The Dissertation Committee for Oliver Christopher Jensen
certifies that this is the approved version of the following dissertation:

Secure Protocols for Contactless Credit Cards and Electronic Wallets

Committee:

Mohamed G. Gouda, Supervisor

Lorenzo Alvisi

Lili Qiu

Vijay Garg

Secure Protocols for Contactless Credit Cards and Electronic Wallets

by

Oliver Christopher Jensen, B.A., M.S.Comp.Sci.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2017

This thesis is dedicated to those who fight for privacy in our increasingly public world.

Acknowledgments

To my parents, Michael and Karin, thank you for instilling, encouraging, and suffering through an unceasing curiosity about how things work, and for providing me with a proverbial springboard of opportunities growing up. To my brother, David, thank you for showing me that computers are indeed interesting tools, and for introducing me to the concepts of security and privacy at an early age by insisting I communicate with you via PGP when I was seven. As a result of those efforts, no prying eyes will ever know what secrets and sensitive information we were privy to at that age... at least for several minutes, until they guess my passphrase "`mA4Tn`".

There have been many teachers and professors over the years who have been instrumental in helping me become the person I am today. I am sincerely grateful to two in particular, who have both had a profound impact: John Barry Smith, teacher of Mathematics at the International School of Geneva, who instilled within me a love of learning and a general refusal to accept mediocrity from myself; Philip Mulry, professor of Computer Science at Colgate University, whose unending enthusiasm made even the dryest of topics captivating.

Special thanks to professors Jaime Spacco and Vijay Ramachandran at Colgate University, and professors Vitaly Shmatikov and Lorenzo Alvisi at the University of Texas at Austin. You lit (and more importantly, stoked and maintained) the fire of interest I felt towards higher learning. You showed me that research was interesting, and helped me find my direction. I wouldn't be where I am today were it not for your contributions.

A heartfelt thanks to my friends: Richard McPherson, for always being there when I needed a sounding-board, a voice of reason, a companion, or anything else; Megan Merner, for her constant love and support, and for endlessly reading and proof-reading this document.

Finally, to my advisor, Mohamed Gouda, thank you for your unending patience, guidance, and support. Your skillful approach toward both writing and expression has guided my abilities in these areas. Your encouragement and resolve were invaluable when setbacks and discouragement loomed. You saw me through to the completion of this endeavor, and for that I am extremely grateful.

Secure Protocols for Contactless Credit Cards and Electronic Wallets

Publication No. ____________________

Oliver Christopher Jensen, Ph.D.
The University of Texas at Austin, 2017

Supervisor: Mohamed G. Gouda

The contactless credit card protocol in use today is insecure. The credit card industry has chosen to use the NFC channel for contactless transactions. However, reliance on NFC's short range has led to poor assumptions in the contactless credit card protocol. For example, the card assumes (sometimes incorrectly) that its ability to receive a solicitation implies the cardholder's intent to purchase. In this dissertation, we examine the protocol currently in use, and present a family of three replacement protocols to defend against its deficiencies.

First, we consider "outsider" attacks (e.g. eavesdropping, skimming attacks, relay attacks, and attacks facilitated by compromised points of sale) and design our first protocol to defend against these attacks. We call this protocol the Externally Secure CC Protocol, and design it using stepwise refinement.

This protocol makes use of single-use "charge tokens" verifiable by the bank, while minimizing computation that needs to occur on the card.

Second, we identify two attacks which may be carried out by malicious retailers: Over-charge attacks and Transparent Bridge attacks. Both attacks are predicated on the customer's lack of participation in the protocol, and involve modifying or replacing a charge after it has been confirmed by the customer. We look to Electronic Wallet applications (such as Android Pay and Apple Wallet), which provide a channel between customer and card. We augment the Externally Secure CC Protocol using this channel to construct the Secure CC Protocol, binding charge tokens to a given price, and thus stymieing both outsider and malicious retailer attacks.

The Secure CC Protocol supports a property known as *linkability*: while only the bank can verify charge tokens, tokens from the same card can be recognized as such by the retailer. This property is also supported by the (insecure) protocol in use today, and is commonly used by retailers to construct marketing profiles on their customers. However, linkability has serious consumer privacy consequences, so we consider the converse property of *unlinkability*, where a retailer cannot identify different purchases as having been made by the same card. We require that our unlinkable protocol make use of existing infrastructure, so as not to require retailer cooperation. In response, we design the Unlinkable Wallet Protocol, leveraging techniques from the Secure CC Protocol to guard against malicious outsiders and retailers, while tunneling secure and unlinkable charge tokens through the protocol in use today.

Table of Contents

List of Figures

Chapter 1

Introduction

Near Field Communication (NFC) is short-range wireless channel which permits two devices to communicate wirelessly, allowing for arbitrary computation on both devices. A distinctive feature of NFC is that one of the two devices may eschew having its own power source (termed a “passive device”), drawing power from the other device in order to perform computation and respond [7]. This feature makes NFC particularly useful for use in contactless integrated circuit cards (often termed “Proximity Cards”) [19], where a person may authenticate through demonstrating ownership of a physical token (by bringing it close to a reader). Such applications are commonly used for access control to buildings, parking garages, and public transit [12, 35, 43, 45].

These same properties also make NFC an attractive channel for use in contactless payment protocols, of which common examples include MasterCard’s PayPass^{TM} [33] and Visa’s payWave^{TM} [44]. Use of NFC has also been standardized more generally in EMV’s Contactless Specifications for Payment Systems [10].

1.1 Security Concerns

The protocol in use today by contactless credit cards falls short of addressing even basic security issues. It employs no authentication, allowing any device to effectively pose as a point of sale. Furthermore, it transmits sensitive information (the credit card number, expiration date, etc.) wirelessly in plaintext. This transmission of data is then considered sufficient to allow an arbitrary charge to occur. This leads to a number of potential security vulnerabilities.

The contactless credit card protocol equates the ability to communicate with an intent to pay. Relying on NFC's short range, this is intended as a stand-in for equating *proximity* with intent to pay. However, much work has been done to dissociate the two concepts [2, 6, 14, 15, 32], such as by proxying and relaying communications over greater distances.

Even the notion that proximity is tantamount to an intention to pay is flawed when used in a protocol without authentication. For example, skimming a contactless credit card is easy, because a phone can be moved close to a pocket containing a credit card. Simply instruct an NFC-capable phone to send out solicitation messages claiming to be a point of sale, and any contactless credit cards within range will respond with their information. Indeed, there is an Android application [18] which simplifies this process down to clicking a button. This skimmed data can then be replayed to a real point of sale in order to perform a purchase on behalf of the skimmed credit card.

Eavesdropping is likewise easy. NFC range is quite short, greatly limiting the possible location of an eavesdropper. However, constructing an eavesdropping antenna out of an inexpensive NFC tag is a simple procedure, as described in Section 2.3. Concealing such an antenna within a few centimeters of (or even attached to) a point of sale is sufficient to harvest credit card information, due to the plaintext nature of the messages being sent. Furthermore, work has also been done to increase the range at which NFC communication may be captured [2, 23], exacerbating the issue.

A credit card's response to a solicitation is likewise overly permissive in how it may be used: while a point of sale may display a given price to the customer, there is no protocol-level assurance that the card will be charged the price on the screen. Instead, a credit card's response provides a point of sale with the ability to charge any arbitrary single purchase to the credit card. This charge need not even come from the same retailer or establishment which solicited it.

In this dissertation, we present a family of three replacement protocols to defend against the deficiencies of the contactless credit card protocol in use today. The first two (described in Chapters 3 and 4) answer to security concerns, and the third (described in Chapter 8) answers to privacy concerns.

In Chapter 3, we construct the Externally Secure CC Protocol. We tackle the problems that are raised by malicious outsiders: parties external to the customer and the retailer. We find that the contactless credit card protocol in use today is vulnerable to eavesdroppers, skimming attacks, relay attacks, and attacks facilitated by compromised points of sale. We construct a replacement protocol to transmit single-use charge tokens which leak no sensitive information, nullifying their value outside of the current transaction. These charge tokens are also bound to a randomized challenge included in the solicitation, and thus cannot be skimmed ahead of time.

In Chapter 4, we construct the Secure CC Protocol. We explore the overly permissive nature of the credit card's response, and approach the protocol from the perspective of a malicious retailer. We enumerate two attacks which a malicious retailer may perform: over-charge attacks and transparent bridge attacks. These attacks affect both the contactless credit card protocol in use today, as well as the Externally Secure CC Protocol described in Chapter 3. Both attacks involve using the credit card's response to authorize a purchase which differs from the one accepted by the customer. We extend the Externally Secure CC Protocol to limit the utility of charge tokens for use outside of the current (confirmed) transaction.

1.2 Privacy Concerns

In Chapter 6, we discuss the benefits and drawbacks associated with an additional property: *linkability.* This property allows a retailer to identify multiple purchases made with the same credit card. Naturally, any protocol which transmits a credit card number (or any other unique and constant identifier) maintains this property. This includes all protocols discussed thus far.

Linkability is valuable to retailers, as it enables the construction of purchasing profiles, which can then be used for marketing purposes or sold to interested third parties. However, this behavior may be undesirable to consumers concerned about their privacy: purchasing habits can reveal extremely sensitive information [8].

Our third protocol, described in Chapter 8 and termed the Unlinkable Wallet Protocol, renders retailers unable to identify purchases made using the same credit card (in addition to protecting customers from malicious outsiders and malicious retailers). While the Externally Secure CC Protocol and the Secure CC Protocol require significant retailer cooperation in modifying or replacing all point of sale devices, the Unlinkable Wallet Protocol makes use of existing point of sale infrastructure, requiring no modification to point of sale devices and thus no retailer cooperation.

1.3 Related Work

Related work in this area falls into a number of broad categories: the security of the NFC channel itself, breaking the distance and range assumptions built into NFC, the legitimate use of phones as credit card payment devices, the malicious use of phones as NFC relay or credit card skimming and cloning devices, and other NFC payment or communication research.

