امنیت بیشتر با SSL

فهرست

۱ مقدمه‌ای بر رمزنگاری

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

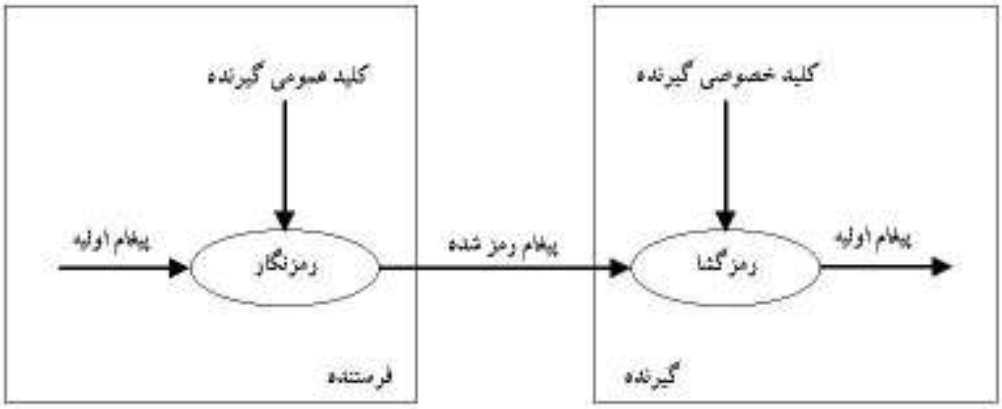
- رمزنگاری متقارن[1]
- رمزنگاری نامتقارن[2]

در روش رمزنگاری متقارن کلید رمزنگاری و کلید رمزگشایی هر دو یکسان هستند یا به‌سهولت ازروی 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. ازمزایای این روش سهولت پیاده‌سازی و سرعت بالای آن را می‌توان نام برد.

شکل ۱ – رمزنگاری متقارن

در روش رمزنگاری نامتقارن هر موجودیتی دو کلید مرتبط به‌هم با نام‌های کلید عمومی و کلید خصوصی دارد که به‌دست آوردن آنها از روی همدیگر به‌لحاظ محاسباتی‌تقریبا 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 درصورتی‌که 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 بازکردن پیغام استفاده می‌کنند تا مطمئن شوند که پیغام از طرف او بوده‌است. 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.

[1] Symmetric Cryptography
[2] Asymmetric Cryptography



شکل ۲ – رمزنگاری نامتقارن

- **محرمانگی**[۳]

منظور از محرمانگی آن است که اطلاعات ردوبدل شده توسط موجودیت‌های غیرمجاز قابل فهم نباشد. محرمانگی از طریق رمزکردن اطلاعات ارسالی با یک کلید متقارن تصادفی به‌دست می‌آید الگوریتم‌های متقارن به‌لحاظ سرعت بیشتری که دارند در رمزکردن 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 کل اطلاعات را رمزگشایی می‌کند.

- **تمامیت داده‌ها**

منظور از تمامیت داده‌ها دریافت داده‌ها به همان صورت ارسال شده‌است. تمامیت با محاسبهٔ مقدار Hash پیغام ارسالی و رمزکردن حاصل با کلید خصوصی فرستنده حاصل می‌شود (امضای دیجیتالی). خصوصیت تابع Hash آن است که پیغام‌های متفاوت، مقادیر Hash متفاوت دارند و پیداکردن دو پیغام که دارای مقدار Hash یکسان باشند به‌لحاظ محاسباتی غیرممکن است. امضای دیجیتالی حاصل رمزکردن مقدار Hash با کلید خصوصی فرستنده است که به‌همراه 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.

