A Security Based Comparison of the Apple Pay and Google Pay Mobile Payment Platforms

Wesam Elhanafi, Alexander Lotero, Jacob Goldfischer, Paul Mauro
we2032@nyu.edu, aal562@nyu.edu, jeg535@nyu.edu, pmm9694@nyu.edu
New York University Tandon School of Engineering

Introduction

Mobile payment, “a money payment made for a product or service through a portable electronic device such as a tablet or cell phone,” [1] has seen a strong push from mobile hardware companies in the last decade. Beginning with Google Wallet in 2011, the predecessor to the modern Google Pay, companies like Google, Apple, and Samsung have been partnering with financial institutions, developing the necessary wireless protocols, and encouraging consumers to digitize their payment cards for use within their respective mobile applications. Apple Pay was released to the public in October of 2014 and Google’s current Google Pay service was released in January of 2018.

The motivation behind the mobile pay movement is one of improved user experience. By eliminating the physical card in people’s wallets, customers no longer have to worry about losing their card and discovering mysterious charges on their bank statements. Customers also no longer have to memorize card numbers in order to speed-up the online checkout experience. By implementing the entire payment process onto a smartphone, mobile device manufacturers inch closer to their goal of making your mobile phone the only necessity to have on your person at all times.

Despite the numerous benefits that come with the switch to the mobile payment scheme, a new way to pay means an entirely new attack surface. The security of a user’s financials is no longer linked to the confidentiality of a single card within their wallet, but now depends on the security status of their device, the integrity of the protocols in place to process transactions, and the policies chosen by the device manufacturers. Our work will focus on the latter two: the protocols used to perform the transactions and the mobile payment policies enacted by device manufacturers.

Background

While other options exist in the market, the two largest players are Apple Pay (for IOS devices) and Google Pay (for Android devices), both of which utilize the near-field communication (NFC) technology for data transmission. NFC technology, an evolution of the radio frequency identification (RFID) technology, enables mobile devices to transfer information to a point-of-sale terminal either through touch or placing the devices within centimeters of one another [2]. Both devices, the mobile and the point-of-sale terminal, must be outfitted with NFC chips to enable the transaction.

The biggest difference between Apple Pay and Google Pay is their adopted policy for user data. Both services provide user card confidentiality by issuing a digital replacement for the card for use in transactions to ensure that the seller never sees the user's original card data. Apple issues what they call a Device Account Number, a device-specific identifier stored on the mobile device in specialized hardware isolated from the rest of the system called the Secure Element [3]. While Apple’s databases do store portions of this information, Apple claims to never keep the user’s full card information or their Device Account Number on their servers. Similarly, Google generates a digital card that lacks any resemblance to the user’s original card. This newly issued card is then charged for transactions and then that same amount is taken out of the user’s original card by Google. The user’s original card information is stored, in its entirety, within Google databases along with transaction histories [4].

Both organizations highlight their security infrastructure and emphasize the importance of protecting customer data, but as with all network-based services, vulnerabilities have been discovered.

Researchers have bypassed lock-screens to access Apple Pay transactions [5], discovered single key and initialization vector reuse in Samsung phones [6], and unsecure payment cards created upon linking Google Pay and Paypal accounts [7].

Related Work

Mobile payment systems in their modern form, i.e. transactions completed through close-proximity, wireless communication, is a new and developing field. Google Wallet, the first of these modern mobile payment systems, was only launched eleven years ago in 2011. Since then, competition from the likes of Apple and Samsung, as well as further developments in the near field communication (NFC) standard, has caused a unique evolution from Google's original vision. All this to say, analysis performed on these services even just a couple of years ago, may no longer reflect their most up-to-date implementations.

Earlier work by Kazan [15] in 2015 pitted Apple's one year old service against Google's four year old service, at the time still known as Google Wallet. The researcher approached the comparison from a technological innovation and commercialization perspective. While similar to our work in that considerations included hardware developments and policy decisions, our approach is one of security techniques. Specifically, what hardware design and policy decisions were made in accordance to the respective companies' security goals.

