AES cryptography algorithm document

شرح رمز

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. استاندارد رمزنگاری پیشرفته گونه‌ای از Rijndael است که اندازه بلاک ثابت ۱۲۸ بیتی و اندازه کلید ۱۲۸، ۱۹۲ و ۲۵۶ بیتی دارد. در مقابل، مشخصه Rijndael *per se* الگوریتم با اندازه کلید و اندازه بلاکی تعیین می‌شود که می‌تواند هر ضریبی از ۳۲ بیت، با حداقل ۱۲۸ و حداکثر ۲۵۶ بیت باشد.

استاندارد رمزنگاری پیشرفته روی ماتریسی ۴*۴ از بایت‌ها با ترتیب ستونی، که *state* نامیده می‌شود، عمل می‌کند، اگرچه برخی نسخه‌های Rijndael اندازه بلاک بزرگتر و ستونهای بیشتری در state دارند. بیشترین محاسبات AES در یک finite field خاص انجام می‌گیرد.

اندازه کلید استفاده شده در رمز AES ، تعداد تکرارهای چرخه‌های تبدیل (transformation) را تعیین می‌کند که ورودی، با نام متن عادی (plaintext) را به خروجی نهایی با نام متن رمز شده (ciphertext) تبدیل می‌نماید. تعداد چرخه‌های تکرار به صورت زیر است:

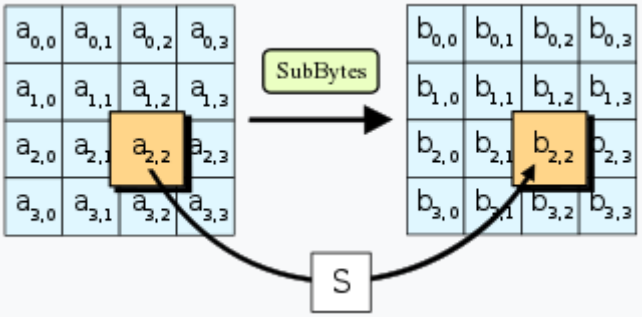
- ۱۰ چرخه تکرار برای کلیدهای ۱۲۸ بیتی.
- ۱۲ چرخه تکرار برای کلیدهای ۱۹۲ بیتی.
- ۱۴ چرخه تکرار برای کلیدهای ۲۵۶ بیتی.

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.

شرح کلی الگوریتم

1. بسط کلید - (KeyExpansion) کلیدهای چرخه از کلید رمز با استفاده از زمانبندی کلید Rijndael مشتق می‌شود.
2. چرخه اولیه
   1. - AddRoundKeyهر بایت از state با کلید چرخه توسط xor بیت به بیت ترکیب می‌شود.
3. چرخه‌ها
   1. -SubBytesمرحله جانشین سازی (substitution) غیر خطی که هر بایت با بایت دیگری بر اساس یک جدول جستجو (lookup table) جایگزین می‌شود.
   2. -ShiftRowsمرحله جابجاسازی (transposition) که هر سطر از stateبه صورت تکراری در چند مرحله معین شیفت می‌یابد.
   3. -MixColumnsفرایند در هم ریختن (mixing) ستون‌ها که روی ستون‌های state عمل می‌نماید و چهار بایت از هر ستون را ترکیب می‌نماید.
   4. AddRoundKey
4. مرحله آخر
   1. SubBytes
   2. ShiftRows
   3. AddRoundKey

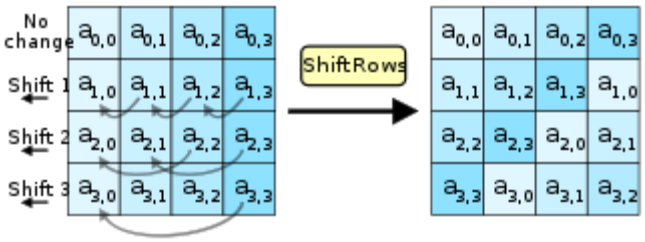
مرحله SubBytes



در SubBytes هر بایت در state با مقداری در جدول جستجو با ۸ بیت ثابت جایگزین می‌شود، $S;\ b_{ij} = S(a_{ij})$

در مرحله SubBytes، هر بایت در ماتریس *state* با استفاده از substitution box 8 بیتی Rijndael S-box با یک SubByte جایگزین می‌گردد S-box. استفاده شده از معکوس فزاینده (multiplicative inverse) روی $GF(2^8)$ 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.