[3] Confidentiality

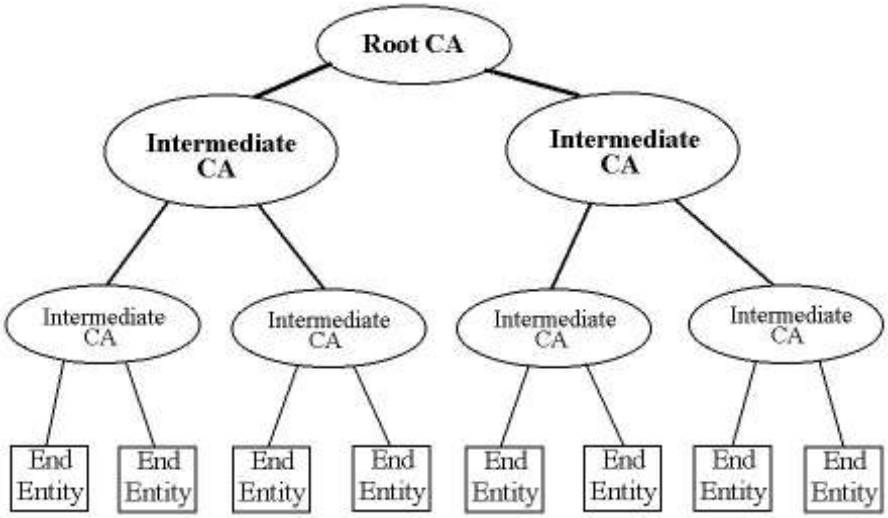
- **تصدیق پیام و موجودیت**

تصدیق موجودیت بر اساس میزان اطمینان از مالکیت انحصاری یک کلید خصوصی توسط آن موجودیت که توسط یک گواهینامهٔ دیجیتالی[۴] مشخص می‌شود و میزان اعتبار گواهینامهٔ مربوطه انجام می‌شود. درحالی‌که در اکثر سیستم‌های جاری تصدیق موجودیت براساس کد کاربری و کلمهٔ عبور انجام می‌شود گواهینامه‌ها راه جدیدی برای تصدیق موجودیت پیشنهاد می‌کند که اساس آن داشتن کلید خصوصی مرتبط با کلید عمومی موجود در آنهاستدر عمل معمو لاً ترکیب دو روش ذکرشده استفاده می شود. تصدیق پیغام با استفاده از آزمون تمامیت داده‌ها انجام می شود. درواقع تصدیق پیغام و تمامیت داده‌ها لازم و ملزوم یکدیگرند. تمامیت داده‌ها درصورتی‌که مبدا پیغام تصدیق نشود ارزش چندانی ندارد و تصدیق مبدا پیغام ارزش ندارد اگر تمامیت داده‌ها حفظ نشود.

- **سرویس‌گر گواهینامه**[۵]

سرویس‌گر گواهینامه وظیفهٔ صادرکردن، مدیریت و فسخ‌کردن گواهینامه را برعهده دارد. سایر موجودیت‌های دخیل باید گواهینامهٔ این سرویس‌گر را به‌خوبی بشناسند (کلید عمومی‌اش را بدانند). این سرویس‌گر می‌تواند وظایف خود را با صادرکردن گواهینامه‌های مخصوص سرویس‌گرهای گواهینامه به موجودیت‌های دیگری در سطح پایین تر محول کند و به این ترتیب سلسله مراتب سرویس‌گر گواهینامه را تشکیل دهد. هدف از این کار مدیریت ساده‌تر (پیاده‌سازی سیاست‌های مختلف) و کارآیی بیشتر می باشد. دنبالهٔ مرتب گواهینامه‌ها از آخرین شعبه تا سرویس‌گر ریشه را زنجیرهٔ گواهینامه می‌نامند. هر گواهینامه شامل نام وامضای صادرکنندهٔ آن می‌باشد. گواهینامهٔ سرویس‌گر ریشه توسط خودش امضا می شود. امضای یک گواهینامه ضامن حفظ تمامیت آن است.

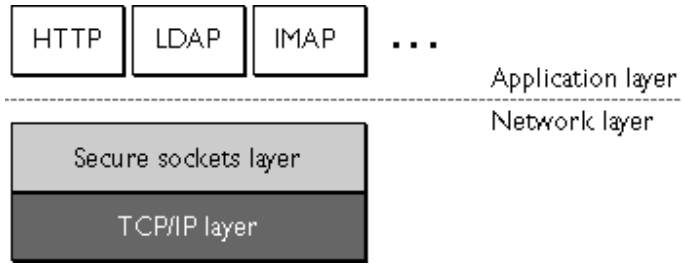
[4] Digital Certificate
[5] Certificate Authority (CA)



شکل ۳ – سلسله مراتب صدور گواهینامه

۲ پروتکل SSL[۶]

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



شکل ۴ – پروتکل SSL

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

[6] Secure Socket Layer

- اثبات هویت سرویس دهندهٔ 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.
- اثبات هویت سرویس‌گیر SSL : این قابلیت به سرور اجازه می‌دهد که هویت سرویس‌گیر را تأیید نماید.
- اتصال امن SSL : این قابلیت به دو نرم‌افزار سرور و سرویس‌گیر اجازه می‌دهد که اطلاعاتشان را به‌صورت رمزشده مبادله نمایند.

