
بسم الله الرحمن الرحيم

پیک ساماریوم

محمد مهدی اقدسی ۴۰۰۵۲۱۰۸۱

علی شکوهی ۴۰۰۵۲۱۴۷۷

دانیال یگانه ۴۰۰۵۲۲۳۳۷

آخرین ویرایش: ۱۵ دی ۱۴۰۳ در ساعت ۱۷ و ۵۷ دقیقه

فهرست مطالب

۱ سوالات تئوری

۱.۱ سوال اول

DTLS چیست؟

پروتکل **امنیت لایه انتقال دیتاگرام (DTLS)** یک پروتکل ارتباطی است که برای تأمین امنیت برنامه‌های مبتنی بر دیتاگرام طراحی شده است. این پروتکل یک گسترش از پروتکل **امنیت لایه انتقال (TLS)** است، اما بر روی **پروتکل دیتاگرام کاربر (UDP)** به جای TCP ساخته شده است. این پروتکل مناسب برنامه‌هایی است که به انتقال سریع داده نیاز دارند و حساس به تأخیرها می‌باشند، مانند بازی‌های آنلاین، استریم ویدئو و VoIP.

ویژگی‌های کلیدی DTLS:

- **امنیت:** DTLS با رمزگذاری بسته‌های داده، حفظ حریم خصوصی و یکپارچگی داده را تأمین می‌کند و مانع از جاسوسی، دستکاری و جعل پیام‌ها می‌شود.
- **مبتنی بر دیتاگرام:** برخلاف TCP، UDP تضمین نمی‌کند که بسته‌ها به ترتیب یا حتی به مقصد برسند. DTLS با مدیریت ترتیب‌دهی مجدد بسته‌ها و از دست رفتن بسته‌ها، تجربه‌ای مطمئن و امن از ارتباط را فراهم می‌کند.
- **کارایی:** DTLS از تأخیرهای مربوط به پروتکل‌های جریان داده جلوگیری می‌کند و آن را برای برنامه‌های زمان واقعی ایده‌آل می‌سازد.

جریان پیام در DTLS:

مرحله هم‌آهنگی

فرآیند هم‌آهنگی در DTLS به دلیل غیرقابل اعتماد بودن و نبود ترتیب در UDP کمی پیچیده‌تر از TLS است. مراحل اصلی شامل موارد زیر است:

ClientHello

- کاربر یک پیام **ClientHello** ارسال می‌کند که شامل نسخه پروتکل، مقادیر تصادفی، شناسه نشست، مجموعه‌های رمزنگاری و روش‌های فشرده‌سازی است.

ServerHello

- سرور با یک پیام **ServerHello** پاسخ می‌دهد که شامل فیلدهای مشابه **ClientHello** است. این پیام همچنین گواهی سرور (Certificate message) را شامل می‌شود و ممکن است گواهی کاربر نیز درخواست شود.

ServerKeyExchange

- اگر لازم باشد، سرور یک پیام **ServerKeyExchange** ارسال می‌کند. این پیام در صورتی استفاده می‌شود که گواهی سرور شامل اطلاعات کافی برای تکمیل تبادل کلید نباشد.

CertificateRequest (اختیاری)

- سرور ممکن است از کاربر بخواهد که یک گواهی ارائه دهد اگر نیاز به احراز هویت دو طرفه باشد.

ServerHelloDone

- سرور یک پیام **ServerHelloDone** ارسال می‌کند تا نشان دهد که مرحله پیام‌های هماهنگی کامل شده است.

گواهی کاربر (اختیاری)

- اگر سرور گواهی درخواست کرده باشد، کاربر پیام **Certificate** خود را ارسال می‌کند.

ClientKeyExchange

- کاربر یک پیام **ClientKeyExchange** ارسال می‌کند که حاوی کلید عمومی کاربر یا پیش رمز عبور لازم برای برقراری یک کلید مشترک است.

CertificateVerify (اختیاری)

- اگر نیاز به احراز هویت کاربر باشد، کاربر یک پیام **CertificateVerify** ارسال می‌کند تا گواهی خود را تایید کند.

پایان یافته

- پس از اینکه همه پیام‌های قبلی رد و بدل شدنده ر دو کاربر و سرور یک پیام **Finished** برای یکدیگر ارسال می‌کنند. پیام **Finished** رمزگذاری شده و اطمینان حاصل می‌کند که هم‌آهنگی موفقیت آمیز بوده و تمام پیام‌ها دستکاری نشده‌اند.

مرحله انتقال داده

پس از تکمیل هم‌آهنگی، داده‌ها می‌توانند به صورت امن بین کاربر و سرور منتقل شوند. داده‌ها رمزگذاری می‌شوند و DTLS صحت و اصالت بسته‌ها را تضمین می‌کند.

داده‌های کاربردی

- این مرحله شامل تبادل داده‌های رمزگذاری شده کاربردی بین کاربر و سرور است. DTLS مدیریت ترتیب‌دهی مجدد بسته‌ها، از دست رفتن و تکثیر بسته‌ها را برای اطمینان از انتقال مطمئن داده‌ها بر روی UDP به عهده دارد.

مرحله کلید مجدد (اختیاری)

برای جلسات طولانی مدت، ممکن است نیاز به کلید مجدد برای حفظ امنیت باشد. این شامل تکرار بخشی از هم‌آهنگی برای برقراری کلیدهای رمزنگاری جدید بدون قطع کردن جلسه جاری است.

درخواست کلید مجدد

- هر یک از طرفین می‌توانند با ارسال یک پیام **ClientHello** جدید، درخواست کلید مجدد کنند و فرآیند هم‌آهنگی مجدداً برای برقراری کلیدهای جدید آغاز می‌شود.

مرحله خاتمه

در نهایت، جلسه می‌تواند توسط کاربر یا سرور خاتمه یابد.

CloseNotify

- طرفی که می‌خواهد جلسه را خاتمه دهد، یک هشدار **CloseNotify** ارسال می‌کند. طرف دیگر با ارسال پیام **CloseNotify** خود پاسخ می‌دهد و ارتباط بسته می‌شود.

۲.۱ سوال دوم

حمله Downgrade در TLS زمانی رخ می‌دهد که مهاجم سعی می‌کند ارتباط را به نسخه‌ای قدیمی‌تر یا الگوریتم‌های ضعیف‌تر TLS کاهش دهد تا بتواند امنیت ارتباط را نقض کند. این حمله به مهاجم اجازه می‌دهد تا از ضعف‌های نسخه‌های قدیمی‌تر TLS بهره‌برداری کند.