Jawale et al. performed a deep dive into Apple Pay's security architecture in 2016 [14]. Their analysis discusses the state of the service and its security features. Their work identified vulnerabilities in the implementation and the potential improvements to combat these vulnerabilities. While a stepping stone of our work, this research predates Apple's security focused hardware found within the Secure Element. This new chip, isolated from the main processor and the OS, redefined the security mechanisms of the service and thus warrant a reevaluation of the service's security posture.

Finally, a survey administered by Huh et al. showed that the primary reason for users deciding against the use of mobile payment services, including Apple Pay and Google Pay, was security concerns [16]. The researcher's survey attempts to understand both a) are the average smartphone users comfortable with mobile payment services? and b) if not, why not? Much like Jawale's work [14] before it, Huh's survey was offered before the introduction of Apple's Secure Element. This isolated hardware directly addresses the surveyed user's biggest concern: "storing card information on their phones is less secure than physically carrying cards inside their wallets."

Our work attempts to provide an up-to-date analysis of the two largest mobile payment services available today. By approaching the topic through a comparative lens, between Apple Pay and Google Pay, our discussion provides additional context through insight into the mobile payment industry as a whole. By understanding the best that these services have to offer today, researchers can make informed decisions about where these services might be in future iterations.

Apple Pay Security[8]

We will briefly identify the main security components implemented by Apple to enable mobile payments at Point of Sale systems using Apple Pay. We outline Apple's implementation of mobile payments to Point of Sale within Apple Pay. Apple Pay security begins with the Secure Element, a financial industry certified chip that runs the Java Card Platform and is included in all Apple mobile devices. Next is the NFC controller responsible for the Near Field Communication protocols and routing communication between the Application Processor and the Secure Element. The NFC controller routes communication between the Secure Element and retail POS systems. In order to initiate transactions, users must first be authenticated by the Secure Enclave. The Secure Enclave is an additional, isolated processor with a hardware filter that cannot be accessed by the main Application Processor. Users manage Apple Pay through an application included with IOS called Wallet. Within the Wallet app, users can add and manage credit/debit cards, view transaction histories, etc. As for Apple's infrastructure, dedicated Apple Pay servers handle the setup and provisioning of credit / debit cards for use within the Wallet app. The Wallet app permits management of device account numbers and the "digital card" issued to replace the original card for transactional privacy. Apple Pay servers communicate with the mobile device and the payment network.

The mobile payment process begins with card provisioning within the Wallet app. Card details can be either typed manually or obtained by the app by requesting the user take a picture of the card. Note, the picture is not stored on the device beyond the provisioning process. Once card details are received by the app, the data is encrypted and sent using TLS 1.2 to the Apple Pay Servers and the card issuer for approval. The Device Account Number, the number serving as a newly issued digital card, is isolated from the OS on the Secure Element and is never backed up to iCloud. This protects against any potential malware on the iPhone accessing this Device Account Number. Any card added in the Wallet app is linked to the user through their iCloud account. iCloud accounts require two-factor authentication to be enabled in order to manage payment cards added to Apple Pay.