پروتکل SSL از دو پروتکل جزئی‌تر تشکیل شده‌است : SSL Record Protocol و SSL Handshake Protocol. پروتکل رکورد SSL فرمت انتقال داده‌ها را تعریف می‌کند. پروتکل ارتباطی SSL نیز نحوهٔ استفاده از پروتکل رکورد SSL برای مبادلهٔ پیغام‌های سرویس‌گیر و سرور SSL را تعریف می‌نماید. هدف از مبادلهٔ این پیغام‌ها، دستیابی به اهداف زیر است:

- اثبات هویت سرور به سرویس‌گیر.
- انتخاب الگوریتم رمزنگاری توسط سرویس‌گیر و سرور به‌طوری‌که هردو طرف آن را پشتیبانی کنند.
- اثبات هویت سرویس‌گیر به سرور (اختیاری).
- استفاده از رمزنگاری کلید عمومی برای تولید کلید اشتراکی.
- ایجاد اتصال رمزنگاری‌شدهٔ SSL.

۲-۱ پروتکل ارتباطی SSL

پروتکل SSL ترکیبی از رمزنگاری کلید عمومی و کلید اشتراکی را استفاده می‌نماید. رمزنگاری کلید اشتراکی سریع‌تر می‌باشد، درمقابل رمزنگاری کلید عمومی سرویس تأیید هویت مطمئن‌تری را ارائه می‌دهد. یک جلسهٔ SSL با پیغام SSL Handshake آغاز می‌گردد. این فاز پروتکل به سرور اجازه می‌دهد که هویت خودش را با استفاده از تکنیک کلید عمومی به سرویس‌گیر اثبات نماید؛ سپس به

سرویس‌گیر و سرور اجازهٔ ساخت کلید اشتراکی را می‌دهد. این کلید برای به رمز درآوردن داده‌ها، رمزگشایی داده‌ها و اطمینان از جامعیت پیغام مورد استفاده قرار می‌گیرد. این فاز می‌تواند شامل اثبات هویت سرویس‌گیر به سرور نیز باشد. این مراحل به‌طور خلاصه در زیر آورده شده‌اند:

- سرویس‌گیر برای سرور شمارهٔ نسخهٔ SSL مورد استفاده، الگوریتم رمز، دادهٔ تصادفی تولیدشده و دیگر اطلاعاتی را که سرور برای ارتباط SSL با سرویس‌گیر نیاز دارد ارسال می‌نماید.
- سرور نیز اطلاعاتی که در مرحلهٔ اول گفته شد به‌علاوهٔ گواهی خود را برای سرویس‌گیر ارسال می‌نماید. اگر نیاز به تأیید هویت سرویس‌گیر باشد سرور درخواست گواهی سرویس‌گیر را نیز ارسال می‌کند.
- سرویس‌گیر با استفاده از اطلاعاتی که توسط سرور ارسال شده هویت وی را بررسی می‌نماید. اگر هویت سرور اثبات نشد به کاربر اعلام می‌کندکه امکان ایجاد اتصال رمزنگاری وجود ندارد.
- سرویس‌گیر با استفاده از داده‌هایی که تابه‌حال مبادله شده و نیز با توجه به الگوریتم توافقی یک کلید محرمانهٔ اولیه[۷] ایجاد کرده و آن را با کلید عمومی سرور به رمز درمی‌آورد و فرم رمزشدهٔ آن را برای سرور ارسال می‌نماید.
- اگر سرور درخواست اثبات هویت سرویس‌گیر را کرده باشد سرویس‌گیر علاوه بر کلید محرمانهٔ اولیه یک پیغام امضاء شده به‌همراه گواهی خود برای سرور ارسال می‌نماید.
- اگر سرور درخواست اثبات هویت سرویس‌گیر را کرده باشد سرور سعی در تأیید هویت سرویس‌گیر می‌نماید. اگر هویت سرویس‌گیر تأیید نشود اتصال خاتمه می‌یابد؛ در غیر این‌صورت سرور از کلید خصوصی خودش برای رمزگشایی کلید محرمانهٔ اولیه استفاده می‌نماید.
- سرور و سرویس‌گیر با استفاده از کلید محرمانهٔ اولیه کلید جلسه را تولید می‌نمایند که یک کلید اشتراکی و متقارن می‌باشد. اطلاعاتی که در طول یک جلسهٔ SSL مبادله می‌گردد با استفاده از این کلید رمزنگاری و رمزگشایی می‌شود؛ همچنین با استفاده از آن می‌توان از جامعیت اطلاعات مبادله‌شده یعنی از تغییر نیافتن محتوای پیغام در طول انتقال اطمینان حاصل کرد.