انواع حمله Downgrade:

۱. **Protocol Downgrade**: کاهش نسخه پروتکل TLS به نسخه‌ای قدیمی‌تر مانند TLS 1.0 یا SSL

۲. **Cipher Suite Downgrade**: تغییر الگوریتم‌های رمزنگاری قوی به الگوریتم‌های ضعیف‌تر مانند RC4 یا DES

۳. **Feature Downgrade**: کاهش ویژگی‌های امنیتی مانند حذف Perfect Forward Secrecy (PFS)

روش‌های مقابله با حمله Downgrade:

۱. پیکربندی سرور و کلاینت به گونه‌ای که تنها نسخه‌ها و الگوریتم‌های قوی را پشتیبانی کنند.
۲. استفاده از مکانیزم‌های امنیتی مانند TLS_FALLBACK_SCSV که از کاهش ناخواسته نسخه جلوگیری می‌کند.
۳. استفاده از HSTS (HTTP Strict Transport Security) برای اطمینان از استفاده از TLS در ارتباطات.
۴. به‌روز نگه داشتن نرم‌افزارها و سرورها برای حمایت از آخرین نسخه‌های TLS.

۳.۱ سوال سوم

گواهی ریشه (Root Certificate) یک گواهی دیجیتال است که توسط یک مرجع صدور گواهی (Certificate Authority-CA) معتبر صادر شده و به عنوان پایه‌ای برای اعتبارسنجی دیگر گواهی‌ها عمل می‌کند. این گواهی‌ها به‌صورت پیش‌فرض در سیستم‌عامل‌ها و مرورگرهای وب ذخیره شده‌اند و به کاربران امکان می‌دهند تا هویت سرورها و سرویس‌های آنلاین را تایید کنند.

تأیید گواهی ریشه از سمت کاربر:

۱. **ذخیره‌سازی در مخزن گواهی‌های معتبر**: سیستم‌عامل و مرورگرها دارای یک مخزن از گواهی‌های ریشه معتبر هستند.
۲. **اعتبارسنجی زنجیره گواهی‌ها**: هنگام اتصال به یک سرور، گواهی سرور توسط زنجیره‌ای از گواهی‌ها تا گواهی ریشه دنبال می‌شود تا اطمینان حاصل شود که هر گواهی توسط گواهی بالاتر آن اعتبارسنجی شده است.
۳. **بررسی اعتبار و انقضای گواهی**: سیستم بررسی می‌کند که گواهی ریشه معتبر و منقضی نشده باشد.
۴. استفاده از پروتکل‌های امنیتی مانند OCSP (Online Certificate Status Protocol) برای بررسی وضعیت گواهی‌ها به صورت آنلاین.

۴.۱ سوال چهارم

Samy Kamkar یا سامی کامکار یک هکر و توسعه‌دهنده نرم‌افزار آمریکایی است که به خاطر فعالیت‌های امنیتی و برخی حملات معروف خود شناخته می‌شود. او به عنوان یک محقق امنیتی در زمینه امنیت وب و دستگاه‌های موبایل فعالیت می‌کند و چندین ابزار امنیتی و حملات معروف را توسعه داده است.

فعالیت‌های معروف سامی کامکار

۱. **حمله Cross-Site Scripting (XSS) در MySpace**: در سال ۲۰۰۵، او حمله‌ای XSS بر روی شبکه اجتماعی MySpace انجام داد که منجر به ایجاد پروفایل خود با نام "Samy" شد و به سرعت به ویروسی شدن پروفایل او منجر گردید.
۲. **Pineapple WiFi**: توسعه دستگاه‌هایی برای تست و تحلیل امنیت شبکه‌های WiFi.
۳. **Phantom**: یک ابزار برای سوءاستفاده از آسیب‌پذیری‌های DNS.

۴. **برنامه‌های مختلف امنیتی**: توسعه ابزارها و برنامه‌های مختلف برای تحلیل امنیت و تست نفوذ.

۲ سوالات عملی

۱.۲ سوال اول

برای استخراج جریان پیام TLS، نیازمند برنامه‌ای هستیم که بتواند با یک سرور ارتباط برقرار پیدا کرده و رازهای رمزنگاری را در مکانی ذخیره کند، و به برنامه‌ای نیاز داریم که بتواند بسته‌های ردوبدل شده را ضبط کرده و در فایلی ذخیره بکند تا بعدا بتوانیم به آنها دسترسی داشته باشیم. برای ارتباط با سرور از OpenSSL و برای ضبط بسته‌ها از tcpdump استفاده می‌کنیم.

اول دستور زیر را اجرا می‌کنیم تا بسته‌های ردوبدل شده را ضبط کنیم:

```
sudo tcpdump -i any host google.com -w output.pcap
```

آپشن‌های مورد استفاده:

- -i any: ترافیک را بر روی تمامی اینترفیس‌ها ضبط می‌کند.
- host google.com: بسته‌هایی را دریافت می‌کند که مرتبط با سرور ما (در اینجا google.com) می‌باشد.
- -w output.pcap: بسته‌های ضبط شده را در داخل فایل output.pcap ذخیره می‌کند.

سپس دستور زیر را اجرا می‌کنیم تا یک ارتباط با سرور مورد نظر (google.com) برقرار کنیم:

```
openssl s_client -connect google.com:443 -keylogfile /home/daniyeg/sslkeys.log -tls1_3
```

آپشن‌های مورد استفاده:

- s_client: برای برقراری ارتباط از آن استفاده می‌شود.
- -connect google.com:443: مقصد را مشخص می‌کند که در اینجا پورت ۴۴۳ (مربوط به HTTPS) در سایت google.com می‌باشد.
- -keylogfile /home/daniyeg/sslkeys.log: مشخص می‌کند که رازهای رمزنگاری این نشست در کدام فایل باید ذخیره شوند.
- -tls1_3: مشخص می‌کند که تنها از TLS 1.3 برای برقراری ارتباط استفاده شود.

