مجله علمی- پژوهشی "پدافند الکترونیکی و سایبری"
سال ششم، شماره ۱، بهار ۱۳۹۷، ص ۳۵-۱۷

راه‌حل پیشنهادی جلوگیری از حمله رله در گوشی تلفن‌همراه با استفاده از TEE

سمیرا طارمی[۱]، محمدعلی دوستاری[۲*]، صادق حاجی محسنی[۳]، محسن مقصودلو[۴]، مریم میابی جغال[۵]

۱و۵- دانشجوی کارشناسی ارشد، ۲- استادیار، ۳ و۴- کارشناس ارشد، مهندسی فناوری اطلاعات، دانشکده فنی و مهندسی، دانشگاه شاهد

(دریافت: ۹۴/۱۰/۲۳، پذیرش: ۹۵/۰۸/۱۰)

چکیده

نرم‌افزار کیف‌پول گوگل که از فناوری NFC استفاده می‌کند، در سال‌های اخیر توجه بسیاری از کاربران را جهت پرداخت، به خود جلب کرده است. این نرم‌افزار بر روی سیستم‌عامل تلفن‌همراه بدون درنظرگرفتن تمهیدات امنیتی نصب و اجرا می‌شود و از المان‌امن به‌عنوان محل مناسبی جهت ذخیره اطلاعات مالی کاربران استفاده می‌نماید. بنابراین، این نرم‌افزار به‌راحتی در معرض ریسک‌های امنیتی فراوانی از جمله حمله‌رله قرار می‌گیرد. در نتیجه، تأمین امنیت در کاربرد مذکور همواره یکی از مهم‌ترین چالش‌ها محسوب می‌شود. یکی ازمکانیزم‌های مناسب برای تأمین امنیت، اجرا و نصب ایزوله نرم‌افزار است. اما پیش از اجرا و نصب ایزوله نرم‌افزار ابتدا باید گوشی تلفن‌همراه به محیط امنی مجهز گردد که این محیط امن توسط محیط اجرای قابل اعتماد (TEE) تأمین می‌گردد. در این مقاله، یک روش احرازهویت دوطرفه متقارن بین المان‌امن و ترمینال بر روی تلفن‌های مجهز به پردازنده با دو محیط اجرایی (TEE/REE) جهت مدیریت اجرای نرم‌افزار کیف‌پول گوگل پیشنهاد شده است. روش پیشنهادی نه تنها سبب بهبود امنیت نرم‌افزار کیف‌پول گوگل در برابر حمله رله می‌گردد بلکه نرم‌افزارهای مشابه نیز می‌توانند با استفاده از این روش امنیت خود را در برابر این نوع حملات فراهم نمایند.

واژه‌های کلیدی: فناوریNFC، کیف‌پول گوگل، المان‌امن، حمله رله، محیط اجرای مورد اعتماد(TEE).

۱- مقدمه

یکی از جدیدترین نمونه‌های فناوری بی‌سیم[۱]"ارتباط میدانی کوتاه بُرد"(NFC)[۲] است که امکان تبادل اطلاعات در فاصله کم را فراهم می‌کند [۱۱]. این فناوری در سطح سخت‌افزار، دارای چهار جزء اصلی شامل: کنترولر NFC[۳]، کنترولر میزبان[۴]، آنتن NFC[۵] و المان امن (SE)[۶] می‌باشد [۱۲]. فناوری NFC مطابق با استانداردهای حوزه RFID[۷] و کارت هوشمند[۸]طراحی شده است. مشکلات ناشی از کاربردهای RFID تا حدودی در فناوری NFC نیز وجود دارد و لازم است راه‌حل‌های امنیتی مناسبی برای حل آن‌ها درنظر گرفته شود [۱۱]. اگرچه بُرد ارتباطی[۹]NFC محدود است و این مسئله به‌عنوان یکی از نقاط قوت NFC در تأمین امنیت محسوب می‌شود اما NFC به تنهایی یک ارتباط امن قابل اعتماد شمرده نمی‌شود [۱۷]. حملاتی مانند حمله جعل محتوا[۱۰]، حمله استراق سمع[۱۱] و حمله رله[۱۲] ازجمله حملات موجود برای سوءاستفاده در این فناوری می‌باشند [۹].

تأمین امنیت در گوشی‌های هوشمند[۱۳]با قابلیت NFC، مهم‌ترین مسئله در محیط پرداخت است و با وجود پژوهش‌های انجام‌شده هنوز چالش‌هایی در این زمینه وجود دارد. در نرم‌افزار کیف‌پول[۱۴]، که پرداخت را از طریق واسط NFC انجام می‌دهند، اطلاعات مالی کاربران در المان‌امن، که محل ذخیره‌سازی قابل اعتماد است، ذخیره می‌شود. عدم به‌کارگیری مجوزهای لازم برای دسترسی به المان‌امن و استفاده از نرم‌افزارهای مختلف از

*رایانامه نویسنده مسئول: doostari@shahed.ac.ir

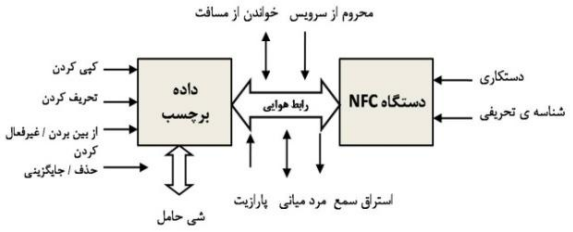
1- Wireless (Wifi)
2- Near Field Communication
3- NFC-Controller
4- Host-Controller
5- NFC-Antenna
6- Secure Element
7- Radio Frequency Identification
8- Smart Card
9- Communication Board
10- Fake Content Attack
11- Eavesdropping Attack
12- Relay Attack
13- Smart Phone
14- Application Wallet

توسعه‌دهندگان[1] متفاوت، سبب می‌گردد مهاجمان[2] با روش‌های مختلفی به‌دنبال به‌دست‌آوردن اطلاعات مالی ذخیره شده باشند. موارد فوق از طرفی سبب آسیب‌پذیری و ناامنی کل سیستم و از طرف دیگر، سبب عدم روی‌آوردن کاربران به فناوری جدید می‌شود. بنابراین، باید به‌دنبال راهی برای امن و قابل اعتمادکردن محیط کاری کاربران به‌خصوص در بحث پرداخت بود.

با توجه به گستردگی استفاده از نرم‌افزار کیف‌پول گوگل[3]، در این مقاله یک روش احراز هویت جدید بر روی تلفن‌های همراه مجهز به فناوری TEE[4] جهت متمایزکردن برنامه معتبر از برنامه‌های مخرب وجلوگیری از حمله رله بر روی کیف‌پول گوگل پیشنهاد شده است. در ادامه در بخش دوم، سناریوی حمله رله، در بخش سوم، حمله رله بر روی کیف‌پول گوگل، در بخش چهارم، راه‌حل‌های موجود جهت جلوگیری از حمله رله، در بخش پنجم، راه‌حل پیشنهادی مقاله حاضر، در بخش ششم، احرازهویت دوطرفه متقارن پیشنهادی، در بخش هفتم، روش پیشنهادی نصب نرم‌افزار کیف‌پول گوگل، در بخش هشتم، جلوگیری از حمله رله با استفاده از TEE، در بخش نهم پیاده‌سازی نمونه آزمایشگاهی و درنهایت، نتیجه‌گیری از مقاله و اهداف آتی توسط نویسندگان مقاله، مطرح و مورد بررسی قرارگرفته است.

۲- حملات رله

نمای کلی از حملات احتمالی به سیستمNFC، در شکل (۱) نشان داده شده است. واسط غیرتماسی همواره یکسان است. دستگاهNFC می‌تواند توسط مالک یا کاربر خود یا بدون اطلاع مالک و توسط رخنه‌گر[5] دستکاری شود [۱۲].



شکل (۱): نمای کلی از حملات به سیستم‌های NFC [۱۲]

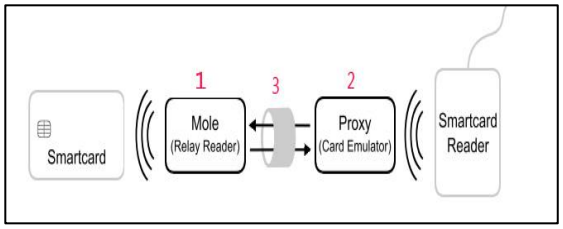
۲-۱- حمله‌های رله مبتنی بر NFC

حمله‌رله زیرمجموعه‌ای از حملات رابط‌هوایی[6] است. رابط هوای بدون تماس سبب می‌شود این حمله بدون دسترسی فیزیکی و پنهانی انجام شود. در این حمله، مهاجم با استفاده از کانال ارتباطی دیگری (رله)، به‌عنوان واسط، سعی می‌کند به سیستم نفوذ نماید. شایان ذکر است که این حمله بدون اطلاع قربانی[7] آغاز می‌شود. در واقع، مهاجم از ضعف دستگاه‌های موجود به‌منظور شروع ارتباط، بدون دخالت کاربر، استفاده می‌کند. درنتیجه، دریافت و ارسال اطلاعات از کارت قربانی به سمت ترمینال[8] و بالعکس، از طریق دو دستگاه غیرقانونی که یکی نزدیک به کارت قربانی و دیگری نزدیک به ترمینال است انجام می‌گیرد. این حمله محرمانه‌بودن سیستم NFC را تحت تأثیر قرار می‌دهد [۱ و ۱۲]. همان‌طورکه در شکل (۲) نشان داده شده است حمله‌رله به سه جزء اصلی نیاز دارد:

۱- یک دستگاه کارتخوان[9](Moleیا Leech) که نزدیک به کارت قربانی قرار می‌گیرد.

۲- یک دستگاه شبیه‌سازکارت[10] (Proxy یا Ghost) که برای ارتباط با کارتخوان واقعی استفاده می‌شود.

۳- یک کانال ارتباطی سریع که بین این دو دستگاه قرار دارد.



شکل (۲): ارتباط رله بین کارت هوشمند و کارتخوان [۱]

۲-۲- نسل بعدی: حمله رله مبتنی بر نرم‌افزار

حمله‌رله مبتنی بر NFC، در فاصله فیزیکی نزدیک (کم‌تر از۱m) رخ می‌دهد و همین مسئله احتمال حمله را کاهش می‌دهد. اما پژوهش‌های اخیر نشان‌می‌دهد به‌جای دسترسی به المان‌امن، دستگاه از طریق واسط خارجی(بدون تماس)، می‌توان به پردازشگر برنامه دستگاه از طریق واسط داخلی دسترسی پیدا کرد و وجود نرم‌افزار اصلی بر روی پردازشگر دستگاهی که به آن حمله شده، کافی است [۱].

قربانی با نصب برنامه کاربردی مخرب و اجرای آن، این امکان را به مهاجم می‌دهد تا از طریق این نرم‌افزار، المان‌امن دستگاه را فعال کرده و از طریق شبکه ارتباطی، اطلاعات را به دستگاهی که کارت‌هوشمند را شبیه‌سازی کرده است، ارسال نماید. حمله‌رله

1- Developer's
2- Attacker's
3- Google wallet
4- Trusted Execution Environment(TEE)
5- Hacker
6- Air Interface

7- Victim
۸- منظور از کارتخوان یا Terminal موجودیت خارج از کارت است که با کارت یا موجودیت‌های روی کارت ارتباط برقرار می کند.
9- Reader
10- Card Emulator

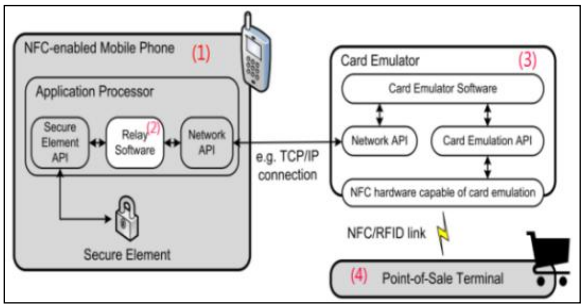
نرم‌افزاری دارای محدودیت فاصله فیزیکی و تشخیص فاصله دور برای ارسال نیست. هم‌چنین در کشورهایی که پرداخت از طریق تلفن‌همراه دارای محدودیت است، مهاجم با تشکیل شبکه‌ای از قربانیان از هریک، مقداری پول کسر می‌نماید[۶]. همان‌طورکه در شکل (۳) نشان داده شده است، حملات رله مبتنی بر نرم‌افزار متشکل از چهار بخش می‌باشد:

۱) تلفن‌همراه (تحت کنترل مالک/ کاربر قانونی).

۲)نرم‌افزار رله (تحت کنترل مهاجم).

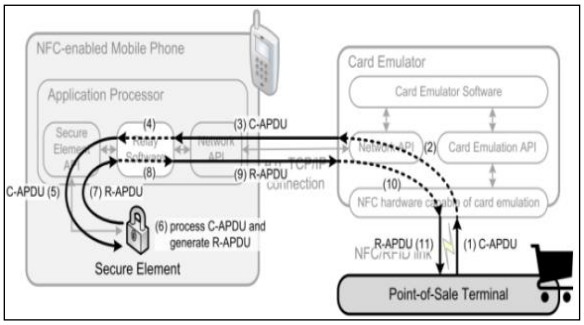
۳) شبیه‌ساز کارت (تحت کنترل مهاجم).