[7] Premaster Key

- سرویس‌گیر به‌وسیلهٔ پیغامی به سرور اطلاع می‌دهد که بقیهٔ پیغام‌ها بوسیلهٔ کلید جلسه رمز خواهند شد؛ سپس یک پیغام رمزشده به نشانهٔ پایان یافتن فاز ارتباطی ارسال می‌نماید.
- سرور نیز به سرویس‌گیر پیغامی می‌فرستد که به نشانهٔ پایان یافتن فاز ارتباطی می‌باشد.
- فاز ارتباطی SSL به اتمام رسیده و کلید جلسهٔ SSL به اشتراک گذاشته شده‌است. سرویس‌گیر و سرور از این کلید جلسه برای به رمز درآوردن، رمزگشایی و اطمینان از جامعیت داده‌ها استفاده می‌نمایند.

۲-۲ اثبات هویت سرور

نرم‌افزار سرویس‌گیر SSL به تأیید هویت سرور نیاز دارد. همان‌طور که در مرحلهٔ ۲ از فاز ارتباطی SSL گفته شد سرور گواهی خود را برای سرویس‌گیر ارسال می‌کند تا هویت خودش را اثبات نماید. در مرحلهٔ ۳ سرویس‌گیر با استفاده از گواهی سرور هویت وی را بررسی می‌نماید. نرم‌افزار سرویس‌گیر SSL برای تأیید هویت سرور باید از چهار سؤال زیر پاسخ مثبت دریافت کند:

- آیا گواهی سرور معتبر است؟ سرویس‌گیر تاریخ جاری را با زمان اعتبار گواهی مقایسه می‌کند. اگر گواهی منقضی شده‌باشد فرآیند تأیید هویت قطع می‌گردد.
- آیا صادر کنندهٔ گواهی در لیست مراکز صدور گواهی قابل اعتماد سرویس‌گیر می‌باشد؟ هر سرویس‌گیر SSL لیستی از مراکز صدور گواهی قابل اعتماد خود را دارد. این لیست معین می‌کند که سرویس‌گیر گواهی چه سرورهایی را می‌پذیرد. بنابراین نام صادرکنندهٔ 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 آن‌که خود سرویس‌گیر بخواهد آن را به لیست خود اضافه نماید.
- آیا کلید عمومی مرکز صدور گواهی امضای الکترونیکی صادرکننده را تأیید می‌کند؟ سرویس‌گیر با استفاده از کلید عمومی مرکز صدور گواهی که از گواهی به‌دست آورده‌است 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 دامنهٔ سرور مطابقت دارد؟ این مرحله تأیید می‌کند که سرور واقعاً در همان شبکه‌ای واقع شده که در نام دامنهٔ گواهی ذکر شده‌است.

اگر پاسخ به تمام سؤالات فوق مثبت بود هویت سرور تأیید می‌گردد و در غیر این‌صورت اتصال SSL ایجاد نخواهدشد.

