



امنیت بیشتر با SSL

شرحی کامل بر این پروتکل امنیتی
هومن آتشبار ، حسین عسگری
تدوین : گروه امنیتی سیمرغ

Written By:

Hossein - Asgary

Security Magazine 
www.simorgh-ev.com

فهرست

- ۱- مقدمه‌ای بر رمزنگاری..... ۲
- ۲- پروتکل SSL..... ۵
- ۱-۲ پروتکل ارتباطی SSL..... ۶
- ۲-۲ اثبات هویت سرور..... ۷
- ۳- ایجاد کردن کلید و گواهی با استفاده از دستور openssl..... ۸
- ۱-۳ دستور genrsa..... ۹
- ۲-۳ دستور rsa..... ۹
- ۳-۳ دستور req..... ۱۰
- ۴-۳ دستور x509..... ۱۱
- ۴- استفاده از SSL برای امن کردن سرویس‌ها..... ۱۳

مقدمه‌ای بر رمزنگاری

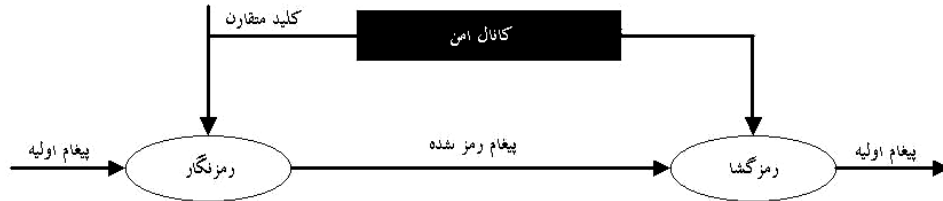
در شبکه‌های باز سه مشکل عمده در زمینه تبادل اطلاعات بین موجودیت‌ها خودنمایی می‌کند. این مشکلات عبارتند از محرمانگی داده‌ها، تمامیت داده‌ها و تایید هویت طرف‌های ارسال کننده و دریافت کننده. روش‌های رمزنگاری برای فائق آمدن بر این مشکلات طراحی شدند. دو روش رمزنگاری عمده عبارتند از:

- رمزنگاری متقارن^۱
- رمزنگاری نامتقارن^۲

در روش رمزنگاری متقارن کلید رمزنگاری و کلید رمزگشایی هر دو یکسان هستند یا به‌سهولت از روی هم قابل محاسبه هستند. اولین مشکل این روش تبادل کلید است که باید از طریق یک کانال امن صورت گیرد. مشکل دوم آن است که هر دو موجودیت باید یک کلید مشترک باهم داشته باشند. مشکل سوم نیز سختی ورود موجودیت‌های جدید به سیستم می‌باشد. آزمایش‌های این روش سهولت پیاده‌سازی و سرعت بالای آن را می‌توان نام برد.

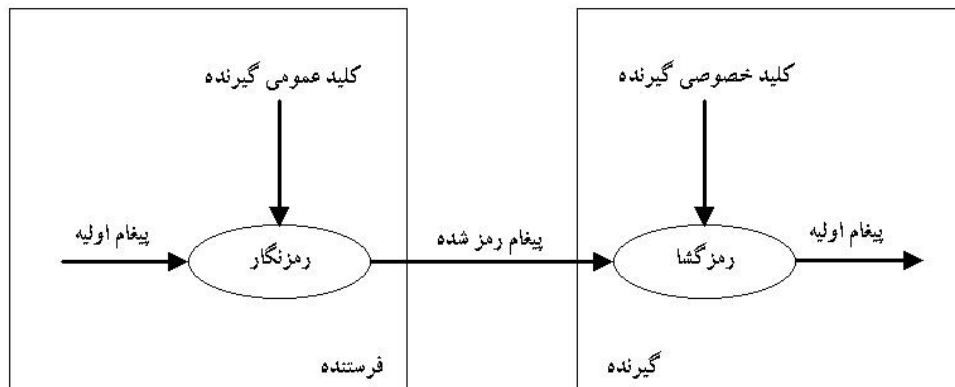
¹ Symmetric Cryptography

² Asymmetric Cryptography



شکل ۱- رمزنگاری متقارن

در روش رمزنگاری نامتقارن هر موجودیتی دو کلید مرتبط به هم با نام‌های کلید عمومی و کلید خصوصی دارد که به دست آوردن آنها از روی همدیگر به لحاظ محاسباتی تقریباً غیرممکن است. داده‌هایی که با یکی از این دو کلید رمز شود با کلید دیگر رمزگشایی می‌شود. کلید خصوصی محرمانه تلقی شده و نزد موجودیت می‌ماند اما کلید عمومی منتشر می‌شود. بنابراین در صورتی که دیگران بخواهند اطلاعاتی برایش ارسال کنند که فقط خود وی بتواند بخواند آن را با کلید عمومی‌اش رمز می‌کنند و می‌فرستند. اگر خود آن موجودیت بخواهد پیغامی را امضاء کند آن را با کلید خصوصی‌اش رمز می‌کند و دیگران از کلید عمومی متناظر آن برای باز کردن پیغام استفاده می‌کنند تا مطمئن شوند که پیغام از طرف او بوده است. مشکل اصلی این روش تطبیق کلید عمومی با موجودیت است؛ یعنی بتوان اطمینان حاصل کرد که K کلید عمومی موجودیت X است. برای حل این مشکل زیرساخت کلید عمومی ابداع شد.