۴) دستگاه کارتخوان.



شکل (۳): سناریوی حمله رله [۱]

فرض براین است که نرم‌افزار رله، امتیاز لازم جهت دسترسی به المان‌امن و ارتباطات شبکه‌ای را دارا می‌باشد؛ به این معنا که هم‌تراز با نرم‌افزار کیف‌پول گوگل است. نرم‌افزار رله دستورات (APDU)[1]را دریافت کرده، آن‌ها را به المان‌امن ارسال می‌کند. سپس پاسخ‌های دریافت‌شده از المان‌امن را از طریق سوکت شبکه به شبیه‌ساز کارت ارسال می‌نماید[۱]. جریانی از دستورات رله‌شده بین کارتخوان کارت‌هوشمند و المان‌امن در شکل (۴) نشان داده شده است.



شکل(۴): جریان دستورات بین کارتخوان و المان‌امن [۱]

1- Application Protocol Data Unit(Apdu)

۲-۳- المان امن و دسترسی به آن

المان امن جهت ذخیره و مدیریت اطلاعات حساسی مانند رمز عبور[۲]، اطلاعات حساب‌بانکی و برنامه‌های پرداخت الکترونیکی استفاده می‌شود؛ بنابراین، المان‌امن باید از حافظه داخلی تلفن همراه جدا باشد. این قطعه، با فناوری کارت‌هوشمند ریز پردازنده مطابقت داشته و شامل پردازنده، حافظه فرار، حافظه غیرفرار و ... می‌باشد [۱۵]. در برنامه‌های کاربردی NFC از المان‌امن جهت ذخیره‌سازی امن اطلاعات استفاده می‌شود [۱۲].

روش‌های مختلفی جهت کنترل دسترسی به المان‌امن پیشنهاد شده است که همه آن‌ها به سیستم‌عامل تلفن‌همراه که بر روی پردازنده مرکزی اجرا می‌شود، تکیه می‌کنند و این نقص قابل توجهی است. بنابراین، در تمام موارد، المان‌امن به‌طور کورکورانه به تصمیم‌گیری پردازنده(المان ناامن) در مورد دسترسی اعتماد می‌کند. بر همین اساس، برنامه‌ای که مجوز لازم را به صورت غیرقانونی دریافت کرده باشد (به اصطلاح تلفن همراه را روت کرده باشد)، می‌تواند به راحتی بررسی‌های امنیتی را دور زده و کنترل المان‌امن را به‌دست گیرد [۱۸].

۳- اعمال حمله به سیستم پرداخت دنیای واقعی

به منظور بررسی حمله‌رله مبتنی‌بر نرم‌افزار، این حمله بر روی سیستم پرداخت موجود یعنی کیف‌پول گوگل مورد بررسی قرار می‌گیرد. به دلایل ذیل، نرم‌افزار کیف‌پول گوگل برای بررسی انتخاب شده است:

-کیف‌پول گوگل توسط بسیاری از کاربران استفاده می‌شود [۱].

-کیف‌پول گوگل براساس استانداردهای پرداخت EMV[3] طراحی شده و توسط ترمینال فروش[4] که تراکنش‌های کارت‌اعتباری بدون تماس PayPass[5]را پشتیبانی می‌کند، استفاده می‌شود [۱۸].

-کد منبع‌باز آندروید[6] در دسترس عموم می‌باشد. بنابراین، به‌راحتی می‌توان پشته[7] نرم‌افزار NFC و رابط برنامه کاربردی[8] المان‌امن را کشف نمود [۱۸].

۱-۳- کیف‌پول گوگل

کیف‌پول گوگل از نرم‌افزار آندرویدی با واسط‌کاربری (UI)[9] و

2- Personal Identification Number (Pin)
3- Europay Mastercard Visa
4- Point-Of-Sale Terminal
5- PayPass Contactless Credit Card
6- Open Source Code
7- Stack
8- Application Programming Interface
9- User Interface

اپلت‌های جاواکارت[1]، تشکیل شده است و در گوشی‌های هوشمند با قابلیت NFC والمان‌امن اجرا می‌شود. این بسته نرم‌افزاری جهت عملیات بانکی و پرداخت، توسعه داده شده است [۵].

در ابتدا، کانال امنی براساس پروتکل کانال امن۰۲(SCP02[2]) بین المان امن و سرور[3] برقرار می‌شود. این سرور، مدیریت المان امن را برعهده دارد. نرم‌افزارکیف‌پول گوگل،رمز عبور را در المان امن شخصی‌سازی می‌کند. به منظور حمله رله موفق، لازم است مؤلفه روی کارت کیف‌پول گوگل[4] انتخاب شده و کیف‌پول باز گردد. سپس، کارت پرداخت پیش‌فرض از طریق حالت داخلی المان امن در دسترس قرار می‌گیرد. در این حمله، به هیچ تعاملی توسط کاربر نیاز نمی‌باشد[۱].

۳-۲- اکوسیستم پرداخت کیف‌پول گوگل

اکوسیستم پرداخت گوگل از ذی‌نفعان مختلفی تشکیل شده است. ذی‌نفعان مختلف در این اکوسیستم سبب می‌شوند تا اطلاعات مشتری به راحتی مورد حمله قرار نگیرد [۱۵]. شرکت‌های مختلفی که می‌توانند در این محصول با گوگل همکاری داشته باشند عبارتند از:

- بانک‌های صادرکننده[5]
- شبکه‌های پرداخت[6]
- سیستم‌های فروش[7]
- شرکت‌های تولیدکننده تراشه[8]
- تولیدکنندگان گوشی تلفن همراه[9]
- اپراتورهای تلفن‌همراه[10]
- فروشندگان[11]
- شرکت‌های کارت پیش‌پرداخت[12]

فناوری NFC یک جزء کلیدی است که برروی کل اکوسیستم کیف‌پول گوگل تأثیر می‌گذارد. با توجه به این نکته که فناوری NFC برنامه‌هایی از قبیل بلیط حمل‌ونقل و ... را پشتیبانی می‌کند، همین مورد شانس، رشد کیف‌پول گوگل را افزایش می‌دهد [۶].

۴- راه‌حل‌های موجود جهت جلوگیری از حمله رله

در این بخش، به بررسی سه راه‌حل جهت جلوگیری از حمله‌رله می‌پردازیم. این راه‌حل‌ها در حال حاضر موجود هستند.

۴-۱- وقفه‌های ترمینال فروش[13]

این روش به‌راحتی قابل پیاده‌سازی می‌باشد، اما قابل اعتماد نیست. در این راه‌حل، جهت جلوگیری از حمله رله، تراکنش‌های پرداخت در ترمینال فروش، باید در طی زمان مشخصی انجام شوند. در واقع، ترمینال و گوشی‌هوشمند باید طی وقفه‌های کوتاه از پیش تعیین‌شده دستورات و پاسخ‌های لازم را ارسال نمایند. درنتیجه، تراکنش‌های طولانی‌تر از وقفه‌های تعیین‌شده یا باید متوقف گردند یا به‌طور کلی رد شوند. این راه‌حل، از حمله رله در فواصل طولانی جلوگیری می‌نماید اما از حمله‌رله‌ای که در فواصل کوتاه و کانال‌های ارتباطی سریع رخ می‌دهد و می‌تواند طبق وقفه‌های از پیش تعیین‌شده عمل نماید، جلوگیری نمی‌نماید. مشکل این نرم‌افزارها داشتن سیاست‌هایی همانند سیاست‌های نرم‌افزار رله می‌باشد. بنابراین، حمله موفقی را نصیب مهاجم می‌نماید [۱۸].

۴-۲- اعتبارسنجی رمز عبور توسط مؤلفه روی کارت کیف‌پول گوگل

رمز عبور کیف‌پول در سطح نرم‌افزار، اعتبارسنجی می‌شود و مؤلفه روی کارت کیف‌پول گوگل، این رمز عبور را اعتبارسنجی نمی‌کند. این مؤلفه توسط دستورات قفل‌کردن و بازکردن ساده کنترل می‌شود. بنابراین، در این راه‌حل، اعتبارسنجی رمز عبور توسط مؤلفه روی کارت در المان امن مدیریت می‌شود. در این صورت، نرم‌افزار مهاجم برای انجام یک حمله موفقیت‌آمیز به دانستن رمز عبور کیف‌پول نیاز دارد [۱۸].

برنامه‌های کاربردی بسیاری بر روی رایانه شخصی (Spyware &Keylogger) وجود دارند که اجازه می‌دهند مهاجم کلیدهای ضربه زده‌شده توسط کاربر را ضبط کند و اطلاعاتی مانند رمز عبور یا اطلاعات حساب بانکی را به سرور مهاجم ارسال نماید. اما این عمل در دستگاه تلفن‌همراه دشوار است به این دلیل که

1- Javacard Applets
2- Secure Channel Protocol 02
3- Server
4- Google Wallet On-Card Component
5- Issuing Banks
6- Payment Networks
7- Point Of Sale Systems
8- Semiconductor Companies
9- Mobile Handset Manufacturers
10- Mobile Operators
11- Merchants
12- Prepaid Card Companies
13- Timeouts Of POS Terminals

نرم‌افزارهای کاربردی معمولاً از طریق بازارهای نرم‌افزاری آنلاین توزیع می‌شود که کنترل برنامه‌های کاربردی را انجام می‌دهند. علاوه بر این، برنامه‌های کاربردی معمولاً اجازه دسترسی به صفحه نمایش ورودی/خروجی هنگامی‌که در پیش‌زمینه اجرا می‌شوند را نمی‌دهند. چون امروزه دستگاه‌های تلفن‌همراه معمولاً دارای یک حسگر بر روی بُرد خود هستند لذا راه‌های دیگری برای استخراج ورودی در دستگاه تلفن‌همراه وجود دارد. حسگرها می‌توانند همانند کانال جانبی[1] برای به‌دست‌آوردن ورودی کاربر استفاده شوند و امکان حمله بر روی دستگاه تلفن‌همراه عملی گردد [۱۹]. با این حال، این روش بسیار مشکل‌تر از ارسال دستور ساده بازکردن مؤلفه روی کارت کیف‌پول گوگل است [۱۸].

۴-۳- غیرفعال‌کردن ارتباط داخلی اپلت‌های پرداخت

گوشی‌های تلفن‌همراه به‌منظور تشخیص ارتباطات داخلی(پردازشگر برنامه) و خارجی (واسط بدون تماس) المان امن، می‌توانند به اجزای امن پیشرفته‌ای مجهز شوند. جهت دسترسی به واسط ارتباطی می‌توان برحسب هر برنامه و حتی واحد داده ارسالی برنامه، قواعدی را درنظر گرفت. به علاوه، المان امن این قدرت را دارد تا ارتباط با یک اپلت را به صورت کامل غیرفعال نماید [۱۸]. هم چنین، این قابلیت می‌تواند حالت کارکرد داخلی برنامه‌های پرداخت را غیرفعال نماید. در نتیجه، از حمله‌رله مبتنی‌بر نرم‌افزار ممانعت می‌نماید. اما این راه‌حل عملی نمی‌باشد چرا که المان امن نمی‌تواند برای نرم‌افزار پرداخت امن بر روی دستگاه (به‌عنوان مثال، پرداختEMV بر روی مرورگر تلفن‌همراه) در آینده استفاده شود [۱۸].

۵- راه‌حل پیشنهادی: محیط اجرای قابل اعتماد

TEE محیط امن و مجموعه‌ای از اجزای سخت‌افزاری و نرم‌افزاری است که امکانات ضروری برای پشتیبانی از برنامه‌ها را فراهم می‌آورد. TEE تضمین می‌کند کدها و داده‌ها در هنگام بارگذاری از جنبه‌های محرمانگی[2] و یکپارچگی[3] محافظت می‌شوند. TEE محیط اجرای جداگانه‌ای است که در کنار سیستم‌عامل‌تلفن‌همراه که با نام Rich OS[4] شناخته می‌شود، اجرا می‌گردد و سرویس‌های امنی را برای محیط RichOS فراهم می‌نماید. فضای اجرایی که توسط TEE فراهم می‌شود سطحی بالاتر از امنیت Rich OS را ارائه می‌کند. TEE اجرای ایزوله‌شده[5]، ذخیره‌سازی امن، راه‌اندازی یکپارچه،شناسایی دستگاه و قابلیت احرازهویت را تأمین می‌نماید [۱۴]. این محیط می‌تواند در برابر حملات راه دور، نرم‌افزاری و سخت‌افزاری مقاومت نماید [۲۳]. امنیتی که TEE فراهم می‌کند بستگی به پیاده‌سازی آن دارد، مثلاً: TEE که با ARMTrustedzone پیاده‌سازی شده است در برابر دست‌کاری سخت‌افزاری حساس‌تر است [۱۳]. دستگاه‌های تلفن‌همراه مجهز به TEE پتانسیل جایگزین‌شدن با توکن‌ها[6] را دارند [۷].

