گزارش فنی امنیت داده و شبکه

دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف

آرین احدی نیا، روزبه پیراعیادی، آرین یزدان‌پرست

استاد درس: جناب آقای دکتر امینی

بهار ۱۴۰۲

فهرست مطالب

۱ مقدمه

با توجه به افزایش اطلاعات در گردش و همچنین افزایش نیاز به استفاده از ابزارهای پیام‌رسان، امنیت این ابزارها به چالشی بدل شده است. در این پروژه قصد داریم با استفاده از برخی پروتکل‌های پیشنهادی و همین طور برخی از پروتکل‌های مرسوم بر برخی از این چالش ها فائق آییم.

۲ پروتکل‌های مورد استفاده

برای پیاده‌سازی امن پروژه ما از سه پروتکل امنیتی استفاده کرده‌ایم. پروتکل اول به نام HTTPSL برای برقراری ارتباط اولیه استفاده می‌شود که در بخش (۱.۲) به تفصیل آن را توضیح داده‌ایم. سپس پس از برقراری ارتباط اولیه امن از این پروتکل برای بدست آوردن یک کلید متقارن برای افزایش سرعت و عملکرد ارتباط با سرور استفاده می‌کنیم. این پروتکل نیز در بخش (۲.۲) به تفصیل توضیح داده شده است. هر دوی این پروتکل‌ها برای برقراری ارتباط امن کارخواه با کارگزار هستند. از پروتکل جغجغه دوطرفه برای برقراری ارتباط امن انتها به انتها بین دو کاربر برای رمزگزاری پیام‌هایشان استفاده می‌شود. این پروتکل در یک بخش مجزا در بخش (۳) توضیح داده شده است.

HTTPSL ۱.۲

این پروتکل از جهاتی شبیه به HTTPS است. این پروتکل بدون حالت است و پس از ارسال درخواست و دریافت پاسخ اتصال را قطع می‌کند. استفاده از این پروتکل برای ارسال درخواست از طرف کارخواه به سمت کارگزار و دریافت پاسخ آن است.

این پروتکل کاملا مبتی بر رمزنگاری غیرمتقارن و در اختیار داشتن کلید عمومی مخاطب برای رمزنگاری و گواهی کلید عمومی برای امضای دیجیتال است. در این روش پیام و مهر زمانی و امضای دیجیتال آن را کنار هم دیگر قرار می‌دهیم و کل آن را با استفاده از کلید عمومی مخاطب رمز می‌کنیم. در تمام این مراحل با استفاده از base64 پیام را به کاراکترهای قابل ارسال تبدیل می‌کنیم. از مهرزمانی نیز برای اطمینان از تازه بودن پیام استفاده می‌شود.

$$A \longrightarrow B : E(PU_B, M||E(PR_A, H(M))||TS)$$

SSE: Symentric Session Encryption ۲.۲

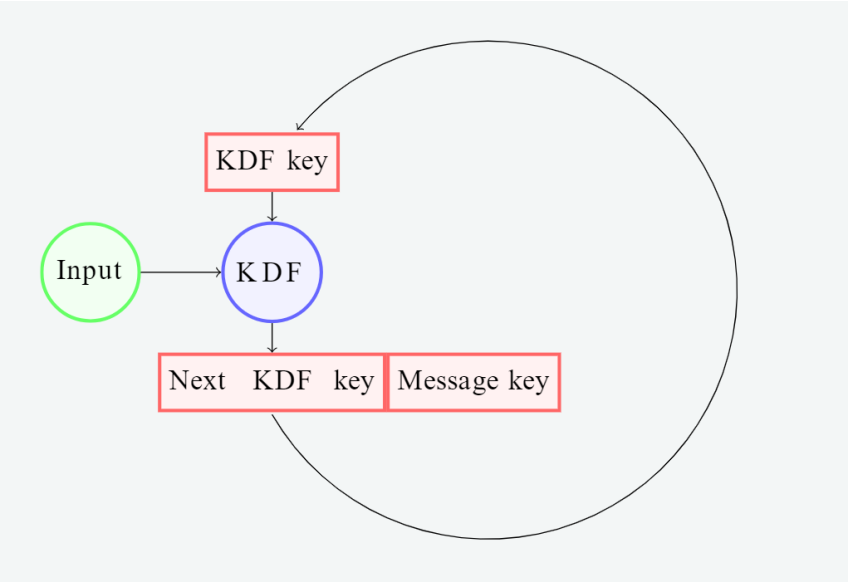
این پروتکل بر مبنای امضا با رمزنامتقارن و رمزنگاری با رمز متقارن AES است. پس از تبادل کلید با HTTPSL می‌توانیم با استفاده از یک کلید AES که به صورت امن تبادل شده است، با استفاده از یک کلید متقارن رمزنگاری کنیم و در کنار آن امضا با استفاده از کلید نامتقارن قرار دهیم. همچنین مهر زمانی نیز در این پروتکل جهت بررسی تازگی پیام‌های ارسال شده قرار می‌دهیم. از پروتکل برای ارتباط هر دو طرف با هم استفاده می‌شود. همچنین برای اینکه پیام برای سرور قابل شناسایی باشد، توکن به صورت رمز شده با کلید عمومی سرور در کنار این پیام ارسال می‌شود.