Relating to channel security, Haselsteine and Breitfuß provide a broad survey of various attacks and defenses applicable to protocols built on the NFC channel [17]. Their focus is on channel-layer defenses, shielding NFC communication itself from attackers, and suggesting that NFC participants perform a key-exchange protocol such as Diffie-Hellman [5], then using this key to encrypt subsequent communication.

Similarly, Madlmayr et al. analyze the state of NFC communication [31], focusing on not only the security and privacy of communications, but also the continued operability of device and host controller. They enumerate and discuss the viability and consequences of a number of attacks, limiting discussion to channel security. Kortvedt further explores the problem of eavesdropping on NFC communications [25], suggesting various improvements such as using a symmetric encryption solution with strong mutual authentication, with “Over-the-Air Programming” (OTA) as a solution for key management.

These works [17, 25, 31] focus on channel security, and thus are effective against channel attacks such as eavesdropping. However, confidentiality and integrity of the channel cannot provide authentication or assurance of correct operation of the receiver. As such, these approaches are not effective at protecting NFC credit card payments.

Hanke discusses a practical relay attack against ISO 14443 (NFC “proximity cards”) [15], demonstrating that such attacks are viable and can be invisible to the application layer.

Kfir et al. explore maximizing the distance over which a relay attack may occur using only readily available equipment [23]. They explore the problem from both sides, focusing on increasing the distance between the two relay devices and their subsequent communication endpoints. For example, increasing the distance between the relay and the tag (in our case, the credit card) increases the distance from which credit card information may be skimmed.

Similarly, Brown et al. explore maximizing eavesdropping range [2]. They find that range appears to be determined primarily by background noise, and that with proper equipment they can achieve an eavesdropping range of 90cm. This represents a much larger sphere volume in which an eavesdropping node may be located.

On the topic of using mobile phones legitimately for contactless credit card transactions (i.e. the concept of an Electronic Wallet), Roland et al. discuss the relative merits and weaknesses inherent when mobile devices emulate NFC cards in [37] and [39]. These works discuss several attack vectors (including relay attacks, denial-of-service attacks, and unauthorized usage attacks), and analyze APIs provided by mobile phones. In particular, multiple Electronic Wallet applications anchored to the same secure element is highlighted as potentially problematic. A proposed defense is to store sensitive information remotely, and access this data on demand via an authorized relay application.

In [40], Roland et al. discuss relay attacks targeting Electronic Wallet applications in general, describing a relay attack involving malware in which relay attackers need not even be within NFC range of the victim's phone. In subsequent work [41], Roland et al. investigate the feasibility of relay attacks on Google Wallet (now called Android Pay) in particular. They relay transmissions out-of-band over a TCP connection and suggest three primary countermeasures: short point of sale timeouts, Wallet Application PIN verification, and strong limits on the capabilities afforded to applications interfacing with the phone's NFC API.

Drimer and Murdoch present an attack on credit card payment systems [6], which we described in Section 4.1.2 as the Transparent Bridge attack. This attack relies on the ability to perform out-of-band real-time proxying and relaying of messages between two parties. Drimer et al. implement this attack against EMV "Chip" cards, demonstrating its practicality. They recommend defending against such attacks via distance bounding protocols. Hancke et al. present such a distance bounding protocol for RFID / NFC tags in [16]. This protocol seeks to infer an upper bound on distance between communicating endpoints, based on response times and the speed of light.

Anderson discusses the move towards using mobile phones as payment devices [1], and predicts that such devices (programmable by the end-user) would make excellent platforms from which to conduct relay attacks on payment protocols. In [13], Francis et al. discuss the ability for NFC capable mobile phones to operate as skimming platforms, They propose countermeasures to prevent NFC mobile phones from being used as such, with the intention of raising the difficulty bar.

In addition, Francis et al. find that out-of-band real-time proxying and relaying of general NFC communication is possible in [14]. They demonstrate two NFC devices communicating over a distance much larger than NFC range, by using phones to relaying NFC communications over Bluetooth. While Drimer et al. perform the Transparent Bridge attack with EMV Chip cards, this result indicates that the attack applies to contactless credit cards as well.

Francis et al. propose to use location information such as GPS coordinates in order to detect and defend against this relaying of messages, which in turn would render the Transparent Bridge attack infeasible. However, location information can be unreliable or unavailable, and as such, one cannot rely on its availability and correctness. Furthermore, passive NFC tags such as physical contactless credit cards do not have access to location information.

In a followup work [32], Markantonakis et al. construct and demonstrate a practical NFC relay over Bluetooth using mobile phones. They suggest using distance bounding and location information to defend against proxying NFC communication. They also suggest using the NFC tag's UID token from the communication layer's anti-collision operation, but acknowledge that tag UIDs are often randomized for privacy reasons. Finally, they recommend restricting what mobile phone applications may do with regards to NFC communication.

While the distance bounding approaches suggested by these works [14, 32, 6] are reasonable when reading responses directly from chip I/O or a dedicated tag, they do not lend themselves well to a protocol in which Electronic Wallets participate: determining an upper-bound to distance based on response times using the speed of light as a metric becomes a very coarse measurement when dealing with a multitasking operating system on a smart phone, where delays can be variable, and depend on page faults, context switching, and unrelated software. In contrast, our approach does not seek to detect or prevent attacks relying on the proxying or relaying of information. Instead, our protocol aims to render such attacks harmless.

With regard to contactless credit card protocol security, Lee provides some analysis of relay and skimming attacks on NFC credit card transactions [29]. The stated goal of this work is to demonstrate the simplicity of performing these attacks, emphasizing that they are easily performed by the general public (having little-to-no knowledge of NFC or credit card protocols). To this end, he presents an Android application *NFCProxy* [18] which implements these attacks. The application is easily installed, and transforms contactless credit card skimming into no more than a button-push endeavor.

Subsequently, Lifchitz performs a more detailed exploration of the contactless credit card protocol [30], and emphasizes the near total lack of security in contactless credit cards. Roland et al. then demonstrate how such attacks may be used to clone contactless credit cards [38].

In [3], Chen et al. explore using the challenge-response mechanism built into 3G to perform authentication for payment protocols. They propose such a protocol for contactless payments (albeit not for credit card payments), and analyze the risks inherent to using this authentication method. This proposed protocol involves mutual authentication and customer price confirmation (communicated to the phone from the point of sale device via NFC).

In [11], Eun et al. explore the issue of privacy in the face of NFC eavesdroppers, considering mobile payments as a case study. They suggest the creation of an NFC-SEC protocol, complete with key-exchange and public key cryptography, including requirements of *unobservability* (an individual transaction may not be distinguishable from other transactions) and *unlinkability* (two transactions from the same card may not be identifiable as such), while still maintaining *traceability* (it must be possible to ascertain who generated a given set of data in order to troubleshoot problems which may arise). Eun et al. approach this problem from a clean slate perspective and do not constrain themselves to making use of existing infrastructure, imposing a significant barrier to adoption where infrastructure has already been deployed. Furthermore, the *lack* of unlinkability of current credit card transactions is profitable to retailers. As such, a clean-slate unlinkable protocol is unlikely to see adoption for contactless credit card processing.

Taking a more general view of using NFC for secure applications, Coskun et al. discuss a general view of where NFC technology is expected to go [4]. Comparing NFC to Bluetooth and Zigbee, they suggest that NFC will become the channel of choice for anything from unlocking doors, payment systems, identification, etc., and posit an end-goal of having a mobile phone replace everything one would otherwise need to use for such purposes.

Nandakumar et al. discuss an alternate implementation of NFC-like communication for mobile phones [34]. In particular, this work is directed towards providing NFC-like functionality to phones which do not support NFC.

Rather than using RF communication, this work explores using the speaker and microphone to transmit data between two phones over short range. Of particular interest is their use of a self-jamming signal, effectively preventing eavesdroppers. Exploration of a self-jamming NFC carrier-wave would be particularly intriguing.

Chapter 2

The Insecure CC Protocol

Contactless credit card payment systems are rapidly gaining popularity. Such systems allow customers to pay using their credit cards by simply bringing the card close to a point of sale, and without actually swiping or necessarily coming into direct contact with it.*

This chapter describes the protocol for such payments in use at retailers today. We refer to this protocol as the Insecure CC Protocol, because it employs hardly any protection against fraudulent use.

2.1 Goals

A credit card payment system has five fundamental principals:

1. A **customer** who wants to make a purchase.
2. A **bank** at which the customer has an account.
3. A **credit card** issued by the bank to the customer.

* Portions of this chapter have previously been published in [20]. While most of the contributions in this chapter are my own, acknowledgements are due to Mohamed Gouda for helping construct clear and concise protocol descriptions.

4. A **retailer** from whom the customer wishes to make the purchase.

5. A **point of sale** controlled and initialized by the retailer. It displays the purchase price to the customer, and communicates with both the credit card and its issuing bank to coordinate the transaction.

The underlying goal of any credit card payment protocol is to enable the customer and the retailer to negotiate a transaction, after which the customer's bank debits the appropriate funds from the customer's account and issues a payment to the retailer.

Traditional magnetic-stripe credit card systems have been in operation in this space for many years, but they face several important drawbacks: it is easy to accidentally de-magnetize your credit card, and dirty or corroded contacts on the point of sale can make even a well-magnetized card difficult to read. As a result, it is not at all uncommon for a retailer to need to swipe a credit card multiple times before a successful read occurs.

The primary goal of the Insecure CC Protocol is to solve these drawbacks by using a contactless (i.e. wireless) communication channel. However, due to the reduction of control that a contactless solution presents, this protocol has a secondary goal as well: to prevent a malicious actor from cloning a credit card simply by querying its contents.

2.2 Design of the Protocol

The Insecure CC Protocol uses the NFC channel to transmit messages between the credit card and the point of sale. This decision is based on the following factors:

- NFC is a wireless channel, and thus it is unaffected by card demagnetization or read errors due to dirty or corroded contacts.
- NFC has a very short range (under 10 cm), mitigating many of the privacy concerns commonly associated with wireless channels.
- NFC supports communication with unpowered (termed "passive") devices, allowing the credit card to forego having its own power source.
- Even passive NFC devices such credit cards can perform complex computation while being wirelessly powered by the point of sale.