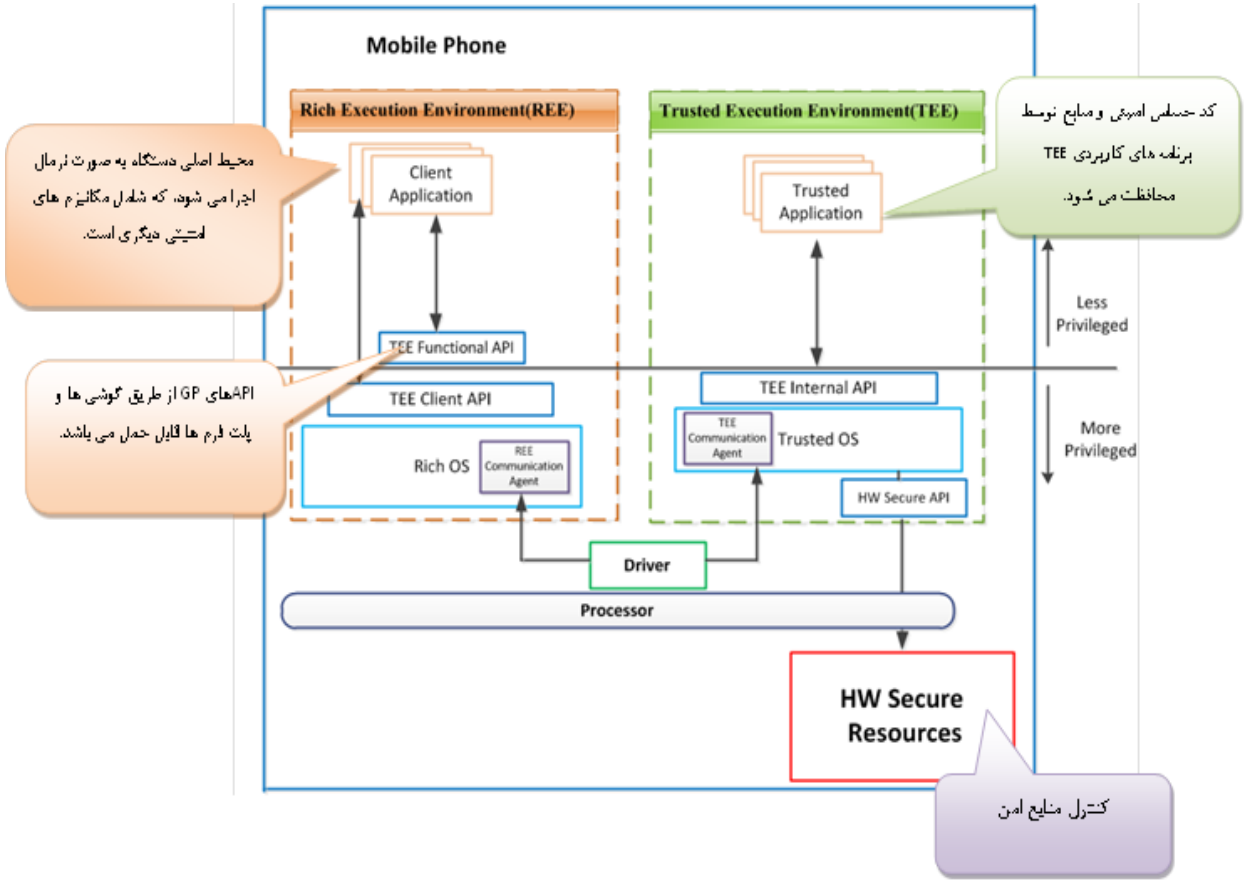
تلاش‌های استانداردسازی جدیدی مانند گلوبال پلتفرم (GP)[7]و ابتکارات پیاده‌سازی متن بازی مانند OP-TEE [۵] و Trusted Little Kernel [۲۱]، این پتانسیل را دارند که در استفاده گسترده از TEE توسط توسعه‌دهندگان نرم‌افزار، رشد و ترقی کنند[۱۶]. شکل (۵) معماری سه بلاک اصلی دستگاه تلفن همراه است که توسط GP استاندارد شده است وشامل (۱) پلتفرم سخت‌افزاری، (۲) محیط اجرای Rich و (۳) محیط اجرای قابل اعتماد می‌باشد [۱۳].

در TEE نرم‌افزار قابل اعتماد (TA)[8] برنامه‌هایی هستند که قابلیت اجرای ایزوله‌شده را دارند. ایزوله نرم‌افزاری و رمزنگاری در TEE، سبب می‌شود هر TA از سایر TAها مستقل باشد و هیچ TA ای نتواند به منابع TA دیگر دسترسی پیدا کند. برنامه‌های کاربردی که در TEE اجرا می‌شوند از دسترسی برنامه‌های کاربردی که بر روی Rich OS اجرا می‌شوند، محفوظ هستند. در واقع TEE لایه امنیتی بین Rich OS دستگاه (سطح امنیتی ضعیف) و المان امن (سطح امنیتی بالا) را فراهم می‌کند [۱۹]. برنامه مشتری (CA)[9] برنامه‌ای است که در Rich OS اجرا می‌شود و از طریق TEE Client API به سرویس‌های ارائه‌شده توسط TA در TEE،دسترسی پیدا می‌کند[۲۳]. CA با آغاز جلسه با TA ارتباط برقرار کرده و شروع به ارسال دستور می‌کند. هنگامی‌که TA دستورحاوی پیام را دریافت می‌نماید در ابتدا آن‌را تجزیه و تحلیل کرده، سپس پردازش‌های لازم را انجام می‌دهد و

1- Side-Channel
2- Confidentiality
3- Integrity
۴-Rich OS (در اصطلاح پلتفرم جهانی) به سیستم عامل دستگاه تلفن همراه اشاره می‌نماید که غیرقابل اطمینان است و در مقابل حملات آسیب‌پذیر می‌باشد.

5-Isolated Execution
6- Token
7- Global Platform(GP)
8- Trusted Application(TA)
9- Client Application(CA)

درنهایت پاسخ را به CA ارسال می‌کند [۲۲]. همان‌طورکه در جدول (۱) برآورد شده است در مقایسه بین المان امن TEE، المان‌امن کاملاً استاندارد شده است. تنها اشکال فنی آن، محدودیت منابع و کارایی پایین است [۱۴].



شکل (۵): معماری سه بلاک اصلی دستگاه تلفن همراه از دید نرم‌افزاری [۲۳]

جدول (۱): مقایسه المان امن و TEE [۱۴]

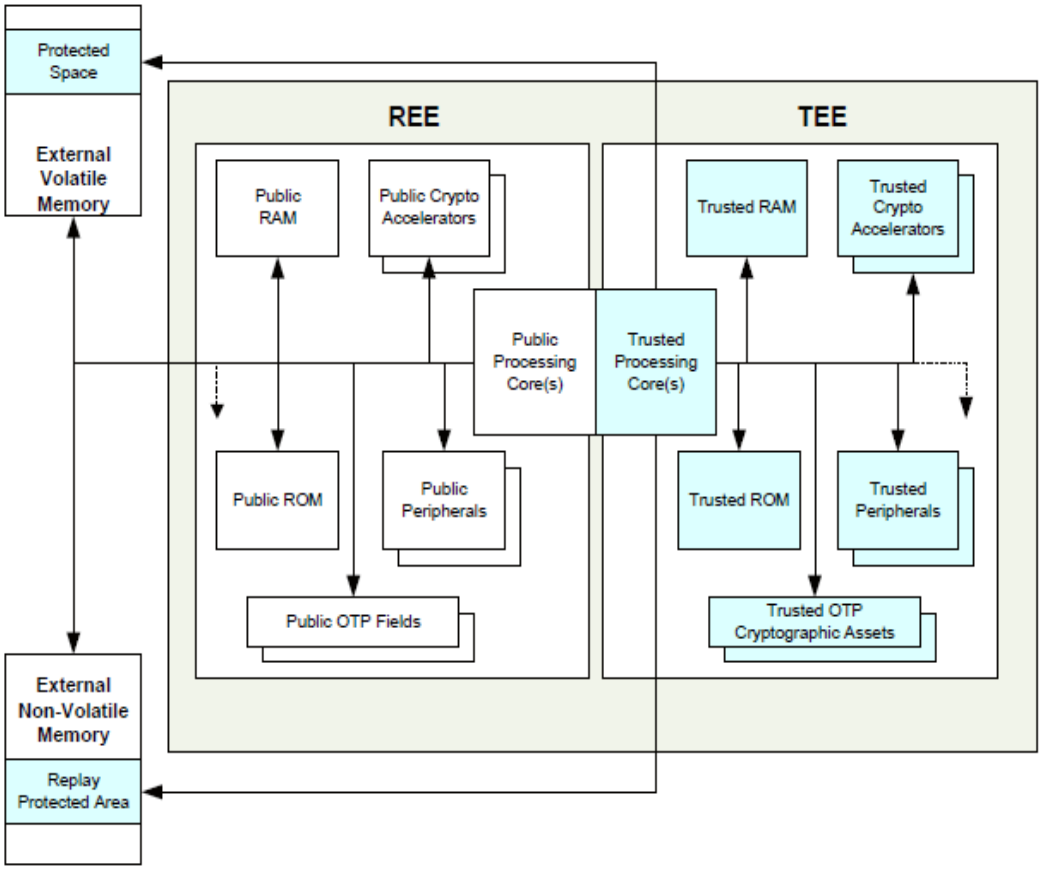
	المان امن	TEE
منابع بالا	خیر(منبع محدود)	بله
سطح امنیت	بالا	متوسط
کارایی	پایین	بالا
رابط کاربری	بدون رابط کاربری یا بسیار محدود	متوسط
سرعت پردازش	پایین	بالا
هزینه اضافی	بالا	پایین

در شکل (۶)، معماری سخت‌افزاری TEE که توسط GP استاندارد شده است، نشان داده می‌شود. TEE دارای سخت‌افزار مختص به خود می‌باشد. بدین معنی که برخی از بخش‌های سخت‌افزاری صرفاً توسط TEE قابل دسترس هستند. به عنوان مثال: قسمتی از حافظه به عنوان حافظه امن برای TEE در نظرگرفته می‌شود و ممکن است Rich OS حتی از اختصاص چنین حافظه‌ای به TEE اطلاعی نداشته باشد.

با توجه به این مسئله که ارتباط بین REE و TEE مستعد حمله می‌باشد، برای حل این مشکل در[۱۰] کانال امن SeCReT[1] پیشنهاد شده است. با توجه به توضیحات داده‌شده و استفاده از فناوری TEEیک پروتکل پرداخت جدید به‌نام EMV-TLS،

1- Secure Channel Between Rich Execution Environment and Trusted Execution Environment

معرفی شده است. امنیت این پروتکل جدید توسط سیستم TEE بهبود یافته است زیرا TEE، اعتماد را برای ورودی/ خروجی صفحه‌کلید، صفحه‌نمایش که برای نمایش اطلاعات و کلیدهای رمز عبور استفاده می‌شود، اجبار می‌کند و این بهبود سبب پوشش راه‌حل دوم موجود در جلوگیری از حمله رله درضبط اطلاعات ورودی کاربر می‌شود.



شکل (۶): معماری سخت‌افزاری TEE [۲۳]

۵-۱- ذخیره‌سازی امن

هم پلتفرم[1] و هم برنامه‌های تلفن‌همراه (همانند کیف‌پول گوگل) نیاز به ذخیره‌سازی امن دارند. پیش‌شرط ذخیره‌سازی امن، پایداری کلید دستگاه است. کلید دستگاه باید غیرقابل تغییر باشد و برای مؤلفه‌های سخت‌افزاری و نرم‌افزاری‌ای که مجوز دسترسی دارند، قابل دسترس باشد [۳].

ذخیره‌سازی امن، علاوه بر کلید دستگاه، به الگوریتم‌های رمزنگاری نیز نیازمند است. ذخیره یکپارچه به وسیله ذخیره چنین الگوریتم‌هایی بر روی حافظه غیرفرار تضمین می‌شود. اجرای الگوریتم رمزنگاری در یک محیط ایزوله مانند TEE فراهم می‌شود [۳].

۵- ۲- اجرای ایزوله

برنامه‌هایی مانند سرویس‌های پرداخت و کارت‌های اعتباری باید نسبت به نرم‌افزاری غیرقابل اعتماد به‌طور ایزوله اجرا شوند. به‌طور معمول، مکانیزم‌های ایزوله‌سازی مبتنی‌بر نرم‌افزار، محافظت کافی برای برنامه‌هایی با نیازمندی‌های امنیتی بالا را فراهم نمی‌کند. مکانیزم‌های ذخیره‌سازی امن امکان اجرای قابلیت‌های رمزنگاری (همانند الگوریتم‌های رمزنگاری) را به‌طور ایزوله برای کل سیستم فراهم می‌کند. با این‌که رمزگذاری، رمزگشایی و برخی عملیات رمزنگاری برای بسیاری از برنامه‌ها کافی است، اما برخی سرویس‌های امنیتی نیاز به اجرای ایزوله الگوریتم‌های رمزنگاری مختص برنامه خود را دارند که نمی‌توانند از پیش روی دستگاه نصب شوند. برای اجازه اجرای ایزوله‌شده الگوریتم‌های امنیتی مختص برنامه، سخت‌افزار تلفن‌همراه باید از کد قراردادی اجرای

1- Platform

ایزوله‌شده[1] حمایت کند. برای این منظور محیط TEE، قابلیت-های پردازش و ذخیره‌سازی که برای اجرای ایزوله‌شده ضروری است را فراهم می‌کند [۳].