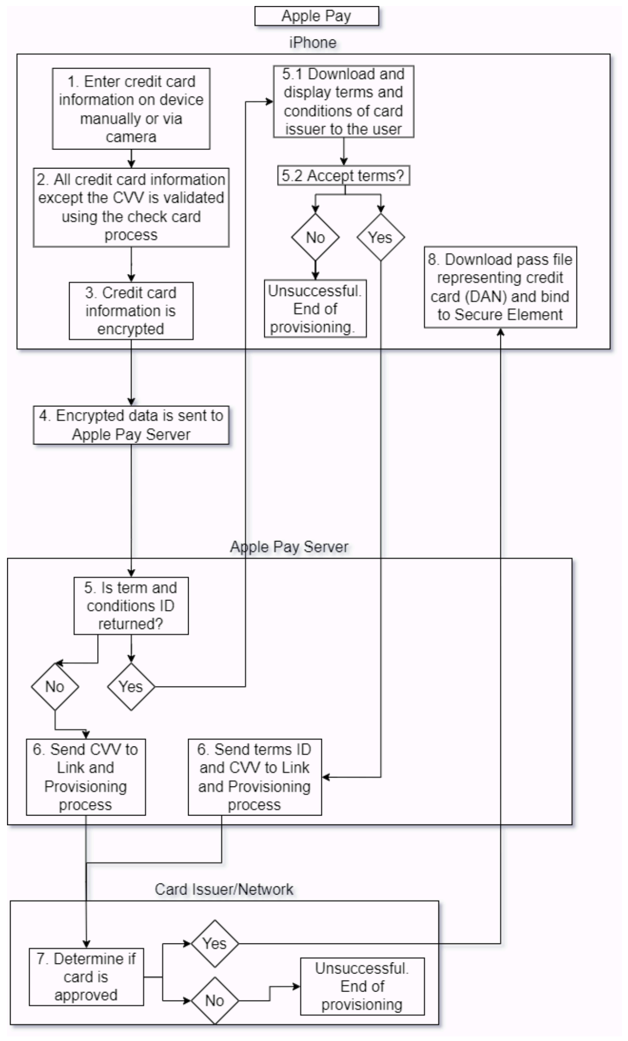
Apple Pay
iPhone
1. Enter credit card information on device manually or via camera
2. All credit card information except the CVV is validated using the check card process
3. Credit card information is encrypted
5.1 Download and display terms and conditions of card issuer to the user
5.2 Accept terms?
No
Yes
Unsuccessful. End of provisioning.
8. Download pass file representing credit card (DAN) and bind to Secure Element
4. Encrypted data is sent to Apple Pay Server
Apple Pay Server
5. Is term and conditions ID returned?
No
Yes
6. Send CVV to Link and Provisioning process
6. Send terms ID and CVV to Link and Provisioning process
Card Issuer/Network
7. Determine if card is approved
Yes
No
Unsuccessful. End of provisioning

Figure 1: Apple Pay process diagram.

Authentication occurs within the Secure Enclave portion of the iPhone and is initiated via Face ID, Touch ID, or device passcode. The process involves communication between the Secure Enclave and the Secure Element via a serial interface.

Next, authorization of each individual transaction involves the Secure Enclave sending the signed transaction data to the Secure Element, including an “authorization random value” (AR). The AR value was generated during card provisioning within the Secure Enclave’s TRNG function. The signed transaction data and AR Value are encrypted at rest using AES within the Secure Enclave’s dedicated secure nonvolatile storage.

The payment process then continues from the Secure Enclave to the POS terminal via the NFC controller. The java applet inside the Secure Element adds a one-time code (payment cryptogram), the device account number, and the transaction data and sends it to the POS terminal. This one-time code is a combination of a transaction counter and a key *K*. The key is a unique identifier provisioned by the java applet during initial card provisioning within the Apple Wallet app. Further, there is a terminal unpredictable number (TUU), generated randomly for NFC communication that protects against easy prediction of this value by an adversary. Both the one-time code’s key *K* and the TUU are provided to the card issuer in order to verify each transaction.

As mentioned previously, the Secure Enclave plays a major role in Apple Pay security. The chip includes a standalone TRNG function, AES encryption engine, Memory Protection Engine, Public Key Accelerator which is used to perform asymmetric cryptography operations, and Secure Non-volatile memory. As an isolated chip designed specifically for security functionality, the Secure Enclave incorporates and enforces security design principles like least common mechanisms and separation of privilege

In summary, Apple Pay implements authorization of credit cards, authenticates users, enables encryption and integrity protection in line with industry standards and affords a high level of identity and privacy protection via dedicated, security-focused hardware.