In the Insecure CC Protocol, the customer indicates the intention to pay by enabling communication between the point of sale and the contactless credit card. This is done by the customer bringing the credit card within a few centimeters of the point of sale. Once within range of each other, the point of sale may send messages to the credit card and receive any resulting responses. The steps involved in this protocol, illustrated in Figure 2.1, are as follows:

1. The point of sale displays the price of the purchase on its screen, while simultaneously attempting to establish communication over NFC.

2. After checking the displayed price, the customer places the credit card within NFC range of the point of sale, and communication between the point of sale and the credit card is established.

3. The point of sale sends a Solicitation message to the credit card.

4. The credit card responds to the solicitation message with a Card Information message, supplying the point of sale with the necessary information to initiate a transaction, and identifying the credit card's issuing bank.

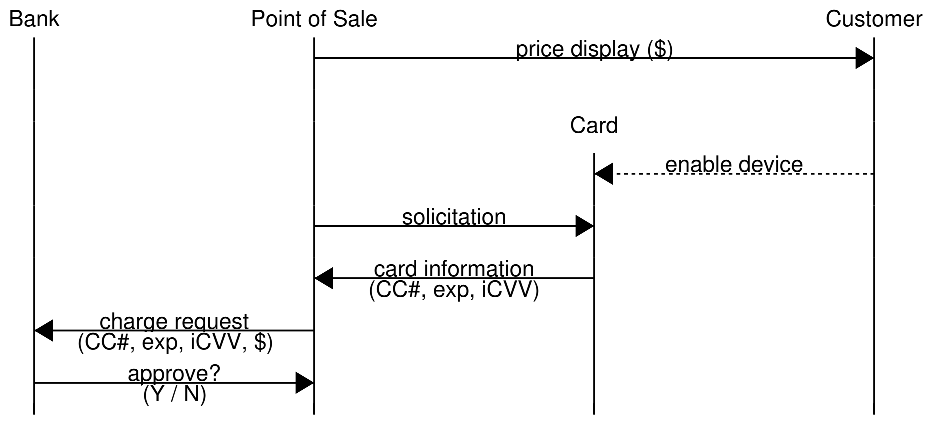
5. Then the point of sale sends a Charge Request message to the bank. This message is sent securely over the Internet.

6. The bank verifies the details of the charge request, and responds to the point of sale with a Approval message, indicating whether the Charge Request has been accepted.

The message contents in the Insecure CC Protocol are as follows:

Solicitation: In practice, the solicitation message actually consists of a number of messages sent in both directions. The purpose of these messages is to exchange information about the credit card type (e.g. *Visa Credit*) and the point of sale model (e.g. *2PAY.SYS.DDF01*), which defines the format of subsequent messages. It is a choreographed dance with a specific (and constant) set of messages for a given model of point of sale

Figure 2.1: The Insecure CC Protocol



and credit card, so we abstract this conversation to a single solicitation message.

Card Information: This message contains all information necessary to coordinate an arbitrary charge to the customer's account. It consists of four components:

- The credit card number, identical to the number printed on the front of the card.
- The credit card's expiration date.
- An *iCVV* ("integrated Card Verification Value"). This iCVV is a security code, similar to the 3-digit number printed on the back of a credit card, but is newly generated for each transaction. It is an element in a pseudo-random sequence generated by a secret seed known only to the credit card and its issuing bank, making it

unpredictable to third parties.

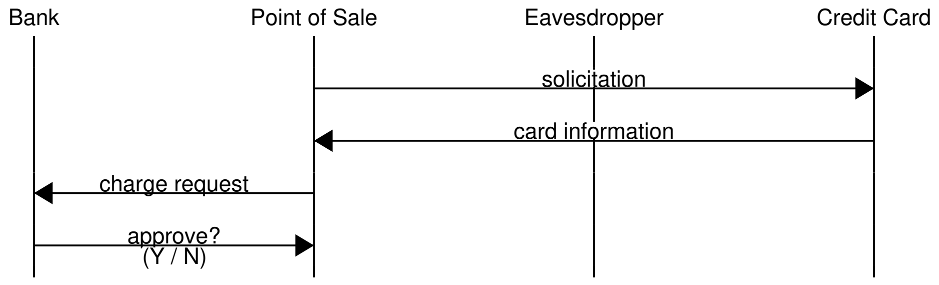
- The issuing bank name. This field is used for routing purposes and is not a component of the subsequent Charge Request. As such, it is not pictured in Figure 2.1.

Charge Request: This message is sent to the bank identified in the card information message, and consists of four components:

- The credit card number, identifying the account to be charged.
- The credit card's expiration date.
- The credit card's iCVV.
- The dollar amount to be charged.

Approval: This message consists of a *response code* determined by the bank, indicating its decision relating to this charge. The bank makes this decision after verifying the information supplied in the Charge Request message, and after performing additional checks such as matching the purchase to a known location of the customer. The most common response codes are the result of a simple approval decision (i.e. "approved" or "declined"), although a number of different codes (e.g. "Pick up card" if the card was reported lost or stolen, etc.) are also supported. We abstract this message as a single bit: whether or not the customer's account has been charged.

Figure 2.2: Eavesdropping



2.3 Attacks on the Protocol

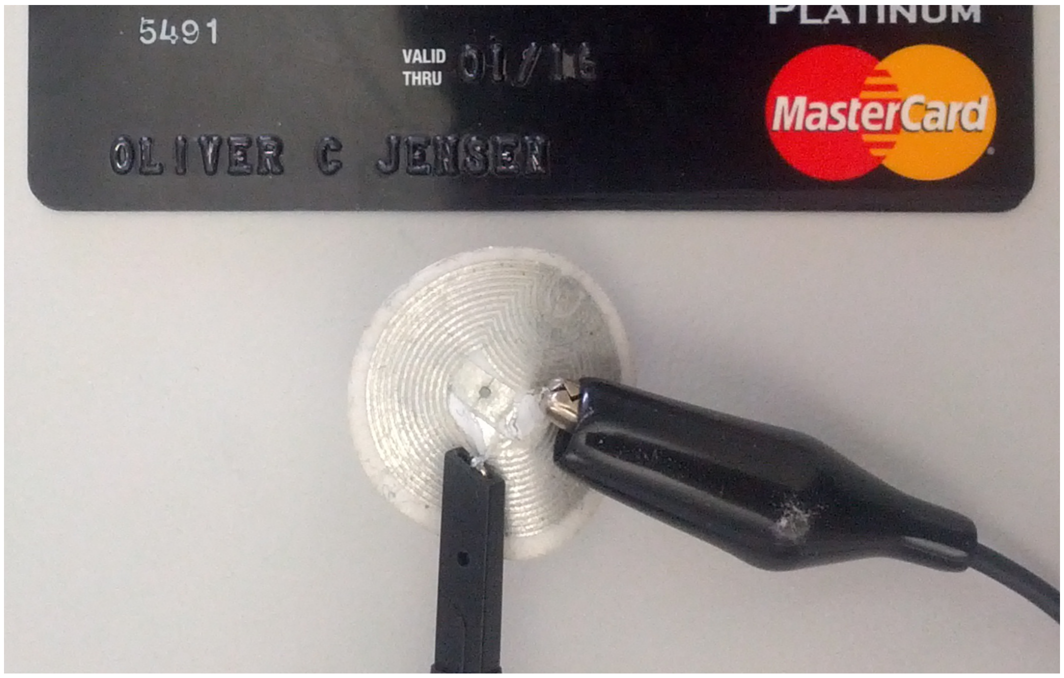
The Insecure CC Protocol is vulnerable to a number of attacks that can be performed by an external adversary. We classify these attacks into four broad categories: eavesdropping, skimming attacks, relay attacks, and attacks facilitated by compromised points of sale. In this section, we describe these attacks in detail.

2.3.1 Eavesdropping

The goal of an eavesdropper is to gain the victim's credit card number and expiration date. Eavesdropping is a passive attack, where the eavesdropper hears all communication between the point of sale and the credit card. (Communication between the bank and the point of sale is securely transmitted over the Internet.) An outline of this attack is shown in Figure 2.2.

We have demonstrated the feasibility of this attack by building a very low form-factor antenna capable of eavesdropping on NFC communications. Similar to the device described in [24], we modified a MIFARE NFC tag to

Figure 2.3: Eavesdropping Antenna (credit card for scale)



act as an antenna by disabling the chip at the center and attaching leads to either side of the coil where they connect to the chip. We then measure the voltage induced in the coil.

In Figure 2.3, we show our NFC eavesdropping antenna next to a credit card for scale. The resulting antenna is paper thin, flexible, approximately three centimeters in diameter and adhesive on one side. As such, it can easily be concealed within range of a point of sale.

By connecting such a makeshift antenna to a software-defined radio (available very inexpensively online) and recording the captured signal with a program like GNU Radio, an eavesdropper can record all transmissions that occur between the point of sale and credit cards. We have written a simple program to read these signal-recordings and decode the messages in each direction.

In the Insecure CC Protocol, an eavesdropper acquires the credit card number, expiration date, and the issuing bank name. (The eavesdropper also acquires the iCVV, but since this is used immediately in the current transaction, the acquired iCVV is of no value.)

