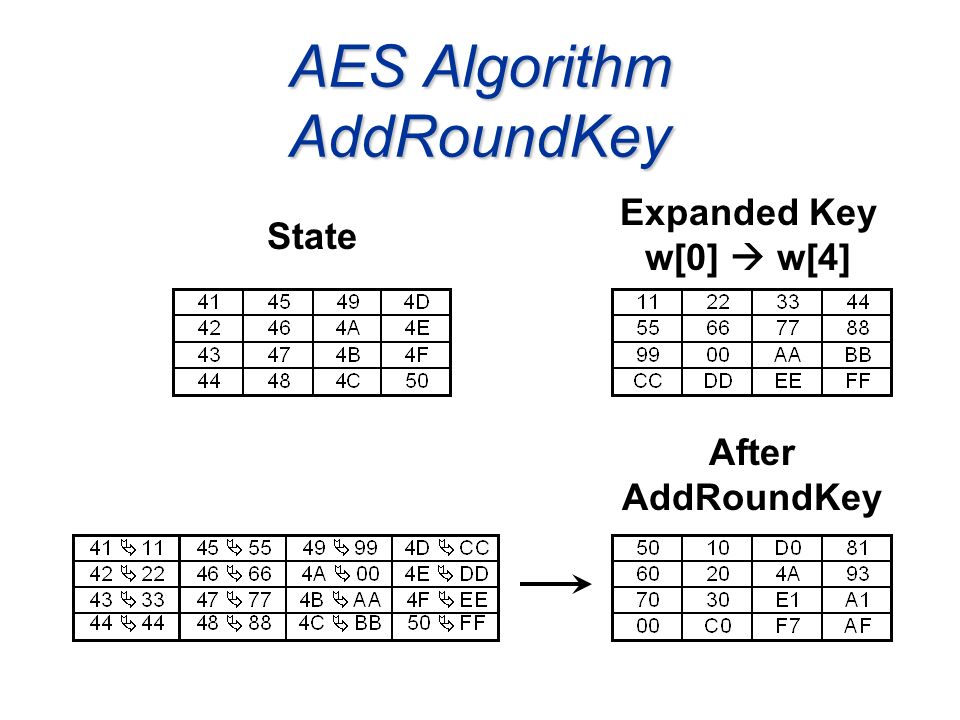
**3. ترکیب ستونی(Mix Column)**

ابتدا بایت های C به شیوه ستونی داخل یک ماتریس چیده می شود. بعد یک ماتریس به اسم D درنظر گرفته که شامل Di تا Di+3 می شود. این چهار سطر حاصل ضرب یک ماتریس در یک بردار هستند. بردار شامل Ci تا Ci+3 است. یعنی هرکدام از این ستون های C را در یک ماتریس ضرب کرده و یک ستون بدست میاد. این ستون ماتریس D است. به دلیل کار با بایت، باید درایه های ماتریس هم به صورت بایت باشند. پس به صورت اعداد هگز نوشته می شوند. هر سطر حاصل یک شیفت به سمت راست سطر قبلی خود است.مقادیر i برابر با 0,4,8,12 خواهد بود. این ضرب ها در میدان های گالوا انجام می شود. میدان های گالوا اساس و پایه رمز های مدرن هستند. حتی رمز های کوانتومی بر مبنای گالوا ساخته می شوند.

**میدان های گالوا**

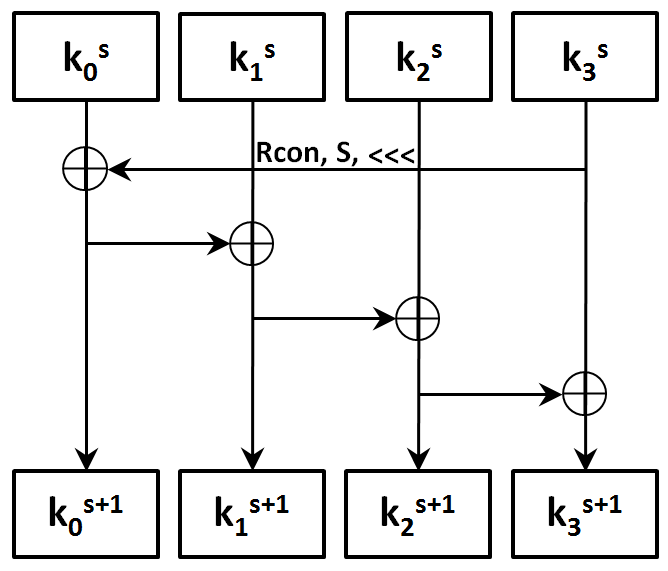
یک بایت شامل 8 بیت a7,…,a0 است. متناظر می شود با چند جمله ای x7,…,x0 است. از x فقط برای نمایش استفاده می شود و مقداری نمی گیرد.