مرحله ShiftRows

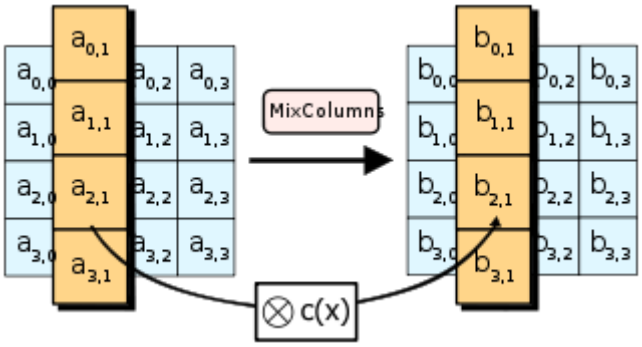


در مرحله **ShiftRows** بایت‌ها در هر سطر از **state** به صورت چرخشی شیفت داده می‌شوند. تعداد مکانهایی که هر بایت شیفت می‌یابد برای هر سطر متفاوت است.

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-1** 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.

مرحله MixColumns



در مرحله **MixColumns** هر ستون از **state** با یک چندجمله‌ای ***C(x)*** ضرب می‌شود.

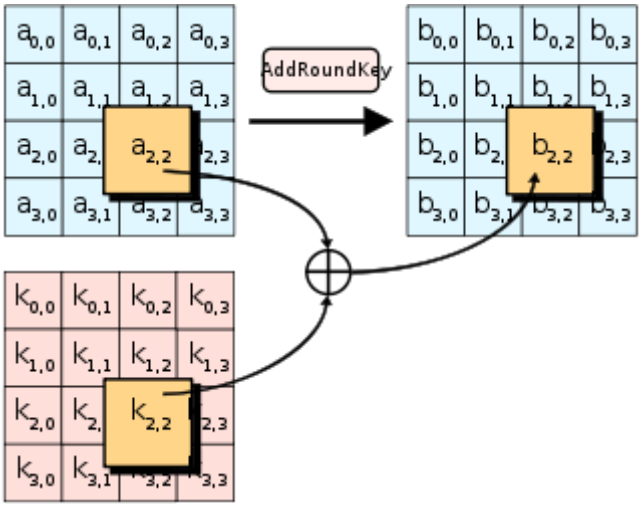
در مرحله **MixColumns** ، چهار بایت از هر ستون **state** با استفاده از تبدیل خطی معکوس ترکیب می‌شوند. تابع **MixColumns** چهار بایت را به عنوان ورودی در نظر می‌گیرد و چهار بایت را به خروجی می‌دهد، که هر بایت ورودی بر هر چهار بایت خروجی تأثیر می‌گذارد. به همراه **ShiftRows** ، مرحله **MixColumns** آشفتگی و پخش **(diffusion)** 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 ۰ **xFF**باشد، **XOR** شرطی با ۰ **x11B**باید انجام شود.

به صورت کلی تر، هر ستون به عنوان یک چند جمله‌ای روی **GF(2^8)** 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.

مرحله**AddRoundKey**



در مرحله **AddRoundKey**هر بایت از **stat** با یک بایت از **subkey** چرخه با استفاده از عمل **XOR (⊕)** ترکیب می‌شود.

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.

بهینه‌سازی رمز

در سیستم‌های با کلمات ۳۲ بیتی یا بزرگتر، این امکان وجود دارد که به وسیله ترکیب مراحل **SubBytes**و **ShiftRows** با مرحله **MixColumns** با تبدیل آنها به دنباله‌ای از جستجوهای‌های جدول، اجرای این رمزنگاری را سرعت بخشید. این امر نیازمند چهار جدول با ۲۵۶ مدخل ۳۲ بیتی و بهره‌گیری از چهار کیلوبایت (۴۰۹۶ کلمه) حافظه (یک کیلوبایت برای هر جدول) است. آنگاه یک چرخه می‌تواند با ۱۶ جستجوی جدول و ۱۲ عمل **OR** انحصاری ۳۲ بیتی و در ادامه با چهار عمل **OR** انحصاری ۳۲ بیتی در مرحله **AddRoundKey** انجام می‌شود .

اگر اندازه جدول چهار کیلوبایتی برای پلت فرم مقصد بزرگ است، جستجوی جدول می‌تواند با یک جدول با ۲۵۶ مدخل ۳۲ بیتی (یعنی ۱ کیلوبایت) با استفاده از تکرارهای چرخشی انجام گردد.