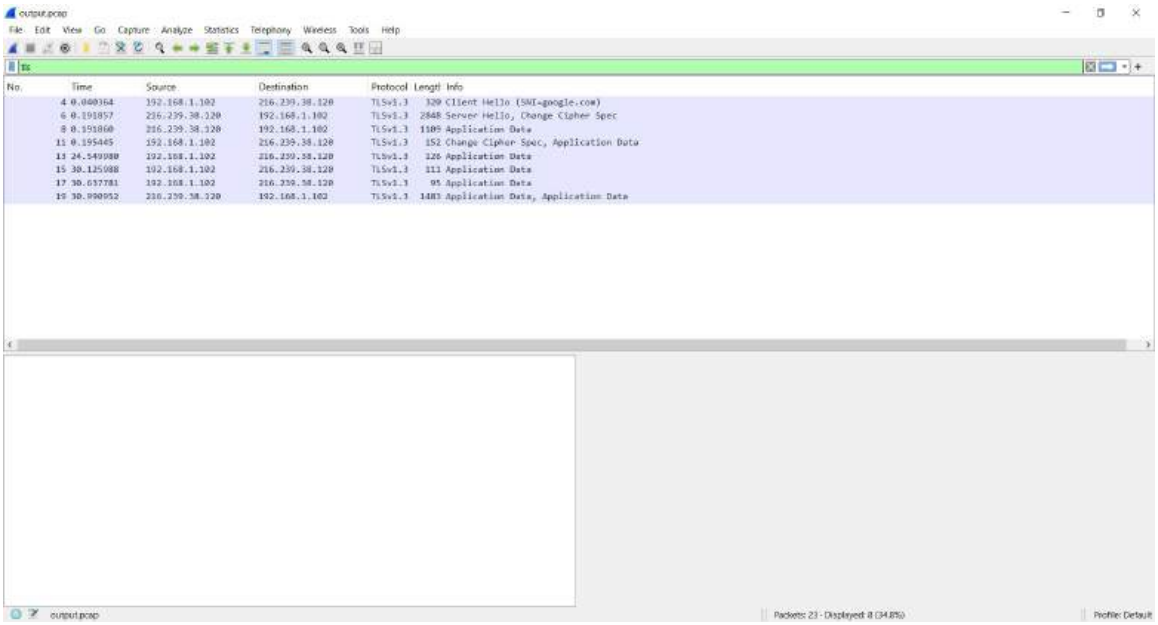
پس از برقراری ارتباط نیز درخواست HTTP ساده زیر را برای سرور می‌فرستیم و ارتباط را قطع می‌کنیم:

```
GET https://google.com HTTP/1.1
```

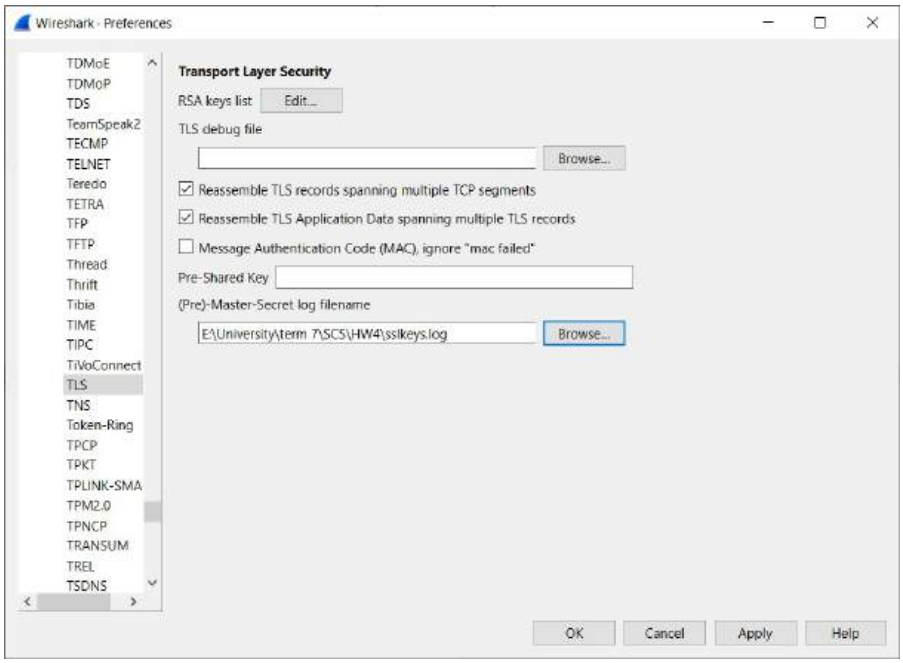
Host: google.com

حالا دارای دو فایل می‌باشیم: output.pcap که شامل بسته‌های ما می‌باشد و sslkeys.log که شامل رازهای رمزنگاری نشست می‌باشد. سپس می‌توانیم این فایل‌ها را در Wireshark باز کنیم تا محتوای آن را به صورت رمزگشایی شده بدست آوریم.

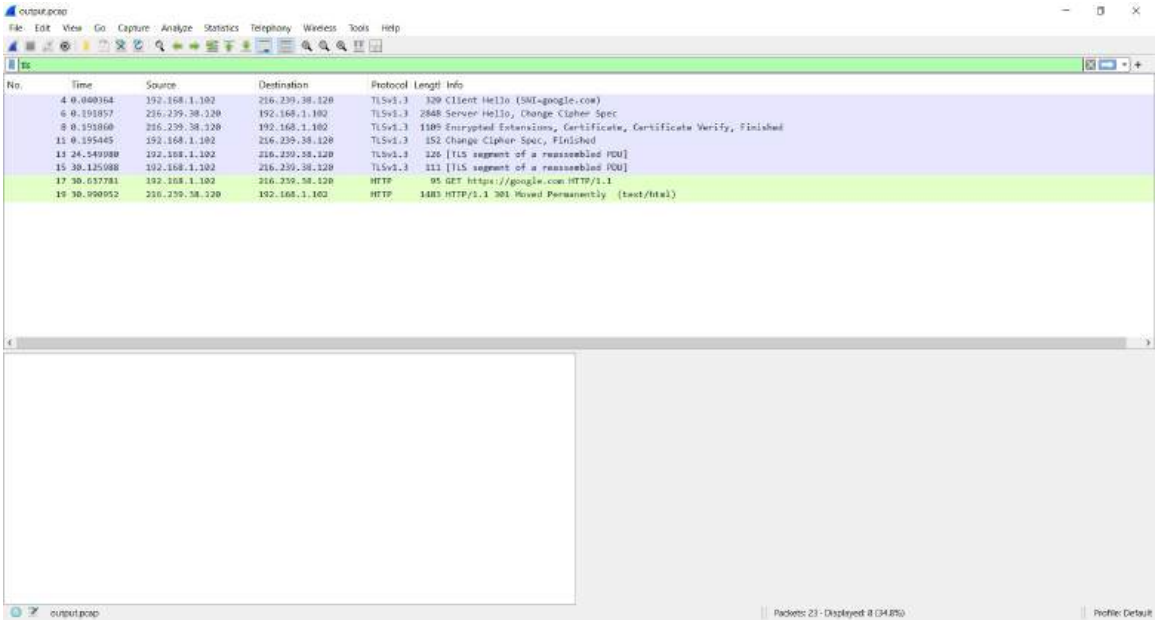
اول فایل output.pcap را در Wireshark باز می‌کنیم. همانطور که مشاهده می‌شود پیام‌های ما رمز شده می‌باشند:



سپس در برنامه Wireshark از منوی بالا به بخش Edit -> Preferences رفته و در منوی بغل Protocols -> TLS را انتخاب کرده و فایل مورد نظر خود را در بخش (Pre)-Master-Secret log filename انتخاب می‌کنیم.



حالا می‌توانیم محتویات پیام‌ها را به صورت رمزگشایی شده ببینیم. می‌توان مشاهده کرد که ۴ پیام برای برقراری ارتباط TLS رد و بدل شده است که در ادامه به محتویات آنها خواهیم پرداخت:



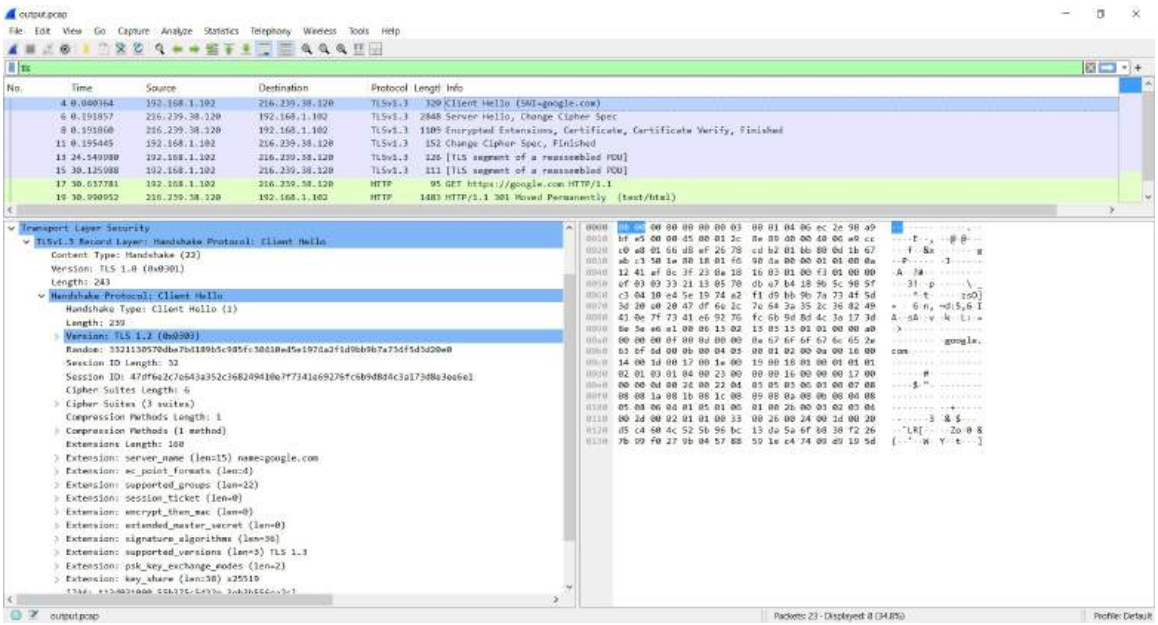
تصویر زیر محتویات پیام اول را نشان می‌دهد که شامل رکورد Client Hello می‌باشد. این رکورد شامل فیلدهای زیر می‌باشد:

- Version: فیلد نسخه که نسخه ارتباطات را مشخص می‌کند. این فیلد در TLS 1.3 استفاده نمی‌شود و همواره دارای مقادیر 0x301 به معنای TLS 1.0 یا 0x303 به معنای TLS 1.2 می‌باشد.
- Random: یک عدد رندوم ۳۲ بایتی که توسط کلاینت تولید شده است و در تولید کلیدهای نهایی برای رمزنگاری نقش دارد.
- Session ID: یک شناسه یکتا که مختص این اتصال می‌باشد و برای سرگیری اتصال در اتصال‌های بعد استفاده می‌شود تا سرعت اتصال مجدد بهبود یافته و میزان اطلاعات تکراری کمتر شود.
- Cipher Suites: الگوریتم‌های رمزنگاری را مشخص می‌کند که کلاینت از آنها پشتیبانی می‌کند و مایل است که در این اتصال از آن استفاده کند.
- Compression Methods: الگوریتم‌های فشرده‌سازی را مشخص می‌کند که کلاینت مایل است از آنها استفاده شود (در این مثال این فیلد خالی می‌باشد که به معنای فشرده نبودن پیام‌ها می‌باشد).

سپس چندین extension رد و بدل می‌شود که محتویات این اکستنشن‌ها برای عملکرد این پروتکل مهم است. اکستنشن‌های مهم در این پیام عبارتند از:

- server_name: دامنه سروری که کلاینت می‌خواهد به آن متصل شود.
- supported_groups: منحنی بیضی و گروه‌های میدان محدودی را که کلاینت برای تبادل کلید از آنها پشتیبانی می کند نشان می‌دهد و مربوط به رمزنگاری بیضوی دیفی-هلمن می‌باشد.
- session_ticket: این فیلد مشخص می‌کند که آیا کلاینت دارای Session Ticket ای می‌باشد یا خیر. از Session Ticket برای کاهش طول فرآیند دست دادن در اتصال مجدد به یک سرور استفاده می‌شود.
- encrypt_then_mac: این اکستنشن مشخص می‌کند که پیام‌ها اول باید رمز شوند و سپس توسط مکانیسم MAC یکپارچگی پیام‌ها تضمین شود.