شکل ۲- رمزنگاری نامتقارن

§ محرمانگی^۳

منظور از محرمانگی آن است که اطلاعات ردوبدل شده توسط موجودیت‌های غیرمجاز قابل فهم نباشد. محرمانگی از طریق رمز کردن اطلاعات ارسالی با یک کلید متقارن تصادفی به دست می‌آید الگوریتم‌های متقارن به لحاظ سرعت بیشتری که دارند در رمز کردن حجم‌های بزرگ اطلاعات مورد استفاده قرار می‌گیرند. کلید متقارن تصادفی نیز با کلید عمومی گیرنده رمز می‌شود و همراه اطلاعات فرستاده می‌شود. گیرنده ابتدا با استفاده از کلید خصوصی‌اش، کلید متقارن تصادفی را می‌یابد و سپس با استفاده از آن کلید اطلاعات را رمزگشایی می‌کند.

³ Confidentiality

§ تمامیت داده‌ها

منظور از تمامیت داده‌ها دریافت داده‌ها به همان صورت ارسال شده‌است. تمامیت با محاسبه مقدار Hash پیغام ارسالی و رمز کردن حاصل با کلید خصوصی فرستنده حاصل می‌شود (امضای دیجیتالی). خصوصیت تابع Hash آن است که پیغام‌های متفاوت، مقادیر Hash متفاوت دارند و پیدا کردن دو پیغام که دارای مقدار Hash یکسان باشند به لحاظ محاسباتی غیرممکن است. امضای دیجیتالی حاصل رمز کردن مقدار Hash با کلید خصوصی فرستنده است که به همراه پیغام اصلی فرستاده می‌شود. در سمت گیرنده برای آزمودن تمامیت داده‌های دریافتی مقدار Hash پیغام دریافتی را حساب می‌کنند و سپس امضای دیجیتالی را با کلید عمومی فرستنده باز می‌کنند و این دو را با هم مقایسه می‌کنند، در صورت تساوی تمامیت داده‌ها احراز می‌شود.

§ تصدیق پیام و موجودیت

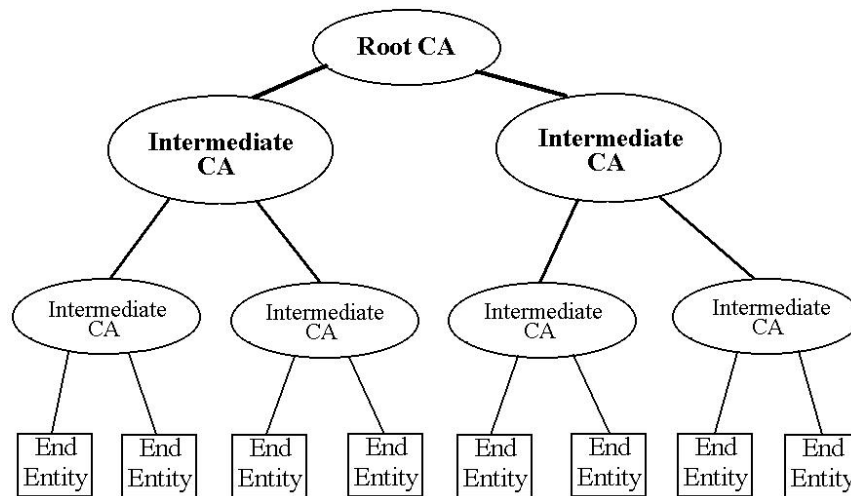
تصدیق موجودیت بر اساس میزان اطمینان از مالکیت انحصاری یک کلید خصوصی توسط آن موجودیت که توسط یک گواهینامه دیجیتالی^۴ مشخص می‌شود و میزان اعتبار گواهینامه مربوطه انجام می‌شود. درحالی‌که در اکثر سیستم‌های جاری تصدیق موجودیت براساس کد کاربری و کلمه عبور انجام می‌شود گواهینامه‌ها راه جدیدی برای تصدیق موجودیت پیشنهاد می‌کند که اساس آن داشتن کلید خصوصی مرتبط با کلید عمومی موجود در آنهاست. در عمل معمولاً ترکیب دو روش ذکر شده استفاده می‌شود. تصدیق پیغام با استفاده از آزمون تمامیت داده‌ها انجام می‌شود. در واقع تصدیق پیغام و تمامیت داده‌ها لازم و ملزوم یکدیگرند. تمامیت داده‌ها در صورتی که مبدا پیغام تصدیق نشود ارزش چندانی ندارد و تصدیق مبدا پیغام ارزش ندارد اگر تمامیت داده‌ها حفظ نشود.

§ سرویس گر گواهینامه^۵

سرویس گر گواهینامه وظیفه صادر کردن، مدیریت و فسخ کردن گواهینامه را برعهده دارد. سایر موجودیت‌های دخیل باید گواهینامه این سرویس گر را به خوبی بشناسند (کلید عمومی اش را بدانند). این سرویس گر می‌تواند وظایف خود را با صادر کردن گواهینامه‌های مخصوص سرویس گرهای گواهینامه به موجودیت‌های دیگری در سطح پایین تر محول کند و به این ترتیب سلسله مراتب سرویس گر گواهینامه را تشکیل دهد. هدف از این کار مدیریت ساده‌تر (پیاده‌سازی سیاست‌های مختلف) و کارآیی بیشتر می‌باشد. دنباله مرتب گواهینامه‌ها از آخرین شعبه تا سرویس گر ریشه را زنجیره گواهینامه می‌نامند. هر گواهینامه شامل نام و امضای صادر کننده آن می‌باشد. گواهینامه سرویس گر ریشه توسط خودش امضا می‌شود. امضای یک گواهینامه ضامن حفظ تمامیت آن است.