$$A \longrightarrow B : E(K_{A,B}, M||E(PR_A, H(M))||TS)||E(PU_B, Token)$$

۳ پروتکل جغجغه دو طرفه

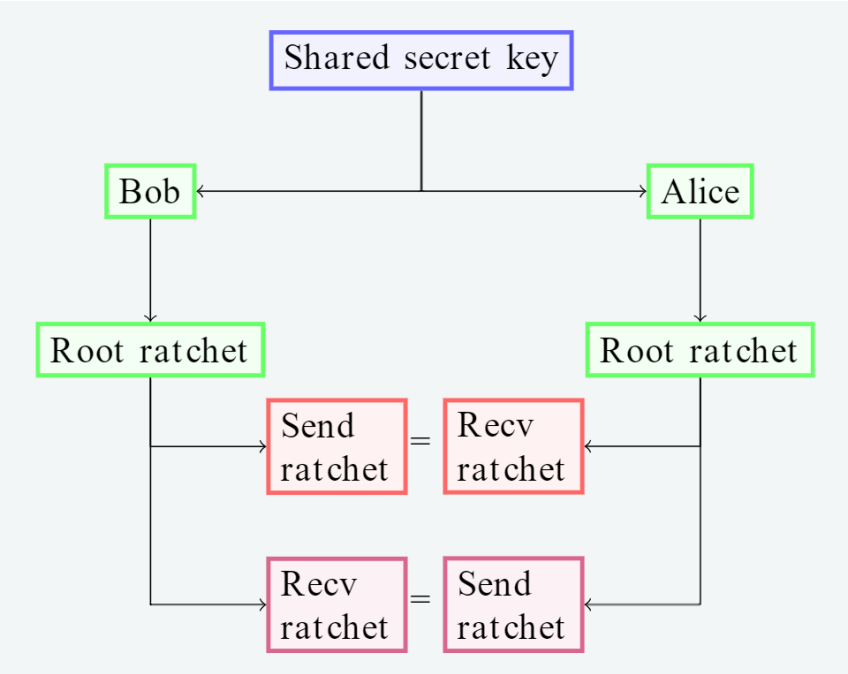
برای برقرار کردن نیازمندی‌های محرمانگی پیشرو و محرمانگی آینده از پروتکل double ratchet استفاده می‌کنیم. [۱]

این پروتکل مبتنی بر ساختاری به نام ratchet است. نمای کلی یک ratchet را می‌توانید در شکل (۱.۳) یک ratchet با هر بار چرخش کلید جدیدی تولید می‌کند. ویژگی مهمی که این ساختار دارد این است که مهاجم با به دست آوردن یکی از کلیدها نمی‌تواند ratchet را به عقب بچرخاند و به کلیدهای قبلی دست پیدا کند. به همین دلیل با استفاده از یک ratchet می‌توان ویژگی محرمانگی پیشرو را فراهم کرد. اما چنین ساختاری ویژگی محرمانگی آینده را ندارد. در اصل اگر یکی از پیام‌ها افشا شود و مهاجم بتواند از آنجا به بعد ورودی‌ها را نیز رصد کند، تمام کلیدها برای او مشخص خواهند بود.