ai ها مقادیر 0 و 1 گرفته و با z2 نشان داده می شود. به همه این چند جمله ای ها Galois Field با ضرایب به پیمانه 2 گفته می شود. یعنی مد 2 گرفته و خروجی 0 یا 1 است. تعداد بیت ها 8 عدد است، پس میدان می شود : GF(28) A(x) = a7x7 + … a0x0

**4. اضافه کردن کلید (Key Addition)**

هر تعداد تکرار در AES داشته باشیم، نیاز داریم که یک کلید بیشتر تولید کنیم چون یک تبدیل لایه صفر داریم. یعنی متن ورودی به محض اینکه وارد می شود با کلید مرحله صفرم XOR بشود، بعد وارد تکرار اول و الی آخر می شود. در هر لایه، متن با کلید مرحله XOR می شود. اگر طول کلید 128 بیت باشد، با 10 مرحله عملیات نیازمند 11 کلید فرعی، برای 192 بیتی با 12 مرحله نیازمند 13 کلید و برای 256 با 14 مرحله نیازمند 15 کلید مرحله است.

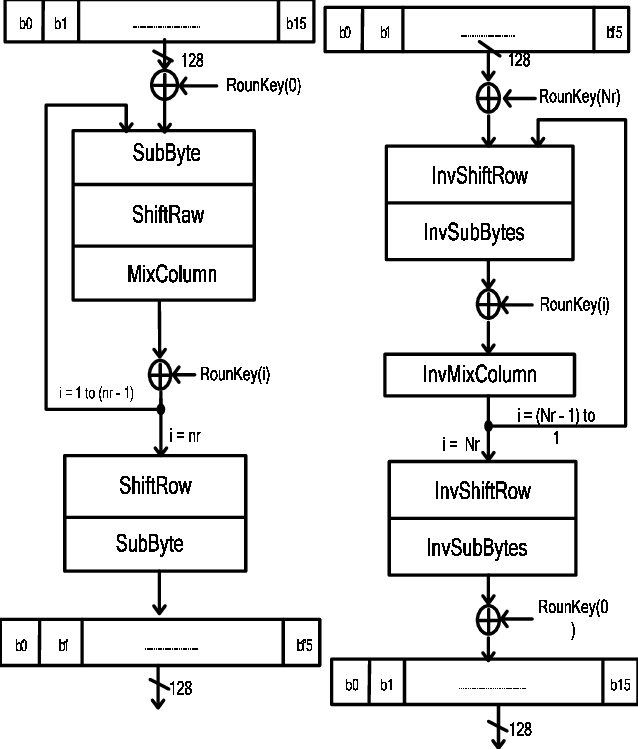
**الگوریتم زمان بند کلید (Key scheduling)**

**128 بیتی :** از کلمه (word) استفاده می کند. بسته به معماری سیستم می تواند 32 یا 64 بیتی باشد. در ابتدا چک می کند چند کلید نیاز است. برای 128 بیتی 11 کلید، و برای 11 کلید، 44 کلمه (word) 32 بیتی نیاز است. پس از w[0] تا w[43] را تولید می کند. کلید مرحله اول یعنی w[0] تا w[3] دقیقا برابر با کلید اصلی است. پس 10 کلید تولید می شود و این به عنوان کلید اضافه در نظر گرفته شده است. برای اینکه w[4] تا w[7] ساخته شود، از یک الگو استفاده می شود. مقدار w[3] به تابع g() داده می شود. بعد مقدار خروجی آن با w[0] XOR می شود و w[4] بدست می آید.

w[4] = w[0]⊕ g(w[3]) و این روال در هر سطر برای کلید مرحله تکرار می شود.

**192 بیتی :** برای 192 بیت 52 کلمه (word) درنظر گرفته می شود. طول کلید ورودی 192 بیت است که 6 کلمه (word) 32 بیتی می شود. اما برای کلید مرحله 128 بیت و 4 کلمه (word) کافی است. از w[0] تا w[51] یعنی 52 کلمه ساخته می شود و بعد 4 تا 4 تا به عنوان کلید های مرحله تعیین می شود. مقدار w[5] را به عنوان ورودی به تابع g() داده و خروجی با w[0]، XOR می شود و این روال ادامه پیدا می کند.