Google Pay Security

In explaining all of the components required for Google Pay to enable mobile payments at POS devices, the conversation focuses on software implementation and architecture. Google Pay can be implemented on many different mobile devices, and with that comes varying levels of hardware security implementations. This is because Google pay runs on Android, an open-source operating system. We will briefly identify all of the main requirements for a mobile transaction but would note that the implementation can vary based on hardware vendor.

Google Pay is implemented in a Trusted Execution Environment (TEE), a separate and isolated area in the phone's hardware. Phone manufacturers can implement a TEE as a separate processor and memory or a virtualized instance on a shared processor. Input and Output protection allows for separation between the TEE and the Application Processor. This is achieved through software implementation that protects the TEE even if the main phone is compromised and rooted. In implementations that allow for a separate processor, like Google's Titan Chip, a separate isolated OS (Trusty) communicates with the application CPU through kernel drivers. [9]

The main components in Google Pay's architecture is the Android Key-store API and Hardware Abstraction Layer. This is where cryptographic keys, biometric data, digital signatures, and hashing capabilities are created and stored. This can be a software only implementation or a combination of hardware and software based on the phone vendor. Other components that make up Google Pay are the Google Pay App and NFC controller [9].

Google Pay mobile payments begin with a user adding their payment card in the Google Pay app. The card number is not stored on the device but is sent to two locations: the Google servers and the payment card issuer. This information is sent encrypted over TLS. Google states that the credit card numbers are stored in encrypted format within their servers. They state that they retain the right to use and share this and other purchase information as dictated in the Google Payments Privacy Notice [10]. The credit card information is also sent to the card issuer to obtain verification, approval, and a token to replace the card number during transactions. Messages sent to the card issuer for authorization are protected with confidentiality, integrity, and authentication in transit. Messages are sent using a ciphersuite that meets financial industry standards. Elliptic Curve Digital Signature Algorithm and SHA-256 are commonly used to provide authentication and integrity of these messages. For symmetric encryption, the AES-256 algorithm is often used running in CTR mode. Google states that it uses OpenSSL to generate asymmetric key pairs [11].

The tokenization process assigns a new virtual account number to the credit card that changes with every transaction. This works to mitigate replay attacks even if the NFC communication is compromised. The actual credit card number is never shared with a merchant POS device.

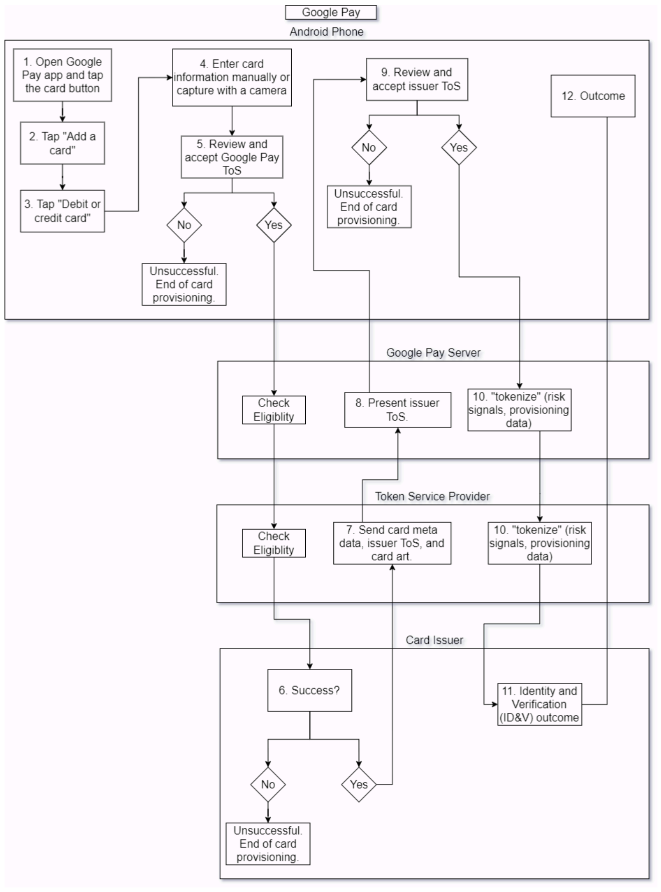
Google Pay
Android Phone
1. Open Google Pay app and tap the card button
2. Tap "Add a card"
3. Tap "Debit or credit card"
4. Enter card information manually or capture with a camera
5. Review and accept Google Pay ToS
No
Yes
Unsuccessful. End of card provisioning.
9. Review and accept issuer ToS
No
Yes
Unsuccessful. End of card provisioning.
12. Outcome
Google Pay Server
Check Eligiblity
8. Present issuer ToS.
10. "tokenize" (risk signals, provisioning data)
Token Service Provider
Check Eligiblity
7. Send card meta data, issuer ToS, and card art.
10. "tokenize" (risk signals, provisioning data)
Card Issuer
6. Success?
11. Identity and Verification (ID&V) outcome
No
Yes
Unsuccessful. End of card provisioning.