۵-۳- معماری نرم‌افزاری TEE

با دقت به معماری TEE، در می‌یابیم که این محیط از منابع امن سخت‌افزاری، Trusted OS، TAها و رابط‌های برنامه کاربردی TEE تشکیل شده است. در ادامه، هریک از این موارد براساس مشخصات GP بیان می‌شود [۲].

۵-۳-۱- مؤلفه‌های نرم‌افزاری TEE

Trusted OS: سیستم‌عامل سبک و کوچکی است که در روی محیط TEE اجرا می‌شود و تکنیک‌های امنیتی برای طراحی آن استفاده شده است [۲۳]. Trusted OS نه تنها مسئولیت نگهداری TAهایی که بر روی آن اجرا می‌شوند را دارد بلکه سرویس‌هایی را نیز برای توسعه‌دهندگان نرم‌افزار فراهم می‌نماید [۲].

نرم‌افزار قابل‌اعتماد: نرم‌افزارهای کاربردی مورداعتماد (TA) یا Trustlet به برنامه‌هایی گفته می‌شود که بر روی محیط TEE قابلیت اجرای ایزوله را دارند [۴]. هر TA با یک شناسه منحصربه‌فرد (UUID)[2] شناخته می‌شود [۲۲]. TAها سرویس‌های امنیتی را برای CAهای خارج از TEE فراهم می‌کنند. مالک TEE[3] توانایی‌های مستقیم و غیرمستقیم جهت مدیریت TAهای درون TEE را دارد [۲۳]. TAها از طریق فراهم‌کنندگان نرم‌افزارهای متعدد تولید می‌شوند. در واقع، TA ابزاری برای فراهم‌کنندگان سرویس[4] است تا سرویس‌های امن خودشان را در گوشی با قابلیت‌های TEE مستقر کنند. همه عملیات‌هایی که برای فراهم‌کنندگان سرویس لازم است، لزوماً نباید توسط TA که در TEE مستقر شده است، انجام شود. چرا که بخشی از آن می‌تواند توسط Rich OS انجام گردد [۴]. برای مثال، TAها می‌توانند نرم‌افزارهای مهمی مانند نرم‌افزار پرداخت باشند [۲]. TAها جهت اجراشدن متکی به TEE Internal API هستند[۴]. ممکن است یک TA برای TA دیگر با استفاده از Internal Client API، به‌عنوان یک CA عمل کند [۲۲]. هم‌چنین، TAها می‌توانند توسط کارخانه سازنده تراشه نصب شوند یا می‌توانند توسط اپراتور شبکه تلفن‌همراه (MNO)[5] در طول حیات دستگاه، بر روی TEE نصب گردند. به طور کلی، TAها باید به منظور نصب‌شدن ویژگی‌هایی که در ادامه بیان شده است را دارا باشند:

- محافظت عقب‌گرد[6]

برای اکثر سرویس‌ها، داده غیرفرار[7] در TEE بسیار مهم است. محافظت عقب‌گرد مختلفی برای داده و کد لازم است که ممکن است در طول زمان به‌روزرسانی گردد، TEE باید مکانیزم‌هایی معینی داشته باشد که فقط از آخرین نسخه استفاده نماید. به‌روزرسانی نسخه TA ممکن است به‌دلیل نقصی باشد که در نسخه‌های نرم‌افزارهای قبلی وجود داشته است. بسیاری از TAها به ذخیره‌کردن اطلاعات به صورت محلی در یک جای امن نیاز دارند (ذخیره‌سازی غیرفرار). هنگامی‌که اطلاعات از فروشگاه خوانده می‌شود، TA باید بتواند آن‌ها را ذخیره کند و مهاجم نباید آن را با نسخه قدیمی جایگزین نماید. اگر معماری TEE بخواهد کد TA را به روش تکه‌ای اجرا کند (که توسط درایور OS برنامه‌ریزی می‌شود) محافظت از پخش در هنگام اجرا TA باید به‌کار گرفته شود این عمل تضمین می‌کند که داده غیرمرتبطی در بین اجراها تکرار نشده است یا این‌که مرحله‌ای خارج از آن‌چه تعریف شده است اجرا نگردد [۸].

- ایزوله

(از طریق خاصیت بانک حافظه[8] که در یک تراشه قرار گرفته است وبه‌عنوان هسته پردازشگر[9] با خصوصیت‌های محیط امن پردازشگر (PSE)[10] برای اجرای مفهومی استفاده می‌گردد). علاوه بر این، هر TA که با مجوزهای مختلف در TEE آپلود[11] می‌شود به غیر از آن‌که جهت استفاده از کد سیستم TEE آپلود شده است باید به صورت منطقی در سطح سیستم TEE همانند TA دیگر به صورت ایزوله‌شده باقی بماند مگر این‌که، مجوز لازم برای تعامل با TA دیگر را داشته باشد [۸].

- زمان‌بندی امن[12]

TA آپلودشده باید در TEE تکه‌تکه گردد و به‌صورت تکه‌ای اجرا شود. به‌همین دلیل، لازم است زمان‌بند در درایور حافظه محافظت نشده باقی بماند و فقط کم‌ترین مفاهیم امنیتی در

1- Isolated Execution Of Arbitrary Code
2- Universally Unique Identifier
3- Ownership TEE
4- Service Providers
5- Mobile Network Operators

6- Rollback Protection
7 - Nonvolatile Data
8- Memory Banks
9 - Processor Core
10- Processor Secure Environments
11- Upload
12- Secure Scheduling

RAM امن (جهت اطمینان از یکپارچگی جریان کد TA مورد انتظار) نگهداری گردد[۸].

- ارائه مجوز[1]

به TAها و داده‌های آن‌ها باید مجوز لازم داده شود. داده و کد برنامه ممکن است علاوه بر تضمین یکپارچگی به محافظت محرمانگی، نیاز داشته باشند. به مجموعه مشخصی از TAها می‌توان به‌صورت منحصر به‌فردی رمزی را اختصاص داد. در این‌صورت، یک رمز در میان مجموعه نرم‌افزارها به اشتراک گذاشته می‌شود[۸].

منابع امن سخت‌افزاری: دارای چهار دسته‌بندی می‌باشد که در ادامه بررسی می‌شود:

۱) کلیدهای سخت‌افزاری که کلیدهای پلتفرم نامیده می‌شود.

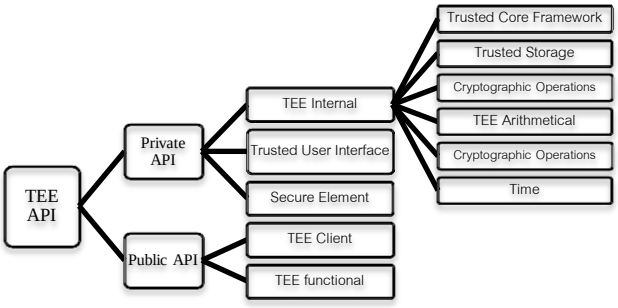
۲) توابع امن همانند ذخیره‌سازی امن، ورود و نمایش امن.

۳) شتاب‌دهنده[2]های رمزنگاری (عملیات رمزنگاری مثل Des، 3Des و ...)

این قسمت با قطعات و بخش‌هایی در ارتباط است که کارهای مربوط به سخت‌افزار را انجام می‌دهند. در واقع این قسمت آماده‌کننده توابع و ... است که ارتباط با قسمت‌های سخت‌افزاری را سهولت می‌بخشد.

۵-۳-۲- رابط‌های کاربردی TEE

کنترل دسترسی که بر روی REE توسط ایزوله سخت‌افزاری انجام می‌شود مبنی بر این است که REE نباید به منابع مورد اعتماد دسترسی یابد. تنها راه دسترسی از طریق رابط کاربردی یا سرویس‌هایی است که توسط محیط TEE فراهم شده است و این عمل سبب می‌شود تا REE در وضعیت کنترل‌شده و محافظت‌شده‌ای به TEE دسترسی یابد [۲۳]. در این قسمت با توجه به مطالعات انجام‌شده دسته‌بندی از رابط‌های کاربردی مربوط به محیط TEE و REE به‌صورت مختصر توضیح داده می‌شود. با دقت در شکل (۷)، دسته‌بندی کلی از APIهای موجود در محیط REE و TEE را می‌یابیم که به دو کلاس TEE APIs عمومی (Public) و خصوصی (Private) تقسیم می‌شوند. Public API واسط‌های ارتباطی بین REE و TEE (بین CAها و TAها) را فراهم می‌نماید و به‌عنوان عامل ارتباطی شناخته می‌شود اما Private API واسط‌های ارتباطی بین TA و نرم‌افزار المان امن و نیز دسترسی به منابع TEE را فراهم می‌نماید [۲].



شکل (۷): دسته‌بندی کلی از APIهای موجود در محیط REE و TEE

در Private API، TEE Internal API واسطی است که توسط آن TA در حال اجرا در TEE، از منابع TEE همانند قابلیت‌های رمزنگاری و ذخیره‌سازی امن استفاده می‌نماید. در واقع، این API سبب می‌شود که TA فعال بهترین استفاده را از محیط TEEداشته باشد [۲].

Trusted user interface API دومین API است که جزء Private API می‌باشد که به منظور جلوگیری از استراق سمع و دستکاری توسط نرم‌افزارهای غیرمجاز، داده‌های ورودی توسط TAکنترل می‌شوند [۲].

TEE Secure Element API نیز در Private API اجازه می‌دهد TA با المان امن ارتباط برقرار کند [۲]. لازم به ذکر است که دسترسی به المان امن فقط از طریق TEE میسر است بدین ترتیب، کانال ارتباطی بین المان امن و TEE می‌تواند به صورت مستقیم باشد که این حالت مبتنی بر ارتباط سخت‌افزاری است وREE نمی‌تواند با حملات نرم‌افزاری در آن تداخل ایجاد کند اما این حالت می‌تواند در معرض حملات سخت‌افزاری باشد. حالت ارتباطی دیگر به صورت غیرمستقیم است که در این حالت ممکن است ارتباط توسط REE قطع شود. برای ایجاد ارتباط امن بر روی هریک از انواع کانال‌ها یک پروتکل امن نیاز می‌باشد [۲۰].

در Public APIs، TEE Client API واسط ارتباطی سطح پایینی است که بین CA و TA ارتباط برقرار می‌کند. هدف این ارتباط در واقع تبادل داده بین TA (مانند: GoogleWallet TA) و CA (مانند: GoogleWallet CA) است. در طول ارتباط اگر مورد مشکوکی تشخیص داده شود، جلسه ارتباطی مربوطه بلافاصله قطع می‌شود [۲].

TEE Functional API مجموعه‌ای از APIهای Rich OS-friendly را ارائه می‌دهد [۲] و اجازه می‌دهد به برخی خدمات TEE از قبیل رمزنگاری یا ذخیره‌سازی قابل اعتماد دسترسی پیدا کنیم [۲۳].

1- Provisioning
2- Accelerator

۶- احرازهویت دو طرفه متقارن[1] پیشنهادی

پیش از پرداختن به حالت‌های پیشنهادی نصب نرم‌افزار کیف‌پول گوگل لازم است فرایند احرازهویت دو طرفه‌ای که به‌طورکلی در احرازهویت بین المان‌امن گوشی‌همراه و ترمینال پیشنهادشده است را مورد بررسی و ارزیابی قرار دهیم.

هدف از احرازهویت، بررسی‌کردن هویت و اعتبار طرف مقابل می‌باشد. در این روال، یک کلید رمز[2] برای انجام عملیات بین ترمینال و المان‌امن به اشتراک گذاشته می‌شود. در احرازهویت دوطرفه پیشنهادی، جهت جلوگیری از دسترسی غیرمجاز به المان‌امن، در مراحل ساخت تراشه المان‌امن مقدار منحصربه فردی با نام شماره سریال المان‌امن (SSN)[3] در تراشه قرار می‌گیرد به‌طوری‌که،SSN هر تراشه با تراشه دیگری متفاوت می‌باشد. انحصاری بودن SSN، امکان ردیابی تراشه را فراهم می‌کند. قبل از این‌که تراشه در روی گوشی هوشمند قرار بگیرد ابتدا مقدار SSN خود را برای دستگاهی به نام ماژول امنیتی سخت‌افزاری (HSM)[4]جهت تولید کلید ارسال می‌نماید. در دستگاه HSM یک الگوریتم و کلید اصلی(Master) امن وجود دارد. الگوریتم دستگاه HSM کلید Master و مقدار SSN را به عنوان ورودی دریافت می‌کند، همان‌طورکه در شبه‌کد مرحله اول مشاهده می‌کنید، دو کلید تولید می‌شود که به این تراشه اختصاص داده می‌شود و به‌صورت امنی بر روی تراشه قرارمی‌گیرد. یکی از کلیدها برای محاسبه (MacKey) و دیگری برای رمزنگاری (EncKey) مورد استفاده قرار می‌گیرد. در صورتی‌که تراشه دیگری با SSN متفاوتی به این الگوریتم و کلید Master داده شود، کلیدهای متفاوت دیگری مختص به تراشه ایجاد می‌گردد.