⁴ Digital Certificate

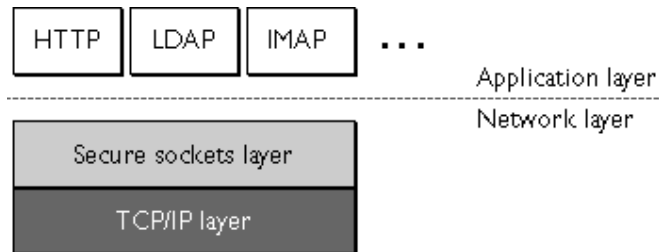
⁵ Certificate Authority (CA)



شکل ۳ - سلسله مراتب صدور گواهینامه

پروتکل SSL^۶

پروتکل TCP/IP برای انتقال و مسیریابی داده‌ها بر روی شبکه مورد استفاده قرار می‌گیرد. پروتکل‌هایی مانند LDAP، IMAP و HTTP در لایه بالایی پروتکل TCP/IP اجرا می‌شوند.



شکل ۴ - پروتکل SSL

همان‌طور که در شکل فوق مشاهده می‌شود SSL پایین‌تر از لایه کاربرد و بالاتر از لایه انتقال به اجرا درمی‌آید. SSL به یک سرور با قابلیت SSL اجازه می‌دهد که هویت خود را به یک سرویس‌گیر SSL اثبات نماید (عکس این عمل نیز امکان پذیر است)؛ همچنین اجازه می‌دهد که دو طرف یک اتصال امن رمزنگاری را ایجاد نمایند و اطلاعاتشان را به صورت کدشده منتقل کنند:

- اثبات هویت سرویس دهنده SSL: این قابلیت به سرویس‌گیر این امکان را می‌دهد که از هویت سرور اطمینان حاصل نماید. نرم‌افزار سرویس‌گیر SSL می‌تواند با استفاده از تکنیک استاندارد رمزنگاری کلید عمومی از اعتبار گواهی و کلید عمومی سرور اطمینان حاصل نماید. همچنین می‌تواند بررسی کند که مرکز صدور گواهی سرور، در لیست مراکز صدور گواهی قابل اعتماد خود قرار دارد یا نه.

⁶ Secure Socket Layer

- اثبات هویت سرویس گیر SSL: این قابلیت به سرور اجازه می‌دهد که هویت سرویس گیر را تأیید نماید.
 - اتصال امن SSL: این قابلیت به دو نرم‌افزار سرور و سرویس گیر اجازه می‌دهد که اطلاعاتشان را به صورت رمز شده مبادله نمایند.
- پروتکل SSL از دو پروتکل جزئی‌تر تشکیل شده است: SSL Record Protocol و SSL Handshake Protocol. پروتکل رکورد SSL فرمت انتقال داده‌ها را تعریف می‌کند. پروتکل ارتباطی SSL نیز نحوه استفاده از پروتکل رکورد SSL برای مبادله پیام‌های سرویس گیر و سرور SSL را تعریف می‌نماید. هدف از مبادله این پیام‌ها، دستیابی به اهداف زیر است:
- اثبات هویت سرور به سرویس گیر.
 - انتخاب الگوریتم رمزنگاری توسط سرویس گیر و سرور به طوری که هر دو طرف آن را پشتیبانی کنند.
 - اثبات هویت سرویس گیر به سرور (اختیاری).
 - استفاده از رمزنگاری کلید عمومی برای تولید کلید اشتراکی.
 - ایجاد اتصال رمزنگاری شده SSL.

پروتکل ارتباطی SSL

پروتکل SSL ترکیبی از رمزنگاری کلید عمومی و کلید اشتراکی را استفاده می‌نماید. رمزنگاری کلید اشتراکی سریع‌تر می‌باشد، در مقابل رمزنگاری کلید عمومی سرویس تأیید هویت مطمئن‌تری را ارائه می‌دهد. یک جلسه SSL با پیام SSL Handshake آغاز می‌گردد. این فاز پروتکل به سرور اجازه می‌دهد که هویت خودش را با استفاده از تکنیک کلید عمومی به سرویس گیر اثبات نماید؛ سپس به سرویس گیر و سرور اجازه ساخت کلید اشتراکی را می‌دهد. این کلید برای به رمز درآوردن داده‌ها، رمزگشایی داده‌ها و اطمینان از جامعیت پیام مورد استفاده قرار می‌گیرد. این فاز می‌تواند شامل اثبات هویت سرویس گیر به سرور نیز باشد. این مراحل به طور خلاصه در زیر آورده شده‌اند:

- سرویس گیر برای سرور شماره نسخه SSL مورد استفاده، الگوریتم رمز، داده تصادفی تولید شده و دیگر اطلاعاتی را که سرور برای ارتباط SSL با سرویس گیر نیاز دارد ارسال می‌نماید.
- سرور نیز اطلاعاتی که در مرحله اول گفته شد به علاوه گواهی خود را برای سرویس گیر ارسال می‌نماید. اگر نیاز به تأیید هویت سرویس گیر باشد سرور درخواست گواهی سرویس گیر را نیز ارسال می‌کند.
- سرویس گیر با استفاده از اطلاعاتی که توسط سرور ارسال شده هویت وی را بررسی می‌نماید. اگر هویت سرور اثبات نشد به کاربر اعلام می‌کند که امکان ایجاد اتصال رمزنگاری وجود ندارد.
- سرویس گیر با استفاده از داده‌هایی که تا به حال مبادله شده و نیز با توجه به الگوریتم توافقی یک کلید محرمانه اولیه⁷ ایجاد کرده و آن را با کلید عمومی سرور به رمز درمی‌آورد و فرم رمز شده آن را برای سرور ارسال می‌نماید.
- اگر سرور درخواست اثبات هویت سرویس گیر را کرده باشد سرویس گیر علاوه بر کلید محرمانه اولیه یک پیام امضاء شده به همراه گواهی خود برای سرور ارسال می‌نماید.

⁷ Premaster Key

- اگر سرور درخواست اثبات هویت سرویس گیر را کرده باشد سرور سعی در تأیید هویت سرویس گیر می‌نماید. اگر هویت سرویس گیر تأیید نشود اتصال خاتمه می‌یابد؛ در غیر این صورت سرور از کلید خصوصی خودش برای رمز گشایی کلید محرمانه اولیه استفاده می‌نماید.
- سرور و سرویس گیر با استفاده از کلید محرمانه اولیه کلید جلسه را تولید می‌نمایند که یک کلید اشتراکی و متقارن می‌باشد. اطلاعاتی که در طول یک جلسه SSL مبادله می‌گردد با استفاده از این کلید رمزنگاری و رمز گشایی می‌شود؛ همچنین با استفاده از آن می‌توان از جامعیت اطلاعات مبادله شده یعنی از تغییر نیافتن محتوای پیغام در طول انتقال اطمینان حاصل کرد.
- سرویس گیر به وسیله پیغامی به سرور اطلاع می‌دهد که بقیه پیغامها بوسیله کلید جلسه رمز خواهند شد؛ سپس یک پیغام رمز شده به نشانه پایان یافتن فاز ارتباطی ارسال می‌نماید.
- سرور نیز به سرویس گیر پیغامی می‌فرستد که به نشانه پایان یافتن فاز ارتباطی می‌باشد.
- فاز ارتباطی SSL به اتمام رسیده و کلید جلسه SSL به اشتراک گذاشته شده است. سرویس گیر و سرور از این کلید جلسه برای به رمز درآوردن، رمز گشایی و اطمینان از جامعیت داده‌ها استفاده می‌نمایند.

اثبات هویت سرور

نرم افزار سرویس گیر SSL به تأیید هویت سرور نیاز دارد. همان طور که در مرحله ۲ از فاز ارتباطی SSL گفته شد سرور گواهی خود را برای سرویس گیر ارسال می‌کند تا هویت خودش را اثبات نماید. در مرحله ۳ سرویس گیر با استفاده از گواهی سرور هویت وی را بررسی می‌نماید. نرم افزار سرویس گیر SSL برای تأیید هویت سرور باید از چهار سؤال زیر پاسخ مثبت دریافت کند:

- آیا گواهی سرور معتبر است؟ سرویس گیر تاریخ جاری را با زمان اعتبار گواهی مقایسه می‌کند. اگر گواهی منقضی شده باشد فرآیند تأیید هویت قطع می‌گردد.
 - آیا صادر کننده گواهی در لیست مراکز صدور گواهی قابل اعتماد سرویس گیر می‌باشد؟ هر سرویس گیر SSL لیستی از مراکز صدور گواهی قابل اعتماد خود را دارد. این لیست معین می‌کند که سرویس گیر گواهی چه سرورهایی را می‌پذیرد. بنابراین نام صادر کننده گواهی را که در فیلد DN از گواهی آمده با لیست مراکز صدور گواهی قابل اعتماد خود مقایسه می‌کند. اگر این نام در لیست نباشد هویت سرور تأیید نمی‌شود مگر آن که خود سرویس گیر بخواهد آن را به لیست خود اضافه نماید.
 - آیا کلید عمومی مرکز صدور گواهی امضای الکترونیکی صادر کننده را تأیید می‌کند؟ سرویس گیر با استفاده از کلید عمومی مرکز صدور گواهی که از گواهی به دست آورده است امضای الکترونیکی گواهی را بررسی می‌نماید. اگر اطلاعات گواهی سرور بعد از امضای آن توسط مرکز صدور گواهی تغییر کرده باشد و یا کلید عمومی مرکز صدور گواهی با کلید خصوصی که برای امضای گواهی سرور استفاده شده مطابقت نکند هویت سرور تأیید نخواهد شد.
 - آیا نام دامنه در گواهی سرور با نام دامنه سرور مطابقت دارد؟ این مرحله تأیید می‌کند که سرور واقعاً در همان شبکه‌ای واقع شده که در نام دامنه گواهی ذکر شده است.
- اگر پاسخ به تمام سؤالات فوق مثبت بود هویت سرور تأیید می‌گردد و در غیر این صورت اتصال SSL ایجاد نخواهد شد.