**256 بیتی :** برای 256 بیتی نیز به همین صورت است. منتهی اگر از w[7] برای XOR استفاده شود، ممکن است پریشان بودن به خوبی رخ ندهد. به همین دلیل از تابع h() برای پریشان تر شدن استفاده و درنهایت 60 کلمه (word) تولید می شود و در آخرین تکرار از h استفاده نمی شود. در اینجا 14 مرحله وجود دارد، پس 15 کلید فرعی ساخته می شود.

**رمزگشایی AES**

برای رمزگشایی لازم است عکس عملیات رمزگذاری انجام شود و هر مرحله از الگوریتم AES برعکس شود. یعنی باید برعکس لایه اضافه کردن کلید، برعکس لایه ترکیب ستونی، برعکس لایه انتقال سطر و برعکس لایه جانشینی بایتی انجام شود. و اینارو هم از پایین به بالا انجام بدیم یعنی باید ترتیب تولید زمانبند کلید برعکس شود (کلید های تولیدی را نگه داشته و به صورت عکس استفاده شوند).

1. **عکس عملیات ترکیب ستونی**

برای عکس شدن عملیات، باید در ماتریس معکوس ضرب شود. هر سطری در ستون که ضرب می شود، سطر اول در ستون اول ضرب می شود مقدار آن می شود یک به پیمانه چند جمله ای تحویل ناپذیر مورد استفاده در AES.

1. **عکس عملیات انتقال سطر**

به صورت بایت نوشته شده و سطرها به صورت دورانی شیفت داده می شوند.

* + سطر اول صفر بایت شیفت به سمت راست.
  + سطر دوم یک بایت شیفت به سمت راست.
  + سطر سوم دو بایت شیفت به سمت راست.
  + سطر چهارم سه بایت شیفت به سمت راست.

1. **عکس عملیات جانشینی بایتی**

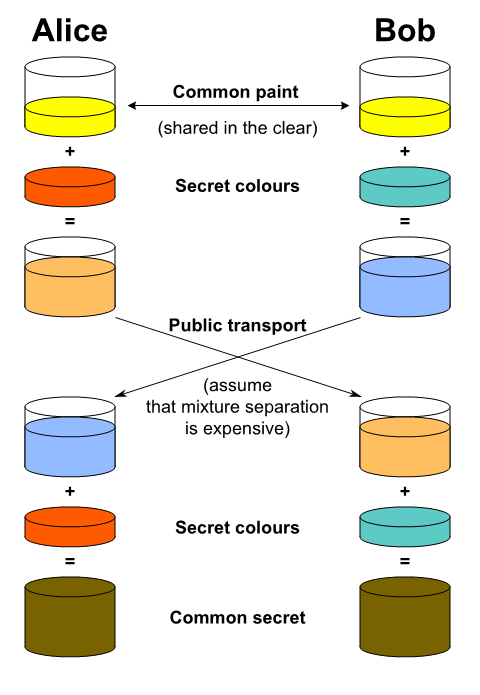
برای برعکس کردن کافیه که از S معکوس استفاده شود تا از ورودی به خروجی برود.

**امنیت رمز AES**

امنیت این رمز 2 به توان طول کلید است، یعنی 2128 یا 2192 یا 2256 . بعدا یک الگوریتم جدید ساخته شد که پیچیدگی عملیاتی آن شد 2126/2 و 2190 و 2254. پس این جستجوی فراگیر تهدیدی علیه امنیت رمز AES محسوب نمی شود. رمز AES دارای امنیت CPA نیست. برای اینکه رمز امنیت CPA داشته باشد، الگوریتم باید تصادفی عمل کند اما AES یک رمز قطعی است.

# **الگوریتم های رمزنگاری مدرن با کلید نامتقارن**

## **1. الگوریتم رمزنگاری Diffie–Hellman**

یکی از مسائل قابل توجه در رمزنگاری نحوه توزیع کلید می باشد. کلید های ایجاد شده می بایست به نحوی بین مبدا و مقصد رد و بدل شوند و این دسترسی می بایست به نحوی انجام شود که کلید مورد نظر توسط افراد بدون صلاحیت قابل دسترسی و بهره برداری نباشد. نحوه ارسال کلید نیز میتواند به نحوی توسط فرایند رمزنگاری و یا انجام برخی از محاسبات انجام پذیرد. الگوریتم دیفی هلمن از این قاعده مستثنی نبوده و با انجام محاسبات لازم ارسال کلید بین مبدا و مقصد را امکان پذیر می سازد.