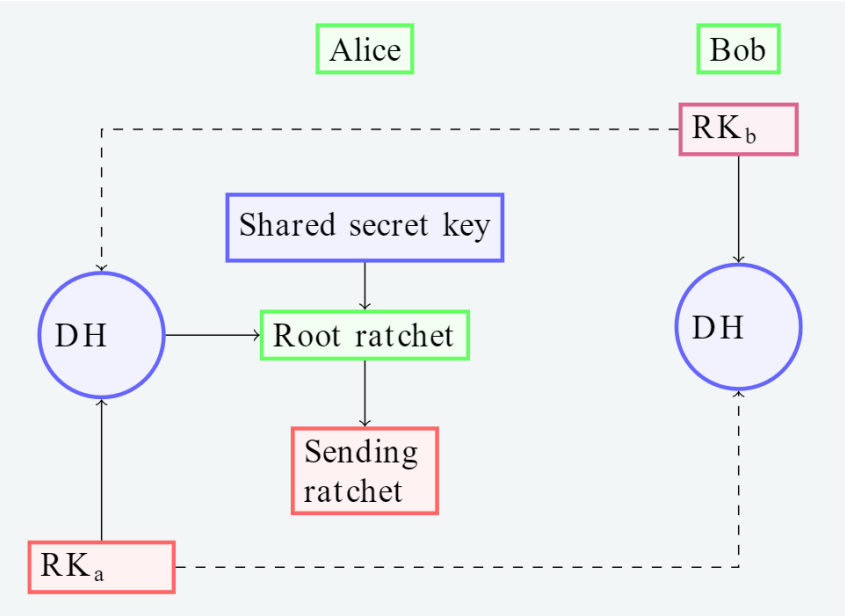
شکل ۱.۳

موضوع دیگری که وجود دارد این است که اگر هر کدام از طرفین تنها یک ratchet داشته باشند، نمی‌توانند آنها را با یکدیگر همگام نگه دارند. در واقع ممکن است نفر اول و دوم در زمان تقریبا یکسانی تصمیم به ارسال پیام به یکدیگر بگیرند و برای این کار هر دو ratchet های خود را یکبار خواهند چرخاند و هرکدام پیام خود را با کلید حاصل از این کار رمز خواهند کرد. به همین دلیل هر کدام از طرفین می‌بایست یک ratchet صرفا برای دریافت و یکی هم صرفا برای ارسال داشته باشند. توجه داشته باشید که ratchet مربوط به ارسال یک طرف باید با ratchet مربوط به دریافت دیگر همگام باشد و برعکس. مقدار اولیه‌ی ratchet ها نیز توسط یک ratchet به نام root تعیین می‌شود. دقت داشته باشید که یک طرف باید اولین مقدار تولیدی توسط root را برای ارسال و دیگری اولین مقدار تولیدی توسط root را برای دریافت در نظر بگیرد. به علت همین عدم تقارن ها در رفتار کسی که شروع کننده‌ی ارتباط است با کسی که نفر دوم به حساب می‌آید، در کد کتابخانه‌های مجزای first_person_ratchet و second_person_ratchet در نظر گرفته شده‌اند.



شکل ۲.۳

برای این که ویژگی محرمانگی آینده را هم به این پروتکل اضافه کنیم، می‌بایست دائما وضعیت ratchet ریشه و به تبع آن ratchet های مربوط به ارسال و دریافت را تغییر دهیم. برای این کار هر بار که فرد می‌خواهد پیام جدیدی ارسال کند، کلید دیفی‌ـ هلمن جدیدی برای خودش تولید می‌کند و با کمک آن کلید جدید وضعیت ratchet هایش را تغییر می‌دهد. برای آنکه طرف مقابل بتواند پیام‌ها را رمزگشایی کند، این فرد می‌بایست کلید عمومی دیفی‌ـ هلمنی که به تازگی تولید کرده است را برای او بفرستد. حالا در طرف دیگر، فرد مقابل که پیام را دریافت کرده است، باید ابتدا با کمک کلید عمومی جدید دریافت شده و کلید قدیمی خود، وضعیت ratchet را تعیین کند. با انجام این کار او نیز می‌تواند پیام را رمزگشایی کند. با انجام چنین کاری برنامه قابلیت break-in recovery خواهد داشت به این معنا که اگر هم مهاجمی به یکی از کلیدها دست پیدا کند، چون وضعیت ratchet ها توسط کلیدهای دیفی‌ـ هلمن جدید تغییر می‌کنند، عملا امکان دسترسی به پیام‌های بعدی را نخواهد داشت. در آخر دقت داشته باشید که تمام این پیام‌ها بر روی پروتکل SSE جابه‌جا می‌شوند و عملا مهاجم بدون در اختیار داشتن کلید جلسه حتی قادر به مشاهده‌ی پارامترهای عمومی دیفی‌ـ هلمن هم نخواهد بود.