با استفاده از رویکرد بایت گرا، ترکیب مراحل**SubBytes** ، **ShiftRows**و **MixColumns**به یک چرخه تنها امکان‌پذیر است

امنیت

تا ماه مه ۲۰۰۹، تنها حملات منتشر شده موفق علیه **AES** کامل، حملات**Side-Channel**در برخی از پیاده‌سازی‌های خاص بود .آژانس امنیت ملی امریکا **(NSA)** همه **AES**های فینالیست، از جمله **Rijndael** را بازبینی کرد، و اظهار داشت که همه آنها برای اطلاعات غیر طبقه‌بندی شده دولت ایالات متحده به اندازه کافی امن است. در ماه ژوئن سال ۲۰۰۳، دولت ایالات متحده اعلام کرد که **AES** می‌تواند برای محافظت از اطلاعات طبقه‌بندی شده مورد استفاده قرار گیرد:

طراحی و قدرت تمام طول کلیدهای الگوریتم **AES** به عنوان مثال(۱۲۸، ۱۹۲ و ۲۵۶) برای محافظت از اطلاعات طبقه‌بندی شده تا سطح محزمانه 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

AESدارای 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 .

حملات شناخته شده

برای رمزنگاران، شکست **(break)** رمزنگاری هر چیزی است که سریع تر از انجام **brute force**(رمزگشایی آزمایشی برای هر یک کلید) باشد. حملات **brute force** با تکنولوژی فعلی نشدنی هستند. بزرگترین حمله موفقیت آمیز و به صورت عمومی شناخته شده **brute force**، در برابر هر گونه رمزنگاری **block-cipher** در برابر **RC5** با کلید ۶۴ بیتی به وسیله **distributed.net** در سال ۲۰۰۶ بود

AESاست شرح جبری نسبتاً ساده دارد در سال ۲۰۰۲، یک حمله نظری، با عنوان "حمله **XSL**"، توسط **Nicolas Courtois** و **Josef Pieprzyk** اعلام شد، که به نظر می‌رسید ضعفی را در الگوریتم **AES** به علت شرح ساده نشان می‌دهد از آن زمان به بعد، مقالات دیگر نشان داده‌اند که که حمله ارائه شده ناکارآمد است .حمله ایکس‌اس‌ال مطالعه شود.

در طول روند**AES** ، توسعه دهندگان الگوریتم‌های محاسباتی درباره **Rijndael** نوشتند، "... ما نگران استفاده از آن... در برنامه‌های کاربردی حساس امنیتی هستیم." با این حال، در ماه اکتبر سال ۲۰۰۰ و در پایان فرایند انتخاب**AES** ، **Bruce Schneier**، توسعه دهنده الگوریتم محاسباتی**Twofish** ، در حالی که فکر می‌کرد حملات موفق دانشگاهی روی **Rijndael**

روزی توسعه داده خواهد شد، نوشت: "من باور ندارم که هیچ‌کسی حمله‌ای را کشف کند که اجازه دهد کسی ترافیک **Rijndael** را بخواند"

در تاریخ ۱ ژوئیه ۲۰۰۹، **Bruce Schneier**، در وبلاگش در مورد یک حمله مرتبط با کلید در نسخه‌های ۱۹۲-بیتی و ۲۵۶ بیتی **AES** خبر داد، که توسط **Alex Biryukov** و **Dmitry Khovratovich**کشف شده بود که این حمله از زمانبندی کلید تا حدودی ساده **AES**استفاده کرده ودارای پیچیدگی 2^{119}است. در دسامبر ۲۰۰۹، این پیچیدگی به$2^{99.5}$بهبود یافته بود. این حمله دنباله حمله‌ای بودکه پیش از آن در سال ۲۰۰۹ توسط **Alex Biryukov**، **Dmitry Khovratovich**و **Nikolić Ivica** با یک پیچیدگی از 2^{96} برای یکی از هر2^{35}کلید بود

در تاریخ ۳۰ ژوئیه ۲۰۰۹ حمله دیگری در وبلاگ **Bruce Schneier** گزارش گردید و به عنوان یک نسخه پیش از چاپ در تاریخ ۳ اوت ۲۰۰۹ منتشر گردید .این حمله جدید، توسط **Alex Biryukov**، **Orr Dunkelman**، **Nathan Keller**، **Dmitry Khovratovich**و**Adi Shamir** ، روی **AES-256** است که با استفاده از تنها دو کلید مربوطه و زمان2^{39} برای بازیابی کلید ۲۵۶ بیتی در نسخه‌های ۹ چرخه‌ای، یا زمان2^{45} برای یک نسخه ۱۰ چرخه‌ای با نوعی قوی تر از حمله مرتبط با**subkey** ، یا زمان 2^{70}برای نسخه ۱۱ چرخه‌ای انجام می‌شود **AES 256** .بیتی از ۱۴ چرخه استفاده می‌نماید، بنابراین چنین حملاتی روی **AES** کامل مؤثر نیست.