شبه کد مرحله اول به شرح زیر است:

```
Procedure GenerateKey (SSN , Masterkey:
Double)                                          (۱)
{
Var  EncKey, MacKey: Double;
EncKey: = Generate_EncKey (SSN , Masterkey);
MacKey:=Generate_MacKey (SSN , Masterkey);
}
Endprocedure

Function Generate_EncKey (SSN , Masterkey:
Double): Double
{
Var TailEnc, EncKey: Double;
TailEnc :=Contact (SSN , 0000000000000001);
EncKey :=3DesEncrypt(TailEnc , Masterkey);
}
return EncKey;

Function Generate_MacKey (SSN , Masterkey:
Double): Double
{
 Var TailMac, MacKey :Double;
TailMac :=Contact (SSN , 0000000000000002);
MacKey :=3DesEncrypt(TailMac , Masterkey);
}
      return MacKey;
```

در مرحله بعد، تراشه‌المان‌امن بر روی گوشی‌هوشمند قرار می‌گیرد. طبق سیاست‌هایی که توسط سیستم‌عامل المان‌امن درنظر گرفته شده است، اجازه خواندن کلیدها از راه‌های مختلف از المان‌امن امکان‌پذیر نمی‌باشد. گوشی‌هوشمند جهت برقراری ارتباط با ترمینال، باید ترمینال را احرازهویت کند. کلیدهای المان‌امن به‌دلیل جلوگیری از حملات ممکن از قبیل: استراق‌سمع و حمله مرد میانی[5] و ... به ترمینال فرستاده نمی‌شود. گوشی هوشمند در ابتدا SSN خود را برای ترمینال ارسال می‌کند. ترمینال با استفاده از الگوریتم و کلید Master به محاسبه کلیدهای المان‌امن می‌پردازد. کلید Master و الگوریتم باید به‌صورت امنی در ترمینال ذخیره شوند و در صورت حمله، کلید Master و الگوریتم باید حذف شوند. در این مرحله ترمینال با استفاده از کلید Master و SSN و الگوریتم، کلیدهای المان‌امن را همانند شبه‌کد مرحله اول تولید می‌نماید.

در مرحله دوم المان‌امن، با درخواست ترمینال، یک عدد تصادفی[6](Rnd_SE) تولید کرده و برای ترمینال ارسال می‌کند، ترمینال دنباله عدد تولیدشده (Value_SE) را با کلید مشترک ایجاد شده رمز می‌کند. هم‌چنین، MAC داده رمزگذاری‌شده را نیز محاسبه کرده و مقدار نهایی را برای المان‌امن می‌فرستد.

شبه کد مرحله دوم به شرح زیر است:

```
Procedure MacTerminal (Rnd_SE, SSN_SE,
EncKey, MacKey :Double)                          (۲)
{
 Var Macdata_terminal, Value_SE, Encdata
:Double;
 Value_SE := Contact (Rnd_SE , SSN_SE );
Encdata :=AES (Value_SE , EncKey );
Macdata_terminal := Mac (Encdata , MacKey);
}
endprocedure
```

در مرحله سوم المان‌امن عدد تصادفی رمزنگاری‌شده توسط ترمینال را اعتبارسنجی می‌کند. برای این منظور، المان‌امن عدد تصادفی ارسالی خود را رمز کرده و با مقدار رمزشده ترمینال مقایسه می‌نماید و درصورتی‌که هر دو با یکدیگر برابر باشند ترمینال مورد اعتماد است.

1- Symmetric
2- Secret
3- Se Serial Namber
4- Hardware Security Module(Hsm)
5- Man In The Middle (Mitm)
6- Random Number

شبه کد مرحله سوم به شرح زیر است:

```
Procedure VerifyTerminal (Macdata_terminal,
EncKey, MacKey, Rnd_SE , SSN_SE : Double)
{
Var SeparateMac, Value, Encdata: Double;
Verify: Boolean;
 Value: = Contact (Rnd_SE , SSN_SE );
 Encdata: = AES (Value , EncKey );
 SeparateMac: = SliceMac(Macdata_terminal);
 If (SeparateMac = = Encdata) then
Verify := true;
 Else
Verify := false;
}
endprocedure
```
(۳)

درمرحله‌چهارم، ترمینال نیز المان امن را احرازهویت می‌کند، بنابراین، یک عدد تصادفی(Rnd_ter) تولید و به همراه SSN تراشه ترمینال برای المان امن ارسال می‌کند، سپس المان امن دنباله عددی (Value_ter) را تولید کرده و با کلید مشترک رمز می‌کند. هم‌چنین، MAC داده رمزشده را نیز محاسبه کرده و هر دو داده را برای ترمینال ارسال می‌نماید.

شبه کد مرحله چهارم به شرح زیر است:

```
Procedure MacSecureElement (Rnd_ter, SSN_ter,
EncKey,  MacKey :  Double)
{
Var Macdata_SecureElement, Value_ter, Encdata
:Double;
Value_ter := Contact (Rnd_ter , SSN_ter);
Encdata :=AES (Value_ter , EncKey );
Macdata_SecureElement := Mac (Encdata ,
MacKey);
}
endprocedure
```
(۴)

در مرحله پنجم، ترمینال عدد تصادفی که توسط المان امن رمزنگاری شده است را اعتبارسنجی می‌نماید. بنابراین، عدد تصادفی ارسالی خود را رمز کرده و با مقدار رمزشده المان امن مقایسه می‌کند و درصورتی‌که هر دو بایکدیگر برابر باشند المان امن گوشی همراه مورد اعتماد است.

شبه کد مرحله پنجم به شرح زیر است:

```
Procedure VerifySecureElement (Macdata_SecureElement,
EncKey, MacKey, Rnd_ter , SSN_ter : Double)
{
Var SeparateMac, Value, Encdata: Double;
Verify: Boolean;
Value = Contact (Rnd_ter , SSN_ter );
Encdata = AES (Value , EncKey );
SeparateMac = SliceMac(Macdata_SecureElement);
If (SeparateMac = = Encdata) then
Verify := true;
Else
Verify := false;
End if
}
endprocedure
```
(۵)

تمام مقادیری که در این میان رد و بدل می‌شوند از قبیل: عدد تصادفی و SSN درصورتی‌که مورد شنود و حمله قرار بگیرند هیچ اطلاعاتی نصیب مهاجم نخواهد شد.

درصورتی‌که، احرازهویت دوطرفه با موفقیت انجام شود ترمینال و گوشی هوشمند شروع به ارسال دستور جهت برقراری تبادل اطلاعات می‌کنند.

لازم به ذکر است درحالتی‌که المان‌امن در سیم‌کارت جاسازی شود، سیم‌کارت گوشی نیز دارای شماره سریال منحصربه فردی با نام ICCID[1] می‌باشد و این شماره منحصربه فرد را می‌توان جهت شناسایی در روال احرازهویت و شناسایی در شبکه‌های مخابراتی استفاده نمود. همچنین، برای اهداف مختلف از IMSI[2] که شماره سریال منحصربه‌فرد گوشی‌همراه است نیز می‌توان استفاده کرد. در واقع، المان امن به هر نحوه که پیاده‌سازی شود، می‌تواند دارای یک شماره سریال معتبر در اکوسیستم پرداخت که شامل فراهم‌کننده سرویس پرداخت، اپراتور و ... است، باشد.

۷- روش پیشنهادی نصب نرم‌افزار کیف‌پول گوگل

کاربران برای نصب برنامه‌های مختلف می‌توانند به بازارها[3] یا مراکزتوزیع برنامه‌های کاربردی مراجعه‌کنند. برخی از پلتفرم‌های تلفن‌همراه فقط اجازه نصب برنامه از یک بازار مرکزی را می‌دهند، این پلتفرم‌ها مدل توزیع متمرکز دارند، در حالی‌که برخی پلتفرم‌های دیگر اجازه نصب از چندین بازار کمکی را می‌دهند. برنامه‌ها بایستی قبل از نصب امضاء رمزنگاری دیجیتال شوند. در پلتفرم‌هایی که مدل توزیع متمرکز دارند، بازار به‌عنوان یک مرجع قابل اعتماد عمل می‌کند. اپراتور بازار، برنامه‌ها را تست و امضاء می‌کند. علاوه بر این، اپراتور بازار می‌تواند توسعه‌دهنده برنامه را احرازهویت کند و هویت توسعه‌دهندگان برنامه را در بسته‌های نصب برنامه توزیع شده قرار دهد.

در پلتفرم‌هایی که مدل توزیع آن‌ها بازارهای کمکی است، ارائه‌دهنده بازار یا توسعه‌دهنده، برنامه را امضاء می‌کند. تضمین امضای بازار کمکی وابسته به قابلیت اعتماد به اپراتور بازار است. امضای توسعه‌دهنده تضمینی برای برقراری الزامات ضروری نیست، بلکه هدف اصلی اثبات اصالت برنامه برای به‌روزرسانی‌های آتی توسط توسعه‌دهنده است.

این امضای توسعه‌دهنده، اعتباری برای برنامه از دیدکاربر ایجاد نمی‌کند. برنامه‌ها براساس امضای بازار شناخته می‌شوند.

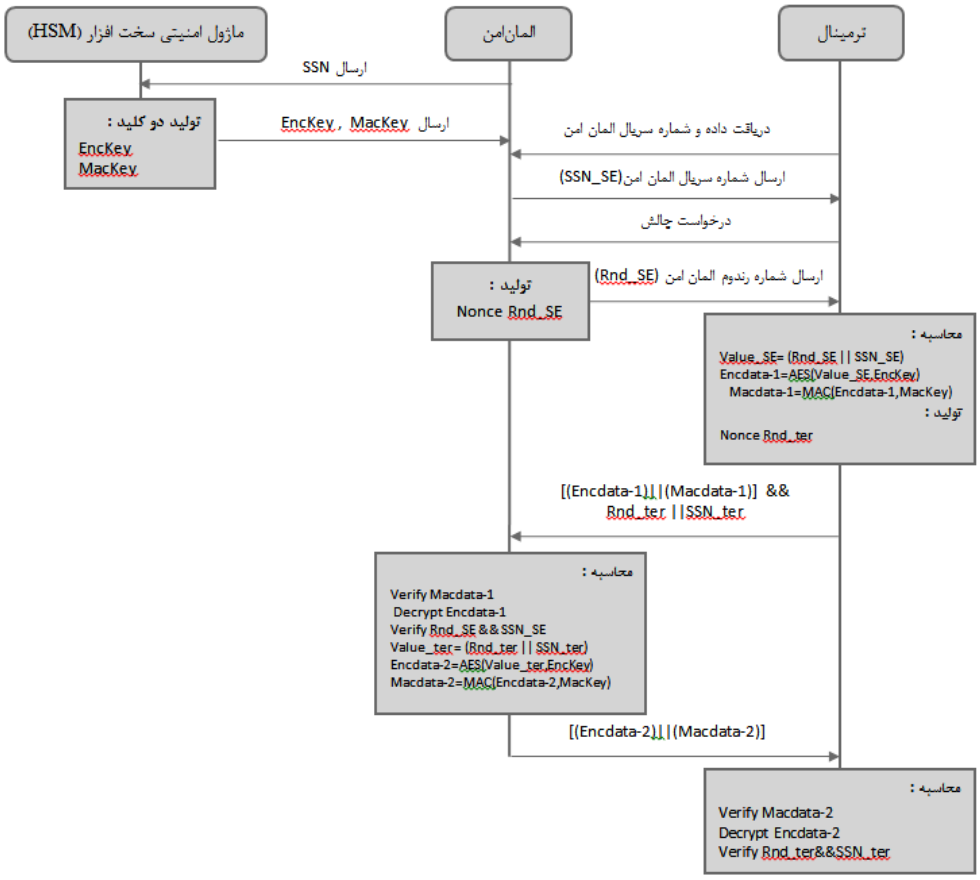
1-Integrated Circuit Card Identifier
2- International Mobile Subscriber Identity
3- Marketpalces

بنابراین، ترکیبی از کلید بازار و شناسه برنامه در بازار یک شناسه منحصر به فردرا فراهم می‌کند. در زمان توزیع برنامه، توسعه‌دهنده، مجوزهایی که برنامه یا سرویس برای دسترسی به واسطها و منابع سیستم نیاز دارد را تعریف می‌کند. توسعه‌دهنده درخواست مجوز را در فایل تنظیمات برنامه یا بسته توزیع شده آن قرار می‌دهد. این فایل با نام Manifest شناخته می‌شود [۳].

زمانی‌که یک برنامه نصب می‌شود، نصب‌کننده برنامه صحت امضای برنامه را بررسی می‌کند و فایل مجوزهای درخواستی برنامه را دریافت می‌کند. سپس یک پایگاه‌داده با توجه به مجوزهای درخواستی و امضاء تولید می‌کند. نصب‌کننده برنامه بررسی می‌کند که مرجعی که برنامه را امضاء کرده حق اعطای چه مجوزهایی را دارد و آیا مجوزهای درخواستی برنامه در این مجموعه قرار می‌گیرد [۳].

