Picking Up My Tab: Understanding and Mitigating Synchronized Token Lifting and Spending in Mobile Payment

Xiaolong Bai, *Tsinghua University;* **Zhe Zhou,** *The Chinese University of Hong Kong;* **XiaoFeng Wang,** *Indiana University Bloomington;* **Zhou Li,** *IEEE Member;* **Xianghang Mi and Nan Zhang,** *Indiana University Bloomington;* **Tongxin Li,** *Peking University;* **Shi-Min Hu,** *Tsinghua University;* **Kehuan Zhang,** *The Chinese University of Hong Kong*

https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/bai

This paper is included in the Proceedings of the 26th USENIX Security Symposium

August 16–18, 2017 • Vancouver, BC, Canada

ISBN 978-1-931971-40-9

Open access to the Proceedings of the 26th USENIX Security Symposium is sponsored by USENIX

Picking Up My Tab: Understanding and Mitigating Synchronized Token Lifting and Spending in Mobile Payment

Xiaolong Bai[1,*], Zhe Zhou[2,3,*], XiaoFeng Wang[3], Zhou Li[4],
Xianghang Mi[3], Nan Zhang[3], Tongxin Li[5], Shi-Min Hu[1], Kehuan Zhang[2,†]
[1]*Tsinghua University,* [2]*The Chinese University of Hong Kong,*
[3]*Indiana University Bloomington,* [4]*IEEE Member,* [5]*Peking University*
bxl12@mails.tsinghua.edu.cn, {zz113, khzhang}@ie.cuhk.edu.hk, {xw7, xmi, nz3}@indiana.edu,
lzcarl@gmail.com, litongxin@pku.edu.cn, shimin@tsinghua.edu.cn

Abstract

Mobile *off-line* payment enables purchase over the counter even in the absence of reliable network connections. Popular solutions proposed by leading payment service providers (e.g., Google, Amazon, Samsung, Apple) rely on direct communication between the payer's device and the POS system, through Near-Field Communication (NFC), Magnetic Secure Transaction (MST), audio and QR code. Although pre-cautions have been taken to protect the payment transactions through these channels, their security implications are less understood, particularly in the presence of unique threats to this new e-commerce service.

In the paper, we report a new type of over-the-counter payment frauds on mobile off-line payment, which exploit the designs of existing schemes that apparently fail to consider the adversary capable of actively affecting the payment process. Our attack, called *Synchronized Token Lifting and Spending* (STLS), demonstrates that an *active* attacker can sniff the payment token, halt the ongoing transaction through various means and transmit the token quickly to a colluder to spend it in a different transaction while the token is still valid. Our research shows that such STLS attacks pose a realistic threat to popular off-line payment schemes, particularly those meant to be backwardly compatible, like Samsung Pay and AliPay.

To mitigate the newly discovered threats, we propose a new solution called *POSAUTH*. One fundamental cause of the STLS risk is the nature of the communication channels used by the vulnerable mobile off-line payment schemes, which are easy to sniff and jam, and more importantly, unable to support a secure mutual challenge-response protocols since information can only be transmitted in one-way. POSAUTH addresses this issue by incorporating one unique ID of the current POS terminal into the generation of payment tokens by requiring a quick scanning of QR code printed on the POS terminal. When combined with a short valid period, POSAUTH can ensure that tokens generated for one transaction can only be used in that transaction.

1 Introduction

The pervasiveness of mobile devices has profoundly changed the ways commercial activities are conducted. Particularly, mobile payment, in which a payment transaction is carried out between a smartphone and a point of sale (POS) system, becomes increasingly popular, with over 1 trillion dollars revenue projected for 2019 [49]. Leading e-commerce providers (e.g., PayPal, Amazon, Google, Alibaba) and smartphone manufacturers (e.g., Samsung, Apple) all come up with their own solutions and competing with each other for market shares. Most of these schemes are designed for *online* use originally, which requires both the payer and the payee to stay connected to the Internet during a transaction, so both parties are notified by the payment service provider once the transaction succeeds. A problem for this approach is that the payer (who in many cases is a grocery shopper) is expected to have a decent network connection (or enough mobile data) whenever she pays. To avoid the delay and extra cost introduced during this process, recently *off-line* payment schemes are gaining traction, which allow a transaction to go through even when the payer's network connection is less reliable. This is achieved by establishing a direct connection between the smartphone and the POS system through Near-Field Communication (NFC), Bluetooth, electromagnetic field, 2D-QR code or even audio signal, and delivering a payment token over this channel. Already many prominent payment schemes have offered this off-line support, including PayPal, Google Wallet, Apple Pay, Samsung Pay and AliPay. What is less clear, however, is the security guarantee they can provide.

Security of mobile off-line payment. Unlike the online payment, in which the payer and the payee do the

*The two lead authors are ordered alphabetically.
†Corresponding author.

transaction through the service provider, the off-line approach relies on the direct communication between the smartphone and the POS system, and therefore can be vulnerable to the eavesdropping attack from a bystander. This is less of an issue for the NFC channel, due to its extremely short communication distance, making the sniffing difficult. More importantly, both NFC and Bluetooth allow convenient bidirectional interactions between the payer (smartphone) and the payee (POS), which helps strengthen the protection: a typical approach is for the POS device to challenge the phone with an "unpredictable number"; the number is then used by the phone to generate a payment token with a short validity period. This thwarts the attempt to use the token in a different transaction.

In practice, however, the POS systems armed with NFC or Bluetooth are expensive and less deployed, and cheaper and more backwardly compatible alternatives are widely adopted. For example, Samsung Pay supports *Magnetic Secure Transmission* (MST), which can work on those using magnetic stripes, like credit-card readers. PayPal and AliPay (an extremely popular Chinese payment service, with 190 million users) both transmit the token through QR scanning, an approach widely supported by POS machines. Also, AliPay and ToneTag [51] utilize audio signals, a low-cost solution that needs only a sound recorder on the payee side. A problem here is that all such channels (electromagnetic field, QR code and audio) are one-way in nature, making the above challenge-response approach hard to implement. To address this issue, these schemes employ *one-time payment token*, together with a short valid time, to defend against the eavesdropping attack. The idea is that once the token is observed, it cannot be used again and will soon expire, and is therefore useless to the adversary. The effectiveness of this protection, however, becomes questionable in the presence of an *active* attacker, who is capable of disrupting a transaction to prevent the token from being spent, which allows him to use it in a different transaction within the validity period. This was found to be completely realistic in our study.

Our attacks. In this paper, we report our security analysis of two leading mobile off-line payment schemes: Samsung Pay and AliPay. Our study reveals surprising security vulnerabilities within these high-profile schemes, affecting hundreds of millions of users around the world: we found that both approaches are subject to a new type of over-the-counter payment frauds, called *Synchronized Token Lifting and Spending* (STLS), in which an adversary sniffs the payment token, manages to halt the ongoing transaction and transmits the token to a colluder to spend it in a different transaction while the token is still valid. Oftentimes, such an attack can also seamlessly trigger a retry from the payer to let the original transaction go through without arousing any suspicion. More specifically, in Samsung Pay, we show that the attacker can pick up the magnetic signals 3 meters away using a sensor, and then automatically *jam* the wireless signals produced by a mobile POS (mPOS) using a jammer (a commercial device), causing a disruption between its communication with the payment provider. As a result, the payment token is prevented from being delivered to the provider and instead, recovered, demodulated and then spent in a different transaction (at a different location). After this is done, the attacker stops the jamming, which enables a retry from the shopper to complete the transaction. We found that this attack can be realistically implemented, as demonstrated in a video we posted online [1] (Section 3.1). A similar attack also succeeds on Alipay, over the audio channel: we utilized a recorder to capture the token transferred through sound and again the jammer to disrupt the payment transaction, before using the token for another transaction (Section 3.2).

Alipay also supports payment through QR code scan. In our research, we studied two payment scenarios: peer-to-peer transfer and pay through POS. In the first case, the payer uses his phone to scan the payee's QR code displayed by her phone. Our study shows that a malicious payer device or the one infected with an attack app can not only steal the token from the payee's screen, which can later be used for over-the-counter payment (through POS), but also stealthily force the payee device to refresh its screen, causing it to generate a new token, therefore preserving the original one for an unauthorized purchase (Section 3). When it comes to POS-based payment, we discovered that a malicious app running on the payer's device can stealthily halt the transaction by strategically covering a few pixels on the screen when it is displaying the QR token to the POS machine. In the meantime, we found that it is feasible to acquire the image of the code from the reflection on the glass of the QR scanner's scan window (captured by the phone's front camera) (Section 3.3). Again, the demos of these attacks are available online [1].