Alice
Bob
RK$_b$
Shared secret key
DH
Root ratchet
DH
Sending ratchet
RK$_a$

شکل ۳.۳

۴ پیاده‌سازی کارخواه

۱.۴ ایجاد حساب کاربری

هر کاربر می‌تواند با وارد کردن نام‌کاربری و رمزعبور یک حساب کاربری جدید ایجاد کند. برای هر کاربر یک جفت کلید نامقتارن در زمان ثبت‌نام ایجاد می‌شود و کلید عمومی آن به سرور اعلام می‌گردد. این کار با دستور `register` انجام می‌شود. پس از آن کاربر می‌تواند با دستور `login` وارد شود.

۲.۴ نمایش کاربران آنلاین

برای نمایش کاربران آنلاین سرور به نحوی که در (۳.۵) توضیح داده شده است عمل می‌کند و لیست کاربران آنلاین را در اختیار سایرین قرار می‌دهد. کاربر می‌تواند با دستور `list_onlines` به لیست کاربران آنلاین دسترسی داشته باشد.

۳.۴ ارسال و دریافت پیام

هر کاربر می‌تواند با استفاده از دستور `send` پیام مورد نظر خود را ارسال کند. پیام وی با استفاده از پروتکل جغجغه دوطرفه رمز می‌شود.

۴.۴ نگهداری پیام‌ها و کلید به صورت امن

برای هر کاربر یک کلید مشتق از رمزعبور با استفاده از درهم‌آمیزی رمز وی با الگوریتم SHA256 بدست می‌آوریم. این کلید ۲۵۶ بیتی را می‌توانیم به عنوان کلید AES استفاده کنیم و تمام اطلاعات موجود در سیستم کاربر را با آن رمز کنیم.

۵.۴ ایجاد و مدیریت گروه

کاربر می‌تواند با استفاده از دستور create_group یک گروه جدید با مدیریت خودش ایجاد کند و با استفاده از دستور add_user_to_group و remove_user_from_group کاربران دیگر را به این گروه اضافه و یا از آن حذف کند. برای ارسال پیام در گروه مشابه گفتوگوهای دونفره عمل می‌شود و پیام برای تک تک مخاطبین آن رمز می‌شود. در حال حاضر این بهترین روش مبادله پیام انتها–به–انتها رمزشده است و روش کارآمدی برای رمزنگاری به صورتی که محرمانگی پیشرو و آینده را تضمین کند وجود ندارد.

۶.۴ تایید صحت نشست از طریق کانال امن جانبی

در زمان تبادل کلید بین دو کاربر، چهار ایموجی با استفاده از درهم‌آمیخته شده کلید مشترک دو طرف تولید کرده و آن را به کاربر نمایش می‌دهیم. برای این درهم‌آمیزش استثنا از SHA128 استفاده می‌کنیم که طول مشتق حاصله کمتر باشد، از ۱۶ بایت حاصله، بیت‌های بخش‌پذیر بر چهار را انتخاب می‌کنیم و ایموجی متناظر با آن را تولید می‌کنیم. احتمال تداخل بین این دو از آنجایی که ایموجی از ۶ بیت پایین هر یک از بایت‌ها حاصل می‌شود، برابر با $۲^{۲۴}$ است.

۷.۴ تازه‌سازی کلیدهای نشست

تازه‌سازی کلیدهای نشست، شامل سه بخش می‌شود.