۳ ایجادکردن کلید و گواهی با استفاده از دستور openssl

دستور openssl یک برنامه برای استفاده از امکانات توابع کتابخانه‌ای OpenSSL[۸] 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
- محاسبهٔ چکیدهٔ پیغام
- رمزکردن و رمزگشایی به‌وسیلهٔ الگوریتم‌های رمز
- تست‌کردن سرور و سرویس‌گیر SSL و TLS
- مدیریت پیغام‌های رمزشدهٔ 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 عدهٔ معدودی از آنها برای بسیاری از کاربران کفایت می‌کند. برای راه‌اندازی یک سرویس مبتنی بر SSL لازم است که کاربر یک کلید خصوصی به‌همراه یک CSR تولید کند، سپس این درخواست را برای یک مرکز صدور گواهی ارسال نماید که در مقابل یک گواهی ازطرف مرکز صدور

[8] http://www.openssl.org

گواهی برای وی فرستاده شود. یک راه دیگر این است که این درخواست توسط خود کاربر امضا شود[9]. درهرحالت با استفاده از دستورات genrsa، rsa، req و x509 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 تولیدشده از این کلید خصوصی نیز می‌شود. این امر باعث می‌شود که کلید خصوصی یا گواهی درصورت 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 داشته باشد. لذا کاربران باید با سنجیدن تمام جوانب کار درمورد استفاده از این امکان تصمیم‌گیری کنند.

کلمهٔ passphrase توسط گزینه‌های passin- و passout- (به‌ترتیب برای ورودی و خروجی) مشخص می‌شود. اگر از گزینه‌های passin- یا passout- استفاده نشود passphrase مربوطه از ورودی استاندارد دریافت می‌شود. در هرکدام از دستورات openssl که نیاز به وارد کردن یک passphrase باشد می‌توان یکی از ترکیب‌های زیر را به‌عنوان پارامتر این دو گزینه به‌کار برد:

- pass:*passphrase* – از عبارت *passphrase* استفاده می‌شود.
- env:*var* – از مقدار متغیر محیطی *var* استفاده می‌شود.
- file:*pathname* – از سطر اول فایل *pathname* برای خواندن passphrase استفاده می‌شود. اگر از یک فایل برای دو گزینهٔ 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 شمارهٔ معرف[10] 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.

[9] self-signed certificate or root certificate
[10] descriptor

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

۳-۱ دستور genrsa

این دستور برای تولید کلید خصوصی RSA به‌کار می‌رود. فرم کلی این دستور به‌شکل زیر می‌باشد:

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

گزینه‌های متداول:

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

دستور زیر یک کلید خصوصی به‌طول ۱۰۲۴ بیت و بدون passphrase تولید می‌کند:

openssl genrsa -out rsakey.pem 1024

دستور زیر یک کلید خصوصی به‌طول ۱۰۲۴ رمزشده با الگوریتم DES3 و یک passphrase خاص تولید می‌کند:

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

۳-۲ دستور rsa

این دستور برای مدیریت کلیدهای RSA به‌کار 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 به‌شکل زیر می‌باشد:

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 *filename*	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 *arg*	اگر کلید ورودی رمزشده باشد passphrase آن با این گزینه مشخص می‌شود.
-out *filename*	فایلی را که کلید خصوصی در آن نوشته‌شده مشخص می‌کند. اگر از این گزینه استفاده نشود کلید خروجی در خروجی استاندارد نوشته می‌شود.
-passout *arg*	اگر کلید خروجی رمزشده باشد passphrase آن با این گزینه مشخص می‌شود.
-des|-des3|-idea	از یکی از الگوریتم‌های DES، DES3 یا IDEA برای رمزکردن کلید خصوصی استفاده می‌کند.
-text	فیلدهای کلید خصوصی را علاوه بر حالت کدشده به‌صورت متنی ساده نیز به خروجی می‌فرستد.
-nooout	با استفاده از این گزینه فرم کدشدهٔ 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

۳-۳ 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 به‌شکل 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 می‌کند که می‌تواند PEM، NET یا DER باشد. مقدار پیشفرض آن PEM است.
-in *filename*	فایل ورودی را که شامل یک CSR است مشخص می‌کند. اگر از این گزینه استفاده نشود CSR ورودی از ورودی استاندارد خوانده می‌شود.
-passin *arg*	اگر CSR ورودی رمزشده باشد passphrase آن با این گزینه مشخص می‌شود.
-out *filename*	فایلی را که CSR در آن نوشته‌شده مشخص می‌کند. اگر از این گزینه