ایجاد کردن کلید و گواهی با استفاده از دستور openssl

دستور openssl یک برنامه برای استفاده از امکانات توابع کتابخانه‌های OpenSSL⁸ از طریق shell می‌باشد. با استفاده از این دستور می‌توان کارهای زیر را انجام داد:

- ایجاد کلیدهای RSA، DH و DSA
- ایجاد گواهی، CSR و CRL در استاندارد x.509
- محاسبه چکیده پیغام
- رمز کردن و رمز گشایی به وسیله الگوریتم‌های رمز
- تست کردن سرور و سرویس گیر SSL و TLS
- مدیریت پیغام‌های رمز شده S/MIME

دستور openssl دارای فرمت زیر است:

`Openssl command [command_opts] [command_args]`

پارامتر *command* مقادیر متعددی می‌تواند داشته‌باشد که نوع عملکرد دستور را تعیین می‌کند. البته استفاده از عده معدودی از آنها برای بسیاری از کاربران کفایت می‌کند. برای راه‌اندازی یک سرویس مبتنی بر SSL لازم است که کاربر یک کلید خصوصی به همراه یک CSR تولید کند، سپس این درخواست را برای یک مرکز صدور گواهی ارسال نماید که در مقابل یک گواهی از طرف مرکز صدور گواهی برای وی فرستاده شود. یک راه دیگر این است که این درخواست توسط خود کاربر امضا شود⁹. در هر حالت با استفاده از دستورات `genrsa`، `req` و `rsa` تمام این کارها را می‌توان انجام داد.

قبل از اینکه به بررسی این دستورات بپردازیم به یک نکته بد نیست اشاره شود. هنگام تولید کلید خصوصی می‌توان از یک `passphrase` برای برقراری امنیت بیشتر استفاده نمود. در این حالت برای استفاده از این کلید خصوصی باید `passphrase` مربوطه را وارد کرد. این روال شامل تمام گواهی‌های تولید شده از این کلید خصوصی نیز می‌شود. این امر باعث می‌شود که کلید خصوصی یا گواهی در صورت دزدیده شدن قابل استفاده نباشد. از طرف دیگر اگر از چنین کلیدی برای راه‌اندازی یک سرویس استفاده شود هنگام اجرا کردن سرویس باید `passphrase` مربوطه را وارد کرد. این بدان معناست که چنین سرویسی به‌طور اتوماتیک نمی‌تواند اجرا شود و هر بار هنگام اجرا شدن یک نفر باید برای وارد کردن این `passphrase` حضور فیزیکی داشته‌باشد. لذا کاربران باید با سنجیدن تمام جوانب کار درمورد استفاده از این امکان تصمیم‌گیری کنند.

کلمه `passphrase` توسط گزینه‌های `passin` و `passout` (به ترتیب برای ورودی و خروجی) مشخص می‌شود. اگر از گزینه‌های `passin` یا `passout` استفاده نشود `passphrase` مربوطه از ورودی استاندارد دریافت می‌شود. در هر کدام از دستورات `openssl` که نیاز به وارد کردن یک `passphrase` باشد می‌توان یکی از ترکیب‌های زیر را به عنوان پارامتر این دو گزینه به کار برد:

- `pass:passphrase` - از عبارت `passphrase` استفاده می‌شود.
- `env:var` - از مقدار متغیر محیطی `var` استفاده می‌شود.

⁸ <http://www.openssl.org>

⁹ self-signed certificate or root certificate

- `file:pathname` - از سطر اول فایل `pathname` برای خواندن `passphrase` استفاده می‌شود. اگر از یک فایل برای دو گزینه `passin` و `passout` - به‌طور همزمان استفاده شود، از سطر اول برای خواندن `passphrase` ورودی و از سطر دوم برای خواندن `passphrase` خروجی استفاده می‌شود.

- `fd:number` - از فایلی که شماره^{۱۰} معرف آن `number` باشد برای خواندن `passphrase` استفاده می‌شود.

- `stdin` - `passphrase` مربوطه از ورودی استاندارد خوانده می‌شود. این عملکرد پیشفرض سیستم است، پس نیازی به استفاده از `stdin` نمی‌باشد.

در قسمت‌های بعدی دستورات `req`، `rsa`، `genrsa` و `x509` به‌همراه بعضی از گزینه‌های متداول آنها توضیح داده می‌شود. اطلاعات بیشتر در مورد این دستورات در صفحات `manual` آنها وجود دارد.

دستور genrsa

این دستور برای تولید کلید خصوصی RSA به کار می‌رود. فرم کلی این دستور به شکل زیر می‌باشد:

```
openssl genrsa [-out filename] [-passout arg] [-des] [-des3] [-idea] [-f4] [-3] [-rand file(s)] [numbits]
```

گزینه‌های متداول:

<code>-out filename</code>	نام فایل خروجی را که کلید خصوصی در آن نوشته می‌شود مشخص می‌کند. اگر از این گزینه استفاده نشود خروجی در <code>stdout</code> نوشته می‌شود.
<code>-des -des3 -idea</code>	از یکی از الگوریتم‌های <code>DES</code> ، <code>DES3</code> یا <code>IDEA</code> برای رمز کردن کلید خصوصی استفاده می‌کند.
<code>-passout arg</code>	در صورتی که از یکی از الگوریتم‌های <code>DES</code> ، <code>DES3</code> یا <code>IDEA</code> برای رمز کردن کلید استفاده شود برای مشخص کردن <code>passphrase</code> فایل خروجی از این گزینه استفاده می‌شود.
<code>numbits</code>	طول کلید را مشخص می‌کند. مقدار پیشفرض آن ۵۱۲ است.