در فرایند نصب نرم‌افزار مجوزهای درخواستی به برنامه برای استفاده از منابع تخصیص داده می‌شود. این تخصیص مجوزها می‌تواند براساس امضای برنامه توسط مرجع قابل اعتماد باشد و یا کاربر خود اجازه برخی مجوزهای درخواستی را صادر کند. درواقع، مجوزهایی که دسترسی به اطلاعات خاص سیستم را فراهم می‌کنند، منحصراً براساس امضاء می‌باشند و مجوزهایی که دسترسی به اطلاعات کاربر را فراهم می‌کنند با اجازه کاربر اعطا می‌شوند. پس از نصب برنامه فایل اجرایی برنامه، مجموعه مجوزهای تخصیص داده شده و شناساگر برنامه در پایگاه داده برنامه[1] ذخیره می‌شود [۳].

در شکل (۸) روال احرازهویت به صورت نمودار توالی نمایش داده شده است.

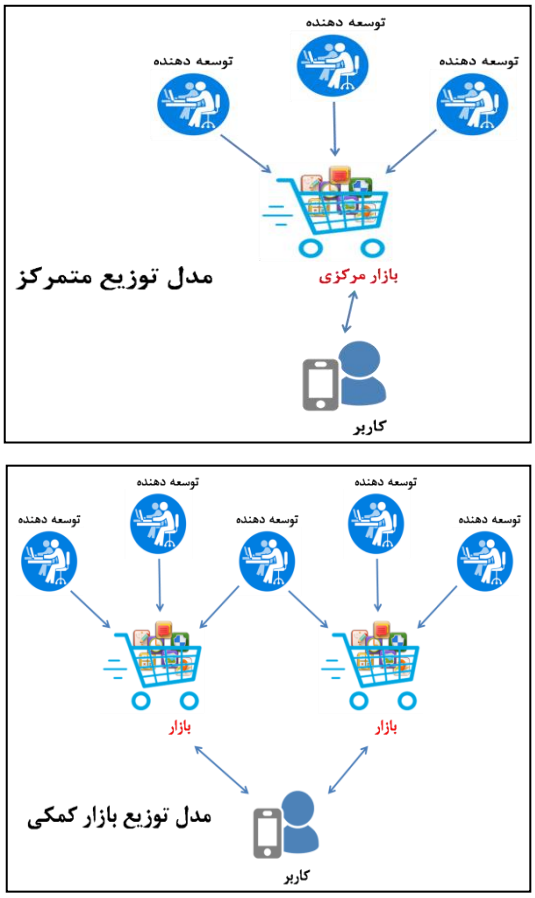


شکل (۸): نمودار توالی احراز هویت دوطرفه پیشنهادی

همان‌طورکه در شکل (۹) نشان داده شده است یک نرم‌افزار می‌تواند به دو روش: ۱- مدل توزیع متمرکز، ۲- مدل توزیع بازار کمکی در روی گوشی هوشمند کاربر نصب گردد. در روش اول، نرم‌افزار توسط یک بازارمرکزی (در این‌جا بانک) در روی گوشی کاربر نصب و راه‌اندازی می‌شود. در این روش، بازار مرکزی مسئول تمام اتفاقات در روی گوشی کاربر است و تمامی برنامه‌ها و اپلت‌هایی که باید به صورت TA نصب شوند تحت‌نظر آن انجام می‌شود. اما در روش دوم، کاربر نرم‌افزار کیف‌پول گوگل را می‌تواند از بازارهای مختلفی دریافت نماید. در حقیقت، روش دوم

1- Application Database

نسبت به روش اول این مزیت را دارد که نیازی به یک بازار مرکزی نیست و لازم نیست فقط یک بازار مرکزی نرم‌افزار مورد اعتماد را در روی گوشی هوشمند نصب کند. پیشنهاد نویسندگان این مقاله جهت نصب ایزوله نرم‌افزار کیف‌پول گوگل، استفاده از روش اول یعنی مدل توزیع متمرکز می‌باشد.



شکل (۹): دو روش نصب نرم‌افزار در روی گوشی هوشمند

۸- جلوگیری از حمله رله با استفاده TEE

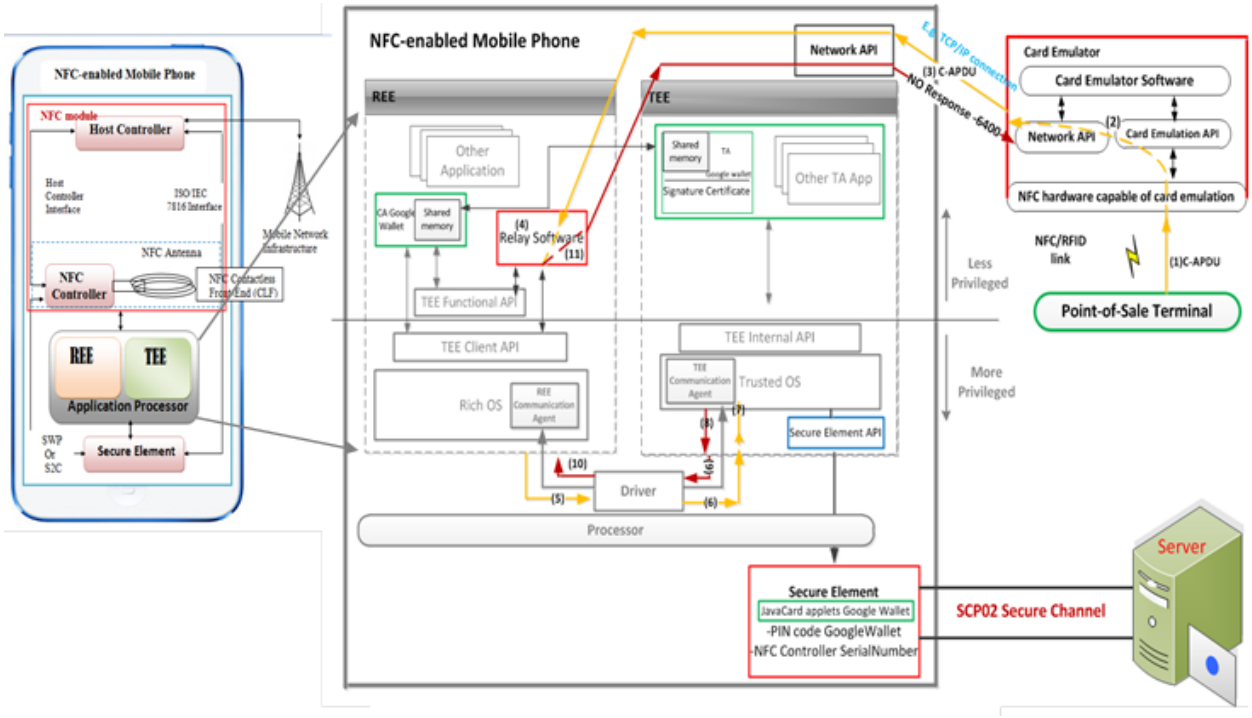
در این قسمت به بررسی راه‌حل پیشنهادی جهت جلوگیری از حمله رله می‌پردازیم. لازم به ذکر است که جلوگیری از حملات دیگر ازجمله حمله Dos[1] و ... جزء اهداف پژوهش حاضر نمی‌باشند. راه‌حل پیشنهادی استفاده از فناوری TEE جهت اجرای ایزوله نرم‌افزارها است. با توجه به این موضوع که سیستم عامل، محیط اجرا و نصب پیش‌فرض برنامه‌های کاربردی بر روی گوشی تلفن‌همراه می‌باشد. همین مسئله سبب می‌شود کیف‌پول گوگل بر روی سیستم‌عامل نصب شود؛ نرم‌افزار رله نیز در سیستم‌عامل گوشی نصب می‌شود و بنابراین، نرم‌افزار رله به راحتی می‌تواند به المان امن دسترسی پیدا کند.

باتوجه به توضیحات داده شده و بررسی المان امن و مدل‌های مختلف پیاده‌سازی آن، می‌توان به این نتیجه رسید که تمامی مدل‌های مختلف پیاده‌سازی المان امن به تصمیم‌گیری کنترل دسترسی سیستم‌عامل اعتماد می‌کنند. بنابراین، یک برنامه مخرب (همانند نرم‌افزار رله) به‌راحتی می‌تواند بررسی‌های امنیتی که توسط سیستم‌عامل انجام می‌گیرد را دور زده و شروع به تعامل، ارسال و دریافت APDUها با المان امن نماید و حمله موفقی را نصیب مهاجم کند. از آن‌جایی‌که TEE در گوشی تلفن همراه ناحیه ایده‌آلی برای قراردادن موتور تطابق و فرایندهای مرتبط جهت احرازهویت است، بنابراین، امنیتی که توسط این محیط افزوده می‌شود می‌تواند به محافظت داده بپردازد و همانند سپری در برابر برنامه‌های مخرب که در سیستم‌عامل تلفن‌همراه واقع شده‌اند، عمل نماید. قرارگرفتن نرم‌افزار کیف‌پول گوگل در TEE سبب می‌شود فرایند احرازهویت کاربر با گرفتن رمز عبور، به‌صورت رمزشده در داخل المان امن بررسی و درصورت صحیح‌بودن تائید شود. همچنین، با فرایند اعتبارسنجی تراکنش‌ها که توسط TEE پشتیبانی می‌شود امکان نمایش اطلاعات صحیح وجود دارد و اعتبارسنجی تراکنش‌ها بدون اجازه کاربر ممکن نمی‌باشد.

با توجه به این مسئله که نرم‌افزار رله هیچ تغییری در داده‌های دریافت شده ایجاد نمی‌کند و صرفاً آن‌ها را انتقال می‌دهد این حمله را می‌توان با اعتبارسنجی رمز عبور توسط المان امن و همچنین قراردادن نرم‌افزار کیف‌پول گوگل در TEE و عدم دسترسی به المان امن جلوگیری نمود. در واقع، با قراردادن نرم‌افزار کیف‌پول گوگل به‌عنوان یک TA می‌توان از هر فعالیت مخرب توسط برنامه‌های کاربردی تلفن‌همراه جلوگیری کرد. هم‌چنین، ایزوله نرم‌افزاری و رمزنگاری در داخل TEE، سبب می‌شود نرم‌افزار رله نتواند به داده‌های کیف‌پول گوگل دسترسی یابد. پشتیبانی از ایزوله سخت‌افزاری، سبب می‌شود TA نرم‌افزار کیف‌پول گوگل در TEE از نرم‌افزار رله که بر روی Rich OS در حال اجرا و فعالیت است محافظت شود. این عمل در واقع استفاده از هدف اجرای ایزوله‌شده نرم‌افزارها در محیط TEE می‌باشد. همان‌طورکه در شکل (۱۰) می‌بینید در معماری پیشنهادشده CA نرم‌افزار کیف‌پول گوگل در Rich OS قرار می‌گیرد و TA این نرم‌افزار که توسط منبع مورد اعتماد امضاء شده و دارای گواهینامه[2] گردیده است در TEE مستقر می‌شود و گواهینامه‌های آن به صورت امنی در حافظه مدیریت‌شده توسط TEE نگهداری می‌شود.

1- Denial Of Service

2- Certificate



شکل (۱۰): معماری پیشنهادی محیط TEEجهت جلوگیری حمله رله

در پردازنده‌هایی که قابلیت پیاده‌سازی TEE را پشتیبانی می‌کنند (از قبیل ARMTrustZone)[۱] حافظه فیزیکی به دوناحیه قابل‌اعتماد (Trusted) و غیرقابل‌اعتماد (UnTrusted) تقسیم می‌شود. حافظه UnTrusted می‌تواند به‌عنوان حافظه مشترک درنظر گرفته شود که توسط هر دو محیط به منظور ایجاد ارتباط استفاده می‌گردد. در حالت کلی، TEE از یک حافظه امن(Secure) و یک حافظه به‌اشتراک گذاشته‌شده (Shared) استفاده می‌نماید. محیط REE اجازه دسترسی به حافظه Secure را ندارد [۱۳].

حافظه مشترک(Shared Memory) بین CA و TA نرم‌افزار کیف‌پول گوگل باعث می‌شود CA و TA در حین ارتباط به میزان داده‌های زیادی با سرعت بالا و کارائی خوب دسترسی پیدا کند. در ضمن، این ویژگی توسط Trusted OS پیاده‌سازی می‌شود و به‌عنوان Shared Memory شناخته می‌گردد. پروتکل این ارتباط توسط طراحان TA نرم‌افزار کیف‌پول گوگل طراحی و توسط TEE Client API و TEE Internal API فعال و به‌کار گرفته می‌شود. در استفاده از حافظه مشترک باید مراقبت‌های لازم با جنبه‌های امنیتی درنظر گرفته شود چراکه CA و TA نرم‌افزار کیف‌پول گوگل دارای پتانسیل‌های لازم جهت تغییر محتویات حافظه آن‌هم به‌صورت هم‌زمان می‌باشند. درایور[۲] در حقیقت سوئیچ و جابه‌جایی بین این دو محیط را مدیریت می‌کند.

پروتکل SCP02 پروتکل کانال امنی است که توسط GP استاندارد شده است. با دقت در معماری که در ادامه آمده است در می‌یابیم که المان‌امن با سرور نرم‌افزار کیف‌پول گوگل از طریق کانال امن SCP02 ارتباط امنی را برقرار می‌نماید. هدف از برقراری کانال امن تصدیق هویت موجودیت‌های خارج المان امن و اطمینان از جامعیت داده‌ها[۳] است.