Mitigation. Our findings highlight the fundamental weaknesses of today's mobile off-line payment schemes: one-time token is insufficient for defending against an active adversary capable of stealthily disrupting a payment transaction (which is found to be completely realistic); also given the error-prone nature of the channels (magnetic field, sound, QR scan) those schemes use, the validity period of their payment tokens needs to be sufficiently long to allow multiple retries, which leaves the door open for the STLS attack. To mitigate the newly discovered threats and enhance the security protection of the off-line payment, we designed and implemented a new solution, called *POSAUTH*, which authorizes the payer to use a payment token only at a specific transaction. POSAUTH is meant to be easily deployed, without changing hard-

ware of today's POS systems. More specifically, each POS terminal presents a QR code carrying its unique ID. For each transaction, the payer is supposed to scan the code to generate the payment token, which is bound to the terminal. This binding, together with the valid period, ensures that the token can only be used in the current transaction (see Figure 1).

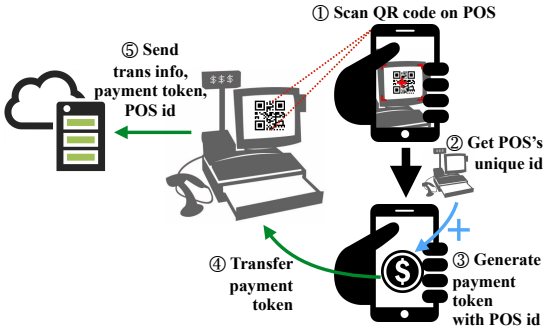

Figure 1: The work-flow of POSAUTH

Contributions. The contributions of the paper are outlined as follows:

• *New findings and understandings*. We report the first study on the STLS threat to mobile off-line payment. Our research brings to light surprising security vulnerabilities within high-profile payment solutions, which subject these schemes to the new type of payment frauds. Such STLS attacks are found to be completely realistic, with serious consequences, leading to unauthorized spending of the payer's token. The findings demonstrate the challenges in protecting these off-line payment schemes in the presence of an *active* adversary.

• *New protection*. We made a first step towards practically mitigating these STLS attacks through a new design that binds the payer's payment token to a specific POS terminal, without changing the hardware of existing systems. We implemented this design, which is found to be effective and efficient in defending against the threat.

Roadmap. The rest of the paper is organized as follows: Section 2 provides background information for our study; Section 3 elaborates the STLS threats on Samsung pay and Alipay; Section 4 presents our protection mechanism; Section 5 discusses the limitations of our study and potential future research; Section 6 compares our work with related prior studies and Section 7 concludes the paper.

2 Background

In this section, we describe how mobile payment works, the current protection in place and potential security risks. Further we present the assumptions made in our research.

Mobile off-line payment. Since 1999, when Ericsson and Telnor Mobil phones were first used to purchase movie tickets, mobile payment has gained considerable popularity in the past decades, and is expected to be used by 90 percent of smartphone users in 2020[42]. Today, a typical payment transaction through mobile devices includes three parties: the payer, the payee and the payment service provider. Depending on the role played by the provider, a payment transaction can be *online* or *off-line*. Figure 2 illustrates the work-flows of both payment methods. A prominent example of mobile online payment is *Mobile wallet*, a scheme provided by PayPal, Amazon and Google. A payment process through Mobile wallet involves registration of a user's phone number and acquisition of a PIN for authentication. In a transaction, the user enters the PIN to validate the payment that will be charged to her account based upon the credit card or other information (stored in her mobile device) given to the service provider during the transaction.

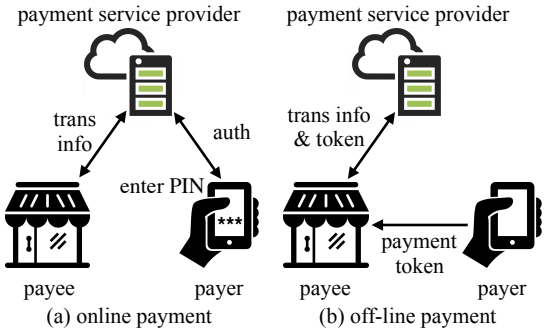

Figure 2: The work-flows of online and off-line mobile payment methods

By comparison, a mobile off-line payment happens directly between the payer and the payee, with the provider communicating with only one of these two parties in the transaction. Oftentimes, the payer is a grocery shopper, with a smartphone carrying a shared secret with the provider and the payee controls a POS device for communicating with the provider. An off-line transaction starts when the payee creates a charge request through entering payment information (e.g., amount, payment method) into a POS terminal. Then, the payer is supposed to run her payment app to establish a communication channel with the POS for transmitting a token. Some of these channels are described in Table 1. Such a token is generated using a secret in the payer's mobile digital wallet, the current time and the challenge from the payee when it is available, and other credential data.

Upon receiving the token, the POS terminal forwards the token as well as other transaction information to the payment service provider for verification. From the token, the provider first recovers the owner information and then verifies its authenticity (whether it is issued by the owner) and liveness (whether it has been used before and whether

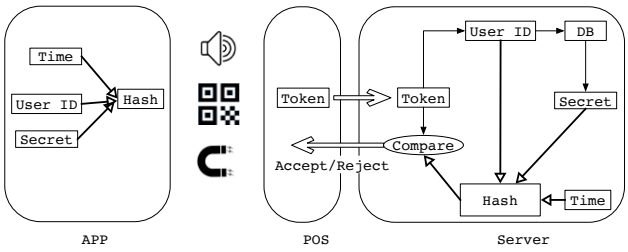

Figure 3: Mobile off-line payment transaction flow.

Channel	Provider Examples
NFC	Apple, Google
Bluetooth	Bridg[5]
MST	Samsung Pay
Audio	Alipay, ToneTag
QR code	Paypal, Alipay, Wechat

Table 1: Example of off-line payment channels and the payment service providers using these channels

it is issued recently, within the validity period) of the token. If so, the provider continues to check the balance of the owner's account to determine whether the transaction can proceed. A transaction approval or denial is then issued to the POS terminal, depending on whether all these checks are passed. The process is shown in Figure 3.

Payment security. The security guarantee of an off-line payment scheme is mainly built upon the protection of the payment token, which is essentially the proof for a payment request, typically in the form of a hash-based message authentication code (HMAC) over its generation time and other information. The token is delivered to the provider by the payee through a secure channel. Less protected here is the direct link between the payer and the payee, which could be monitored by the adversary present at the scene of the payment. For the transaction going through NFC and Bluetooth (see Table 1), a random number generated by the payee can serve to challenge the payer and ensure that the token is bound to a specific POS terminal. For other channels, however, existing payment schemes do not use this challenge-response approach (due to the complexity and unreliability of the channels) and instead, rely on one-time token: a token, once received by the provider, is recorded to make sure that it will not be used again. Also, each token is ephemeral, with a short valid period attached to it, based upon its generation time specified in its content.

This protection apparently only considers the threat from a passive adversary, who does nothing to interfere with the execution of a transaction. The situation can be very different for an active one. In the case that the transaction can actually be disrupted, which stops the delivery of the token to the provider, the observed one-time token can then be stolen and used for a different transaction. Also this attack cannot be prevented by checking the liveness of the token, since the validity period often has to be set sufficiently long to tolerate the errors in a normal token transmission. As an example, a payer needs 5 -10 seconds to place her phone before the QR code can be reliably recognized. As a result, often a payment token has more than one minute of living time, which as shown in our study (Section 3), is often long enough for successfully spending it on a different transaction, with the help of a colluder in the attack.

Adversary model. In our study on the payment through electromagnetic field (Samsung Pay) and audio signals, we consider an adversary who is either physically present at the payment scene or capable of placing her attack devices (including sniffer and jammer) there. This is completely realistic, given the small sizes of the devices, as illustrated in Figure 7(a). In QR-code based payment, we no longer require the presence of attack devices. Instead, we assume that the payer's phone is infected with an attack app, which *does not* have system privileges but needs camera, Bluetooth and network permissions, which are commonly requested by legitimate apps.

3 STLS Attacks

In this section, we report our security analysis on Samsung Pay as well as the Audio Pay and QR Pay techniques utilized by other popular mobile off-line payment services such as Alipay and Wechat. Our study shows that they are all subject to the STLS attacks: an adversary can realistically disrupt payment transactions, steal payment tokens and spend them without proper authorization. This security hazard affects hundreds of millions of mobile users worldwide. We contacted all affected service providers and some of them have already acknowledged the importance of the findings. We are now helping them fix the discovered problems and enhance their protection. Following we elaborate this study.

3.1 Samsung Pay

Samsung Pay is a popular token based mobile payment service available on the smartphones manufactured by Samsung Electronics. It is characterized by a unique POS-device communication technique, when compared with other payment services like Apple Pay and Android Pay, called *Magnetic Secure Transmission (MST)*, which has been acquired from LoopPay in 2015 [44]. In this paper we focus on the security protection of MST, even though Samsung Pay also supports NFC.