دستور زیر یک کلید خصوصی به طول ۱۰۲۴ بیت و بدون `passphrase` تولید می‌کند:

```
openssl genrsa -out rsakey.pem 1024
```

دستور زیر یک کلید خصوصی به طول ۱۰۲۴ رمز شده با الگوریتم `DES3` و یک `passphrase` خاص تولید می‌کند:

```
openssl genrsa -out rsakey.pem -passout pass:enter-pass-here -des3 1024
```

دستور rsa

این دستور برای مدیریت کلیدهای `RSA` به کار می‌رود. با استفاده از این دستور می‌توان کلیدها را از یک فرمت به فرمت دیگر تبدیل کرد، محتوای آنها را تغییر داد، یا مقدار فیلدهای مختلف کلید را مشاهده نمود. فرم کلی این دستور به شکل زیر می‌باشد:

¹⁰ descriptor

openssl rsa [-inform PEM|NET|DER] [-outform PEM|NET|DER] [-in *filename*] [-passin *arg*] [-out *filename*] [-passout *arg*] [-sgckey] [-des] [-des3] [-idea] [-text] [-noout] [-modulus] [-check] [-pubin] [-pubout]

گزینه‌های متداول:

-inform	فرمت کلید ورودی را تعیین می‌کند که می‌تواند PEM, NET یا DER باشد. مقدار پیشفرض آن PEM است.
-outform	فرمت کلید خروجی را تعیین می‌کند که می‌تواند PEM, NET یا DER باشد. مقدار پیشفرض آن PEM است.
-in <i>filename</i>	فایل ورودی را که شامل یک کلید خصوصی است مشخص می‌کند. اگر از این گزینه استفاده نشود کلید ورودی از ورودی استاندارد خوانده می‌شود.
-passin <i>arg</i>	اگر کلید ورودی رمز شده باشد passphrase آن با این گزینه مشخص می‌شود.
-out <i>filename</i>	فایلی را که کلید خصوصی در آن نوشته شده مشخص می‌کند. اگر از این گزینه استفاده نشود کلید خروجی در خروجی استاندارد نوشته می‌شود.
-passout <i>arg</i>	اگر کلید خروجی رمز شده باشد passphrase آن با این گزینه مشخص می‌شود.
-des -des3 -idea	از یکی از الگوریتم‌های DES, DES3 یا IDEA برای رمز کردن کلید خصوصی استفاده می‌کند.
-text	فیلدهای کلید خصوصی را علاوه بر حالت گذشته به صورت متنی ساده نیز به خروجی می‌فرستد.
-noout	با استفاده از این گزینه فرم گذشته فیلدهای کلید خصوصی در خروجی ظاهر نمی‌شوند.
-modulus	قسمت modulus کلید را در خروجی ظاهر می‌کند.

دستور زیر passphrase یک کلید خصوصی را پاک می‌کند:

```
openssl rsa -in inkey.pem -passin file: pass-file-here -out outkey.pem
```

دستور زیر یک کلید خصوصی را رمز می‌کند (passphrase از ورودی استاندارد خوانده می‌شود):

```
openssl rsa -in inkey.pem -des3 -out outkey.pem
```

دستور زیر محتویات یک کلید خصوصی را نشان می‌دهد:

```
openssl rsa -in inkey.pem -text -noout
```

دستور req

از این دستور برای مدیریت CSR استفاده می‌شود، هرچند می‌توان از آن برای تولید کلید خصوصی و گواهی

نیز استفاده نمود. فرم کلی این دستور به شکل زیر می‌باشد:

```
openssl req [-inform PEM|DER] [-outform PEM|DER] [-in filename] [-passin arg] [-out filename] [-passout arg] [-text] [-noout] [-verify] [-modulus] [-new] [-rand file(s)] [-newkey rsa:bits] [-newkey dsa:file] [-nodes] [-key filename] [-keyform PEM|DER] [-keyout filename] [-[md5|sha1|md2|mdc2]] [-config filename] [-x509] [-days n] [-asn1-kludge] [-newhdr] [-extensions section] [-reqexts section]
```

گزینه‌های متداول:

-inform	فرمت CSR ورودی را تعیین می‌کند که می‌تواند PEM, NET یا DER باشد. مقدار پیشفرض آن PEM است.
---------	---