این الگوریتم توسط Whitefield Diffie و Martin Hellman در سال 1976 ابداع شد. این الگوریتم میتواند برای ارسال کلید از یک کانال عمومی استفاده کند که نیاز به هیچگونه محدودیتی در استفاده از آن نیست و این کانال میتواند برای عموم قابل دسترس باشد. این کانال برای عموم قابل شنود بوده و انتقال کلید براساس انجام محاسبات بین مبدا و مقصد می باشد.

در مورد این الگوریتم میتوان گفت :

1 - کلید های عمومی و اختصاصی کاربران با زوج (P,S) شناخته میشود که میتواند موقتی باشد.

2 - برای رمزنگاری از تابع F استفاده میگردد که به ازای دو ورودی x,y خروجی مورد نظر را به صورت F(x,y) تولید کند.

ساختار کلی رمزنگاری در الگوریتم دیفی هلمن به شرح زیر است :

1. آلیس کلید عمومی خود را برای باب ارسال میکند.
2. باب کلید عمومی خود را برای الیس ارسال میکند.
3. آلیس تابع F(Sa,Pa) را محاسبه میکند. الیس زمانی میتواند این کار را انجام دهد که کلید اختصاصی Sa را در اختیار داشته باشد.
4. باب تابع F(Sb,Pb) را محاسبه میکند. باب زمانی میتواند این کار را انجام دهد که کلید اختصاصی Sb را در اختیار داشته باشد.

خصوصیت اساسی این سیستم رمزنگاری کلید عمومی را میتوان در تابع زیر عنوان نمود.

F(Sa,Pb) = F(Sb,Pa)

این خصوصیت موجب می شود بر اساس محاسبه انجام شده ,کلید های بدست امده برای طرفین دارای مقداری یکسان باشد.

**رمزنگاری**

الگوریتم رمزنگاری را میتوان به شرح زیر عنوان نمود :

1 - آلیس و باب بر روی عدد اول بزرگ p,g توافق میکنند که عدد p بسیار بزرگ خواهد بود. نیازی نیست که این دو عدد مخفی نگه داشته شوند.

2 - آلیس و باب دو عدد اول Xa , Xb را به صورت تصادفی انتخاب می کنند که کوچکتر از p می باشند. این دو عدد به عنوان کلید خصوصی می باید مخفی نگه داشته شوند.

3 - آلیس کلید قابل ارسال را با استفاده از فرمول Ya= (g^Xa) mod p و بطور مشابه باب کلید قابل ارسال را با استفاده از فرمول Yb= (g^Xb) mod p محاسبه میکند . اعداد بدست آمده از طریق یک کانال ناامن برای طرفین ارسال خواهد شد.

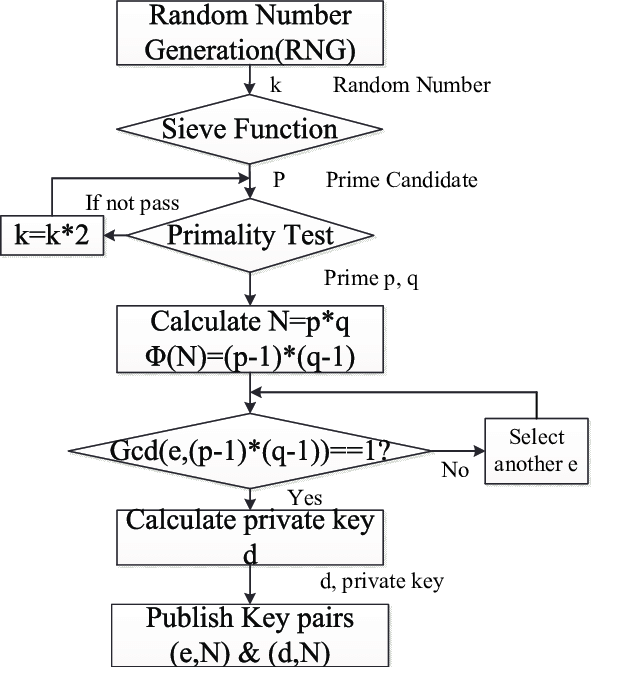
4 - آلیس کلید مورد نظر را با استفاده از فرمول Za= (Yb^Xa) mod p و به طور مشابه باب با استفاده از فرمول

Zb=(Ya^Xb) mod p کلید مورد نظر را استخراج میکند.