در نوامبر ۲۰۰۹، اولین حمله تشخیص کلید روی نسخه‌ای از **AES-128** کاهش یافته به ۸ چرخه، به عنوان یک نسخه پیش از چاپ منتشر گردید .این حمله تشخیص کلید، نسخه‌ای بهبود یافته از حملات **rebound** یا **start-from-the-middle** برای جایگشت‌ها **(permutations)**مشابه **AES** است، که دو چرخه متوالی از جایگشت را به عنوان کاربردی از سوپر **Sbox** در نظر می‌گیرد. این حمله روی نسخه ۸ چرخه‌ای **AES-128** با پیچیدگی زمانی 2^{80}و یک پیچیدگی حافظه2^{32} عمل می‌کند.

در ژوئیه ۲۰۱۰ **Vincent Rijmen** مقاله طعنه آمیزی برای حملات**"chosen-key-relations-in-the-middle"** روی **AES-128** منتشر کرد

اولین حمله بازیابی کلید روی **AES** کامل توسط**Andrey Bogdanov** ، **Dmitry Khovratovich**و **Christian Rechberger** بود و در سال ۲۰۱۱ منتشر شد .حمله بر اساس **bicliques** بوده و با ضریب ۴ سریعتر از **brute force** است. برای **AES-192**و**AES-256** ، به ترتیب به $2^{189.7}$و$2^{254.4}$ عمل نیازمند است.

حملات غیر مستقیم

حملات غیرمستقیم به انگلیسی(**Side-Channel attacks**)به رمز مورد نظر حمله نمی‌کنند اما به پیاده‌سازی رمز بر روی سیستم‌هایی که سهواً اطلاعات را فاش می‌نمایند، حمله می‌کنند. چندین حملات شناخته شده روی پیاده‌سازی‌های خاصی از **AES** وجود دارد.

در آوریل سال ۲۰۰۵، **D.J. Bernstein**یک حمله **cache-timing** را اعلام کرد که او این حمله را برای شکستن یک سرور سفارشی که از رمزگذاری **AES** کتابخانه **OpenSSL** استفاده می‌نمود، بکار گرفت .این حمله نیاز به بیش از ۲۰۰ میلیون پیام رمز نشده داشت .سرور سفارشی به گونه‌ای طراحی شده بود که تا حد ممکن اطلاعات زمانی را انتشار می‌داد (سرور گزارشی از تعداد چرخه‌های انجام گرفته توسط عملیات رمزگذاری را باز می‌گرداند)؛ با این حال، همانگونه که **Bernstein** نشان داد، کاهش دقت مهرهای زمانی **(timestamp)** سرور، یا حذف آنها را از پاسخ سرور، این حمله را متوقف نمی‌نماید: مشتری به سادگی با استفاده از زمان رفت و برگشت بر اساس ساعت محلی خودش، استفاده نموده و اختلالات افزایش یافته را به وسیله میانگین گیری روی تعداد زیادی از نمونه‌ها جبران می‌نماید .

در اکتبر سال ۲۰۰۵، **Dag Arne Osvik**، **Adi Shamir**و **Tromer Eran** مقاله‌ای را ارائه دادند که چندین حمله **cache-timing** را روی **AES** نشان می‌داد یک حمله قادر به دست آوردن تمام کلید **AES** پس از ۸۰۰ عمل آغازسازی رمزگذاری، در مجموع

۶۵ میلی ثانیه، بود. این حمله نیازمند آن است که مهاجم قادر به اجرای برنامه‌هایی بر روی همان سیستم یا همان پلت فرمی که **AES** در حال انجام است، باشد.

در دسامبر ۲۰۰۹، حمله‌ای روی برخی از پیاده‌سازی‌های سخت‌افزاری منتشر شد که از تجزیه و تحلیل خطای تفاضلی **(differential fault analysis)** استفاده می‌کند و اجازه می‌دهد کلیدی با پیچیدگی از 2^{32}را بازیابی نمود

در نوامبر ۲۰۱۰ **Endre Bangerter** ، **David Gullasch** و **Stephan Krenn** مقاله‌ای را چاپ نمودند که روشی عملی برای بازیابی نزدیک به بی درنگ کلیدهای رمز-**AES-128**بدون نیاز به متن رمزنگاری یا متنی را توضیح می‌داد. این روش همچنین روی پیاده‌سازی‌های **AES-128** که جداول فشرده سازی را استفاده می‌نماید، مانند**OpenSSL** ، عمل می‌کند .همانند برخی از حملات قبلی، این حمله نیازمند این قابلیت است که بتواند روی سیستمی که رمزگذاری **AES** را انجام می‌دهد، اجرا شود، که اینکار می‌تواند توسط آلودگی به تروجان انجام گیرد.