Figure 2: Google Pay process diagram.

Once the credit card is added and a token is obtained, Google Pay is ready to use. The first step is authentication where the owner of the device is verified against the encryption keys stored in the TEE. Android specifies the level of security required for facial recognition cameras and fingerprint sensors. To be considered compatible with Android, manufacturers must meet the requirements presented in the Android Compatibility Definition Document (CDD). The manufacturers are required to meet the CDD architectural security and spoofability standards [13].

Comparisons

The mobile payment platforms we researched share many similarities in user experience and security, but with key differences found in payment authorization and data collection. Both Apple Pay and Google Pay interact with the user in the same way, an additional app on the device where users control their payment information. Apple Pay requires users to link an iCloud account, and Google Pay requires a user to register with a Google account. Both services make use of NFC technology to send transactions to POS terminals without sending the actual card number. However, there are major differences in how transaction data is encrypted and how the encryption is implemented.

Apple has the advantage of controlling the hardware as well as software, so it can store sensitive data like device account numbers and keys in its Secure Enclave and generate those keys with its Secure Element. This also allows Apple to forgo the need to store card numbers in their servers. Apple can communicate with the card issuers using their device account numbers on the device. Google attempts to create a similar secure environment through software with its TEE, but inevitably needs to store card numbers on their servers. Google does store the cards in an encrypted format.

In comparing the two payment systems, we are limited in scope up until the POS terminal. We will not describe how the payment network and card issuers interact with each other or how the acquirers communicate with these entities. This is outside of the control of Apple Pay and Google Pay.

Conclusion

While varying approaches were taken to allow for mobile payments at POS systems, there are many similarities. Both implementations allow for secure transactions that meet or exceed current financial industry standards. Regardless of whether Apple's "walled garden" approach or Google's open source approach is better suited for mobile payments, they both share an inherited financial infrastructure that decides the fate of transactions. As we have learned time and again throughout this course, infrastructure is forever. Regardless of how secure each system is made, they will rely heavily on the financial system that they have very little control over. As was seen with recent attacks on Apple Pay and Google Pay, the compromise was actually fraud or scam type attacks by adding stolen credit card information to the mobile wallet apps and not an actual compromise of the platforms [17], [18]. Despite all this, it is the humble opinion of this research paper that mobile payment systems, like Apple Pay and Google Pay, will only continue to grow in popularity and use.