Samsung Pay features a high compatibility to existing POS terminals which work with magnetic-stripe card. Merchants need no modification to their out-dated POS

terminal to support this kind of innovative mobile phone payment. A Samsung phone stores a piece of secret key inside KNOX, a secure container. When the Samsung Pay user (the payer) is going to pay at a POS, she launches the app and chooses a card she is going to pay with and then passes the app's verification with either password or fingerprint. Then the app (inside KNOX) immediately generates a token for the user by HMAC a piece of message containing the transaction counter, the primary account number (PAN) [1] using the secret key, assembles all information in the same format as the magnetic tracks on conventional credit cards, and starts to broadcast that information over the MST channel by modulating electric current passing through a MST antenna. Any POS terminal, if magnetic card is supported, will receive the token through its magnetic head and then process it in the exact same way as if the user is swiping a magnetic-stripe credit card. The track data with token and other information encapsulated will be passed to the service provider via POS's network for further transaction processes including token verification, translating to real PAN, balance verification, and the transaction result will be returned to the POS terminal to notify the payee if the transaction is approved or not.

Understanding MST. MST is a patented technique (US8814046 [20]) that first appears in LoopPay Fob and LoopPay CardCase and it is compatible with any existing POS terminal.

The security protection of MST pretty much depends on the property of electromagnetic field, which is considered to be a near-field communication channel. Specifically, the strength of electromagnet signal quickly attenuates as the distance to the source r grows, at the rate of $1/r^3$. On LoopPay's home page, it is claimed: "LoopPay works within a 3-inch distance from the read head. The field dissipates rapidly beyond that point, and only exists during a transmission initiated by the user" [33]. A similar claim is also made by Samsung Pay: "Due to the short-range nature of MST, it is difficult to capture the payment signal" [43].

Eavesdropping MST signal. However, we found in our research that this distance based protection does not work as stated by those claims, which has also been reported by other research [6, 3]. Fundamentally, the distance that allows electromagnetic field signal sniffing feasible is determined by a signal-noise-ratio (SNR) at that distance and the capability for the sniffing antenna to pick up the signal. Our study shows that instead of 3 inches (< 0.08 meters) as claimed by the MST document, a small loop antenna at the size of a small bag (as illustrated in Figure 4) can effectively collect the signal at least 2 meters away from the source. More importantly, the signal captured at this distance still carry enough information for decoding, in a realistic noise environment. For example, Figure 5 a) and b) compare the signal received by our loop antenna (2 meters away from the source) with the theoretically received ones, as discovered in a real-world grocery store. As we can see here, the signal still largely preserved the coding information and can therefore be used for decoding using our later proposed decoding method.

[1] a virtualized one instead of original credit card number.

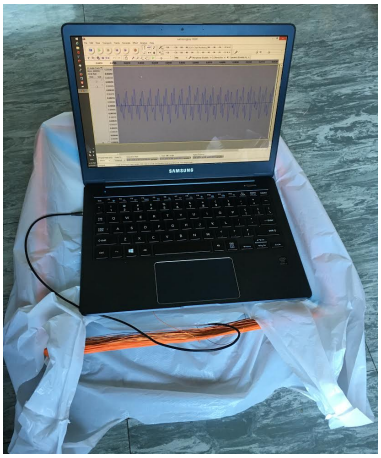

Figure 4: Sniffing devices.

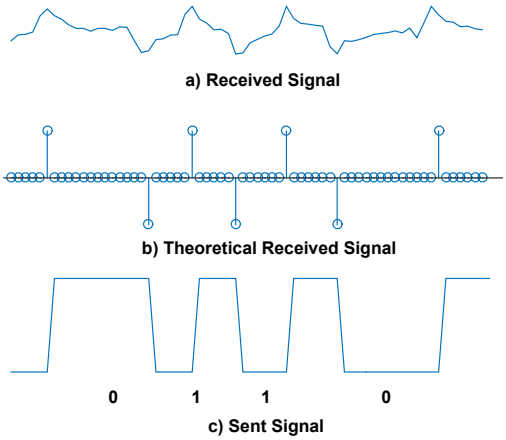

Figure 5: Comparison between original signal and our received one in 2 meters.

Signal decoding. In our research, we decode such signal according to impulse polarity changes. Specifically, MST uses differential Manchester encoding, in which the polarity flips once for the symbol '0' while twice for the symbol '1' (Figure 5 c)). Although our antenna cannot directly sense the magnetic field, it is able to capture the polarity flips, because the current generated by the antenna is the derivative of the magnetic filed (a flip's derivative is an impulse, as compared in the Figure 5 b) and c)).

The captured signal is then decoded using a band pass filter (BPF), a synchronization detection module and a symbol judgment module. BPF allows only frequency

components from 0.3 kHz to 10 kHz to pass, which effectively reduces the out-of-band noise. The synchronization detection module identifies the start and the end positions of each symbol. It sequentially enumerates all the sample points and determines whether a given sample point is an apex and whether it exceeds a threshold: if so, the point is chosen as the start point of the first symbol. Then the module chooses an apex with maximum strength around its theoretical end position (based upon the symbol duration) as the end point of the first symbol (also the start point of the second symbol). The process repeats until the apex's strength is under the threshold, which indicates that the valid signal ends. In this way, all the symbols' start and end positions are determined. The symbol judgment module decides whether a given symbol represents '0' or '1' by comparing the polarities of the start and end point. If the start point and end point have the same polarity, the symbol represents '1', otherwise, it is '0'.

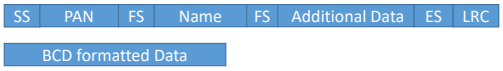

Figure 6: The track format Samsung Pay uses.

The symbols generated by the symbol judgment can be easily translated to a text string in accordance with the ANSI/ISO ALPHA data format (designed for magnetic card track 1) or the ANSI/ISO BCD data format (for track 2 and 3) [2], as shown in Figure 6.

(a) (b) (c)

Figure 7: A commercial jammer and a mPOS.

mPOS jamming. As mentioned earlier, Samsung Pay and LoopPay utilize one-time token, which effectively defends against passive attacks: up to our knowledge, none of the prior exploit attempts [6, 3] can succeed, because a used token cannot be used again. A fundamental issue here, however, is that the protection does not work against an *active* adversary and interfering with an ongoing transaction is much more realistic than one thought, as discovered in our research. Specifically, we found that mobile POS systems, as shown in Figure 7(b) and Figure 7(c), with over 3.2 million already installations and a over 27 million installations in 2021 by expectation [21], can be easily jammed using a portable commercial device. For example, the device in Figure 7(a) can easily block either WiFi or cellular signal or both at a distance of 3

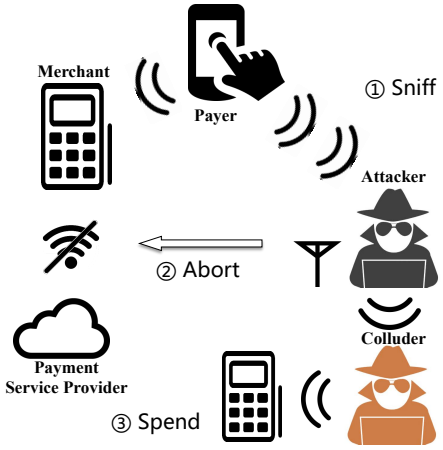

Figure 8: Attack flow for Samsung Pay.

meters, which causes all mPOS transaction to abort. Such a jammer simply broadcasts white noise over the same frequencies as those used by the targeted channels to interfere with legitimate communications. It can be easily switched on and off to target a specific payment step. Note that such jamming does not need to be blind: most mPOS systems are using WiFi, and a temporary disruption of its service, within a few meters, will not affect other mobile users, such as those using smartphones through 3G or 4G; even for the mPOS running on cellular networks, the adversary can jam only their specific cellular formats, e.g., Verizon (CDMA format), without interfering with others, e.g., AT&T users (UMTS format), in a 3-meter peripheral. Further, given the delay for the POS system to restore its connection, the adversary can quickly stop jamming: for example, he can turn on the device for 30 seconds and then leave, and gives his colluder, who receives the token from an unblocked channel, at least 1 minute to spend the token.

The attack. Putting pieces together, the flow for the whole STLS attack on Samsung Pay is illustrated in Figure 8. The attacker runs a small antenna (small enough to be hidden in his backpack) connected to a laptop (also hidden) to monitor the electromagnetic signal around an mPOS terminal. Once a customer opens her digital wallet (on her smartphone) for a payment transaction, the antenna captures the token and in the meantime, the jammer is switched on (which does not affect the communication between the wallet and the mPOS) to block the mPOS's network. The acquired token is then automatically decoded and forwarded through an unjammed channel to a colluder (who might run an app to alert him to the arrival of a token that needs to be spent within a time frame[2]). Such a token is automatically written to a magnetic stripe card, which can be used just like a normal credit card, or

[2]Actually we found that Samsung Pay has a one day time limit for its token [25].

to a MagSpoof device (e.g., for a purchase at an automatic vending machine) to replay the token. The adversary can stop the jamming and walk away after a short period of time, which allows the mPOS to restore its network connection and receive a payment error from the server (which often comes without details). As a result, the payer has to retry with an automatically generated new token to complete the transaction. We are communicating with Samsung to help them address this threat.