- **تازه‌سازی گذرواژه**

علت این که تازه‌سازی گذرواژه در نظر گرفته شده است، این است که پیام‌ها و همین‌طور کلیدها با استفاده از کلید مشتق از گذرواژه رمز شده است. پس در صورت افشا شدن گذرواژه امکان دسترسی به آنها وجود خواهد داشت. به همین دلیل لازم است تا در زمان تازه‌سازی گذرواژه تمام این موارد رمزگشایی و مجددا رمزگذاری شوند.

- **تازه‌سازی کلید نشست** از آنجا که در ابتدا هر نشست مجددا handshake صورت می‌گیرد و کلید نشست جدیدی تعیین می‌شود، برای تازه‌سازی کلیدهای نشست کافیست کارخواه را از حساب کاربری خارج کنیم تا مجددا وارد شود.

- **تازه‌سازی کلید نامتقارن** در خواست تازه‌سازی کلید نامتقارن با کلید خصوصی کاربر رمز خواهد شد تا بقیه افراد مزاحم نتوانند بی جهت کلیدهای قبلی فرد را باطل کنند. دقت داشته باشید که در اینجا فرض شده است که کلید نامتقارن کاربر برای مهاجم افشا شده است. اما موضوع این جاست که مهاجمی که کلید برایش افشا شده است، تمایلی به ابطال کلید نامتقارن فعلی نخواهد داشت.

۵ پیاده‌سازی کارگزار

۱.۵ ثبت‌نام و احراز هویت کاربران و نگهداری اطلاعات مربوطه به صورت امن

برای ذخیره اطلاعات کاربران به صورت امن در سرور از یک پایگاه داده SQLite ذخیره می‌کنیم. ذخیره‌سازی رمز عبور کاربران به این صورت است که رمز عبور در کنار یک رشته تصادفی[۱] قرار می‌گیرد و کل آن به صورت درهم‌آمیخته[۲] در پایگاه داده در کنار آن رشته تصادفی ذخیره می‌گردد.

۲.۵ ارسال پیامها به مقصد

سرور از ارتباط خود با هر یک از طرفین استفاده می‌کند تا پیام‌ها را بین دو طرف مبادله کند. فی مابین سرور بررسی می‌کند که آیا مقصد پیام یک شخص موجود در سیستم هست یا خیر. همچنین در صورت ارسال پیام در گروه سرور بررسی می‌کند که فرد فرستنده و گیرنده حتما در گروه حاضر باشند.

۳.۵ ارائه لیست کاربران آنلاین به کارخواه

سرور در زمان دریافت هر پیام از سمت یک کاربر، زمان آخرین دسترسی آن کاربر به سرور را بروز می‌کند و زمانی که کارگزار درخواست برای دیدن کاربران آنلاین انجام می‌دهد، لیستی از کاربرانی را که در دو دقیقه اخیر با سرور ارتباط داشته‌اند را به وی ارائه می‌دهد.

۴.۵ ارسال و دریافت پیامهای مربوط به ساخت کلید نشست

پیام‌های مربوط به تولید نشست نیز همانند رد و بدل شدن پیام بین دو کاربر بین آن‌ها رد و بدل می‌شود.

۵.۵ نگهداری اطلاعات گروه‌ها

لیست گروه‌ها، مدیر گروه و کاربران هر گروه در پایگاه داده در سمت سرور ذخیره می‌شود.

---

[۱] salt

[۲] hash

۶ برآورده شدن نیازمندی‌های امنیتی

در این بخش به بررسی برآورده شدن هر یک از نیازمندی‌های امنیتی می‌پردازیم و برآورده شدن هر یک از آنها را توجیه می‌کنیم.

۱.۶ رمزنگاری انتها به انتها

همان طور که گفته شد، تمام پیام‌های ارسالی توسط پروتکل جغجغه‌ی دوطرفه رمز می‌شوند، و حتی کارگزار هم امکان مشاهده‌ی پیام‌ها را ندارد.

۲.۶ تازگی کلید

بعد از ورود کاربر به حساب کاربری یک handshake بین کاربر و کارگزار اتفاق می‌افتد تا یک کلید جلسه بین آنها مشخص شود. در حین انجام عملیات login و سپس عمیات handshake هر دو طرف nonce که ارسال کرده‌اند را دریافت می‌کنند. به همین جهت هر دو می‌توانند از تازگی پیام‌های ارسالی اطمینان حاصل کنند. با این کار کارخواه می‌تواند از تازگی کلید نشست و همین‌طور تازه بودن token اطمینان حاصل کند. همچنین سرور نیز می‌تواند مطمئن باشد که درخواست تعیین کردن کلید نشست تازه است تا مهاجمان نتوانند با تکرار تعداد زیادی از این پیام‌ها کارگزار را وادار به تولید کلیدهای نشست کنند.