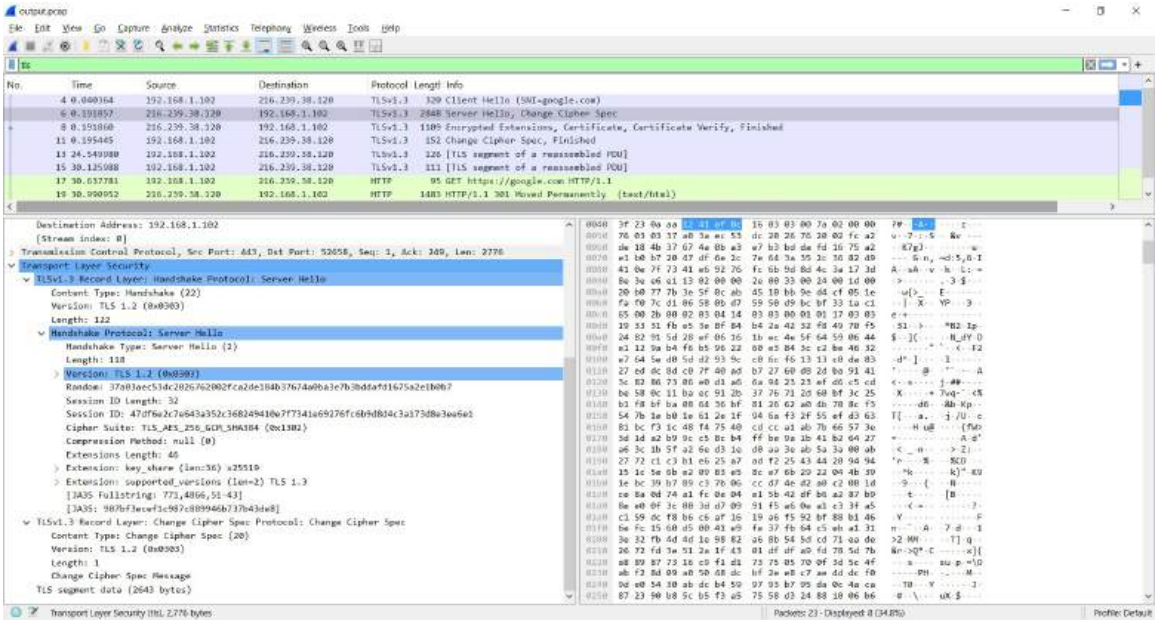
- signature_algorithms: الگوریتم‌های امضای دیجیتالی که کلاینت از انها پشتیبانی می‌کند.
- supported_versions: نسخه‌های TLS پشتیبانی شده توسط کلاینت. در این اکستنشن نسخه‌های واقعی پروتکل تعیین می‌شود.
- key_share: اطلاعات مربوط به توافق کلید کلاینت در روش‌های مبتنی بر دیفی-هلمن.



تصویر زیر محتویات پیام دوم را نشان می‌دهد که شامل دو رکورد می‌باشد. اولین رکورد یعنی Server Hello شامل فیلدهای زیر و اکستنشن‌های می‌باشد:

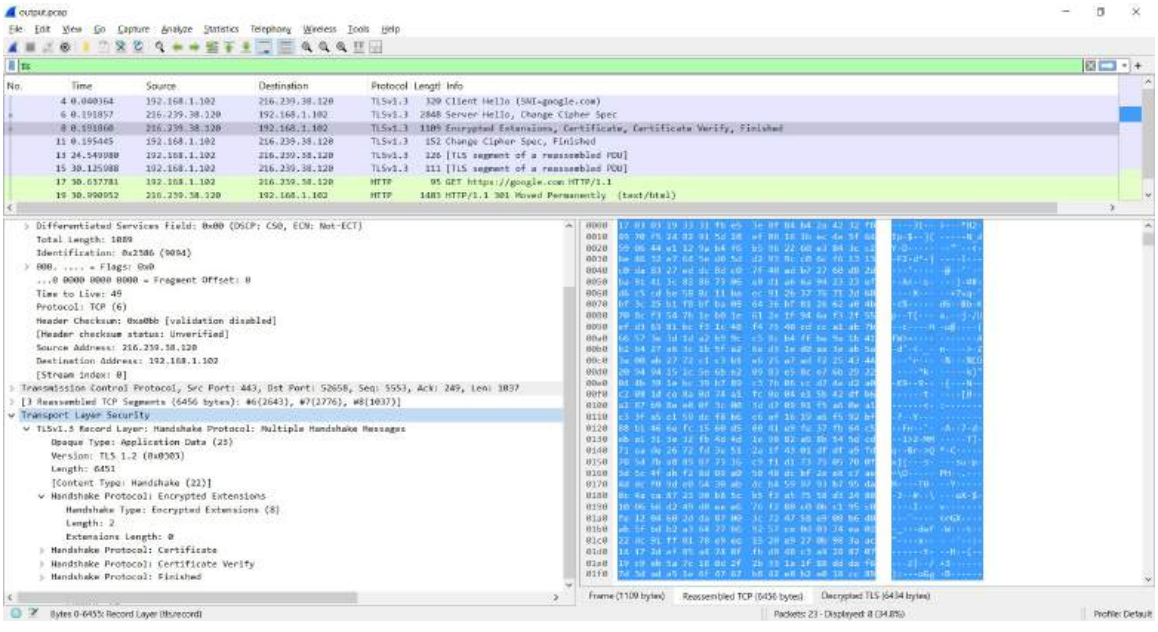
- Random: یک عدد رندوم ۳۲ بایتی که توسط سرور تولید شده است و در تولید کلیدهای نهایی برای رمزنگاری نقش دارد.
- Session ID: شناسه یکتایی که توسط کلاینت تولید شده و توسط سرور تائید شده است.
- Cipher Suite: الگوریتم رمزنگاری که قرار است از آن در این اتصال از آن استفاده شود.
- Compression Method: الگوریتم فشرده‌سازی که قرار است از آن استفاده شود (در این مثال از فشرده‌سازی استفاده نشده است).
- supported_versions: نسخه پروتکل TLS که از آن استفاده می‌شود. نسخه واقعی پروتکل در این فیلد مشخص شده است که TLS 1.3 می‌باشد.
- key_share: اطلاعات مربوط به توافق کلید سرور در روش‌های مبتنی بر دیفی-هلمن.

رکورد بعدی Change Cipher Spec می‌باشد که شامل فیلدی نبوده و صرفا به کلاینت اطلاع می‌دهد که از اینجا به بعد اطلاعات رد و بدل شده به صورت رمزی فرستاده می‌شوند.