3.2 Audio Pay

Similar to MST-based payment, the schemes based upon the audio channel are equally vulnerable to the STLS attack. Following we elaborate the attack on these schemes.

Analysis of audio pay. Audio is an emerging channel for near-field inter-device communication. Compared to other channels like Bluetooth, Wi-Fi Direct, or NFC, the audio channel is cheap and easy to use, given the fact that every phone is equipped with a microphone and a speaker. The main weakness of this channel is its bandwidth because of its narrow frequency spectrum. Nevertheless, it remains an efficient and convenient way to exchange a small amount of information. In particular, it has been used by multiple payment schemes, including Alipay and ToneTag, to transmit a payment token (from the payer's device to the payee).

Specifically, the payer is supposed to encode her payment information into an audio clip using a modulation scheme like audio frequency-shift keying (AFSK). During the payment transaction, she can play the clip to the merchant's POS device. Upon receiving the audio, the merchant decodes it to recover the payment token from, and then sends the token as well as transaction information to the payment service provider. The provider verifies the payment token and replies with an acknowledgement response if successful. This payment process is illustrated in Figure 9.

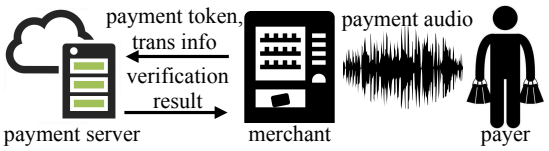

Figure 9: The process of audio pay.

Alipay has widely adopted audio pay on mobile vending machines. In this scenario, after the payer selects an item, the vending machine will ask the payer to play her payment audio. Upon receiving the audio, the machine decodes the payment token from the audio and sends it to the payment service provider through cellular network to proceed the transaction. To produce the payment audio, Alipay encodes the payment token into a carrier sound by AFSK. While the carrier sound can be heard by a human being, the payment token is encoded at the frequency of 17.2kHz - 18.4kHz, which is beyond the absolute threshold of human hearing. But such a modulation scheme also enables the sniffing attack since there is nearly no noise at this frequency range, and the token can be broadcast with low lost. Here, we elaborate our attack to audio pay as below.

The attack. Again, this payment scheme is vulnerable to an STLS attack involving audio recording and WiFi or cellular signal jamming with the device shown in Figure 7(a). Specifically, before the payment transaction starts, a nearby attacker turns on a signal jamming device to block signals and prevents the merchant from communicating with the payment service provider. When the payer plays her payment sound to the merchant (mobile vending machine in Alipay), the attacker records the sound within a proper distance. Since the transaction is aborted by the signal jamming, the recorded payment token is not spent. Then the attacker can replay the recorded sound to make another purchase. The attack is illustrated in Figure 10.

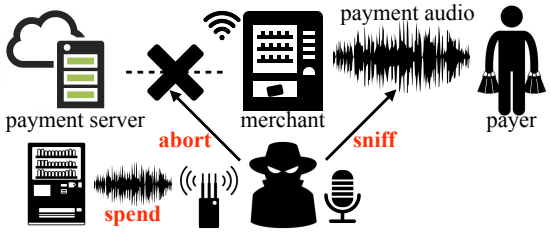

Figure 10: The attack against audio pay.

We implemented this attack against a real-world vending machine. The attack demo is posted online [1]. In this attack, the attacker uses a free iOS app called SpectrumView[35] to record the payment audio signal at a distance of 30cm from the payer's phone. [3] With such low cost, the attacker is still able to successfully launch the STLS attack. Although the the token has a limited valid period (90 seconds), our attack demonstration shows that such time window is sufficient for attackers.

3.3 QR Pay

Mobile payment through QR code is quickly gaining popularity in recent years. A plenty of retailers (like Walmart and Starbucks), financial organizations (like Chase, Paypal and Alibaba) and social network apps (like WeChat) have developed or adopted QR payment. So far, three payment schemes have been proposed to support different payment scenarios [13]:

[3] The attack device is small and can be placed stealthily and closely to merchant device, e.g., within 30cm to a vending machine. Token recording and transmitting can be fully automated without attacker's attendance. Hence, the threat is realistic.

•*Buyer-to-Large retailer transactions (B2L)*. A QR code representing the payment token is generated by the payer's mobile payment app (like WeChat, ChasePay and AliPay) and then picked up by payee's POS scanner and transferred to service provider (see Figure 11). Since a special POS scanner has to be deployed by the payee, this scheme is usually seen in large retailers, like Walmart, Target and Starbucks.

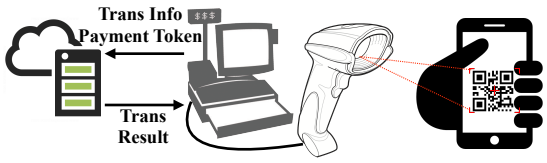

Figure 11: The work-flow of QR pay.

•*Buyer-to-Small business transactions (B2S)*. The payer scans the QR code presented by the payee using a mobile app to get payee's merchant ID, inputs the right amount and then authorizes the payment. In this case, the QR code can be printed on a paper as the merchant ID is usually permanent.

•*Peer-to-Peer transactions (P2P)*. A user (or payer) with payment app wants to transfer money to another user (or payee). The payee generates a QR code to be scanned by the payer. After the scanning process, the money is transferred [4].

In this work, we evaluate whether STLS attack can succeed for the first and third transaction scenario, i.e., whether the payment token can be stolen at one place and spared at another place. We skipped the second scenario, as no payment token is generated by the user (payment is sent directly to the service provider). We focus on the off-line mode, for which the token is generated off-line and no confirmation by the payer is required when the token is about to be consumed. We discuss the online payment option in Section 5.

Security analysis of payment app. Different from the native payment apps, like Samsung Pay and Apple Pay, which protect the payment token through hardware means (e.g., Secure Enclave for Apple Pay and KNOX for Samsung Pay), the third-party payment apps, like WeChat and Alipay, cannot shield the payment token against the OS-level attack (e.g., malicious app with root privilege). Though the OS-level attacks can cause devastating consequences, their victim base is usually small.

As a result, the defense employed by the third-party apps is largely targeting malicious apps with non-root permissions. For instance, AliPay claims that it can prevent another app from taking screenshot to steal the QR code [5]; its payment token is one-time and short-lived; it is capable of detecting mobile trojan app and phishing QR code.

Challenges for STLS attack on QR code. Unlike the MST and audio channels, QR code is a visual sign, which cannot be jammed and sniffed by the nearby physical device. Carrying out STLS attack under this scenario seems impossible at first sight, but through a set of novel techniques, we show such attack is completely realistic. Our only assumption is that a malicious app has been installed on the payer's mobile device with camera, bluetooth and network permissions granted. This app plays a similar role as the nearby physical device used in MST and audio attack. More importantly, our app is a **non-root app** and **does not trigger any abnormal behavior** vetted by the payment app (e.g., taking screenshots of QR code). The key stage of this attack is stealing the payment token while halting the ongoing transaction, and we elaborate two approaches for this step, one attacking the QR scanner of the POS machine and another attacking the payment app during P2P transactions.

3.3.1 Attack POS-based Payment

Attack overview. Our attack can be launched when a user shows the payment QR code to the POS scanner. In particular, the malicious app attempts to steal the QR code from the glass reflection of the POS scanner when the payer's screen is close to it. In the meantime, it disrupts the display of the original QR code to abort the ongoing transaction, by masking the payer's screen. The token (stored in photos captured by the front camera of payer's phone) is exfiltrated to attacker through network and spent at another store after unmasking, like MST and audio attack. To avoid arousing payer's suspicion, the attack app actively profiles the context (e.g., the foreground app and activity) and the actual attack is initiated when the context matches the payment context. We elaborate the four key attack components (sniff, abort, profile and exfiltration) below. A demo of this attack can be found in [1].

Sniffing QR code. For the attack app, direct capture of QR code is not feasible as screen scraping is prevented by the payment app. However, the reflection of the QR code on other objects cannot be controlled by the payment app and we exploit this observation to build this attack component. It turns out the glass window of the POS scanner can serve our purpose perfectly. As illustrated in Figure 12, a typical scanner is composed of a glass window, a camera and a light source. When the screen of payer's phone is close enough to the scanner, the reflection

[4] Some payment app reverses the scheme (payer shows QR code to payee). Our attack is valid for both case.

[5] For example, an Android app can set a window flag `FLAG_SECURE` when initiating an activity window to avoid screen scraping [15]. AliPay uses this flag to protect its QR code.

of the QR code will appear on the scanner glass and the attack app can capture it by taking photos with phone's front camera. The scanner glass is an ideal object here because of its brightness is significantly different from payer's screen: mobile payment apps always increase screen brightness to ease the recognition of the QR code while the light source of the scanner is much darker to avoid glare. As such, a "one-way mirror" is constructed for attack app to pick up the QR code.