5 - Za = Zb کلید مشترک محسوب شده و برای عملیات رمزنگاری متقارن قابل استفاده می باشد.

(عملگر ^ به معنی توان رساندن می باشد)

## **2. الگوریتم RSA \_ Rivest Shamir Adleman**

در رمزگذاری سه تابع داریم :

* تابع تولید کلید
* تابع رمزگذار
* تابع رمزگشا

**تابع تولید کلید**

یک پارامتر ورودی دارد : تعداد بیت های اعدادی است که انتخاب شده و با m نشان داده می شود. این تعداد بیت ها در امنیت تاثیر دارد.

1. اعداد اول p و q را به صورت تصادفی انتخاب کرده که باید فرد و m بیتی باشند. چون اگر کوچک باشند امنیت کافی نیست.
2. مقدار n برابر است با p×q
3. Φ(n) برابر است با (p-1) × (q-1)
4. مقدار e به صورت تصادفی از مجموعه {2, …, Φ(n)-1} انتخاب می شود.
5. معکوس ضربی e در ZΦ(n) محاسبه و مقدار آن d در نظر گرفته می شود. برای محاسبه از الگوریتم اقلیدس تعمیم یافته استفاده می شود.
6. کلید رمزگذار (n,e) و رمزگشا (d) ساخته می شود.

برای امنیت قابل قبول حداقل باید تعداد بیت ها 4096 بیت باشد.

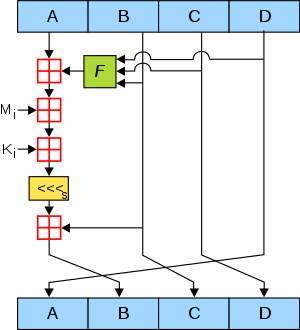
* الگوریتم رمزگذاری :
* الگوریتم رمزگشایی :

**دسته بندی حمله ها به رمز RSA**

* حملات ریاضی
  + تجزیه عدد به عوامل اول
* حمله مربوط به قرارداد
* حملات سومی عمدتا بخاطر ضعف در پیاده سازی، انتخاب مقادیر و برقراری ملزومات امنیتی هسته.
  + حمله شکل پذیری
  + قضیه باقی مانده چینی

# **الگوریتم های درهم سازی**

## **1. الگوریتم رمز MD5 \_ Message Digest 5**

الگوریتمی برای خلاصه سازی عبارات می باشد. یک عبارات با هر طولی می تواند به عنوان ورودی انتخاب شود و در خروجی خلاصه ای از اثر انگشت دیجیتالی آن را با طول 128 بیت در اختیار داشت. این الگوریتم در سال 1991 توسط پروفسور رونالد ریوست اختراع شد.

الگوریتم MD5 برای داشتن سرعت بالا در ماشین های 32 بیتی طراحی شده است در عین حال احتیاجی به جانشینی ها در جداول بزرگ ندارد. این الگوریتم را با کد های بسیار کمی می توان نوشت. الگوریتم MD5 توسعه ای از الگوریتم MD4 می باشد با این تفاوت که MD5 کمی کندتر از MD4 عمل می کند اما در طراحی آن بسیار محافظه کارانه عمل شده است.

MD5 به این دلیل طراحی شد که حس کردند MD4 به عنوان سرعت بالایی که داشت پذیرفته شده و از امنیت بالایی در شرایط بحرانی برخوردار نمی باشد. MD4 برای سرعت بالا طراحی شده ولی احتمال شکست آن در رمز کردنی موفق وجود دارد. MD5 کمی در سرعت کند شده با این تفاوت که بیشترین امنیت را داراست. این الگوریتم حاصل تاثیر دادن نظرات تعدادی از استفاده کنندگان MD4 به همراه مقادیری تغییر در ساختار الگوریتم برای افزایش سرعت و قدرت آن می باشد.

MD5 یک الگوریتم هش کردن (Hash function) می باشد. در الگوریتم های هش برای همه ی ورودی های با طول متفاوت، طول کد تولید شده ثابت و غیرقابل بازگشت می باشد. بنابراین در حقیقت روشن نیست که یک عبارت معادل یک رشته MD5 چیست. در MD5 نیز فایل های ورودی با هر اندازه و هر فرمتی که باشند، کد خروجی همیشه برابر با ۱۲۸ بیت (۱۶ بایت) خواهد بود که معمولا آن را در قالب ۳۲ عدد هگزادسیمال (مبنای ۱۶) نمایش می دهند.