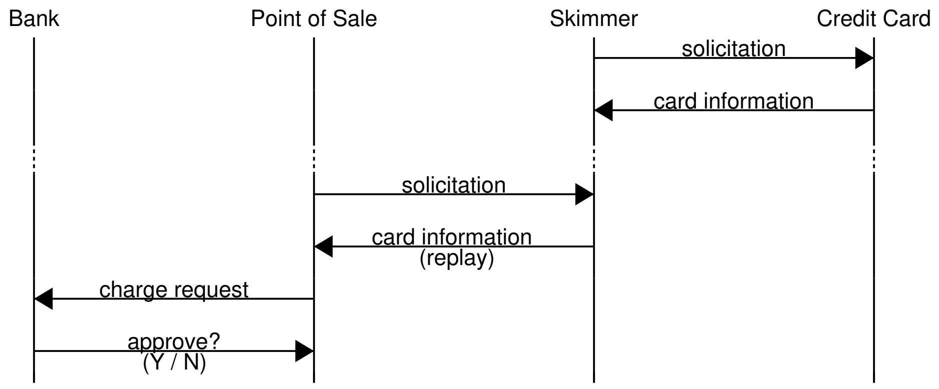
2.3.2 Skimming

The goal of a skimmer is to perform a purchase on behalf of the victim, without the victim's knowledge or consent. First, the skimmer masquerades as a point of sale to the victim's credit card, acquiring the credit card number, expiration date, issuing bank name, and the next iCVV. Subsequently, the skimmer masquerades as a credit card to a legitimate point of sale, making a purchase on behalf of the victim by replaying the skimmed credit card information and iCVV. An outline of this attack is shown in Figure 2.4.

This attack can be performed using a smart-phone with NFC capabilities. An Android application called *NFCProxy*[†] automates this attack and is freely available online. While *NFCProxy* is not listed in the Google Play store, it can be downloaded from SourceForge and installed on the phone in a matter of minutes.

[†]*NFCProxy*[18], presented at Defcon 20, can be downloaded here: `http://sourceforge.net/projects/nfcproxy/`

Figure 2.4: Skimming



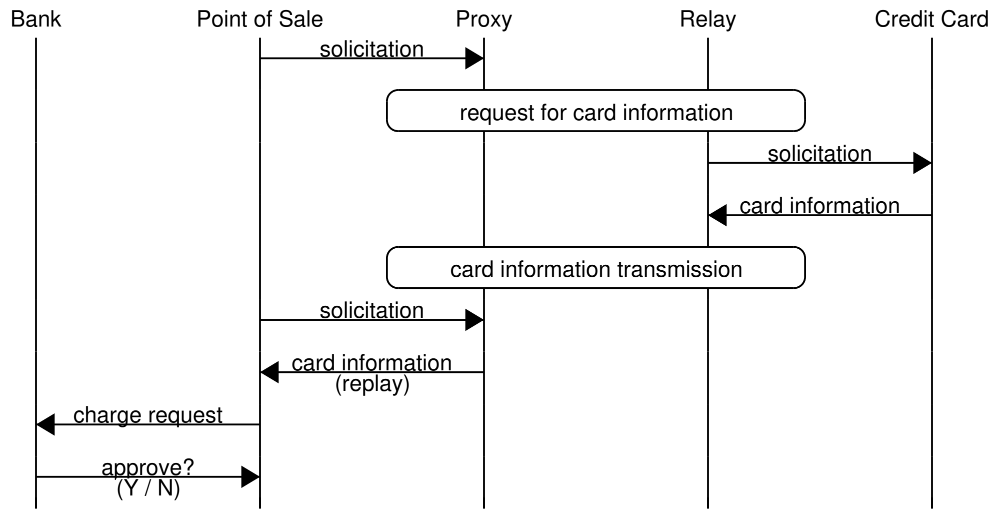
With *NFCProxy* running, the skimmer brings its phone briefly within range of an NFC credit card to acquire the credit card information and the iCVV. When the skimmer wishes to perform the illegitimate purchase on behalf of the victim, the skimmer moves its phone within range of a point of sale as though it were a credit card.

In the Insecure CC Protocol, a skimmer may perform a single purchase (limited by the lack of subsequent iCVVs). The skimmer must take care to perform this purchase before the credit card holder makes a purchase of their own, as this would invalidate the skimmed iCVV. (The skimmer also gains all information that an eavesdropper would learn.)

2.3.3 Relay Attacks

Much like the skimmer, the goal of the relay attacker is to perform purchases on behalf of the victim, without the victim's knowledge or consent.

Figure 2.5: Relay Attack



Multiple devices may be used to relay skimmed credit card information across a separate channel, effectively breaking the assumption of proximity built into NFC. This attack can also be performed using the *NFCProxy* Android application, described in Section 2.3.2. An outline of this attack is shown in Figure 2.5.

In implementation, a relay attack is very similar to a skimming attack, with the exception that the skimmer is separated into two entities, called "proxy" and "relay". These two entities are spatially disparate, but are connected through an out-of-band communication channel. The relay positions its phone near the victim's credit card, while the proxy approaches a point of sale. Whenever the proxy is ready to make a purchase, it sends a message to the relay requesting fresh values from the victim's credit card.

The relay skims the credit card and forwards the card's information back to the proxy, enabling the proxy to make a purchase on behalf of the victim. These messages may be transmitted over any communication channel, but are most easily sent over a wireless LAN.

In the Insecure CC Protocol, a proxy may perform multiple purchases if the relay remains in proximity of the victim's credit card, querying it for fresh iCVVs for every purchase.

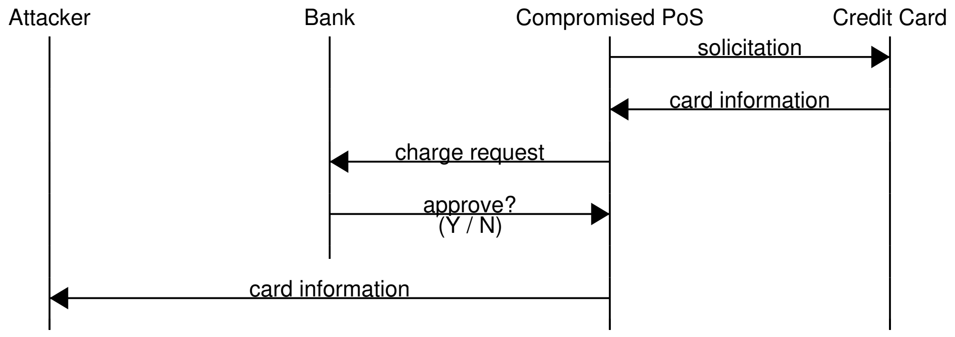
2.3.4 Attacks Facilitated by Compromised Points of Sale

The Insecure CC Protocol implicitly trusts the ability of a retailer to keep its data secure. By allowing persistent sensitive information (e.g. the credit card number and expiration date) to be transmitted to a device under the retailer's control, this protocol invites attacks on the retailer's own systems. We use this category of attacks to refer to any attack which involves the point of sale or merchant performing (possibly unintentional) actions leading to credit card theft[‡]. For example, a point of sale might be compromised and re-programmed to transmit credit card information to an attacker after every successful purchase. An outline of this attack is shown in Figure 2.6.

Point of sale compromise is far from a theoretical threat: these attacks

[‡] When discussing compromised points of sale, we consider only devices which correctly adhere to the protocol. We explicitly exclude what we term *malicious* points of sale, those which may perform arbitrary actions such as refusing to randomize random values, etc. Defending against a malicious point of sale in any payment setting is much more involved, and is discussed in Chapter 4.

Figure 2.6: Compromised Point of Sale



have become alarmingly commonplace, to the point that they have been covered by mainstream news sources including the New York Times [9] and the Wall Street Journal [36][42].

In November through December 2013, unauthorized access was gained to an estimated 40,000 points of sale used by U.S. retailer Target. This data breach was widely publicized, as it resulted in the compromise of *40 million* credit and debit cards, and other personal information of 70 million customers as reported in the Wall Street Journal [36].

More recently, in September 2014, Home Depot confirmed a similar data breach. According to subsequent investigations, it appears that the same malware as was used against Target was at the heart of the Home Depot breach. During this breach, attackers stole *56 million* credit and debit cards, also as reported in the Wall Street Journal [42].

Other recent victims of compromised points of sale include other retailers (e.g. Neiman Marcus in July through October of 2013, in which attackers stole an estimated 1.1 million credit and debit cards [9]), grocery stores (e.g. Supervalu in June through July of 2014, in which it is estimated that attackers stole “millions” of credit and debit cards [22]), as well as restaurants (e.g. P.F. Chang’s in September 2013 through June 2014, in which attackers stole an estimated 7 million credit and debit cards [27]).

Since 2014, headlines regarding data theft from compromised points of sale have diminished, not because the attacks have slowed, but because they have become so prevalent they are no longer headline-worthy.

Chapter 3

The Externally Secure CC Protocol

In Chapter 2 we described the protocol in use today for contactless credit card transactions and showed that this protocol is vulnerable to four types of external attacks. In this chapter, we show how to modify the protocol in order to defend against these attacks. This modified protocol is called the Externally Secure CC Protocol.*

3.1 Goals of the Protocol

The Externally Secure CC Protocol has two goals. First, this protocol should defend against the four categories of external attacks: eavesdropping, skimming attacks, relay attacks, and attacks facilitated by compromised points of sale. Second, we construct this protocol to minimize the computation which occurs on the card itself.

Contactless credit cards today are capable of only very limited computation and do not engage in any cryptographic operations. We hypothesize that the limited computational power on credit cards is to reduce costs to a

* Portions of this chapter have previously been published in [20]. While most of the contributions in this chapter are my own, acknowledgements are due to Mohamed Gouda for helping construct clear and concise protocol descriptions.

minimum: credit cards must be disposably cheap to manufacture so that they can be immediately replaced on suspicion of compromise or theft. As a result, we seek to minimize any additional computation required to be executed on the card.

3.2 Design of the Protocol

In designing the Externally Secure CC Protocol, we do not add any computationally expensive operations such as signature verification, or even hashing on the credit card: beyond its current function, we restrict a credit card to performing only basic arithmetic, indexing, XOR, and similarly inexpensive operations. We design the protocol using a process called *stepwise refinement*:

1. First, we define the protocol in terms of an abstract function H.

2. We then identify two desired properties of function H, namely **H1** and **H2**, and show that if function H satisfies these two properties, then the protocol is not vulnerable to the four classes of attacks described in Section 2.3.

3. Next, we define function H in terms of two abstract functions F and G. We then identify three properties, namely **F1**, **F2**, and **G1**, and prove that if function F satisfies properties **F1** and **F2** and function G satisfies property **G1**, then function H satisfies the two desired properties **H1** and **H2**.