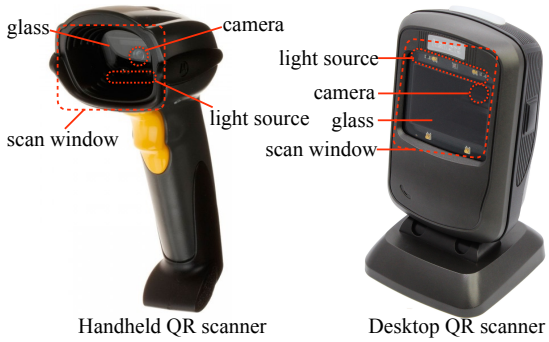

Figure 12: A QR Scanner.

•*Assessment of sniffing.* Whether QR code can be successfully captured by the phone's front camera depends on several factors, including the horizontal distance between the QR code and the front camera (d_{cq}), the side length of the QR code (l_q), the vertical distance between the glass and the phone screen (d_{gs}), and the front camera's angle of view (AOV). Figure 13 illustrates these factors for the most common case that the scanner and the phone are parallel. The ideal approach to assess these factors is to run experiments by simulating all their possible combinations and check whether the QR code can be recovered, which cannot be done within reasonable time. Instead, we compute their theoretical value range for the successful attack.

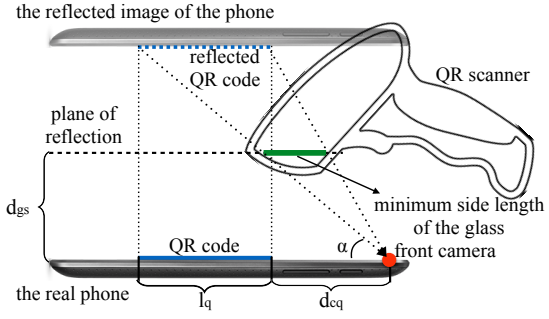

Figure 13: The side view of the phone and the QR scanner during POS-based payment.

In summary, three conditions have to be satisfied. First, the scanner glass should be large enough to reflect the whole QR code. Hence, the minimum side length of the glass should be $d_{cq}/2+l_q-(d_{cq}+l_q)/2$, or $l_q/2$. Second, the horizontal position of the glass should be about the middle point between the QR code and the front camera. Third, the glass should be vertically far enough so it would be within front camera's AOV. In other words, d_{gs} should be at least $(tan(90-AOV/2)\times(l_q+d_{cq}))/2$.

It turns out all the three conditions can be met for the normal payment scenario. For the tested mobile phone (MI 3W), l_q is 3.2cm and d_{cq} is 5.8cm. For Alipay on the tested scanners (Symbol DS-6708SR[50] and NLS-FR40[10]), their glass side lengths are 2.3cm and 4.3cm respectively, which are much larger than the minimum requirement $l_q/2$ (1.6cm). Placing the scanner glass in the middle point is also natural for the payee (as illustrated in Figure 15). Assume AOV for the front camera is 60 degrees, the minimum d_{gs} would be 7.8cm, which is within the suggested working range of the scanner [50, 10].

Aborting ongoing transaction. Signal jamming cannot be used here to disrupt the normal payment process. Instead, we instruct the malicious app to mask the QR code for the disruption.

A QR code has to embed three *positioning markings* (or PM) at its three corners. They are used to ensure that a reader can correctly identify the region and direction of the QR code. If one of these PMs is not displayed, the QR code will not be readable. Our attack app is developed to mask one PM. To this end, the app pops up a floating window covering one PM only. However, showing floating window on top of another app requires a system permission `SYSTEM_ALERT_WINDOW` since Android 6.0 [17]. So, we choose a different approach by commanding the attack app to create an activity which is transparent except the PM region (filled with white pixels) and overlay it on top of the payment app. Such design yields the similar visual effect without asking for any additional permission. When the reflected QR code is captured, the attack app will dismiss the transparent activity and bring back the original QR code window. Figure 14 illustrates an example of the original and masked QR. Figure 15 shows how the masked QR code is scanned and the reflection image of the masked QR code captured by the malicious app with front camera.

Inferring payment activity. To keep the sniffing and jamming process stealthy, our attack app actively infers the running context and moves to the next stage when the context is matched. We exploit a set of side-channel information to learn the context, including the foreground app, its displayed activity and payer's action. The details are described below:

•*Foreground app.* The attack app needs to know when the foreground app is identical to the targeted payment app. The information is not directly available due to the separation between apps. However, it can be inferred by

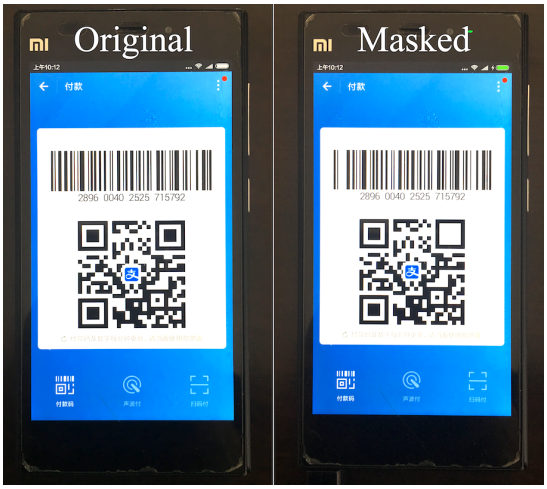

Figure 14: An example of the original and masked QR.

Figure 15: The masked QR code is being scanned and the reflection image of the masked QR code is captured by the malicious app with front camera.

reading a public Linux procfile `/proc/net/tcp6` which logs the opened TCP sockets per app [6]. As long as the targeted payment app is activated , TCP sockets to the servers will be established and the IP addresses expose the app's running status.

•*Displayed activity*. We use the brightness of the screen to determine if the payment app is displaying QR code. Our key observation is that mobile payment apps always increase the screen brightness (say B_{QR}) to the maximum level when showing the QR code. B_{QR} is even higher than the maximum brightness that can be configured by the user. As such, we create a `FileObserver` [14] to monitor the file `/sys/class/leds/lcd-backlight/brightness`. If the brightness logged within the file reaches B_{QR}, the targeted activity is recognized.

[6]We did not use `/proc/net/tcp` since socket information is no longer displayed. `/proc/net/tcp6` shows both IP v6 sockets and IP v4 sockets. The IP v4 addresses are mapped to the IP v6 space in the file.

•*Payer's action*. We use onboard sensors, including accelerometer, gyroscope, and magnetometer, to infer whether the payer is showing her QR code to the merchant. Usually, such action incurs a drastic change of rotation angle of the phone, therefore we can measure the change rate to infer it. In particular, we employ the algorithm in [22] to compute the angles on all axes compared to a fixed position. We begin to monitor the angles when prior conditions are satisfied and record the first observed angle as $A_{initial}$. When the difference between the current angle and $A_{initial}$ exceeds a threshold, the user is recognized as rotating the phone to show the QR code to the merchant.

Exfiltrating QR code. In order to increase the success rate, a series of photos of the reflected QR code are taken during the scanning process. Recovering the QR code from the distorted images on the cell phone is time- and battery-consuming. Therefore, the images are exfiltrated to attackers' server through cellular or WiFi network for further analysis, i.e., mask removal.

Evaluation. We tested our attacks on Alipay (version 10.0.2) and Wechat (version 6.5.4). The testing phone is MI 3W with Android 4.4. The testing QR scanners are Symbol DS-6708SR[50] (hand-held) and NLS-FR40[10] (desktop). And we are able to carry out STLS attack as shown in the demo [1]. We further examined the success rate of our attack by asking 14 users to show the masked QR code (the QR code is masked for 60s) to a desktop QR scanner (NLS-FR40). Among them, 6 (43%) are successfully attacked, suggesting retrieving payment token from the reflected QR code on the scanner's glass window is completely feasible. Some attempts are failed when the user positioned the phone too close to the scanner. The average time of the whole attack is 55s. In particular, the average time to sniff a valid QR code is 16s, and the average time to exfiltrate QR code is 39s. The time to transfer photos to remote server is negligible. Actually, the validity period of a QR code is configured to 90s, based our examination on Alipay and Wechat. This time is sufficient for the attacker to launch the attack and spend the QR code in a different transaction.

Since the attack app does not require any system permissions or any unique system features, the attack is applicable to all Android versions. We are working to implement this attack on other platforms, e.g., iOS. But several issues have to be addressed a priori, e.g., how to mask one PM of the QR code and how to infer the foreground activity, which might need new design of the attack app.

3.3.2 Attack Peer-to-peer Transfer

Attack overview. A user may be attacked even if her device is not infected with any malware. In addition to being used in B2L transaction, a payment QR code

can also be used in P2P transaction, in which the payee presents her QR code to the payer. In this scenario, if there is a malicious app installed on the payer's phone and taking pictures during a P2P transaction, the payee's QR code can be directly harvested. Then the attacker can spend the sniffed QR code in the B2L transaction in another place.