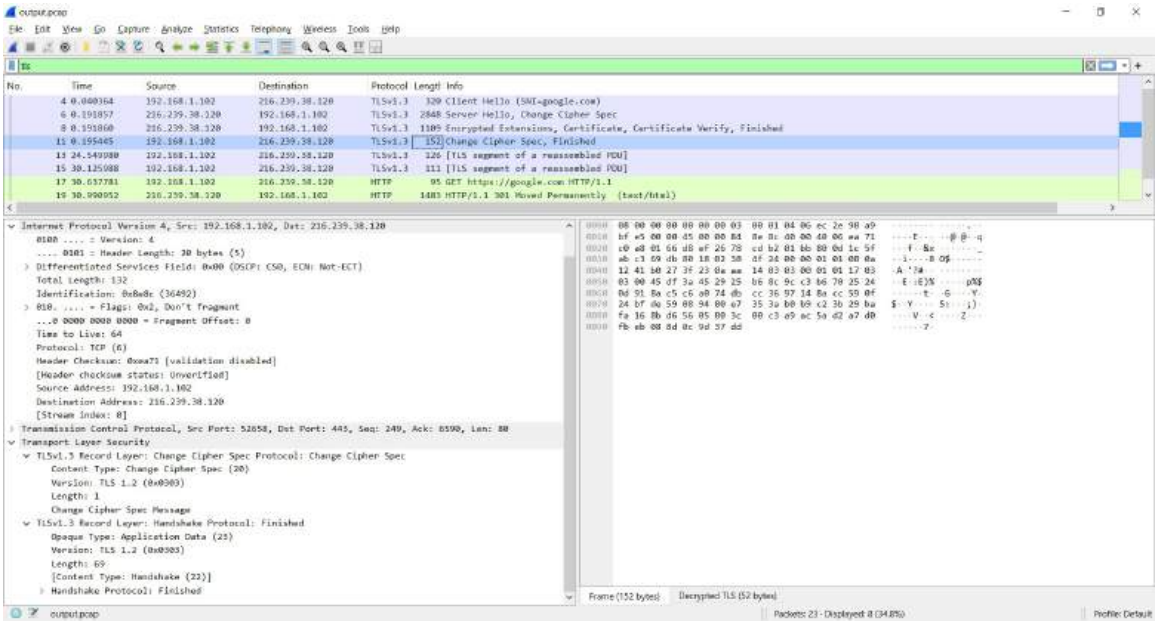


در پیام بعدی که نیز از سمت سرور می‌باشد چندین رکورد به صورت رمزنگاری شده وجود دارد. رکورد اول Encrypted Extensions می‌باشد که اکستنشن‌هایی را به صورت رمزی برای کلاینت می‌فرستد(در اینجا اکستنشنی وجود ندارد). رکورد بعدی صرفا شامل گواهی سرور برای تائید صحت کلید عمومی به اشتراک گذاشته شده در key_share پیام قبل می‌باشد. در رکورد بعدی به نام Cer-tificate Verify سرور در اختیار داشتن کلید خصوصی متناظر با کلید عمومی درج شده در گواهی را به کلاینت اثبات می‌کند. این رکورد شامل یک امضا (که الگوریتم هش آن در این رکورد نیز ذکر شده است) از تمام محتویات دست دادن TLS تا پایان ارسال گواهی می‌باشد که در صورتی که کلاینت بتواند این امضا را تایید کند در دست داشتن کلید خصوصی گواهی برای کلاینت اثبات می‌شود.

در آخر نیز رکورد Finished می‌باشد. این رکورد دارای یک هش (Verify Data) از تمامی پیام‌های ردوبدل شده در طول دست دادن می‌باشد تا کلاینت بتواند صحت اطلاعات پیام‌های دریافتی را با استفاده از مقایسه این هش با هش محاسبه شده توسط خود تایید کند.



تصویر زیر محتویات پیام چهارم و آخر را نشان می‌دهد که در آن دو رکورد TLS از سمت کلاینت به سمت سرور ارسال شده است. رکورد اول Change Cipher Spec می‌باشد که به سرور اطلاع می‌دهد از این به بعد کلاینت پیام‌های خود را به صورت رمز شده ارسال می‌کند. رکورد دوم نیز رکورد Finished می‌باشد که هدف آن در بخش قبلی توضیح داده شد.



در نهایت می‌توان پیام‌های HTTP/1.1 ردوبدل شده را مشاهده کرد.

مزایای TLS 1.3 نسبت به TLS 1.2

افزایش امنیت

- **دست دادن ساده‌تر:** TLS 1.3 لگوریتم‌ها و رمزهای رمزنگاری کمتر امن را حذف کرده و فرآیند دست دادن را ساده‌تر می‌کند که باعث می‌شود در برابر حملات مقاوم‌تر باشد.
- **رمزنگاری با راز رو به جلو:** در این نسخه تنها اجازه استفاده از رمزهایی که رمزنگاری با راز رو به جلو را فراهم می‌کنند داده می‌شود تا اطمینان حاصل شود که کلیدهای نشست حتی در صورت به خطر افتادن کلیدهای بلندمدت امن باقی می‌مانند.
- **encrypt-then-MAC:** به‌صورت پیش‌فرض از روش رمزنگاری-سپس-رمز صحت پیروی می‌کند و داده‌ها را یکپارچه و امن می‌سازد.

بهبود عملکرد

- **کاهش تأخیر:** فرآیند دست دادن به تعداد کمتری رفت و برگشت بین کلاینت و سرور نیاز دارد که منجر به زمان اتصال سریع‌تر می‌شود. در بسیاری از موارد به سرعت یک رفت و برگشت واحد است.
- **از سرگیری با صفر رفت و برگشت 0-RTT:** این ویژگی اجازه استفاده مجدد از نشست قبلی برای اتصالات سریع‌تر بعدی را بدون به خطر انداختن امنیت می‌دهد.

بهبود حریم خصوصی

- **راز کامل به جلو:** هر نشست از یک کلید موقتی جدید استفاده می‌کند که باعث می‌شود داده‌های نشست‌های گذشته در صورت به خطر افتادن کلیدهای بلندمدت غیرقابل دسترس باشند.
- **نمایش کمتر متاداده:** مقدار متاداده‌ی نمایش داده‌شده در فرآیند دست دادن را به حداقل می‌رساند که حریم خصوصی بهتری برای کاربران فراهم می‌کند.