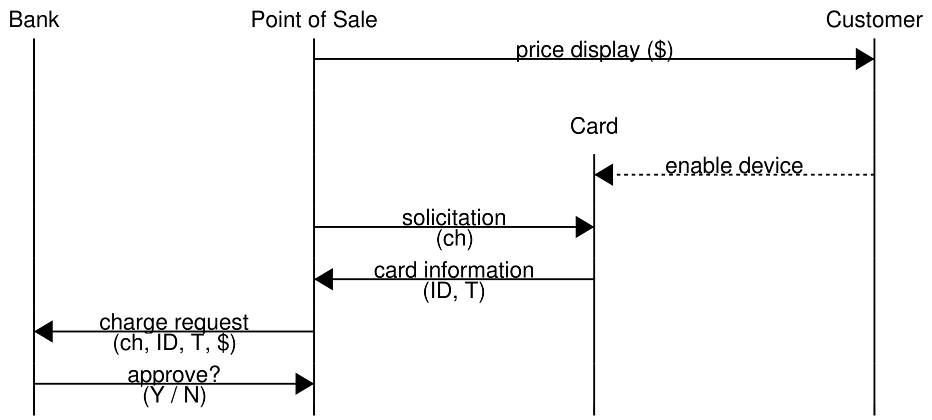
4. We then propose concrete implementations of functions F and G.

5. Finally, we argue that our proposed implementation of function F satisfies properties **F1** and **F2**, and that our proposed implementation of function G satisfies property **G1**.

In so doing, we provide a concrete implementation of the Externally Secure CC Protocol and show that it is not vulnerable to any of the four classes of outsider attacks described in Section 2.3.

The Externally Secure CC Protocol uses the same four messages that are used in the Insecure CC Protocol. However, the contents of these messages have become more involved. In particular, we incorporate a challenge-response mechanism in the Solicitation and Card Information messages. An outline of this protocol is shown in Figure 3.1.

Figure 3.1: Externally Secure CC Protocol



The four messages used in the Externally Secure CC Protocol are as follows:

Solicitation (ch): The point of sale sends a Solicitation message much like in the Insecure CC Protocol. However, this message now includes a random challenge *ch*. This challenge is used by the credit card when constructing its response.

Card Information (ID, T): The credit card's response consists of two primary components, ID and T. They are as follows:

- ***ID***, a Universally Unique Identifier [28], is used to identify the card without revealing the card's information to eavesdroppers or any other party. This value is computed by the credit card manufacturer and stored on the card as a constant.
- ***T = H(info, ch, iCVV)*** is used to authenticate the card's identity. This abstract function H will be defined later through stepwise refinement, but informally we can think of it as similar to a cryptographic hash function: it can be used to verify its arguments while leaking no information about them.

In so doing, *ID* provides identification and T provides authentication. In addition, this message is accompanied by the bank name B, as before, so that the point of sale may route its Charge Request to the proper entity.

Charge Request (*ID*, *T*, *ch*, *$*): After receiving the Card Information message, the point of sale issues a Charge Request to the credit card's issuing bank. The point of sale does not learn the card's private information, but simply forwards the card identification (ID) and authentication (T) to the bank B, specified in the Card Information message. The point of sale also includes the challenge ch (so that the bank can verify that T is a valid response to ch from card ID), as well as the dollar amount to be charged.

Approval (Y/N): The bank maintains an index of ID into its account database. When the bank receives a Charge Request message, it identifies the matching record as specified by ID, looking up $info_{bank}$ and $iCVV_{bank}$. It then calculates $T_{bank} = H(info_{bank},\ ch,\ iCVV_{bank})$ and verifies that $T = T_{bank}$. This step is equivalent to verifying the credit card information and iCVV in the Insecure CC Protocol, and results in a decision to accept or reject the charge request.

3.2.1 Desired Properties of Function *H*

We pit the Externally Secure CC Protocol against the attacks described in Section 2.3 and identify the two properties **H1** and **H2** needed from function H in order for these attacks to be thwarted.

Defending Against Eavesdropping

When eavesdropping on a transaction, the eavesdropper learns a valid (ch, ID, T, B) tuple. The challenge ch, the card identifier ID, and the bank name B are all public information, of no value to the eavesdropper. Only $T = H(\mathit{info},\ ch,\ iCVV)$ contains authentication information useful to an eavesdropper. In order to guarantee that no sensitive information is leaked, we require that function H satisfies the following property:

H1: If $iCVV$ is indistinguishable from random, then $H(\mathit{info},\ ch,\ iCVV)$ is indistinguishable from random.

If function H satisfies property **H1**, then the eavesdropper (ignorant of $iCVV$) cannot distinguish T from random and gains no useful information.

Defending Against Skimming

When sending a Solicitation message to a credit card, the skimmer includes a challenge ch_{skim}, which the card uses to construct its response. From the resulting Card Information message, the skimmer learns ID, $T_{skim} = H(\mathit{info}, ch_{skim}, iCVV)$ and B. When the skimmer attempts to perform a purchase using this information, it will be issued a challenge ch_{pos} by the point of sale. In order to prevent the skimmer from correctly responding to this challenge, we require that function H satisfies the following property:

H2: Given $H(\mathit{info},\ ch,\ iCVV)$, ch, and ch' such that $ch \neq ch'$, one cannot infer $H(\mathit{info},\ ch',\ iCVV)$.

If function H satisfies property **H2**, then the skimmer cannot use the value of T_{skim} in order to construct a response to the point of sale's challenge ch_{pos}, and thus cannot perform purchases on behalf of the credit card.

Defending Against Relay Attacks

The relay attack operates similarly to the skimming attack. When performing a relay attack, the relay provides challenge ch_{relay} to the card. As in the skimming attack, the relay learns ID, $T_{relay} = H(info,\ ch_{relay},\ iCVV)$, and B, and then transmits this information to the proxy. When the proxy attempts to perform a purchase, it is issued a challenge ch_{pos} by the point of sale.

Thus, if function H satisfies the property **H2**, then the proxy cannot use the value of T_{relay} in order to construct a valid response to the point of sale's challenge ch_{pos}, and as a result, cannot perform this purchase on behalf of the credit card.

Defending Against Compromised Points of Sale

A compromised point of sale will result in an attacker learning a valid $(ch,\ ID,\ T,\ B)$ tuple, which the point of sale requires in order to construct a Charge Request message. Note that this is the same information learned by the attacker in the eavesdropping case. As such, if the function H satisfies property **H1**, then the information learned by a compromised point of sale is of no value: no private information about the credit card is leaked, and the authentication token T cannot be reused since the $iCVV$ used in its construction is no longer valid after the charge has occurred.

In summary, in order to defend against the four classes of attacks described in Section 2.3, we require that the $iCVV$ be a pseudorandom value, and that function H upholds the following two properties:

H1: If $iCVV$ is indistinguishable from random, then $H(\mathit{info},\ ch,\ iCVV)$ is indistinguishable from random.

H2: Given $H(\mathit{info},\ ch,\ iCVV)$, ch, and ch' such that $ch \neq ch'$, one cannot infer $H(\mathit{info},\ ch',\ iCVV)$.

3.2.2 Implementing Function *H*

We find it convenient to define function H as a composition of functions F and G as follows:

$$H(\mathit{info},\ ch,\ iCVV) = F(x,\ iCVV) \text{ where } x = G(\mathit{info},\ ch)$$

Once defined in these terms, we posit the properties **H1** and **H2** in terms of functions F and G. We then show that if function F satisfies the properties **F1** and **F2**, and function G satisfies the property **G1**, then function H satisfies the desired properties **H1** and **H2**.

F1: If y is indistinguishable from random, then $F(x, y)$ is indistinguishable from random.

F2: Given only x, or given only y, one cannot infer $F(x, y)$.

G1: Given $G(u, v)$, v, and v' such that $v \neq v'$, one cannot infer $G(u, v')$ without knowledge of u.

Theorem 1. *If function* F *satisfies property* ***F1****, then function* H *satisfies property* ***H1****.*

Proof. Let $x = G(\mathit{info},\ ch)$ for any $\mathit{info},\ ch$. Then $H(\mathit{info},\ ch,\ iCVV) = F(x,\ iCVV)$. Thus if $y = iCVV$ is indistinguishable from random, then by property **F1**, $F(x, y) = H(\mathit{info},\ ch,\ iCVV)$ is indistinguishable from random. □

Theorem 2. *If function* F *satisfies property* ***F2****, and function* G *satisfies property* ***G1****, then function* H *satisfies property* ***H2****.*

Proof. Let $x = G(\mathit{info},\ ch)$ and $x' = G(\mathit{info},\ ch')$ for any info. By property **F2**, evaluating $H(\mathit{info},\ ch',\ iCVV)$ requires knowledge of $x' = G(\mathit{info},\ ch')$. However, by property **G1**, one cannot infer $G(\mathit{info},\ ch')$ without knowledge of info. Further, by property **H1**, no bits of *info* are leaked and thus *info* remains secret. Thus, given $H(\mathit{info},\ ch,\ iCVV)$, ch, and ch' such that $ch \neq ch'$, one cannot infer $H(\mathit{info},\ ch',\ iCVV)$. □

While *info* is considered secret, it cannot be used directly in function G. This is because while most of the data in *info* is unpredictable, many of the bits are not random. For example, in a typical credit card number, the first six digits are taken from the *Issuer Identification Number*, a public value assigned to the issuing bank. Similarly, the attacker can guess the bytes representing the expiration month and year from a very small set of possibilities. Furthermore, the credit card number and expiration date are transmitted as decimal values transliterated into hexadecimal (e.g. "4491" is transmitted as two bytes: `0x44` and `0x91`). As such, inferences may be made about the value of particular bits, since each such byte is drawn from only 154 possible values.

To resolve this problem, we use a keyed hash function known as an HMAC [26] to compute $h_k(\mathit{info})$, using a key k known only to the bank. As *info* does not change over the lifetime of the credit card, $h_k(\mathit{info})$ is stored as a constant on the card. As such, the computation necessary for hashing is not executed on the credit card, and the card requires no knowledge of the bank's secret key k. For convenience we will refer to the output of $h_k(\mathit{info})$ as the constant *khi*.