	استفاده نشود CSR در خروجی استاندارد نوشته می‌شود.
-passout *arg*	اگر CSR رمزشده باشد passphrase آن با این گزینه مشخص می‌شود.
-text	فیلدهای CSR را علاوه بر حالت کدشده به‌صورت متنی ساده نیز به خروجی می‌فرستد.
-nooout	با استفاده از این گزینه فرم کدشدهٔ فیلدهای CSR در خروجی ظاهر نمی‌شوند.
-modulus	قسمت modulus کلید عمومی داخل CSR را در خروجی ظاهر می‌کند.
-new	استفاده از این گزینه باعث ایجاد یک CSR می‌شود و اطلاعات موردنیاز نیز از ورودی استاندارد گرفته می‌شود. برای ایجاد CSR از کلیدی که با گزینهٔ -key مشخص‌شده استفاده می‌شود (اگر مشخص نشده‌باشد یک کلید تولید می‌شود).
-newkey rsa:*bits*	با استفاده از این گزینه یک کلید خصوصی RSA و یک CSR تولید می‌شود. *bits* تعداد بیت‌های کلید را مشخص می‌کند.
-keyout *filename*	این گزینه نام فایلی را که کلید خصوصی در آن نوشته‌می‌شود مشخص می‌کند.
-x509	با استفاده از این گزینه به‌جای CSR یک گواهی root ایجاد می‌شود.
-days *n*	اگر از گزینهٔ -x509 استفاده شده‌باشد *n* میزان اعتبار گواهی را مشخص می‌کند.

دستور زیر با استفاده از یک کلید خصوصی یک CSR تولید می‌کند:

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

دستور زیر یک کلید خصوصی و یک CSR به‌طور همزمان تولید می‌کند:

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

دستور زیر یک کلید خصوصی و یک گواهی root به‌طور همزمان تولید می‌کند:

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

۳-۴ دستور x509

از این دستور برای مدیریت گواهی توسط یک مرکز صدور گواهی استفاده می‌شود. فرم کلی این دستور به‌شکل زیر می‌باشد:

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] [-addreject *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*]

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 می‌کند که می‌تواند PEM، NET یا DER باشد. مقدار پیشفرض آن PEM است.
-in *filename*	فایل ورودی را که شامل یک CSR است مشخص می‌کند. اگر از این گزینه استفاده نشود CSR ورودی از ورودی استاندارد خوانده می‌شود.
-out *filename*	فایلی را که CSR در آن نوشته‌شده مشخص می‌کند. اگر از این گزینه استفاده نشود CSR در خروجی استاندارد نوشته می‌شود.
-text	فیلدهای CSR را علاوه بر حالت کدشده به‌صورت متنی ساده نیز به خروجی می‌فرستد.
-noooout	با استفاده از این گزینه فرم کدشدهٔ فیلدهای CSR در خروجی ظاهر نمی‌شوند.
-modulus	قسمت modulus کلید عمومی داخل گواهی را در خروجی ظاهر می‌کند.
-serial	شمارهٔ سریال گواهی را در خروجی ظاهر می‌کند.
-hash	مقدار hash نام صاحب گواهی را در خروجی ظاهر می‌کند.
-subject	نام صاحب گواهی را در خروجی ظاهر می‌کند.

-issuer	نام صادرکنندهٔ گواهی را در خروجی ظاهر می‌کند.
-email	آدرس پست الکترونیکی صاحب گواهی را در خروجی ظاهر می‌کند.
-startdate	تاریخ صدور گواهی را در خروجی ظاهر می‌کند.
-enddate	تاریخ انقضای گواهی را در خروجی ظاهر می‌کند.
-dates	تاریخ صدور و انقضای گواهی را در خروجی ظاهر می‌کند.
-fingerprint	امضای دیجیتالی گواهی را در خروجی ظاهر می‌کند.
-signkey *filename*	استفاده از این گزینه باعث می‌شود که یک گواهی root با استفاده از کلید خصوصی قرارگرفته در *filename* ایجاد شود.
-keyfrom	فرمت کلید خصوصی ورودی را مشخص می‌کند که می‌تواند PEM یا DER باشد. مقدار پیشفرض آن PEM است.
-days *n*	اگر از گزینهٔ x509- استفاده شده‌باشد *n* میزان اعتبار گواهی را مشخص می‌کند.
-x509toreq	این گزینه یک گواهی را به یک CSR تبدیل می‌کند. از کلید خصوصی داده‌شده در گزینهٔ signkey- برای این کار استفاده می‌شود.
-req	در حالت پیشفرض یک گواهی به‌عنوان ورودی خوانده می‌شود. با استفاده از این گزینه یک CSR در ورودی خوانده می‌شود.
-CA *filename*	گواهی متعلق به مرکز صدور گواهی که برای امضا کردن استفاده می‌شود توسط این گزینه مشخص می‌شود.
-CAkey *filename*	کلید خصوصی مرکز صدور گواهی که برای امضا کردن از آن استفاده می‌شود توسط این گزینه مشخص می‌شود.
-CAserial *filename*	فایل حاوی شمارهٔ سریال certificateها را مشخص می‌کند. مقدار پیشفرض برای نام این فایل عبارت است از نام پایهٔ گواهی مربوط به مرکز صدور گواهی به‌علاوهٔ پسوند srl.
-CAcreateserial *filename*	فایل شمارهٔ سریال certificateها را ایجاد می‌کند.
-extfile *filename*	فایلی را که extentionهای مورد استفاده در آن قرار دارند مشخص می‌کند.
-extentions *section*	برای افزودن یک extention خاص به گواهی از این گزینه استفاده می‌شود.