References

[1]. Grant, M. (2021, September 13). *Mobile payments: What you should know*. Investopedia. Retrieved from https://www.investopedia.com/terms/m/mobile-payment.asp

[2]. Tardi, C. (2022, June 28). *Near Field Communication (NFC) definition*. Investopedia. Retrieved from https://www.investopedia.com/terms/n/near-field-communication-nfc.asp

[3]. *Apple Pay Security and Privacy Overview*. Apple Support. (2021, February 2). Retrieved from https://support.apple.com/en-us/HT203027

[4]. Raghavan, K. (2022, June 28). *Apple pay vs. google pay: How they work*. Investopedia. Retrieved from https://www.investopedia.com/articles/personal-finance/010215/apple-pay-vs-google-wallet-how-they-work.asp#:~:text=Apple%20Pay%20and%20Google%20Pay%20are%20largely%20identical%20offerings.,will%20never%20track%20your%20transactions.

[5]. Arntz, P. (2021, October 1). *Apple pay vulnerable to wireless pickpockets*. Malwarebytes Labs. Retrieved from https://blog.malwarebytes.com/exploits-and-vulnerabilities/2021/10/apple-pay-vulnerable-to-wireless-pickpockets/

[6]. Rogers, J. (2022, February 27). *Android warning as Google pay security flaw leaves millions of phones at risk*. The US Sun. Retrieved from https://www.the-sun.com/money/4783370/android-samsung-security-warning-google-pay/

[7]. Spadafora, A. (2020, February 26). *Google pay users could be left out of pocket by PayPal fraud bug*. TechRadar pro. Retrieved from https://www.techradar.com/news/paypal-bug-let-hackers-defraud-google-pay-users

[8]. Apple. (2022, May). *Apple Platform Security*. Apple Support. Retrieved from https://support.apple.com/en-ie/guide/security/welcome/web

[9]. *Trusty Tee : Android Open Source Project*. Android Open Source Project. (n.d.). Retrieved from https://source.android.com/security/trusty

[10]. Google. (2022, March 28). *Google Payments Privacy Notice*. Google Payments. Retrieved from https://payments.google.com/payments/apis-secure/u/0/get_legal_document?ldo=0&ldt=privacynotice&ldl=en-US

[11]. Google. (n.d.). *Payment data cryptography for merchants | google pay API for Android | google developers*. Google Developers. Retrieved from https://developers.google.com/pay/api/android/guides/resources/payment-data-cryptography

[12]. Android. (n.d.). *Biometrics : Android Open Source Project*. Android Open Source Project. Retrieved from https://source.android.com/security/biometric

[13]. Android. (n.d.). *Android Compatibility Definition document : Android Open Source Project*. Android Open Source Project. Retrieved from https://source.android.com/compatibility/cdd

[14]. A. S. Jawale and J. S. Park, "A Security Analysis on Apple Pay," *2016 European Intelligence and Security Informatics Conference (EISIC)*, 2016, pp. 160-163, doi: 10.1109/EISIC.2016.041. https://ieeexplore.ieee.org/abstract/document/7870214/citations#citations

[15]. Kazan, Erol, "The Innovative Capabilities Of Digital Payment Platforms: A Comparative Study Of Apple Pay & Google Wallet" (2015). 2015 International Conference on Mobile Business. 4. https://aisel.aisnet.org/icmb2015/4

[16]. Huh, J. H., Verma, S., Rayala, S. S. V., Bobba, R. B., Beznosov, K., & Kim, H. (2017, February). I Don't Use Apple Pay because it's less secure...: perception of security and usability in mobile tap-and-pay. In *Workshop on Usable Security, San Diego, CA*.

[17]. Gallagher, W. (2022, April 22). *Fraudsters target Apple pay in credit card scams*. AppleInsider.

Retrieved from https://appleinsider.com/articles/22/04/22/fraudsters-target-apple-pay-in-credit-card-scams

[18]. Tode, C. (n.d.). *Apple pay fraud reflects weakness with credit cards, not platform*. Retail Dive. Retrieved from https://www.retaildive.com/ex/mobilecommercedaily/apple-pay-fraud-reflects-weakness-with-credit-cards-not-platform

[19]. Google. (n.d.). *Overview | device tokenization developer site | google developers*. Google Developers. Retrieved from https://developers.google.com/pay/issuers/tsp-integration/overview