همانطور که ذکر شد ویژگی دیگر این الگوریتم غیر قابل بازگشت بودن یا یک طرفه بودن کد تولید شده می باشد. یعنی نمی توان با استفاده از کد خروجی، ورودی الگوریتم را تشخیص داد. این ویژگی باعث شده است در موارد بسیاری از MD5 برای ذخیره سازی داده ها مانند ذخیره رمز های عبور استفاده شود.

کد های تولید شده توسط الگوریتم MD5 منحصر بفرد نیز می باشند. برای هر فایل ورودی، کد MD5 تولید شده یکتا خواهد بود و به همین دلیل آن را اثر انگشت فایل می نامند. به عنوان مثال اگر یک فایل را به عنوان ورودی به الگوریتم MD5 بدهیم، خروجی آن یک کد ۱۲۸ بیتی می باشد که با کد تولید شده برای هر فایل دیگری متفاوت می باشد و در صورتی که حتی یک بیت از فایل تغییر کند، کد MD5 آن تغییر عمده خواهد کرد و به همین دلیل است که در سایت های دانلود در کنار فایل، کد MD5 آن را نیز برای اطمینان از صحت فایل دانلود شده قرار می دهند. البته به طور مطلق نمی توان گفت که MD5 کد تکراری تولید نمی کند ولی احتمال تکراری بودن کد تولید شده بسیار کم است و می توان گفت صفر است.

**کاربرد MD5**

از آنجا که MD5 برای هر فایل یک اثر انگشت ایجاد می کند می توان از آن برای اطمینان از صحت فایل ها استفاده کرد. به عنوان مثال هنگامی که فایلی را از اینترنت دانلود می کنید اگر در سایت دانلود فایل، کد MD5 فایل هم قرار داده شده باشد می توانید MD5 فایل دانلود شده را با MD5 داده شده در سایت مقایسه کنید و در صورت برابر بودن مطمئن شوید که فایل، صحیح دانلود شده است و یا در بین راه به ویروس آلوده نشده است.

امروزه بحث امنیت یکی از اساسی ترین موضوعات در طراحی نرم افزارها و ذخیره اطلاعات کاربران به شمار می رود. گاها پیش می آید که نیاز به ذخیره سازی اطلاعات به صورت امن و غیرقابل بازگشت وجود دارد. در این موارد می توان اطلاعات را با استفاده از الگوریتم MD5 هش کرد و سپس آنها را در دیتابیس ذخیره نمود. فرض کنید در برنامه هایی که کلمه عبور شما را ذخیره می کنند، رمز عبور به همان صورتی که شما وارد کرده اید در دیتابیس ذخیره شود. در این صورت اگر فردی به دیتابیس دسترسی پیدا کند تمامی رمز های عبور کاربران را می تواند بدست آورد و با استفاده از آن هر اقدامی را انجام دهد.

حتی ممکن است کاربران برای حساب های خود رمزهای عبور مشترک قرار دهند و رمز عبور دزدیده شده، رمز حساب های دیگر وی نیز باشد. برای جلوگیری از این کار بسیاری از برنامه ها رمزهای عبور را ابتدا توسط الگوریتم MD5 هش می کنند و سپس در دیتابیس ذخیره می کنند. در این حالت حتی اگر فردی به دیتابیس دسترسی پیدا کند متوجه نخواهد شد رمز عبور شما چیست؟! شاید در اینجا این سوال مطرح شود که پس برنامه ها چگونه درستی رمز عبور وارد شده را تشخیص می دهند؟ هنگامی که کاربری قصد وارد شدن به برنامه را دارد، باید نام کاربری و رمز عبور خود را وارد نماید.

برنامه، رمز عبور وارد شده ی کاربر را با الگوریتم MD5، هش می کند و سپس مقدار هش شده ی رمز عبور وارد شده را با مقدار ذخیره شده در دیتابیس مقایسه می کند و در صورت برابر بودن بدین معنی است که کاربر کلمه عبور خود را صحیح وارد کرده است.