رمزنگاری مهم‌ترین کاری است که توسط کانال امن SCP02 انجام می‌شود. درواقع، این عمل مانع از آن می‌شود که افراد ناشناخته بتوانند بر روی المان امن واقع در گوشی هوشمند، اپلت یا برنامه‌ای را نصب نماید.

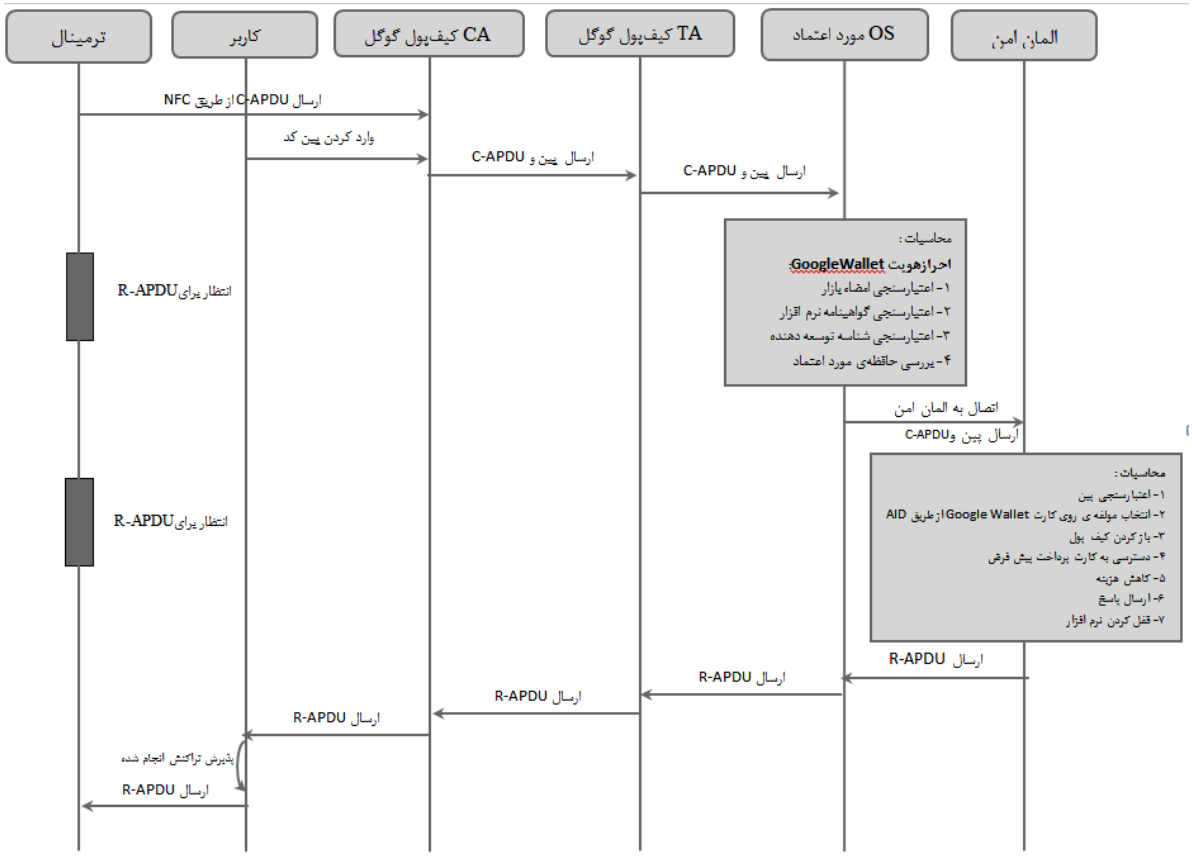
مدیر المان امن (ISD) مسئول تمام اعمالی است که بر روی المان امن انجام می‌شود و درصورتی‌که اپلت کیف‌پول گوگل بخواهد در روی المان امن نصب شود باید زیر نظر ISD نصب شود و این به‌دلیل آن است که ISD دارای کلیدهایی است که قادر به نصب نرم‌افزارهای قابل اطمینان در روی المان‌امن می‌باشد و

۱- نمونه‌ای از پیاده‌سازی سخت‌افزاری TEEدر پردازنده‌های تلفن‌های هوشمند

2- Driver
3- Data Integrity

سرور در واقع عمل مدیریت کارت را از طریق کانال رمزنگاری شده و احراز هویت شده، انجام می‌دهد.

با دقت در این نکته که امنیت المان امن از TEE فراتر است، بنابراین، اپلت‌های جاواکارت و رمز عبور و کلیدهای ضروری نرم‌افزار کیف‌پول گوگل در المان‌امن قرار می‌گیرد. بنابراین، با این عمل هم سطح نرم‌افزار و هم سطح داده‌ها و اطلاعات ضروری از امنیت کافی برخوردار می‌شوند. به‌دلیل این‌که المان امن مستقیماً با TEE در ارتباط است لذا با استفاده از یکی از اهداف TEE مبنی بر اجرای ایزوله‌شده و راه‌اندازی احرازهویت‌شده، نرم‌افزار رله ابتدا باید به‌صورت ایزوله اجرا و توسط Trusted OS احرازهویت شود تا بتواند از طریق API المان امن به اپلت‌های نرم‌افزار کیف‌پول گوگل که در المان‌امن مستقر شده است دسترسی یابد و به سوءاستفاده از اطلاعات کارت‌های پرداخت ذخیره‌شده بپردازد، اما به‌دلیل این‌که هیچ امضای کد و یا رشته درهم‌شده[1] از کد نرم‌افزار رله در موتور تطابق Trusted OS وجود ندارد، لذا، این نرم‌افزار توسط Trusted OS احرازهویت نمی‌شود و دسترسی به آن اعطا نمی‌گردد و در این مرحله با شکست روبرو می‌شود، عملاً حمله متوقف و ممکن است از سمت Trusted OS به‌عنوان نرم‌افزار مخرب شناسایی و بلوکه[2] گردد و یا در حالتی دیگر، ممکن است دستورات و فرمان‌هایی که در دفعات بعدی از طرف این نرم‌افزار برای Trusted OS ارسال می‌گردد مورد بررسی قرار نگیرد. در شکل (۱۱) نمودار توالی پرداخت امن از طریق کیف‌پول گوگل با استفاده از TEE و در شکل (۱۲) نمودار توالی بلوکه‌شدن حمله رله در گوشی تلفن همراه را مشاهده می‌کنید.

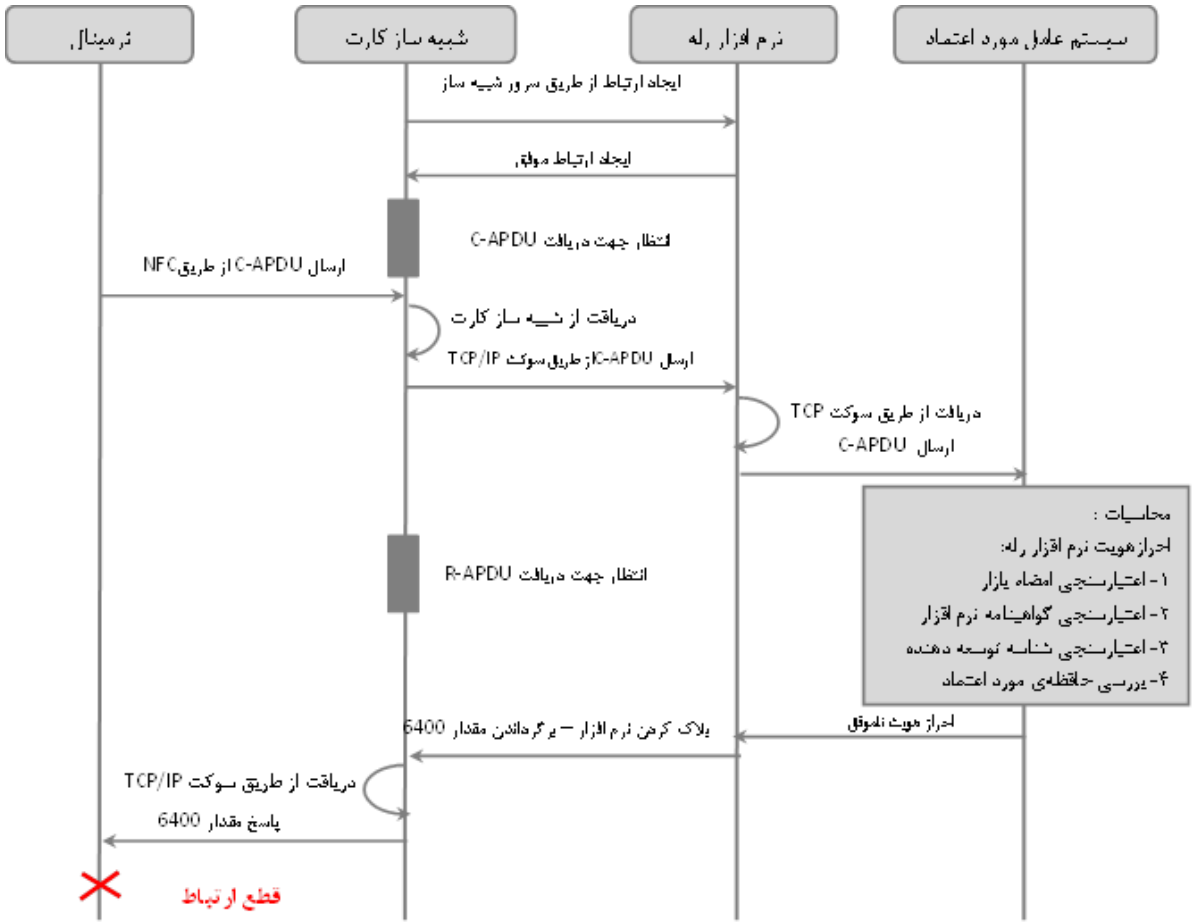


شکل (۱۱): نمودار توالی پرداخت امن از طریق کیف‌پول گوگل با استفاده از TEE

۹- پیاده‌سازی

در این قسمت جهت اثبات عملی‌بودن طرح پیشنهادی، یک نمونه نرم‌افزار آزمایشگاهی بر روی سیستم عامل اندروید پیاده‌سازی شده است. از آن‌جایی‌که ایده مطرح‌شده پیرامون ارتباط نرم‌افزار تلفن همراه (CA_Google Wallet) با TEE و المان امن می‌باشد، ارتباط‌ها با سرور Google Wallet همان ارتباطات نشان‌داده‌شده در معماری پیشنهادی است، در نتیجه، از پیاده‌سازی آن‌ها صرف نظر شده است.

1- Hash
2- Block



شکل (۱۲): نمودار توالی بلوکه‌شدن حمله رله در گوشی تلفن همراه

جهت پیاده‌سازی نرم‌افزار اندرویدی، از محیط برنامه نویسی Android Studio نسخه 2.2.3 استفاده شده است و اجرای برنامه بر روی یک دستگاه تلفن همراه Galaxy note 2 از شرکت سامسونگ آزمایش شده است. برای کار با TEE از یک شبیه‌ساز نرم‌افزاری [۵] استفاده شده است که سخت‌افزار TEE را به‌صورت کامل بر روی سیستم عامل اندروید شبیه‌سازی کرده است. استفاده از شبیه‌ساز TEE شبهه‌ای در اثبات عملی‌بودن ایده پیشنهادشده ایجاد نمی‌کند و درصورتی‌که برنامه نوشته‌شده بر روی یک تلفن همراه مجهز به TEE سخت‌افزاری اجرا شود، روال بدون تغییر در کد برنامه اجرا خواهد شد.

ساختار برنامه اندرویدی شامل یک Activity اصلی است و برای ارتباط با هریک از موارد المان امن و TEE یک کلاس جداگانه تعریف شده است. همچنین، در سیستم عامل اندروید جهت ارسال دستور به TEE از کتابخانه OpenTEE استفاده شده است. در ادامه به‌صورت جداگانه شیوه پیاده‌سازی هریک از این ارتباطات توضیح داده می‌شود.

جهت پیاده‌سازی نمونه آزمایشگاهی، از یک کارت شناسایی هوشمند با نام تجاری آیدین از شرکت پندارکوشک ایمن به‌عنوان المان امن استفاده شده است. اطلاعات کارت‌های اعتباری صاحب کارت و مقدار رمز عبور ("۱۱۱۱") در قالب یک دستور APDU برای المان امن فرستاده می‌شود. درصورتی‌که تعداد کاراکترهای رمز عبور کم‌تر از هشت رقم باشد می‌بایست با مقادیر 0xFF تا رسیدن به طول مورد نظر، پر شود. همچنین، المان امن (کارت آیدین) امکان انجام عملیاتی از جمله احراز صحت پین کاربر، تغییر پین کاربر، خارج‌کردن پین بلاک‌شده، امضای دیجیتال و رمزنگاری (با استفاده از الگوریتم RSA با طول کلیدهای ۱۰۲۴ و ۲۰۴۸ بیتی) را قادر به انجام است.

برای ارتباط برنامه اندرویدی با کارت، لازم است یک دستگاه کارتخوان در دسترس باشد. به همین منظور، از یک دستگاه کارتخوان HID OMNIKEY 3121 استفاده شده است. لازم به ذکر است که کارتخوان‌ها برای ارتباط با سیستم عامل به یک درایور نرم‌افزاری نیاز دارند. در سیستم عامل‌های ویندوز و

لینوکس درایور مربوطه به‌صورت رایگان موجود می‌باشد ولی برای سیستم‌عامل اندروید این درایور به‌صورت رایگان در دسترس نیست و مجوز آن باید خریداری گردد. برای این‌کار از مجوز شرکت SCDROID که توسط شرکت پندارکوشک ایمن خریداری شده است، استفاده می‌شود.