We propose implementations for functions G and F (and thus H) in Figure 3.2. Note that function G is defined as the function which returns those bits of a keyed HMAC $h_k(\mathit{info})$ for which the corresponding bit of *ch* was set to 1. Also note that function F is defined as `XOR`.

In the Insecure CC Protocol, *info* is composed of 96 bits, and $iCVV$ is composed of 32 bits. If we maintain these field-lengths in the Externally Secure CC Protocol, and use an HMAC function which also outputs 96 bits, then our implementation of F and G requires that *ch* must be a 96 bit value with 32 1's and 64 0's. More generally, our implementation requires that $h_k(\mathit{info})$ and *ch* have the same number of bits, and that the number of *1-bits* in *ch* is equal to the number of bits in $iCVV$.

Figure 3.2: Implementation of F, G and H

```
function G(info, ch):
    const khi = HMAC(bank_key, info)
    result = empty list of bits
    for each of the n bits of ch:
        if the n’th bit of ch is 1:
            append n’th bit of khi to result
    return result

function F(x, iCVV):
    return x XOR iCVV

function H(info, ch, iCVV):
    x = G(info, ch)
    return F(x, iCVV)
```

Note that the `XOR` operation satisfies properties **F1** and **F2**, so our implementation of function F (trivially) satisfies them as well.

The output of $G(\mathit{info},\ \mathit{ch})$ is composed of a number of bits of $\mathit{khi} = h_k(\mathit{info})$ selected by the challenge ch. If $\mathit{ch}_1 \neq \mathit{ch}_2$, one cannot infer $G(\mathit{khi},\ \mathit{ch}_2)$ from $G(\mathit{khi},\ \mathit{ch}_1)$ without knowledge of khi, because the results are composed of different bits of khi selected by the challenges ch_1 and ch_2. These bits are indistinguishable from random to any party without knowledge of info and the bank's secret key k. These bits are then masked by the $iCVV$ and as such are not learned by any party. As a result, our implementation of function G satisfies the property **G1**.

Chapter 4

The Secure CC Protocol

The Externally Secure CC Protocol described in Chapter 3 focuses on defending the retailer and customer from malicious external parties, termed "outsiders". In this chapter, we examine the problems posed by malicious retailers and focus on how to secure contactless credit cards against them. As will be described shortly, attacks by malicious retailers are particularly pernicious, as they are less easily identified as fraud. Even when these attacks are detected, the resolutions are not always simple.*

4.1 Goals of the Protocol

Recall that when making a payment, the customer first views the price about to be charged on the screen of the retailer's point of sale. Using this information, the customer makes its one and only decision: whether to allow the payment protocol to occur, or not. The underlying assumption that the customer makes is that the price displayed on the screen is equal to the price

* Portions of this chapter have previously been published in [21]. While most of the contributions in this chapter are my own, acknowledgements are due to Tyler O'Meara for suggesting the Transparent Bridge attack in the context of contactless credit cards, and to Mohamed Gouda for helping construct clear and concise protocol descriptions.

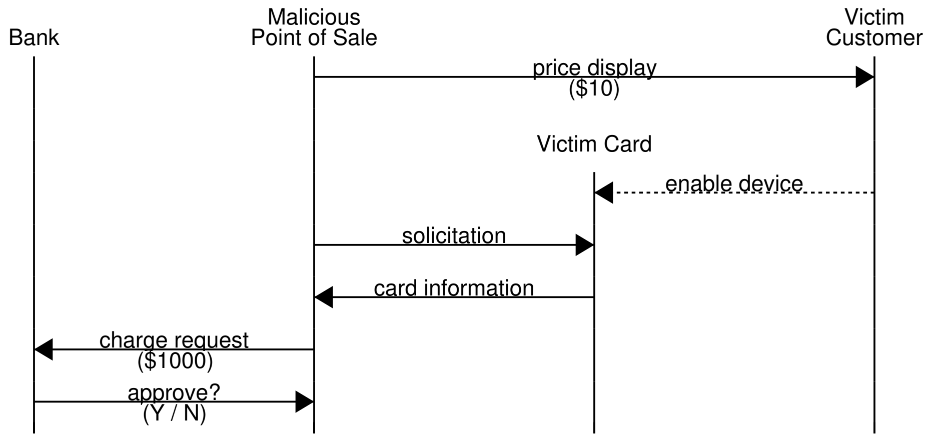
which will be charged to the credit card. This need not be the case. The information displayed on the screen is merely an assurance in the informal sense: the numbers displayed to the customer *should* reflect the dollar amount which will subsequently be sent with the Charge Request message to the bank, but there is no mechanism in place to require this. Two attacks emerge as a result. The goal of the Secure CC Protocol presented in this chapter is to extend the protection provided by the Externally Secure CC Protocol to defend against these two additional attacks.

4.1.1 The Over-charge Attack

An Over-charge attack, illustrated in Figure 4.1, is characterized by the malicious point of sale displaying one price to the customer and then sending a higher price to the bank (in the Charge Request message of the CC Protocols). As a result, the customer authorizes and expects to be charged one amount, but is instead charged an arbitrarily higher amount. Since the customer is uninvolved in the protocol besides the initial step of allowing the protocol to occur, there is no mechanism ensuring that the price displayed to the customer matches the price that the (malicious) point of sale sends to the bank.

Should a customer become aware of an over-charge when reviewing the credit card's monthly statement, the customer may file a charge-back request with his bank, nullifying the payment as fraudulent. As a result, while the amount by which the customer may be overcharged is unconstrained by the protocol, it should be relatively small for the attack to ultimately be successful.

Figure 4.1: Over-charge Attack



For example, it is easy to notice a gas station charge for \$500.00 instead of \$21.87 on a monthly statement, and the resulting investigation would be uncomplicated. However, should the struggling business choose to increase charges by 5%, the resulting gas station charge of \$22.96 could very easily be overlooked. Even were it to be noticed, the victim customer may have difficulty proving the discrepancy.

4.1.2 The Transparent Bridge Attack

A more interesting attack is described by Drimer and Murdoch [6]. The authors consider a man-in-the-middle attack, perpetrated by a malicious retailer and an accomplice with specialized equipment. This attack involves four parties: a victim customer, a malicious retailer, a malicious customer, and a victim retailer. The malicious retailer and the malicious customer collude to perform this attack.

The malicious customer is issued with a special device, capable of relaying all messages it receives from a point of sale to the malicious retailer in real time. Similarly, it can relay any responses it receives from the malicious retailer back to this point of sale. As a result, the malicious customer and malicious retailer can together form a bridge between the victim credit card and the victim retailer's point of sale. The attack is illustrated in Figure 4.2 and runs as follows:

1. First, the victim customer attempts to make a relatively inexpensive purchase from the malicious retailer. Simultaneously, the malicious customer prepares to make a relatively expensive purchase from a victim retailer.

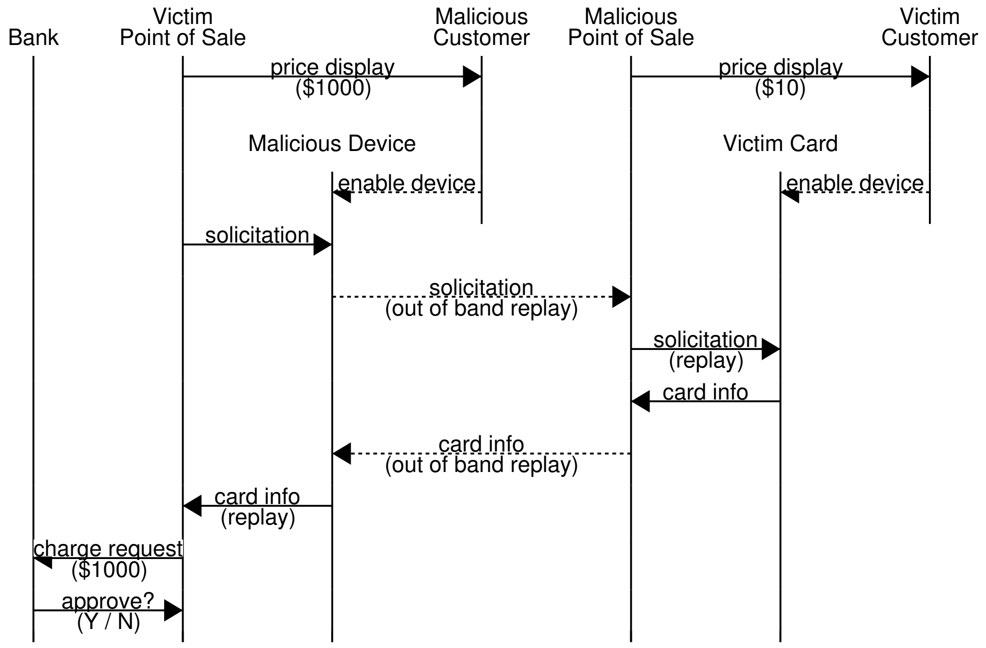
2. The victim retailer's point of sale issues a Solicitation message to the malicious customer, who relays it to the malicious retailer.

3. The malicious retailer then forwards this Solicitation message to the victim credit card.

4. The victim credit card responds with a Card Information message to the malicious point of sale, who relays this message to the malicious customer.

5. The malicious customer forwards this Card Information message to the victim retailer's point of sale.

Figure 4.2: Transparent Bridge Attack



6. The victim retailer's point of sale issues a Charge Request message to the victim credit card's bank, charging the victim customer for the expensive purchase.

In this attack, all messages are transparently relayed between the victim retailer's point of sale and the victim customer's credit card. As a result, the victim customer expects to be charged for an inexpensive purchase at the malicious retailer, but is instead charged for an expensive purchase at the victim retailer. The malicious retailer loses the inexpensive sale, but acquires the merchandise from an expensive purchase in exchange.