In particular, the malicious app on payer's phone brings itself to the foreground and takes picture when it discovers that the payment app on the same phone is in the QR code scanning mode. The original P2P transaction is disrupted by the malicious app by initiating a bluetooth paring process. The QR code is decoded in the payer's phone and transfered to the remote attacker to be spared (different from the prior attack, the QR code of the payee is not masked and therefore can be directly decoded on the phone). We elaborate the steps for activity inference and transaction disruption below (the other steps are straightforward or similar to the prior attack). Figure 16 and Figure 17 illustrate the normal process for P2P transaction and the attack scenario.

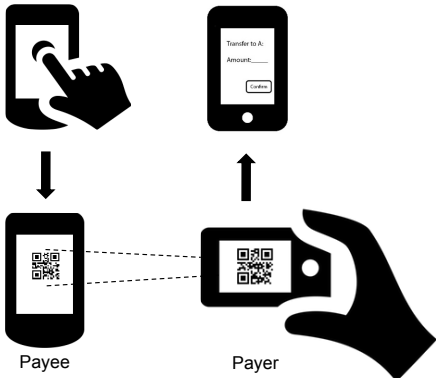

Figure 16: Work flow of P2P transfer.

Inferring payment activity. The attack app on payer's phone needs to learn whether the payment app is on top and in scanning mode. We use the same methodology to infer the foreground app. To detect the scanning mode, our app frequently pings the status of the back camera by invoking a system API `camera.open()` at every 100 milliseconds. If the API returns an error code, the back camera is highly likely occupied by the payment app and the scanning mode is identified (only this mode uses camera).

Interfering P2P transaction. Once the scanning mode is inferred, the attack app will bring an activity (with identical UI to the payment app) to the foreground by sending an intent. Different from the prior attack in which we have no control over the POS scanner, we can block payer's app scanner through intent injection. The attack app keeps scanning QR code until it is successfully decoded. Finally, the malicious app destroys its scan activity to restore in-

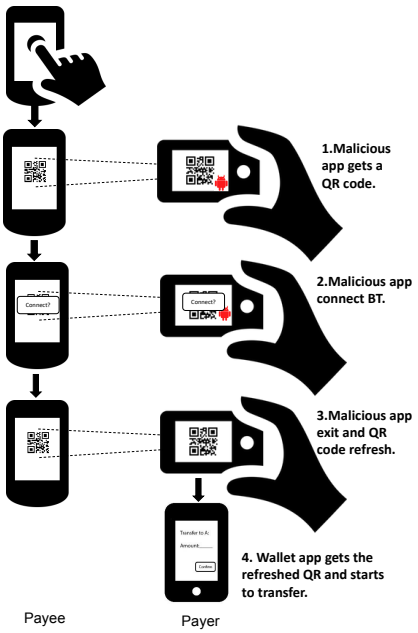

Figure 17: Work flow of attack against P2P transfer.

terface of the payment app. Though the payment token can be obtained by the attacker, it might be invalid to be spent by the attacker in a B2L transaction as the same token could be used earlier by the P2P transaction. We address this problem by forcing the payee's app to refresh the payment token. Since the payment app works in offline mode, both the old and new payment tokens are valid (if used within its lifetime, e.g., around 90 seconds for Alipay). A big challenge here is to alter the behavior of payer's phone without being discovered where there is no attack app installed. In the end, we found that bluetooth pairing could be used for this purpose.

Specifically, the attack app launches a pairing request to a nearby bluetooth device (highly likely to be the payer's phone) by calling an API `createBond()`. A window asking the user to confirm the pairing will be prompted on payee's phone. The attack app immediately cancels the pairing process by calling `cancelPairingUserInput()` API[7]. The pairing process is interrupted and the confirmation window on the payee's phone will disappear. The duration for this step is very short and it is nearly impossible to be observed by the user, as shown in our demo[1]. When the pairing confirmation window disappears, the payment app is brought to foreground and it will refresh the UI together with the token based on its logic.

Evaluation. We successfully launched the attack on a Samsung GT-S7562 (as payee) and a Galaxy Nexus (as payer). The total attacking time is 8s, including activity starting, QR code capturing (5s), bluetooth pairing

[7]This is a hidden API which can be invoked by java reflection.

requesting (3s), and activity exiting. During the bluetooth pairing, the confirmation window showed less than 1 second. We reported the attack to Alipay, and after that, they removed the functionality of payment QR code P2P transfer.

4 Transaction Point Authorization

Our study shows that STLS threat is completely realistic to mobile off-line payment schemes. The fundamental problem behind today's mobile off-line payment schemes is: one-time token is insufficient to protect against an active attacker who is not only able to sniff payment tokens but also capable of disrupting an ongoing legitimate transaction. In addition, our attacks have demonstrated that the validity period of offline payment tokens is sufficiently long for the adversary to transmit the token to a colluder to spend it in a different transaction. To mitigate the STLS thread and enhance the security of the off-line payment schemes, we propose a new solution called POSAUTH. In this section, we elaborate the design and implementation of POSAUTH, and evaluate its effectiveness and efficiency.

Design and implementation. The indispensable steps in a STLS attack include sniffing the payment token, halting the ongoing transaction, and spending the sniffed token in a different transaction. If any one of these steps fails, the STLS attack cannot succeed, which means that we can defend by undermining any step. Due to the broadcast nature of audio and MST channels, it is difficult to defend against sniffing on these channels. Though QR code is a unicast channel, it is still feasible for an adversary to sniff in certain scenarios (like in our attacks). As a result, our defense cannot rely on preventing the payment token from being sniffed. Similarly, it is difficult to prevent an active attacker from halting the ongoing transaction in mobile payment scenes, especially those with mobile POS systems. The only option left is to prevent a token from being used in a different transaction other than the original transaction between the payer and the merchant.

This observation inspires the design of POSAUTH, which binds a payment token to a specific transaction and authorizes the payer to use it only in the same transaction. Actually, we bind the payment token to the POS terminal on which it is going to be spent by the payer. POSAUTH is meant to make such binding easily deployable without changing the hardware. In particular, each POS terminal is attached with a QR code that contains its unique ID (PID). Before the payer starts to transmit her payment token to the POS terminal, she is required to scan the QR code. Upon receiving the POS terminal's unique ID, the payer encodes the PID into her payment token. In this way, the payment token is indeed bound to the terminal. As stated in Section 2, the payment token in the mobile off-line payment schemes is typically in the form of a HMAC over timestamp and other identity information. To prevent an attacker from replacing the encoded PID in the sniffed payment token, we encode the PID by integrating it within the one-way HMAC computation.

When the POS terminal receives the payment token, it sends the payment token as well as its PID to the payment service provider. The provider checks the consistency between the payment token and the PID. If they are bound, the transaction is allowed. If not, the transaction is supposed to be halted and the token's owner (the payer) should be warned about the risks of token being stolen. In this way, even if the payment token is stolen, it can not be spent on another POS terminal. It is unrealistic to assume that the attacker could pay to the same POS terminal in the mean time when the payer is still paying. We can further require that, if a payment token is spent, the payer's tokens with earlier timestamp should be invalid, in order to prevent a stolen payment token being spent on the bound POS terminal in the short period after the payer finishes payment with a refreshed payment token.

To understand whether POSAUTH can properly protect current mobile off-line payment schemes, we implemented a prototype of POSAUTH on Alipay QR Pay. More specifically, in Alipay QR Pay, the payment token is a string of 18 decimal numbers, consisting of a constant prefix of 2 digits, a suffix of 6-digit Time-based One-time Password (TOTP) computed from a pre-configured seed and the current timestamp, and a middle-10-digit encrypted identity (EID), which is generated by encrypting the payer's unique identity (or account number) in a customized symmetric encryption algorithm with the TOTP as its encryption key. In our POSAUTH implementation, we encode a PID into the QR Pay payment token by concatenating it with the timestamp in the TOTP computation. Upon receiving the payment token and PID, the server computes a set of valid TOTPs with the preconfigured seed, a set of valid timestamps, and the PID. And it checks whether the TOTP in the received payment token belongs to a valid one. If valid, the token is then bound to the POS terminal.

Evaluation. We mainly evaluate the time overhead during each transaction introduced by POSAUTH, because obviously POSAUTH does not introduce much other overhead like upgrading costs, power consumption etc.

Comparing with the existing transaction schemes, POSAUTH adds only one QR scanning step and slightly modifies the token generation algorithm while the remaining steps are all the same, which brings extra time consumption in 2 steps. For the time consumed by token generation (modifying algorithms), we consider it negligible since it is a simple operation to integrate the PID into a token generation algorithm (e.g., concatenating it with the timestamp during the TOTP computation in Alipay),

therefore we focus on the extra scanning step. To assess the extra time overhead introduced by POSAUTH, we implemented an app to scan a QR code and recorded the time spent between user clicking the button and QR code decoding module returning result. We prepared a QR code containing 18 digits that is enough to accommodate the POS terminal ID. Then we measure how much time a scan costs. We scanned 10 times and the average time is 3.8 seconds for a Galaxy Nexus.