-outform	فرمت CSR خروجی را تعیین می کند که می تواند PEM, NET یا DER باشد. مقدار پیش فرض آن PEM است.
-in filename	فایل ورودی را که شامل یک CSR است مشخص می کند. اگر از این گزینه استفاده نشود CSR ورودی از ورودی استاندارد خوانده می شود.
-passin arg	اگر CSR ورودی رمز شده باشد passphrase آن با این گزینه مشخص می شود.
-out filename	فایلی را که CSR در آن نوشته شده مشخص می کند. اگر از این گزینه استفاده نشود CSR در خروجی استاندارد نوشته می شود.
-passout arg	اگر CSR رمز شده باشد passphrase آن با این گزینه مشخص می شود.
-text	فیلدهای CSR را علاوه بر حالت گذشته به صورت متنی ساده نیز به خروجی می فرستد.
-noout	با استفاده از این گزینه فرم گذشته فیلدهای CSR در خروجی ظاهر نمی شوند.
-modulus	قسمت modulus کلید عمومی داخل CSR را در خروجی ظاهر می کند.
-new	استفاده از این گزینه باعث ایجاد یک CSR می شود و اطلاعات مورد نیاز نیز از ورودی استاندارد گرفته می شود. برای ایجاد CSR از کلیدی که با گزینه key مشخص شده استفاده می شود (اگر مشخص نشده باشد یک کلید تولید می شود).
-newkey rsa:bits	با استفاده از این گزینه یک کلید خصوصی RSA و یک CSR تولید می شود. bits تعداد بیت های کلید را مشخص می کند.
-keyout filename	این گزینه نام فایلی را که کلید خصوصی در آن نوشته می شود مشخص می کند.
-x509	با استفاده از این گزینه به جای CSR یک گواهی root ایجاد می شود.
-days n	اگر از گزینه x509 استفاده شده باشد n میزان اعتبار گواهی را مشخص می کند.

دستور زیر با استفاده از یک کلید خصوصی یک CSR تولید می کند:

```
openssl req -new -key key.pem -out req.pem
```

دستور زیر یک کلید خصوصی و یک CSR به طور همزمان تولید می کند:

```
openssl req -newkey rsa:1024 -keyout key.pem -out req.pem
```

دستور زیر یک کلید خصوصی و یک گواهی root به طور همزمان تولید می کند:

```
openssl req -x509 -newkey rsa:1024 -keyout key.pem -out cert.pem
```

دستور x509

از این دستور برای مدیریت گواهی توسط یک مرکز صدور گواهی استفاده می شود. فرم کلی این دستور به شکل زیر می باشد:

```
openssl x509 [-inform DER|PEM|NET] [-outform DER|PEM|NET] [-keyform DER|PEM] [-CAform DER|PEM] [-CAkeyform DER|PEM] [-in filename] [-out filename] [-serial] [-hash] [-subject] [-issuer] [-nameopt option] [-email] [-startdate] [-enddate] [-purpose] [-dates] [-modulus] [-fingerprint] [-alias] [-noout] [-trustout] [-clrtrust] [-clrreject] [-addtrust arg] [-addrject arg] [-setalias arg] [-days n] [-signkey filename] [-x509toreq] [-req] [-CA filename] [-CAkey filename] [-CAcreateserial] [-CAserial filename] [-text] [-C] [-md2|md5|sha1|mdc2] [-clrext] [-extfile filename] [-extensions section]
```

گزینه های متداول:

-inform	فرمت CSR ورودی را تعیین می کند که می تواند PEM، NET یا DER باشد. مقدار پیشفرض آن PEM است.
-outform	فرمت CSR خروجی را تعیین می کند که می تواند PEM، NET یا DER باشد. مقدار پیشفرض آن PEM است.
-in filename	فایل ورودی را که شامل یک CSR است مشخص می کند. اگر از این گزینه استفاده نشود CSR ورودی از ورودی استاندارد خوانده می شود.
-out filename	فایلی را که CSR در آن نوشته شده مشخص می کند. اگر از این گزینه استفاده نشود CSR در خروجی استاندارد نوشته می شود.
-text	فیلدهای CSR را علاوه بر حالت گذشته به صورت متنی ساده نیز به خروجی می فرستد.
-noout	با استفاده از این گزینه فرم گذشته فیلدهای CSR در خروجی ظاهر نمی شوند.
-modulus	قسمت modulus کلید عمومی داخل گواهی را در خروجی ظاهر می کند.
-serial	شماره سریال گواهی را در خروجی ظاهر می کند.
-hash	مقدار hash نام صاحب گواهی را در خروجی ظاهر می کند.
-subject	نام صاحب گواهی را در خروجی ظاهر می کند.
-issuer	نام صادرکننده گواهی را در خروجی ظاهر می کند.
-email	آدرس پست الکترونیکی صاحب گواهی را در خروجی ظاهر می کند.
-startdate	تاریخ صدور گواهی را در خروجی ظاهر می کند.
-enddate	تاریخ انقضای گواهی را در خروجی ظاهر می کند.
-dates	تاریخ صدور و انقضای گواهی را در خروجی ظاهر می کند.
-fingerprint	امضای دیجیتالی گواهی را در خروجی ظاهر می کند.
-signkey filename	استفاده از این گزینه باعث می شود که یک گواهی root با استفاده از کلید خصوصی قرار گرفته در filename ایجاد شود.
-keyfrom	فرمت کلید خصوصی ورودی را مشخص می کند که می تواند PEM یا DER باشد. مقدار پیشفرض آن PEM است.
-days n	اگر از گزینه x509- استفاده شده باشد n میزان اعتبار گواهی را مشخص می کند.
-x509toreq	این گزینه یک گواهی را به یک CSR تبدیل می کند. از کلید خصوصی داده شده در گزینه signkey- برای این کار استفاده می شود.
-req	در حالت پیشفرض یک گواهی به عنوان ورودی خوانده می شود. با استفاده از این گزینه یک CSR در ورودی خوانده می شود.
-CA filename	گواهی متعلق به مرکز صدور گواهی که برای امضا کردن استفاده می شود توسط این گزینه مشخص می شود.
-CAkey filename	کلید خصوصی مرکز صدور گواهی که برای امضا کردن از آن استفاده می شود توسط این گزینه مشخص می شود.
-CAserial filename	فایل حاوی شماره سریال certificate را مشخص می کند. مقدار پیشفرض برای نام این فایل عبارت است از نام پایه گواهی مربوط به مرکز صدور گواهی به علاوه پسوند .srl.