ساده‌سازی پروتکل

- **رمزهای ساده‌تر:** تعداد رمزهای پشتیبانی‌شده را کاهش داده و گزینه‌های منسوخ و کمتر امن را حذف می‌کند.
- **مشخصات تمیزتر:** پروتکل ساده‌تر و آسان‌تر برای پیاده‌سازی به صورت صحیح است که احتمال وجود اشکال و آسیب‌پذیری را کاهش می‌دهد.

۲.۲ سوال دوم

در این بخش، به پیاده‌سازی طرح تسهیم راز Shamir برای تقسیم یک رمز عبور ۵۱۲ بیتی به ۱۰ سهم با آستانه ۵ پرداخته شده است. هدف این طرح این است که اگر حداقل ۵ نفر از سهم‌ها جمع شوند، بتوانند راز اصلی را بازسازی کنند و در صورت جمع شدن کمتر از ۵ سهم، اطلاعاتی از راز قابل استخراج نباشد.

بخش اول: شرح طرح تسهیم راز Shamir

Shamir's Secret Sharing یک روش رمزنگاری مبتنی بر نظریه اشکال است که به شما امکان می‌دهد یک راز را به چندین سهم تقسیم کنید به طوری که حداقل تعداد مشخصی از این سهم‌ها برای بازسازی راز اصلی نیاز باشد. این روش از خصوصیات چندجمله‌ای برای اطمینان از امنیت راز استفاده می‌کند.

مراحل طرح Shamir:

۱. **نمایش راز به عنوان یک عدد بزرگ:** رمز عبور ۵۱۲ بیتی به عنوان یک عدد بزرگ در نظر گرفته می‌شود.
۲. **ایجاد یک چندجمله‌ای با درجه t-1 (در اینجا ۴):** ضرایب چندجمله‌ای به طور تصادفی انتخاب می‌شوند.
۳. **محاسبه سهم‌ها:** هر سهم به صورت نقطه‌ای (x، y) روی این چندجمله‌ای محاسبه می‌شود.
۴. **بازیابی راز:** با داشتن حداقل t سهم، می‌توان از طریق روش‌های ریاضی مانند تداخل لگرانژ، چندجمله‌ای را بازسازی کرد و راز را بازیابی نمود.

بخش دوم: پیاده‌سازی طرح تسهیم راز Shamir با Python

در ادامه، پیاده‌سازی ساده‌ای از طرح تسهیم راز Shamir با استفاده از Python ارائه شده است.

شرح کد:

- **mod_inv(a, p)**: محاسبه معکوس ماژولی عدد 'a' در میدان عددی 'p' با استفاده از قضیه کوچک فرما.
- **generate_polynomial(secret, threshold, prime)**: ایجاد یک چندجمله‌ای با درجه 'threshold - ۱' که ضریب آزاد آن برابر راز است و بقیه ضرایب به صورت تصادفی انتخاب می‌شوند.
- **evaluate_polynomial(coefficients, x, prime)**: محاسبه مقدار 'y' برای یک مقدار خاص 'x' در چندجمله‌ای داده شده.
- **split_secret(secret, n, threshold, prime)**: تقسیم راز به 'n' سهم با آستانه 'threshold' با استفاده از چندجمله‌ای تولید شده.
- **reconstruct_secret(shares, prime)**: بازسازی راز اصلی از طریق سهم‌ها با استفاده از تداخل لگرانژ.

تست و تأیید عملکرد کد

برای اطمینان از صحت عملکرد کد، مراحل زیر انجام شده است:

۱. **تولید راز تصادفی ۵۱۲ بیتی:** یک عدد تصادفی ۵۱۲ بیتی به عنوان راز تولید شده است.
۲. **تقسیم راز به ۱۰ سهم با آستانه ۵:** راز به ۱۰ سهم تقسیم شده است.
۳. **انتخاب تصادفی ۵ سهم برای بازسازی راز:** ۵ سهم به طور تصادفی انتخاب شده‌اند.
۴. **بازسازی راز و مقایسه با راز اصلی:** راز بازسازی شده با راز اصلی مقایسه شده و صحت آن تایید گردیده است.

بخش سوم: گزارش و تحلیل

توضیحات کلی

در این پروژه، از زبان برنامه‌نویسی Python برای پیاده‌سازی طرح تسهیم راز Shamir استفاده شده است. هدف اصلی این پیاده‌سازی، تقسیم یک رمز عبور ۵۱۲ بیتی به ۱۰ سهم با آستانه ۵ بوده است. این طرح اطمینان می‌دهد که اگر حداقل ۵ نفر از سهم‌ها جمع شوند، راز اصلی قابل بازسازی است و در صورت جمع شدن کمتر از ۵ سهم، راز اصلی تقریباً غیرقابل استخراج باقی می‌ماند.

مزایا و معایب طرح Shamir

مزایا:

- **امنیت بالا:** بازسازی راز نیازمند حداقل تعداد مشخصی از سهم‌ها است و اگر تعداد سهم‌های جمع‌شده کمتر از آستانه باشد، اطلاعاتی از راز قابل استخراج نیست.
- **قابلیت انعطاف‌پذیری:** امکان تنظیم تعداد کل سهم‌ها و آستانه بازسازی راز به دلخواه.
- **سادگی پیاده‌سازی:** پیاده‌سازی ساده و قابل درک با استفاده از مفاهیم ریاضی پایه.

معایب:

- **نیاز به حافظه زیاد:** برای رمزهای بسیار بزرگ، اندازه چندجمله‌ای و محاسبات ممکن است نیاز به حافظه زیادی داشته باشد.
- **پیچیدگی محاسباتی:** بازسازی راز نیازمند محاسبات پیچیده‌ای مانند تداخل لگرانژ است که ممکن است برای مجموعه‌های بسیار بزرگ بهینه نباشد.

نتایج به دست آمده

با اجرای کد ارائه شده، یک رمز عبور تصادفی ۵۱۲ بیتی تولید شده و به ۱۰ سهم تقسیم گردید. سپس با انتخاب تصادفی ۵ سهم از میان این ۱۰ سهم، راز اصلی بازسازی شد و صحت بازسازی با راز اصلی تایید گردید.