پس اگر برنامه ای که اطلاعات محرمانه شما را ذخیره می کند، بتواند به شما رمز عبورتان را بگوید آن برنامه به هیچ وجه قابل اطمینان نیست. برای اینکار می توانید از قسمت “فراموشی رمز عبور” در سایت هایی که عضو آن هستید استفاده نمایید. اگر رمز عبور شما به صورت هش شده در دیتابیس ذخیره شده باشد امکان بازیابی رمز عبور وجود ندارد و به همین دلیل لینکی برای تغییر کلمه ی عبور برای شما ارسال خواهد شد. اما اگر کلمه ی عبور برای شما ارسال شد مشخص خواهد شد که رمز عبور شما به صورت هش شده ذخیره نشده است.

هش کردن (Hash) روشی برای کد کردن داده با طول رشته ثابت هست که به صورت یکطرفه عملیات رمزنگاری را انجام می دهد و عملا روش متعارفی برای رمزگشایی آن وجود ندارد. البته روش های غیرمتعارف برای این کار وجود دارد. به عنوان مثال در برخی سایت ها دیتابیس بزرگی از رایجترین کلمات و عبارات و مقدار هش شده ی آنها وجود دارد. و در صورتی که مقدار هش شده ی عبارت رایجی را به آن بدهید، اگر در دیتابیس بیاید مقدار ورودی را به شما خواهد گفت.

**روشهای تولید MD5**

در زبان های مختلف برنامه نویسی تابع های متفاوتی در خصوص تولید MD5 از یک عبارت موجود است. همینطور برای سهولت کار نرم افزارها و برخی وب سایت ها ابزارهای در این زمینه ارائه کرده اند. یک ابزار ساده جهت تولید رشته های MD5 از عبارت دریافتی که به صورت آنلاین می باشد، “مولد MD5 کدباکس” است که با کمک آن به راحتی می توانید پیدا کنید.

**رمزگشایی MD5**

الگوریتم MD5 یکی از پرکاربردترین الگوریتم های هش کردن داده ها می باشد که می توان از آن برای اطمینان از صحت فایل ها و ذخیره سازی امن داده ها استفاده نمود. برنامه های بسیاری برای بدست آوردن کد MD5 یک فایل و یا یک متن وجود دارد. شما می توانید به صورت آنلاین در برخی سایت ها، کد MD5 داده ورودی خود را بدست آورید و یا از برنامه های معروفی مانند Md5sum، Md5Checker، WinMd5، WinMd5sum استفاده نمایید. البته ممکن است در تولید MD5 از الگوریتم های ترکیبی مثل چند مرحله ای یا استفاده از کلید اضافی استفاده شده باشد که در آن صورت نرم افزارهای فوق قادر نیستند پیدا کنند معادل رشته چیست.

## **2. خانواده SHA \_ Secure Hash Algorithm**

برای ساختن توابع درهم ساز، می توان از یکسری توابع انحصاری به اسم خانواده توابع SHA استفاده کرد.این خانواده به صورت اختصاصی برای پیاده سازی عملکرد درهم سازی طراحی شده اند.

**SHA-0**

اولین نسل آن را NIST طراحی کرد و معروف به SHA-0 هستند. شباهت زیادی با تابع درهم ساز MD4 دارند. اما در اصل ایراد هایی اساسی در این نسل وجود داشت که سبب ضعف این الگوریتم شده بود. در تعداد کمی از نرم افزار های امنیتی به کار می رود و کاربرد گسترده ای ندارد.

**SHA-1**

به دلیل امکان رخنه در نسل قبل، نسل دوم به اسم SHA-1 معرفی شد. خروجی آن یک دنباله 160 بیتی است.شباهت زیادی به نسل قبل و MD5 دارد که امروزه نیز استفاده می شود. در حال حاضر این نسل پرکاربرد ترین الگوریتم از این خانواده است و در بسیاری از نرم افزار های امنیتی استفاده می شود. در سال ۲۰۰۵ خطاهای امنیتی این الگوریتم در موضوع ریاضیات به کار رفته در آن تشخیص داده شد که نشان می داد ممکن است این الگوریتم شکسته شود. از آن زمان بود که نیاز به یک الگوریتم بهتر در این حوزه احساس شد. اگرچه هنوز این احتمال به واقعیت تبدیل نشده و هیچ گونه حمله ی موفق به این الگوریتم صورت نگرفته است.