This Transparent Bridge attack is particularly interesting because the malicious parties leave no trace with either of the victims: to the victim customer, there is only a record of an expensive purchase at the victim retailer, and to the victim retailer, there is only the customer record of the victim customer. The amount which can be successfully stolen by the malicious retailer is unconstrained, and needs not evade notice: if the discrepancy is noticed and the victim customer files a charge-back request, it will be against the victim retailer (and not the malicious retailer). As such, detected or not, it is one of the two victims that will be left facing the bill, making the Transparent Bridge attack significantly more dangerous than the Over-charge attack described earlier.

Drimer et al. propose a defense against this attack in the context of EMV "chip" cards. However, this solution is not applicable to a contactless credit card protocol.

4.2 Design of the Protocol

Since the attacks described in the previous section (allowing a malicious retailer to exploit a customer) are tied to the retailer's ability to display one price and charge another, our proposed defense against these attacks is built around removing this ability. If using a credit card implemented on a smart phone, the phone's interface provides an additional communication channel between the customer and the credit card. In this case we refer to the card as a "virtual" credit card. The communication channel between the smart phone and the customer can be harnessed to allow the customer to participate in the credit card protocol beyond simply allowing it to occur.

The Externally Secure CC Protocol described in Chapter 3 defines a function H, proves several of its properties, and uses it to defend against external attacks such as skimming and eavesdropping. We note that each property required of this function H is a property enjoyed by common cryptographic hash functions, such as those in the *SHA* family. As such, using a hash function instead of the derived function H does not reduce the security of the Secure CC Protocol. In the context of virtual credit cards, the computational cost of executing a hash function is no longer a concern.

We remind the reader of the following terms:

ch: a fresh, randomly generated challenge value, chosen by the point of sale.

info: the credit card's payment information, consisting of the credit card number and expiration date.

ID: a UUID, uniquely identifying an individual credit card without revealing any information about *info*. This identifier is stored as a constant on the credit card.

iCVV: an unpredictable value freshly generated by the credit card for each transaction (the issuing bank can generate the same sequence of values).

B: the name of the issuing bank, used for the purpose of routing transactions as before.

T: a charge token, to authenticate a single transaction.

The Secure CC Protocol, operating between a point of sale and a virtual credit card, is illustrated in Figure 4.3 and proceeds as follows:

1. The point of sale displays a price *$d* on its screen, inviting the customer to bring a credit card within NFC range.

2. The point of sale sends a Solicitation message to the credit card, including a fresh random challenge *ch* and the price to be charged *$s*. (Recall that if the point of sale is honest, $\$s = \$d$.)

3. The virtual credit card displays the price *$s* on the smart phone to the customer, who can choose to accept or reject it. Rejecting the price aborts the protocol here.

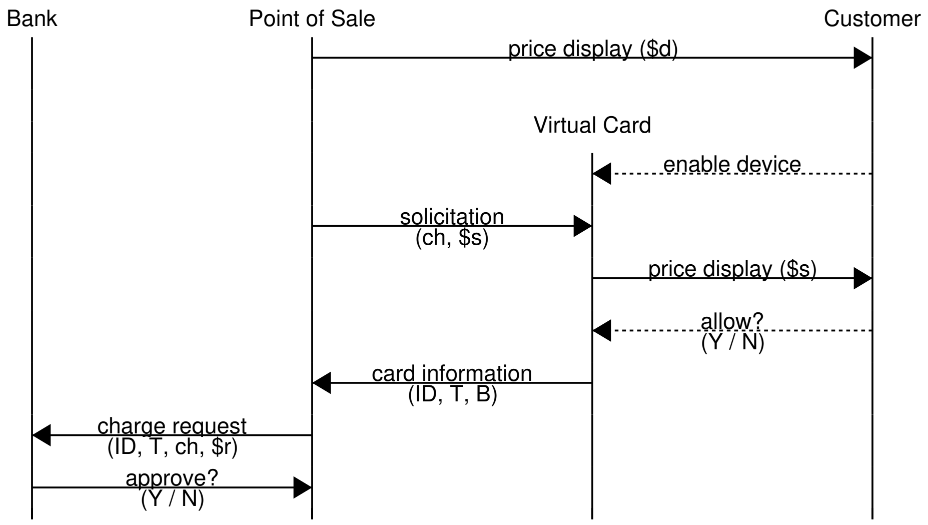
4. If the price is accepted by the customer, the virtual credit card calculates

   $$T = H(info,\ ch,\ \$s,\ iCVV)$$

   and responds to the point of sale with a Card Information message consisting of $(ID,\ T,\ B)$.

5. The point of sale sends a charge request message to the issuing bank (identified by B) consisting of $(ID,\ T,\ ch,\ \$r)$. (Again, if the point of sale is honest, $\$r = \$d$)

Figure 4.3: Secure CC Protocol



6. The bank uses ID to look up $info_{bank}$ and then calculates $iCVV_{bank}$. It then uses the ch and $\$r$ supplied in the Charge Request message to calculate

$$T_{bank} = H(info_{bank},\ ch,\ \$r,\ iCVV_{bank})$$

If $T \neq T_{\text{bank}}$, the bank will decline the charge, otherwise it approves the charge for $\$r$.

When using a physical credit card instead of a virtual one, no communication channel exists between the card and the customer. As a result, the steps above in which the card displays the price sent with the Solicitation message ($\$s$) to the customer and waits for the customer to allow the protocol to proceed cannot occur. Instead, a physical credit card must implicitly assume successful authorization from the customer, effectively skipping step 3. As a result, while not providing protections from malicious retailers to physical credit cards, the protocol maintains backwards compatibility with no loss of functionality or security against malicious outsiders.

We note that a naive implementation of the protocol above might require excessively long timeouts between the point of sale sending its solicitation message and receiving the response. Should long timeouts not be desired, a simple solution would be for the point of sale to send periodic solicitations (with new challenges). The virtual credit card, upon receiving permission from the customer, could then cache this authorization and respond immediately to the subsequent solicitation. Besides noting this particular case, we emphasize that issues such as these are implementation details, the decisions for which are best left to those implementing the protocol.

4.2.1 Defending Against the Over-charge Attack

The Secure CC Protocol prevents the Over-charge attack against customers using a virtual credit card. In step 3, the customer verifies that $\$d = \$s$ through visual comparison. Due to the inclusion of $\$s$ in the hash when generating token T, we gain the assurance that for any charge accepted by the bank, $\$s = \$r$.

Thus, through a transitive argument, the customer can be assured that for any successful charge, $\$d = \$r$. Should the malicious retailer attempt to issue a charge request with some $\$r \neq \$d$, then $T_{bank} \neq T$ and the charge will be declined by the bank.

4.2.2 Defending Against the Transparent Bridge Attack

The Secure CC Protocol makes no attempt to prevent this attack from occurring. Instead, it removes the economic incentive of performing such an attack against customers using virtual credit cards.

In the Transparent Bridge attack, the malicious retailer loses the sale paid by the victim customer, in return for acquiring the purchase made by the malicious customer. In order for the Transparent Bridge attack to be viable, the malicious actors must have something to gain: the value of the malicious customer's purchase must be greater than the value of the victim customer's purchase. When the Secure CC Protocol is used, one of two scenarios occurs:

1. The price associated with the malicious customer's purchase differs from (i.e. is greater than) the price of the victim customer's purchase. The victim customer compares the price displayed by the point of sale and the price displayed by the virtual credit card. The would-be victim customer immediately detects the attack and aborts the transaction.

2. The price associated with the malicious customer's purchase is equal to the price of the victim customer's purchase. The victim customer does not detect this attack, and allows the transaction to occur. The end result: the victim customer paid for the price of what it received, and the victim retailer received the price of what it sold.

As a result, there is no longer any incentive to carrying out this attack, as the only successful instance results in all parties getting paid exactly as much as they would had they been honest.

Chapter 5

Virtual Credit Cards and Electronic Wallets

When describing the Secure CC Protocol in Chapter 4, we referred to "virtual credit cards". In this chapter, we describe them more formally.

5.1 Virtual Credit Cards

Many consumer smart phones today support the ability to communicate over the NFC channel. Indeed, as described in Section 2.3, this ability can be used by skimmers and relay attackers in order to perform fraudulent purchases. Moreover, this ability can be used by an authorized party to perform non-fraudulent purchases.

A virtual credit card is an application on an NFC-capable smart phone which emulates a credit card in a CC Protocol by receiving Solicitation messages and responding with Card Information messages. To emulate a physical credit card, the virtual card needs to have access to the following information:

- The credit card number
- The expiration date
- The next iCVV

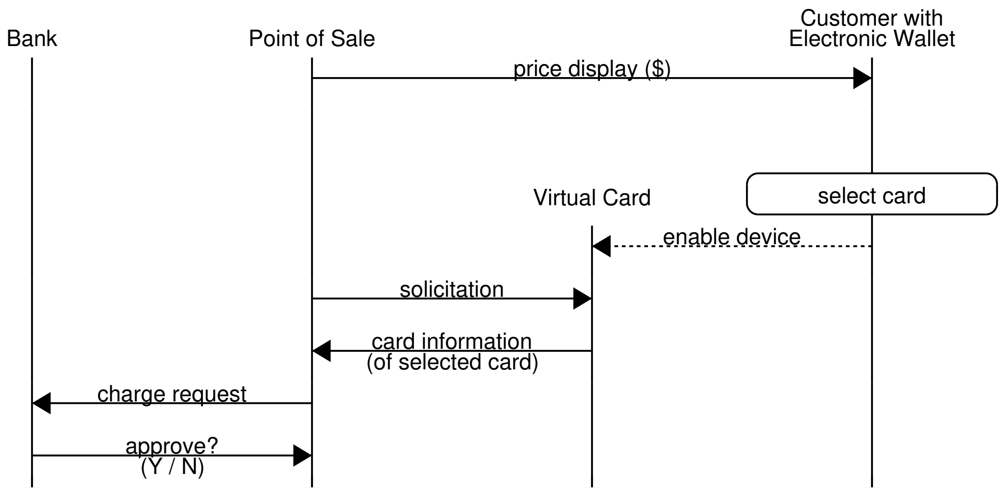
- The issuing bank name

Note that the credit card number, expiration date, and bank name can be input to the virtual credit card by the cardholder. However, in order to be able to generate valid iCVVs, a virtual credit card needs to have access to the seed from which iCVVs are generated. This value is not known to the cardholder, and as such, a virtual credit card cannot normally be created without the cooperation of the physical card's issuing bank.