دستور زیر اطلاعات متنی یک گواهی را نشان می‌دهد:

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

دستور زیر شمارهٔ سریال یک گواهی را نشان می‌دهد:

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

دستور زیر نام صاحب یک گواهی را نشان می‌دهد:

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

دستور زیر امضای دیجیتالی یک گواهی را نشان می‌دهد:

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

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

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

دستور زیر یک گواهی را به یک CSR تبدیل می‌کند:

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

دستور زیر یک CSR را به یک گواهی root تبدیل می‌کند:

```
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 می‌کند:

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

۴ استفاده از SSL برای امن کردن سرویس‌ها

برای امن کردن سرویس‌های مختلف توسط SSL دو حالت وجود دارد:

- سرویس موردنظر دارای امکانات SSL می‌باشد و می‌تواند اطلاعات رمزشده را دریافت و ارسال نماید (مانند سرور وب Apache). در این حالت فقط با پیکربندی سرور مربوطه امکان استفاده از SSL به آن اضافه می‌گردد.
- سرویس موردنظر دارای امکانات SSL نمی‌باشد و قادر به دریافت و ارسال اطلاعات رمزشده نمی‌باشد (مانند سرور IMAP). در این حالت باید از نرم‌افزارهایی تحت عنوان SSL Wrapper

استفاده کرد. این نوع نرم‌افزارها که از متداول‌ترین آنها می‌توان به Stunnel[11] اشاره کرد ترافیک ورودی رمزشده را دریافت کرده پس از رمزگشایی برای سرور می‌فرستند و ترافیک خروجی را از سرور دریافت کرده پس از رمزکردن ارسال می‌کنند.

در هر دو حالت برای کار با SLL نیاز به یک کلید خصوصی و گواهی می‌باشد که آنها را می‌توان بدین طریق ایجاد کرد:

- تولید یک کلید خصوصی:

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

- تولید یک CSR:

```
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 کافیست:

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

در این قسمت برای روشن‌تر شدن موضوع نحوهٔ استفاده از SSL به‌همراه سرور POP3 توضیح داده‌می‌شود.

برای استفاده از SSL به‌همراه سرور POP3 از نرم‌افزار Stunnel استفاده می‌شود. استفاده از Stunnel به دو طریق ممکن است:

- Stunnel اطلاعات رمزشده را از پورت شمارهٔ ۹۹۵ دریافت کرده و آنها را به پورت شمارهٔ ۱۱۰ بفرستد و این اطلاعات توسط سروری که به پورت شمارهٔ ۱۱۰ گوش می‌دهد پردازش شوند. دراین‌صورت دستور زیر را باید در scriptهای بوت سیستم قرار داد:

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

- Stunnel اطلاعات رمزشده را از پورت شمارهٔ ۹۹۵ دریافت کرده و یک سرویس را برای پردازش این اطلاعات رمزگشایی‌شده اجرا کند. دراین‌صورت دستور زیر را باید در scriptهای بوت سیستم قرار داد:

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

[11] http://www.stunnel.org

در هر دو دستور بالا stunnel.pem یک فایل است که به‌ترتیب شامل کلید خصوصی و گواهی سرور (به‌صورت رمزنشده) می‌باشد. در فایل stunnel.pem بعد از کلید خصوصی و نیز بعد از گواهی یک سطر خالی باید وجود داشته‌باشد.