قبل از ارسال دستور، Trusted OS می‌بایست صلاحیت برنامه اندرویدی را برای ارتباط با المان امن تایید نماید. برای احراز صحت برنامه ارتباط گیرنده با المان امن، گواهی منطبق بر کلید خصوصی که برنامه را امضاء کرده است می‌بایست در TEE ذخیره شود. با توجه به این‌که یک برنامه اندرویدی می‌تواند توسط چند مرجع امضاء شود، لذا کد زیر امضای برنامه را خوانده و امکان وجود چند امضاء را بررسی می‌کند:

کد اول به شرح زیر می باشد:

```
byte[] signatureBytes;
PackageInfo packageInfo =
context.getPackageManager().
getPackageInfo(context.getPackageName(),
PackageManager.GET\_SIGNATURES);
for (Signature signature : packageInfo.signatures) {
signatureBytes = signature.toByteArray();
}
```
(۱)

حال باید امضای پیام به همراه چکیده محاسبه‌شده از برنامه برای TEE ارسال شود تا معتبر بودن برنامه به وسیله گواهی امضاءکننده بررسی گردد. برای ارسال داده از CA به TA باید یک حافظه به اشتراک گذاشته شود که CA و TA به آن دسترسی یابند. کد زیر برای ارسال دستور VERIFY_SIGNATURE به TEE استفاده می‌شود.

کد دوم به شرح زیر می باشد:

```
ITEEClient client =
FactoryMethodWrappers.newTEEClient();
ITEEClient.IContext ctx = null;
final String param_TEE_NAME = null; // connect to
the default TEE.
final android.context.Context param_app_context =
getApplicationContext();
try
{
ctx = client.initializeContext(param_TEE_NAME,
param_app_context);
}
catch (TEEClientException e)
{
// handle TEEClientException here.
}
ITEEClient.ISession ses = null;
final UUID param_uuid = new
UUID(0x1234567887654321L,
0x0102030405060708L);
final ConnectionMethod param_conn_method =
ITEEClient.IContext.ConnectionMethod.LoginPubli
c;
final Integer param_conn_data = null;
try {
ses = ctx.openSession(param_uuid,
param_conn_method,
param_conn_data, param_operation);
} catch (TEEClientException e) {
// handle TEEClientException here.
return false;
}
ITEEClient.ISharedMemory sm = null;
byte[] appSignature = new byte[256];
ITEEClient.ISharedMemory param_flags =
ITEEClient.ISharedMemory.TEEC_MEM_INPUT |
ITEEClient.ISharedMemory.TEEC_MEM_OUTPU
T;
try{
sm = ctx.registerSharedMemory(param_byte_array,
param_flags);
} catch (TEEClientException e) {
// handle TEEClientException here.
return false;
}

final int VERIFY_SIGNATURE_COMMANDID =
0x12345678;
try{
ses.invokeCommand(VERIFY_SIGNATURE_COM
MANDID, param_operation);
}catch (TEEClientException e) {
// handle TEEClientException here.
return false;
}
```
(۲)

درصورتی‌که خروجی دستور VERIFY_SINATURE برابر با مقدار صفر باشد، آن‌گاه، TA اجازه می‌دهد تا دستور بعدی برای المان امن ارسال شود.

برای ارسال دستور VERIFY_PIN ابتدا باید مجوز درایور کارتخوان به پروژه اضافه شود. این کار با اضافه‌کردن تگ XML در فایل Manifest.xml انجام می‌شود.

کد سوم به شرح زیر می‌باشد:

```
<meta-data android:name="com.scdroid.ccid.key"
android:value="ccid licence is placed here" />
```
(۳)

در ادامه، فایل ccid_lib.jar به‌عنوان یک ماژول جدید به پروژه اضافه شده تا توابع ارسال دستور APDU به پروژه اضافه شود. ارسال دستور VERIFY_PIN به‌صورت زیر انجام می‌شود:

کد چهارم به شرح زیر می‌باشد:

```
List<USBReader> ReaderList = null;
ReaderList = USBReader.getReaders(this);
if (ReaderList.size() == 0) {
mMessage.setText("no reader connected");
return;
}
```
(۴)

```
USBReader usbReader = ReaderList.get(0);
usbReader.Open();
usbReader.isCardPresent();
usbReader.WaitCardEvent(USBReader.CARD_EV
ENT_DETECED);

//Select Applet APDU
byte[] selectAppletAPDU = new byte[] { 0x00,
(byte) 0xA4, 0x04, 0x00, 0x09, 0xa0, 0x00, 0x00,
0x03, 0x08, 0x00, 0x00, 0x10, 0x00}
byte[] fci =
usbReader.Transmit(selectAppletAPDU);
//Verify PIN
byte[] verifyPin = new byte[]{0x00, 0x20, 0x80,
0x00, 0x08, 0x31, 0x31, 0x31, 0x31, 0xFF, 0xFF,
0xFF, 0xFF};
byte[] result = usbReader.Transmit(verifyPin);
if(Arrays.equals(result, new byte[]{0x90, 0x00}))
return true;
```

پاسخ SW=9000 از کارت، نشان‌دهنده موفقیت در احراز صحت پین می‌باشد. همچنین، پاسخ بازگردانده‌شده SW= 63CX از کارت نشان‌دهنده اشتباه بودن پین و مقدار X، نشان‌دهنده تعداد دفعات باقی‌مانده جهت واردکردن پین نادرست است. درصورتی‌که مقدار X برابر با صفر باشد، پین بلاک شده و درصورت درست وارد کردن مقدار آن پاسخ خطا برگردانده می‌شود. برای خارج‌کردن پین از وضعیت بلاک باید از دستور RESET_RETRY_COUNTER و مقدار PUK استفاده کرد. در انتهای فرایند، برای افزایش امنیت نرم‌افزار باید جلسه ایجادشده بین TA و CA بسته شود.

کد پنجم به شرح زیر می باشد:

```
try {
ctx.releaseSharedMemory(sm);
} catch (TEEClientException e) {
// handle TEEClientException here.
return false;
}
try {
ses.closeSession();                                    (۵)
} catch (TEEClientException e) {
// handle TEEClientException here.
}
try {
ctx.finalizeContext();
} catch (TEEClientException e) {
// handle TEEClientException here
return false;
}
```

۱۰- نتیجه‌گیری

در این مقاله با بهره‌گیری ازفناوری TEE در گوشی‌های‌هوشمند مجهز به فناوری NFC، سعی شد تا از حمله رله که به‌عنوان حملات شناخته‌شده برای نرم‌افزار کیف‌پول گوگل تلقی می‌گردد، جلوگیری شود. همچنین، با بهره‌گیری از احرازهویت دوطرفه متقارن پیشنهادی به‌راحتی می‌توان ترمینال معتبر را شناسایی و احرازهویت نمود و با علم به این‌که تمام دستورات ورودی در ابتدا توسط Trusted OS محیط TEE مورد بررسی قرار می‌گیرد بنابراین، حمله رله از طریق ترمینال قربانی نمی‌تواند حتی احرازهویت معتبر و موفقی را داشته باشد. همچنین، با بهره‌گیری از محیط TEE جهت نصب ایزوله نرم‌افزارکیف‌پول گوگل، کاربران می‌توانند با اطمینان کاملی اطلاعات مالی را در المان امن ذخیره نمایند. در کارهای پیش‌رو، تحلیل و اثبات‌های فراتری از چگونگی تأمین امنیت توسط TEE ارائه خواهد شد.

۱۱- مراجع

[1] S. Taremi, "A solution to increase the security on smart phones against relay attacks," M.S.thesis, Iran, Tehran, Shahed University, 2016 (in Persian).

[2] G. Arfaoui, S. Gharout, and J. Traoré, "Trusted Execution Environments: A look under the hood," in Mobile Cloud Computing, Services, and Engineering (MobileCloud), 2014 2nd IEEE International Conference on, pp. 259-266, 2014.

[3] N. Asokan, L. Davi, A. Dmitrienko, S. Heuser, K. Kostiainen, E. Reshetova, and A. R. Sadeghi, "Mobile Platform security," Synthesis Lectures on Information Security, Privacy, and Trust, A Publication in Morgan & Claypool Publisher, vol. 4, pp. 1-108, 2014.

[4] S. A. Bailey, D. Felton, V. Galindo, F. Hauswirth, J. Hirvimies, M. Fokle, et al., "The trusted execution environment: Delivering enhanced security at a lower cost to the mobile market," Global Platform White Paper, 2011.

[5] J. Bech, "LCU14-103: How to create and run Trusted Applications on OP-TEE," Linaro, Sep. 2014.

[6] A. Chin, R. Palatucci, and J. Powers, "Google Wallet," Available from: http://www.google.com/wallet/, 2014.

[7] J.-E. Ekberg, K. Kostiainen, and N. Asokan, "The Untapped Potential of Trusted Execution Environments on Mobile Devices," IEEE Security & Privacy, vol. 12, pp. 29-37, 2014.

[8] J. E. Ekberg, "Securing Software Architectures for Trusted Processor Environments," Ph.D. Thesis, Aalto University, 2013.

[9] R. Elicitation Vermaas, T. Tervonen, Y. Zhang, and J. Siljee, "The Security Risks of Mobile Payment Applications Using Near-Field Communication," M.S. thesis, Erasmus University Rotterdam, Rotterdam, The Netherlands, 2013.

[10] J. Jang, S. Kong, M. Kim, D. Kim, B. Byunghoon Kang, "SeCReT: Secure Channel between Rich Execution Environment and Trusted Execution Environment," Graduate School of Information Security, 2015.

[11] A. J. Jara, A. F. Alcolea, M. A. Zamora, and A. F. G. Skarmeta, "Evaluation of the security capabilities on NFC-powered devices," in Smart Objects: Systems, Technologies and Applications (RFID Sys Tech), 2010 European Workshop on, pp. 1-9, 2010.

[12] M. Kerschberger, "Near Field Communication: A survey of safety and security measures," Bachelorarbeit thesis, Vienna University of Technology, July 2011.

[13] F. Kvant and M. Kellner, "A Development Environment for ARM TrustZone with GlobalPlatform Support," M.S. thesis, Lund University, LTH, 2014.

[14] B. Lepojevic, B. Pavlovic, and A. Radulovic, "Implementing NFC service security–SE VS TEE VS HCE," presented at the SYMORG Conference, 2014.

[15] M. Mattsson, "Security and Infrastructure for Mobile Phone Payments using Near Field Communication," M.S. thesis, KTH Royal Institute of Technology, 2010.

[16] T. Nyman, B. McGillion, and N. Asokan, "On Making Emerging Trusted Execution Environments Accessible to Developers," in International Conference on Trust and Trustworthy Computing, Springer, pp. 58-67, 2015.

[17] M. Riyazuddin, "NFC: A review of the technology, applications and security," ABI research- Riyazuddin, 2013.

[18] M. Roland, "Applying recent secure element relay attack scenarios to the real world: Google Wallet Relay Attack," NFC Research Laboratory, Technical Report, 2012.

[19] O. Solsjö, "Secure key management in a trusted domain on mobile devices," M.S. thesis, Linköpings Universitet, 2015.

[20] G. Platform, "TEE Secure Element API," Global Platform technical overview, Version 1.1, 2015.

[21] "NVIDIA: Trusted Little Kernel (TLK)," Available from: http://nv-tegra.nvidia.com/, 2015.

[22] G. Platform, "TEE Internal Core API Specification," Global Platform technical overview, Version 1.1, 2014.

[23] G. Platform, "TEE System Architecture," Global Platform technical overview, Version 1.0, 2011.

Journal of Electronical & Cyber Defence
Vol. 6, No. 1, 2017, Serial No. 21

An Innovative Solution for Preventing Relay Attack on Mobile Phones Using TEE

S. Taremi, M. A. Doostari*, S. Hajimohseni, M. Maghsoudloo, M. Mayabi Joghal

*Shahed University

(Received: 13/01/2016, Accepted: 31/10/2016)

ABSTRACT

In recent years, thanks to NFC technology, the GoogleWallet application has attracted the attention of many users for payment purposes. This application is installed and run on mobile platforms without considering enough security. It uses the secure element as a proper place to store users' financial information. As a result, this application is easily susceptible to many security risks such as relay attacks. Therefore, security provision in such systems has always been one of the most important challenges. A suitable mechanism to provide security is installing and running the application in an isolated form. However, before installing and running the application in an isolated form, the mobile phone should be equipped with a secure environment that is provided by Trusted Execution Environment(TEE). The paper proposes a mutual authenticatiom method between secure element and terminal using mobile phones whose processors have two execution environments (TEE/REE) to manage the execution of google wallet application. The proposed method not only improves the security of GoogleWallet application against the relay attack, but also similar applications can use this method to increase security against such attacks.

Keywords: NFC Technology, Google Wallet, Secureelement, Relayattack, Trusted Execution Environment (TEE)

* Corresponding Author Email: doostari@shahed.ac.ir