For a mobile transaction, this time overhead is small, comparing to the time the cashier spends on manipulating the POS terminal, which usually costs 10 or even more seconds. As a result, we conclude that the POSAUTH is a practical defense scheme against the STLS attacks.

5 Discussion

Comparison with online scheme. In most online mobile payment schemes, users are required to confirm the transaction with brief transaction detail prompted, by inputing the password or by pressing their fingerprints. Therefore, our attacks fail in this scenario since such information is usually unavailable to a remote attacker.

However, online schemes are recommended for small business who could not afford a POS terminal. In addition, it requires decent network connection on payer device. Comparing to the off-line schemes which are provided by many large merchants[13] and are able to able to work regardless of payer's connection quality, their adoption is limited so far.

Root Cause. After a careful analysis of all vulnerable payment schemes having been discovered, we conclude that the root cause for STLS attack is the missing of bidirectional communication capabilities when transmitting tokens through near field communication channels. Without such capabilities, mobile off-line payment schemes have to rely on time-restricted one-time token for security, which, as shown in this work, turns out to be ineffective to active attackers. The threat could be mitigated by our defense scheme POSAUTH which provides a light-weight bidirectional communication capability by only requiring a quick scanning of QR code printed on POS terminal.

Comparison between POSAUTH and B2S. Similar to POSAUTH, in the B2S scenario, the merchant also presents a QR code for the payer to scan and pay. The main difference here is that the QR code in B2S is still used as a one-way communication channel, which is vulnerable in the presence of an active attacker. By replacing the merchant's QR code with a malicious one, an attacker can make unauthorized payment with the payer's account [27]. However, POSAUTH is immune to such attacks since the payment token is indeed bound to the ID of a POS terminal (via QR code scanning), and any discrepancy between the POS ID and the payment token would raise an alarm of such attacks to the payment service providers.

6 Related Work

Samsung Pay security. We studied the security of Samsung Pay and showed that it is vulnerable to our STLS attack. Before our work, the security aspect of Samsung Pay was studied by two groups recently as well [6, 3]. These studies showed that sniffing payment token from the MST channel by a passive attacker is feasible, but the proposed techniques did not lead to the successful attack under the real-world settings, as the payment token is one-time and the payer could spend it ahead of the attacker. Instead, our STLS attack employs a jamming device to disrupt the normal transaction to prevent the payment token from being spent by the payer, which ensures that an active attacker is able to spend the victim's payment token in a different transaction.

Data transfer over audio. Several communication products have realized data transmission through audio channel [34, 16]. These techniques encode data into audio signals distinguished by amplitude, frequency, or phase modulation [4]. Our study is the first to investigate the usage of the audio communication channel in mobile payment settings and proposed a realistic attack against such channel.

QR code security. QR code is one of the earliest channel for mobile payment and there have been many works demonstrating how to build a secure payment scheme on top of it [11, 28, 39, 31, 7, 36]. In these payment schemes, QR code is used to encode transaction information [28, 7] or users' payment token [11]. And a user can pay by showing her QR code to the merchant, or scanning the merchant's QR code, or both. In the mean time, attacks [27] have been proposed against QR Pay in B2S transaction, e.g., replacing the merchant's QR code with one associated with the attacker. In this paper, we mainly focus on the scenarios in which a user shows her QR code to the merchant or another user (B2L and P2P transaction). Different from the existing attacks, we are the first to investigate the STLS threats on these scenarios.

Since QR code can carry different types of data, whether and how it can be used to deliver malicious content have been investigated [26, 23]. In fact, an attacker is able to launch attacks including phishing [53], SQL injection [46], and even malicious app installation [55], by encoding malicious content into QR code. In these attacks, an attacker can either use a new malicious QR code or partially modify an existing QR code [24]. In our attack against POS-based payment, we also partially modify the QR code. The difference is that prior attacks still keep the QR code readable but our attack prevents it from being read. In the meantime, defense techniques [54] were

proposed to protect users when scanning an untrusted QR code. Such techniques can not prevent our attacks since the modified QR code has no malicious content.

Similar to POSAUTH, QR code has also been employed to transfer information for authentication schemes [48, 30, 29], given its high usability and low deployment cost. In this work, we show that QR code can be used to protect payment security. Such direction has not been explored before.

Security on other mobile payment schemes. The security of other mobile payment schemes, including contactless NFC payment [40, 41] and online mobile wallet payment [52, 12], has been studied. NFC-based payment has been adopted by the major phone vendors, like Google and Apple. Attack [37, 8, 9] and defense [18, 38, 47] techniques regarding this channel have been investigated, but none of them are similar to our STLS attack. Due to its extremely short communication distance and the challenge-response based bidirectional communication pattern, NFC payment is not affected here.

Users' perception of the emerging mobile payment techniques is also investigated and studied. These studies show that several factors could impede the adoption of mobile payment methods, including their security, usability, and cost [32, 45, 19].

7 Conclusion

In this paper, we present a new threat called STLS which can enable adversaries to attack off-line payment schemes by sniffing payment token, aborting current transaction, and spending the stolen token at other places. We have investigated some leading off-line payment systems in real world and demonstrated that such STLS attacks are completely realistic. We also carreid out security analysis, which reveals some major limitations of existing token protection techniques. Contrary to the closed settings of traditional payment systems, off-line mobile payment solutions have larger attacking surface. Channels between smartphone and POS terminal are susceptible to sniffing attack. Communications between mobile POS and backend servers are built on WiFi or 3G/4G network, thus the ongoing transactions can be disrupted. More importantly, most token delivering channels are one-way only, so tokens cannot be bound to the POS terminal of current transaction. Meanwhile, shortening the token valid period only still cannot guarantee adequate payment security. To mitigate STLS threats, we propose POSAUTH which forces a payment token to include the unique ID of current POS terminal and, when combined with short valid period, is able to confine a token to be used in legitimate transactions only. In the future, we plan to work with merchants and deploy POSAUTH in real-world POS systems.

Acknowledgments

We thank anonymous reviewers for their precious comments. Authors from Tsinghua University are supported in part by Research Grant of Beijing Higher Institution Engineering Research Center. The project is also supported in part by NSFC (Grant No. 61572415), NSF CNS-1223477, 1223495, 1527141, 1618493, ARO W911NF1610127, and the General Research Funds (Project No. 14217816 and 24207815) established under the University Grant Committee of the Hong Kong Special Administrative Region, China.

References

[1] Supporting materials: Stls in mobile payment. https://sites.google.com/site/stlsinmobilepayment/. [Online; accessed 14-Feb-2017].

[2] 21, A. Ansi/iso alpha and bcd data format. http://www.abacus21.com/Magnetic-Strip-Encoding-1586.html. [Online; accessed 19-Jan-2017].

[3] ALVADOR MENDOZA. Samsung Pay: Tokenized Numbers, Flaws and Issues. Tech. rep., 2016.

[4] APPLIDIUM. Audio modem: data over sound. https://applidium.com/en/news/data_transfer_through_sound/, 2013. [Online; accessed 19-Jan-2017].

[5] BRIDG. Bridg. https://www.bridgtheapp.com. [Online; accessed 19-Jan-2017].

[6] CHOI, D., AND LEE, Y. Eavesdropping one-time tokens over magnetic secure transmission in samsung pay. In *Proceedings of the 10th USENIX Conference on Offensive Technologies* (2016), USENIX Association, pp. 52–58.

[7] DE, P., DEY, K., MANKAR, V., AND MUKHERJEA, S. An assessment of qr code as a user interface enabler for mobile payment apps on smartphones. In *Proceedings of the 7th International Conference on HCI, IndiaHCI 2015* (2015), ACM, pp. 81–84.

[8] DRIMER, S., MURDOCH, S. J., ET AL. Keep your enemies close: Distance bounding against smartcard relay attacks. In *USENIX Security* (2007), vol. 2007, pp. 87–102.

[9] EMMS, M., AND VAN MOORSEL, A. Practical attack on contactless payment cards. In *HCI2011 Workshop-Heath, Wealth and Identity Theft* (2011).

[10] FUJIAN NEWLAND AUTO-ID TECH. CO., L. Nls-fr40. http://www.newlandaidc.com/h-pd-j-70-3_10.html. [Online; accessed 19-Jan-2017].

[11] GAO, J., KULKARNI, V., RANAVAT, H., CHANG, L., AND MEI, H. A 2d barcode-based mobile payment system. In *Multimedia and Ubiquitous Engineering, 2009. MUE'09. Third International Conference on* (2009), IEEE, pp. 320–329.

[12] GAO, J. Z., CAI, J., LI, M., AND VENKATESHI, S. M. Wireless payment–opportunities, challenges, and solutions. *Published by High Technology Letters 12* (2006).