**SHA-2**

مجموعه ای از توابع درهم سازی محسوب می شود SHA-224) SHA-256, SHA-384, (SHA-512,. این الگوریتم نسبت به نسل قبل تغییرات اساسی کرده است. این الگوریتم شامل 4 تابع درهم سازی است با چکیده پیام های 512, 384, 256, 224. اگرچه از لحاظ زیادی شبیه نسل قبل است ولی این دست از حملات ذکر شده برای آن پیش بینی نشده است. در حال حاضر به عنوان استاندارد امن در حال استفاده است اما به دلیل شباهت زیاد در طراحی با الگوریتم های شکسته شده دارد، بیم آن می رود که در آینده شکسته شود.

**SHA-3**

مؤسسه‌ ملی استاندارد و فناوری (NIST) در سال ۲۰۰۷ تصمیم گرفت رقابت SHA-3 را برای انتخاب استاندارد جدید درهم سازی برگزار کند. در اواخر سال ۲۰۱۱ و با پایان یافتن دور دوم رقابت، ۵ نامزد از بین ۱۴ نامزد به دور پایانی راه یافتند (BLAKE، Grøstl، JH، Keccak و Skein). سرانجام در دوم اکتبر ۲۰۱۲ و با ارزیابی های مختلفی که توسط محققان رمزنگاری در سراسر دنیا صورت گرفت الگوریتم Keccak به عنوان الگوریتم درهم سازی استاندارد معرفی شد.

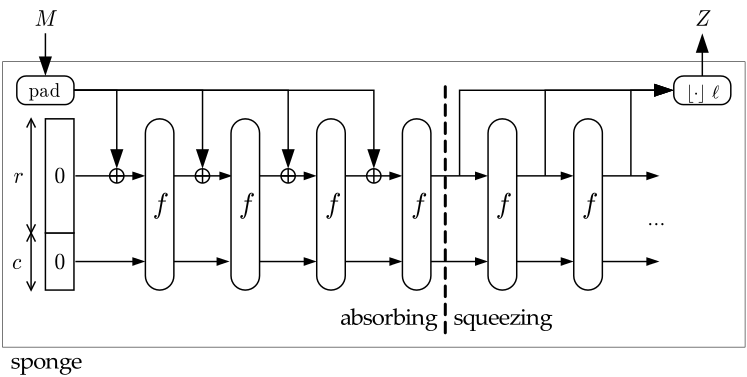
**امنیت خانواده SHA**

امنیت کاربردهای مهمی چون امضای دیجیتال شدیدا به امنیت توابع درهم ساز وابسته است. به همین دلیل وجود یک استاندارد امن برای الگوریتم های درهم سازی ضروری می باشد. در گذشته برخی از توابع درهم ساز مانند MD5 وSHA-1 در کاربردهای مختلف استفاده می شدند اما با شکسته شدن این الگوریتم ها دیگر نمی توان از آنها در کاربردهای حساس و حیاتی استفاده کرد. یک تابع درهم سازِ امن باید داری سه ویژگی مقاومت پیش تصویر، مقاومت پیش تصویر دوم و مقاومت در برابر برخورد باشد.

**از کدامیک استفاده کنیم؟**

مورد اول که هیچی. مورد دوم با جستجوی فراگیر 269 راحت شکسته میشه. در اصل 280 است ولی راهکاری ارائه شد تا جستجوی فراگیر را کاهش دهد. اکنون از SHA-2 استفاده می شود ولی بهتر است از SHA-3 استفاده شود. هر سه SHA از ساختار مرگل دن گارد استفاده می کنند. اما ساختار SHA-3 کاملا متفاوت و مثل اسفنج است.

## **3. الگوریتم(SHA-3) \_ KECCAK**

این تابع به جای مرگل دن گارد، از ساختار اسفنج استفاده می کند. به عنوان مثال تابع هش باید از تک تک قطعات متن یک خلاصه و چکیده به همراه داشته باشد و تحت تاثیر آن قرار بگیرد. اگر یک بیت در متن ورودی تغییر کند، مقدار هش آن باید تغییر زیاد تغییر کند.

یک اسفنج خشک درنظر گرفته، متن را به عنوان یک مایع درنظر گرفته و داخل سطل هایی ریخته و تقسیم می شود. بعد اسفنج را در همه سطل ها فرو کرده و هم زده می شود، جوری که کل اسفنج در کل سطل قرار گیرد. اکنون اسفنج خلاصه ای از محتوای سطل است. بعد داخل سطل بعدی هم زده می شود و الی آخر. ساختار اسفنج به عنوان کلیدی ترین جزء الگوریتم SHA3 است. چکیده پیام ورودی را به صورت تکراری محاسبه می کند و طول ورودی و خروجی هم دلخواه است.