- **سهم‌ها:** ۱۰ سهم تولید شدند که هر کدام شامل یک عدد 'x' و مقدار 'y' هستند.
- **سهم‌های انتخاب شده برای بازسازی:** ۵ سهم به طور تصادفی انتخاب شدند.
- **بازسازی راز:** راز بازسازی شده با راز اصلی برابر بود، که نشان‌دهنده صحت پیاده‌سازی طرح تسهیم راز Shamir است.

کد پیاده‌سازی

کد پیاده‌سازی طرح تسهیم راز Shamir به‌صورت کامل در بخش سوال عملی دوم آورده شده است. این کد شامل توابع محاسبه معکوس ماژولی، ایجاد چندجمله‌ای، ارزیابی چندجمله‌ای، تقسیم راز به سهم‌ها و بازسازی راز از طریق تداخل لگرانژ می‌باشد.

بخش چهارم: نتیجه‌گیری

در این گزارش، به حل سوالات نظری و عملی مرتبط با پروتکل‌های امنیتی و طرح تسهیم راز پرداخته شد. در بخش نظری، مفاهیم اساسی TLS و DTLS، حملات Downgrade و روش‌های مقابله با آن‌ها، گواهی‌های ریشه و نقش Kamkar Samy مورد بررسی قرار گرفتند. در بخش عملی، طرح تسهیم راز Shamir به‌طور کامل پیاده‌سازی و تست شد و نتایج نشان‌دهنده صحت و قابلیت اطمینان این طرح بود.

طرح تسهیم راز Shamir به عنوان یک روش مؤثر برای محافظت از اطلاعات حساس در شرایطی که نیاز به تقسیم‌سازی و توزیع اطلاعات میان چندین نهاد وجود دارد، بسیار کاربردی و مؤثر می‌باشد. این طرح با استفاده از مفاهیم ریاضی پایه، امنیت بالایی را فراهم می‌کند و از دسترسی غیرمجاز به راز اصلی جلوگیری می‌کند.

بخش پنجم: توابع مهم پیاده‌سازی

در این بخش، به توضیح توابع کلیدی پیاده‌سازی در کد Python پرداخته می‌شود.

تابع mod_inv(a, p)

این تابع معکوس ماژولی عدد 'a' را در میدان عددی 'p' محاسبه می‌کند. از قضیه کوچک فرما برای محاسبه معکوس ماژولی استفاده شده است.

تابع generate_polynomial(secret, threshold, prime)

این تابع یک چندجمله‌ای با درجه ‘threshold - ۱’ ایجاد می‌کند که ضریب آزاد آن برابر راز (‘secret’) است و بقیه ضرایب به صورت تصادفی انتخاب می‌شوند.

تابع evaluate_polynomial(coefficients, x, prime)

این تابع مقدار ‘y’ را برای یک مقدار خاص ‘x’ در چندجمله‌ای داده شده محاسبه می‌کند.

تابع split_secret(secret, n, threshold, prime)

این تابع راز را به ‘n’ سهم تقسیم می‌کند که هر سهم شامل یک عدد ‘x’ و مقدار ‘y’ است. حداقل ‘threshold’ سهم برای بازسازی راز نیاز است.

تابع reconstruct_secret(shares, prime)

این تابع راز اصلی را از طریق سهم‌ها با استفاده از تداخل لگرانژ بازسازی می‌کند. حداقل ‘threshold’ سهم برای بازسازی راز لازم است.

بخش ششم: چالش‌ها و راهکارها

در فرآیند پیاده‌سازی طرح تسهیم راز ،Shamir با چالش‌های مختلفی مواجه شدیم که مهم‌ترین آن‌ها عبارت بودند از:

- **انتخاب و تنظیمات صحیح عدد اول بزرگ:** انتخاب عدد اول مناسب که بزرگ‌تر از راز باشد و عملیات ریاضی در آن صحیح انجام شود از چالش‌های اصلی بود. استفاده از اعداد اول بزرگ مانند اعداد Mersenne به دلیل ساده بودن محاسبات و امنیت بالا، انتخاب مناسبی بودند.
- **پیاده‌سازی صحیح تداخل لگرانژ:** اطمینان از صحت پیاده‌سازی روش تداخل لگرانژ برای بازسازی راز نیازمند درک عمیق از مفاهیم ریاضی پایه بود. تست‌های متعدد با داده‌های مختلف به ما کمک کرد تا از صحت عملکرد کد اطمینان حاصل کنیم.
- **مدیریت حافظه و کارایی:** برای رازهای بسیار بزرگ، مدیریت حافظه و بهینه‌سازی عملکرد پیاده‌سازی ضروری بود. استفاده از روش‌های بهینه‌سازی و کاهش پیچیدگی محاسباتی به بهبود کارایی کمک کرد.

راهکارهای مقابله با چالش‌ها:

- **مطالعه دقیق نظریه پشت طرح Shamir:** برای درک بهتر از مفاهیم و الگوریتم‌های مورد استفاده، مطالعه منابع معتبر در زمینه نظریه اشکال و رمزنگاری بسیار مؤثر بود.
- **تست‌های جامع و متعدد:** اجرای تست‌های متعدد با داده‌های مختلف برای اطمینان از صحت عملکرد کد و بازسازی صحیح راز.
- **بهینه‌سازی کد:** استفاده از روش‌های بهینه‌سازی در پیاده‌سازی کد برای کاهش زمان اجرای برنامه و مدیریت بهتر حافظه.

بخش هفتم: پیشنهادات

برای بهبود عملکرد و افزایش دقت طرح تسهیم راز ،Shamir پیشنهاد می‌شود:

- **افزایش تعداد سهم‌های تست شده:** افزایش تعداد سهم‌های تست شده به بیش از ۱۰ سهم برای ارزیابی بهتر و دقیق‌تر عملکرد طرح.
- **استفاده از الگوریتم‌های جایگزین:** بررسی و مقایسه طرح‌های تسهیم راز مختلف مانند طرح‌های مبتنی بر کریپتوگرافی دیگر برای انتخاب بهترین گزینه بر اساس نیازهای امنیتی.
- **بهینه‌سازی محاسبات ریاضی:** استفاده از کتابخانه‌های بهینه‌سازی‌شده برای انجام محاسبات ریاضی پیچیده و افزایش سرعت بازسازی راز.
- **استفاده در کاربردهای عملی:** پیاده‌سازی این طرح در سیستم‌های واقعی برای حفاظت از کلیدهای رمزنگاری و اطلاعات حساس.