[13] GARG, G. Qr code payment introduction. http://scanova.io/blog/blog/2015/04/08/qr-code-payment/. [Online; accessed 19-Jan-2017].

[14] GOOGLE. Fileobserver. https://developer.android.com/reference/android/os/FileObserver.html.

[15] GOOGLE. Flag secure. https://developer.android.com/reference/android/view/WindowManager.LayoutParams.html#FLAG_SECURE. [Online; accessed 19-Jan-2017].

[16] GOOGLE. Google tone. https://chrome.google.com/webstore/detail/google-tone/nnckehldicaciogcbchegobnafnjkcne?hl=en. [Online; accessed 19-Jan-2017].

[17] GOOGLE. System alert window. https://developer.android.com/reference/android/Manifest.permission.html#SYSTEM_ALERT_WINDOW. [Online; accessed 19-Jan-2017].

[18] HALEVI, T., MA, D., SAXENA, N., AND XIANG, T. Secure proximity detection for nfc devices based on ambient sensor data. In *European Symposium on Research in Computer Security* (2012), Springer, pp. 379–396.

[19] HUH, J. H., VERMA, S., RAYALA, S. S. V., BOBBA, R., BEZNOSOV, K., AND KIM, H. I don't use apple pay because it's less secure...: Perception of security and usability in mobile tap-and-pay.

[20] INC, L. System and method for a base-band nearfield magnetic stripe data transmitter. http://www.google.com/patents/US8814046, 2014. [Online; accessed 19-Jan-2017].

[21] INTELLIGENCE, B. mpos us installation base. http://www.businessinsider.com/square-makes-another-play-at-retailers-2017-2. [Online; accessed 16-Feb-2017].

[22] KALEBKE. Gyroscopeexplorer. https://github.com/KalebKE/GyroscopeExplorer. [Online; accessed 19-Jan-2017].

[23] KHARRAZ, A., KIRDA, E., ROBERTSON, W., BALZAROTTI, D., AND FRANCILLON, A. Optical delusions: A study of malicious qr codes in the wild. In *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on* (2014), IEEE, pp. 192–203.

[24] KIESEBERG, P., LEITHNER, M., MULAZZANI, M., MUNROE, L., SCHRITTWIESER, S., SINHA, M., AND WEIPPL, E. Qr code security. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia* (2010), ACM, pp. 430–435.

[25] KOVACS, E. Samsung has one day token lifetime. http://www.securityweek.com/samsung-pay-token-flaw-allows-fraudulent-transactions. [Online; accessed 16-Feb-2017].

[26] KROMBHOLZ, K., FRÜHWIRT, P., KIESEBERG, P., KAPSALIS, I., HUBER, M., AND WEIPPL, E. Qr code security: A survey of attacks and challenges for usable security. In *International Conference on Human Aspects of Information Security, Privacy, and Trust* (2014), Springer, pp. 79–90.

[27] L., K. Hidden risks with 2d qr code payment. https://www.linkedin.com/pulse/20140907174521-104874410-hidden-risks-with-2d-qr-code-payment. [Online; accessed 12-Feb-2017].

[28] LEE, J., CHO, C.-H., AND JUN, M.-S. Secure quick response-payment (qr-pay) system using mobile device. In *Advanced Communication Technology (ICACT), 2011 13th International Conference on* (2011), IEEE, pp. 1424–1427.

[29] LEE, Y. S., KIM, N. H., LIM, H., JO, H., AND LEE, H. J. Online banking authentication system using mobile-otp with qr-code. In *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on* (2010), IEEE, pp. 644–648.

[30] LIAO, K.-C., AND LEE, W.-H. A novel user authentication scheme based on qr-code. *JOURNAL OF NETWORKS 5*, 8 (2010), 937.

[31] LIÉBANA-CABANILLAS, F., RAMOS DE LUNA, I., AND MONTORO-RÍOS, F. J. User behaviour in qr mobile payment system: the qr payment acceptance model. *Technology Analysis & Strategic Management 27*, 9 (2015), 1031–1049.

[32] LINCK, K., POUSTTCHI, K., AND WIEDEMANN, D. G. Security issues in mobile payment from the customer viewpoint.

[33] LOOPPAY. Looppay faq. https://www.looppay.com/faqs/. [Online; accessed 19-Jan-2017].

[34] LTD, A. Chirp. http://chirp.io, 2013. [Online; accessed 19-Jan-2017].

[35] LTD, O. W. R. Spectrumview. http://www.oxfordwaveresearch.com/products/spectrumviewapp/. [Online; accessed 19-Jan-2017].

[36] MA, T., ZHANG, H., QIAN, J., HU, X., AND TIAN, Y. The design and implementation of an innovative mobile payment system based on qr bar code. In *Network and Information Systems for Computers (ICNISC), 2015 International Conference on* (2015), IEEE, pp. 435–440.

[37] MARKANTONAKIS, K., FRANCIS, L., HANCKE, G., AND MAYES, K. Practical relay attack on contactless transactions by using nfc mobile phones. *Radio Frequency Identification System Security: RFIDsec 12* (2012), 21.

[38] MEHRNEZHAD, M., HAO, F., AND SHAHANDASHTI, S. F. Tap-tap and pay (ttp): preventing the mafia attack in nfc payment. In *International Conference on Research in Security Standardisation* (2015), Springer, pp. 21–39.

[39] NSEIR, S., HIRZALLAH, N., AND AQEL, M. A secure mobile payment system using qr code. In *Computer Science and Information Technology (CSIT), 2013 5th International Conference on* (2013), IEEE, pp. 111–114.

[40] ONDRUS, J., AND PIGNEUR, Y. An assessment of nfc for future mobile payment systems. In *Management of Mobile Business, 2007. ICMB 2007. International Conference on the* (2007), IEEE, pp. 43–43.

[41] PASQUET, M., REYNAUD, J., ROSENBERGER, C., ET AL. "payment with mobile nfc phones" how to analyze the security problems. In *2008 International Symposium on Collaborative Technologies and Systems.(see section 2)* (2008).

[42] RAMPTON, J. The evolution of the mobile payment. https://techcrunch.com/2016/06/17/the-evolution-of-the-mobile-payment/. [Online; accessed 16-Feb-2017].

[43] SAMSUNG. Samsung pay faq. http://security.samsungmobile.com/doc/Press_Guidance_Samsung_Pay.pdf. [Online; accessed 19-Jan-2017].

[44] SAMSUNG. Samsung's looppay: What it is, and why you should care. https://www.cnet.com/news/samsungs-looppay-what-it-is-and-why-you-should-care/.

[45] SCHIERZ, P. G., SCHILKE, O., AND WIRTZ, B. W. Understanding consumer acceptance of mobile payment services: An empirical analysis. *Electronic commerce research and applications 9*, 3 (2010), 209–216.

[46] SHARMA, V. A study of malicious qr codes. *International Journal of Computational Intelligence and Information Security 3*, 5 (2012), 21–26.

[47] SHRESTHA, B., SAXENA, N., TRUONG, H. T. T., AND ASOKAN, N. Drone to the rescue: Relay-resilient authentication using ambient multi-sensing. In *International Conference on Financial Cryptography and Data Security* (2014), Springer, pp. 349–364.

[48] STARNBERGER, G., FROIHOFER, L., AND GÖSCHKA, K. M. Qr-tan: Secure mobile transaction authentication. In *Availability, Reliability and Security, 2009. ARES'09. International Conference on* (2009), IEEE, pp. 578–583.

[49] STATISTA. Global mobile payment revenue 2015-2019. https://www.statista.com/statistics/226530/mobile-payment-transaction-volume-forecast/. [Online; accessed 19-Jan-2017].

[50] TECHNOLOGIES, S. Symbol ds6708-dl product reference guide. https://www.zebra.com/content/dam/zebra_new_ia/en-us/manuals/barcode-scanners/ds6707-digital-imager-scanner-product-reference-guide-en-us.pdf. [Online; accessed 19-Jan-2017].

[51] TONETAG. Tone tag. https://www.tonetag.com/about.html. [Online; accessed 19-Jan-2017].

[52] VARSHNEY, U., AND VETTER, R. Mobile commerce: framework, applications and networking support. *Mobile networks and Applications 7*, 3 (2002), 185–198.

[53] VIDAS, T., OWUSU, E., WANG, S., ZENG, C., CRANOR, L. F., AND CHRISTIN, N. Qrishing: The susceptibility of smartphone users to qr code phishing attacks. In *International Conference on Financial Cryptography and Data Security* (2013), Springer, pp. 52–69.

[54] YAO, H., AND SHIN, D. Towards preventing qr code based attacks on android phone using security warnings. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security* (New York, NY, USA, 2013), ASIA CCS '13, ACM, pp. 341–346.

[55] ZHOU, Y., AND JIANG, X. Dissecting android malware: Characterization and evolution. In *Security and Privacy (SP), 2012 IEEE Symposium on* (2012), IEEE, pp. 95–109.