-CAcreateserial filename	فایل شماره سریال certificate را ایجاد می کند.
-extfile filename	فایلی را که extentionsهای مورد استفاده در آن قرار دارند مشخص می کند.
-extentions section	برای افزودن یک extention خاص به گواهی از این گزینه استفاده می شود.

دستور زیر اطلاعات متنی یک گواهی را نشان می دهد:

```
openssl x509 -in cert.pem -noout -text
```

دستور زیر شماره سریال یک گواهی را نشان می دهد:

```
openssl x509 -in cert.pem -noout -serial
```

دستور زیر نام صاحب یک گواهی را نشان می دهد:

```
openssl x509 -in cert.pem -noout -subject
```

دستور زیر امضای دیجیتالی یک گواهی را نشان می دهد:

```
openssl x509 -in cert.pem -noout -fingerprint
```

دستور زیر یک گواهی را از فرمت PEM به فرمت DER تبدیل می کند:

```
openssl x509 -in cert.pem -inform PEM -out cert.der -outform DER
```

دستور زیر یک گواهی را به یک CSR تبدیل می کند:

```
openssl x509 -x509toreq -in cert.pem -out req.pem -signkey key.pem
```

دستور زیر یک CSR را به یک گواهی root تبدیل می کند:

```
openssl x509 -req -in careq.pem -signkey key.pem -out cacert.pem
```

دستور زیر یک CSR را با استفاده از گواهی و کلید خصوصی مرکز صدور گواهی امضا می کند:

```
openssl x509 -req -in req.pem -CA cacert.pem -CAkey key.pem -CAcreateserial
```

استفاده از SSL برای امن کردن سرویس ها

برای امن کردن سرویس های مختلف توسط SSL دو حالت وجود دارد:

- سرویس مورد نظر دارای امکانات SSL می باشد و می تواند اطلاعات رمز شده را دریافت و ارسال نماید (مانند سرور وب Apache). در این حالت فقط با پیکربندی سرور مربوطه امکان استفاده از SSL به آن اضافه می گردد.

- سرویس مورد نظر دارای امکانات SSL نمی باشد و قادر به دریافت و ارسال اطلاعات رمز شده نمی باشد (مانند سرور IMAP). در این حالت باید از نرم افزارهایی تحت عنوان SSL Wrapper استفاده کرد. این نوع نرم افزارها که از متداول ترین آنها می توان به Stunnel¹¹ اشاره کرد ترافیک ورودی رمز شده را دریافت کرده پس از رمز گشایی برای سرور می فرستند و ترافیک خروجی را از سرور دریافت کرده پس از رمز کردن ارسال می کنند.

در هر دو حالت برای کار با SSL نیاز به یک کلید خصوصی و گواهی می باشد که آنها را می توان بدین طریق ایجاد کرد:

- تولید یک کلید خصوصی:

```
openssl genrsa -out key.pem 1024
```

- تولید یک CSR:

¹¹ <http://www.stunnel.org>

```
openssl req -new -key key.pem -out csr.pem
```

- فرستادن CSR به یک مرکز صدور گواهی یا امضا کردن آن توسط کلید خصوصی. برای امضا کردن CSR استفاده از دستور زیر کفایت:

```
openssl x509 -req -in csr.pem -signkey key.pem -out cert.pem -days 365
```

در این قسمت برای روشن تر شدن موضوع نحوه استفاده از SSL به همراه سرور POP3 توضیح داده می شود. برای استفاده از SSL به همراه سرور POP3 از نرم افزار Stunnel استفاده می شود. استفاده از Stunnel به دو طریق ممکن است:

- Stunnel اطلاعات رمز شده را از پورت شماره ۹۹۵ دریافت کرده و آنها را به پورت شماره ۱۱۰ بفرستد و این اطلاعات توسط سروری که به پورت شماره ۱۱۰ گوش می دهد پردازش شوند. در این صورت دستور زیر را باید در script های بوت سیستم قرار داد:

```
stunnel -d 995 -p /usr/local/ssl/certs/stunnel.pem -r localhost:110
```

- Stunnel اطلاعات رمز شده را از پورت شماره ۹۹۵ دریافت کرده و یک سرور را برای پردازش این اطلاعات رمز گشایی شده اجرا کند. در این صورت دستور زیر را باید در script های بوت سیستم قرار داد:

```
stunnel -d 995 -p /usr/local/ssl/certs/stunnel.pem -l /usr/sbin/imapd
```

در هر دو دستور بالا stunnel.pem یک فایل است که به ترتیب شامل کلید خصوصی و گواهی سرور (به صورت رمز نشده) می باشد. در فایل stunnel.pem بعد از کلید خصوصی و نیز بعد از گواهی یک سطر خالی باید وجود داشته باشد.

گروه امنیتی سیمرغ

هومن آتشبار ، حسین عسگری

www.simorgh-ev.com