۳.۶ محرمانگی

در پروتکل HTTPSL تمام پیام‌های با کلیدهای عمومی طرفین (کارخواه و کارگزار) رمز می‌شوند. چون برای این کار از الگوریتم امن RSA استفاده می‌شود، بنابراین نیازمندی امنیتی محرمانگی در این پروتکل برقرار خواهد بود.

در پروتکل SSE نیز پیام‌ها با کمک رمز متقارن و الگوریتم AES رمز می‌شود.

۴.۶ صحت و یکپارچگی

تمامی پیام‌ها چه در پروتکل HTTPSL و چه در پروتکل SSE توسط کلید خصوصی هر فرد، امضا می‌شود و به همین جهت هر تغییری که در پیام به وجود بیاید، این امضا verify نخواهد شد.

۵.۶ حفظ سازگاری

برای این که مطمئن باشیم، ترتیب پیام‌های دریافتی یکسان است از sequence number استفده می‌کنیم. هر چند چون از پروتکل جغجغه‌ی دو طرفه استفاده می‌کنیم، در صورتی که ترتیب پیام‌ها صحیح نباشد، پیام ارسالی رمزگشایی نخواهد شد.

۶.۶ احراز اصالت

همه پیام‌های ارسال شده باید توسط فرستنده امضا شوند. بررسی امضا اثبات خواهد کرد که پیام بدون تغییر پس از امضا به دست ما رسیده است.

۷.۶ عدم انکار

همه پیام‌های ارسال شده باید توسط فرستنده امضا شوند. در زمان دریافت نیز پس از بررسی صحت امضا، در پایگاه داده کنار پیام امضای آن را نیز ذخیره می‌کنیم تا در زمان مقتضی برای اثبات ارسال پیام از امضای آن استفاده کنیم.

۸.۶ کنترل دسترسی

تمام دسترسی‌های کاربران توسط سرور کنترل می‌شود. نحوه بررسی این دسترسی نقش‌ـمبنا است. به این صورت که فردی که دسترسی admin را دارد می‌تواند عملیات حذف و اضافه به گروه را انجام دهد و یا فردی که عضویت در گروه را دارد می‌تواند ارسال پیام در گروه را انجام دهد.

۹.۶ حمله مرد میانی

توجه بفرمایید که تمام ارتباطات توسط کلید و الگوریتم امن رمزنگاری شده‌اند بنابرین رمزگشایی آن و رمزنگاری مجدد آن برای هیچ شخص ثالثی امکان‌پذیر نیست. توجه کنید که در پروتکل SSE نیز کلید نشست توسط پروتکل HTTPSL تبادل شده است و بنابراین هیچ شخص ثالثی نمی‌تواند به این کلید دست پیدا کند.

۱۰.۶ حمله تکرار

در هر دو پروتکل مورد استفاده برای ارتباط، از مهرزمانی استفاده می‌شود که بررسی آن تضمین عدم امکان حمله تکرار را به همراه دارد. توجه بفرمایید که زمان تمام کاربران باید همگام با زمان سرور باشد.

۱۱.۶ استفاده از الگوریتم‌های رمزنگاری امن

الگوریتم‌های مورد استفاده برای رمزنگاری عبارتند از RSA، AES، Diffie–Hellman و HKDF و برای محاسبه درهم‌آمیزش SHA256 استفاده می‌شود که همگی جز الگوریتم‌های امن رمزنگاری هستند.

۱۲.۶ محرمانگی پیش‌رو و محرمانگی آینده

همان‌طور که در (۳) به تفصیل توضیح داده شده است، با استفاده از الگوریتم جغجغه‌ی دوطرفه هر دو طرف محرمانگی را ایجاد می‌کنیم.

References

[1] *Implementing Signal's Double Ratchet algorithm*. nikofil's blog. 2020. URL: `https://nfil.dev/coding/encryption/python/double-ratchet-example/` (visited on 07/01/2023).