5.2 Electronic Wallets

A logical extension of the virtual credit card is the storage and management of multiple virtual cards in a single unit. Applications such as Android Pay and Apple Wallet exemplify this idea, allowing a customer to select the desired credit card in the application, following which the phone emulates the selected credit card. We refer to a collection of virtual credit cards and accompanying software (e.g. the interface for allowing the customer to select a card, password protection, etc.) as an "Electronic Wallet". Figure 5.1 shows how an Electronic Wallet can be used in the CC protocols discussed so far (namely, the Insecure CC Protocol, the Externally Secure CC Protocol, and the Secure CC Protocol), enabling the customer to participate in the protocol as described in the Secure CC Protocol in Chapter 4.

Figure 5.1: Use of an Electronic Wallet in the CC Protocols



Electronic Wallets are advantageous for several reasons:

- Convenience: Electronic Wallets allow a customer to carry an essentially unlimited number of credit cards on their person without taking up extra space.

- Security: Electronic Wallets protect the customer from skimming and relay attacks, because Electronic Wallets are typically programmed not to respond to solicitations without customer consent. (Recall that physical credit cards will always respond to Solicitation messages, regardless of the customer's intention.)

- Interface: Electronic Wallets can provide a rich interface between the customer and the virtual credit card. This interface allows the customer to participate in the protocol.

Chapter 6

Protecting Customer Privacy: Unlinkability

Retailers today enjoy the ability to link multiple purchases to an individual customer by identifying multiple purchases made using the same credit card. We argue that this is problematic from a privacy standpoint and present the prevention of this ability as a goal for a new contactless credit card protocol.

6.1 Linkability and Unlinkability

All CC Protocols discussed so far provide retailers with a property of credit card payments known as *linkability*: retailers are able to link distinct purchases that are made using the same credit card, in order to build purchasing profiles on their customers. Customer purchasing profiles are valuable to retailers, because they allow for more targeted and effective marketing, and can be sold to interested third parties.

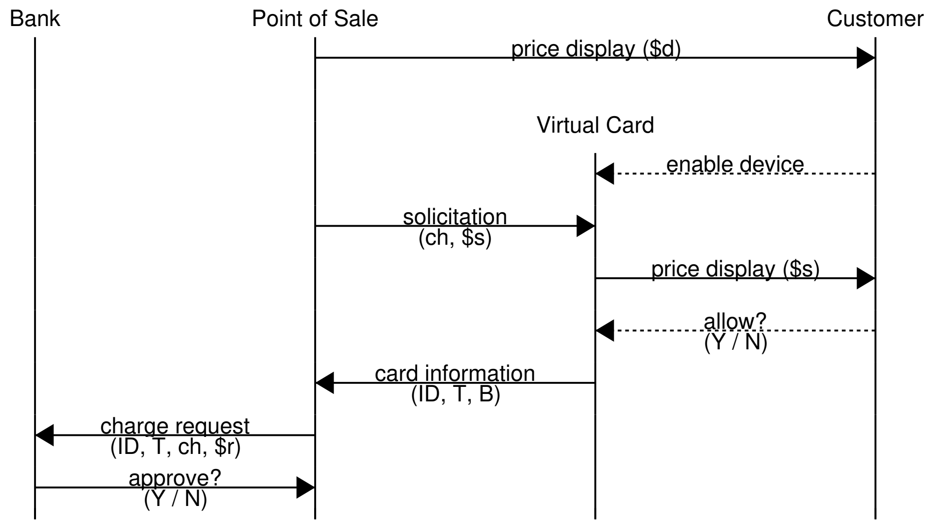
Recall that in the Insecure CC Protocol described in Chapter 2 (and in widespread use today), the credit card discloses its card number, expiration date, and iCVV to the point of sale with every purchase. Therefore, a retailer can link purchases that are made using the same credit card simply by comparing the card numbers of these purchases. Note that any credit card protocol which gives the retailer access to the card number is linkable by definition.

These purchasing profiles can be unpleasant to the customer from a privacy standpoint: purchasing habits can reveal sensitive and personal information [8]. Furthermore, a customer cannot opt-out of this profiling except by avoiding the use of credit cards altogether.

With this in mind, we consider how to provide credit card payments with the converse property: *unlinkability*. Informally, if credit card purchases are unlinkable, then retailers cannot use them to construct purchasing profiles efficiently.

While allowing the retailer access to the card number is sufficient to undermine unlinkability, it is not necessary. For example, consider the Secure CC Protocol, presented in Chapter 4 and shown in Figure 6.1 for convenience. In this protocol, the credit card number is not disclosed, and all authentication data (contained within the value T) is indistinguishable from random to the retailer. However, the charge token contains a constant card identifier ID, required by the bank in order to identify the customer for which T must be verified. No two credit cards have the same ID, and ID does not change, so the retailer can simply link purchases using ID instead of the card number. More generally, if the protocol includes a message received by the retailer from which the customer's identity can be inferred, then the protocol cannot provide unlinkability.

Figure 6.1: Secure CC Protocol



While we wish to conceal the customer's identity from the retailer, it must be disclosed to the bank in order for a charge to be processed. As such, the problem reduces to one where we wish the contents of a message generated by a credit card (the message source) to be opaque to the point of sale (the carrier), but not to the bank (the final destination). A natural approach to this problem is to use cryptography.

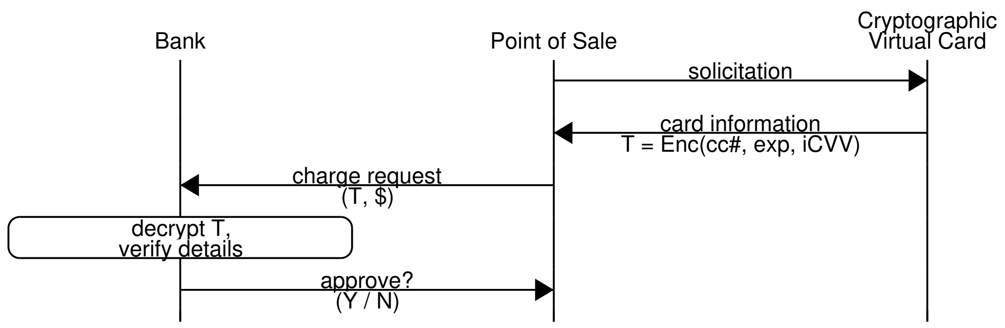
6.2 The Simple Unlinkable Protocol

In considering a cryptographic approach to creating an Unlinkable Credit Card Protocol, let us first consider the simpler approach of using symmetric cryptography. In a cryptographic scheme employing symmetric cryptography, the original sender (the credit card) and the final destination (the bank) share a secret key used for both encryption and decryption. While symmetric cryptography can provide confidentiality of a message from the card to the bank, it cannot provide unlinkability. This is because the message received by the bank must include some method for determining which of its customer's keys the bank should use to decrypt the message. The retailer may then simply use this key identifier to link purchases.

Let us next consider using asymmetric (or "public key") cryptography, a system in which anybody (having access only to publicly available information) may encrypt a message that only the intended recipient may decrypt. Asymmetric cryptography allows the recipient to use a single key in order to decrypt messages originating from any party, sidestepping the key identifier issue above.

We thus present the Simple Unlinkable Wallet Protocol, employing asymmetric encryption to maintain unlinkability. This protocol is illustrated in Figure 6.2, and operates as follows:

Figure 6.2: Simple Unlinkable Wallet Protocol



1. The point of sale sends a Solicitation message to the customer's (virtual) credit card.

2. The customer's card encrypts its credit card information (consisting of the credit card number, expiration date, and iCVV) using the bank's public key, and sends this data (along with the bank name) as a Card Information message.

3. The point of sale receives the Card Information message, appends the price to charge, and sends this data to the bank named in the Card Information message.

4. The bank decrypts the Card Information data to recover the customer's credit card number, expiration date, and iCVV. It then continues exactly as before in the Insecure CC Protocol described in Chapter 2, responding to the point of sale with an Approval message.

In this protocol, the Card Information message consists of an encrypted blob, and the bank name in plaintext. The point of sale learns nothing about the customer besides the name of the card's issuing bank. As a result, this protocol is unlinkable.

Barring any other requirements, asymmetric cryptography provides sufficient primitives for constructing an unlinkable payment protocol. However, asymmetric cryptography requires comparatively large (2048 bit) keys to be secure and results in ciphertexts at least as long as the key. For reasons to be discussed shortly in Section 7.1, our goals for an Unlinkable Wallet Protocol require much shorter message lengths. As a result, using asymmetric cryptography in a secure manner is incompatible with our goals for a new Unlinkable Wallet Protocol.

Chapter 7

Additional Goals for the Unlinkable Wallet Protocol

The Secure CC Protocol presented in Chapter 4 is a significant improvement over the status quo (described in Chapter 2). However, the Secure CC Protocol neglects to provide two properties (in addition to Unlinkability) which we find desirable in a payment protocol: the ability to use existing point of sale infrastructure, and the ability to use any credit card. In this chapter, we define and justify these goals. In Chapter 8 we use these three properties to guide the construction of a new protocol, which we call the Unlinkable Wallet Protocol. This new protocol will defend against malicious outsiders an retailers, while simultaneously satisfying unlinkability, allowing for the use of existing point of sale infrastructure and any credit card.

7.1 Use Existing Point of Sale Infrastructure

While constructing new and better protocols is an effective way to defend against attacks, requiring modification or updates to widely established infrastructure imposes a significant barrier to the adoption of this new protocol. For example, in order to implement the Secure CC Protocol described in Chapter 4, every point of sale must be modified or replaced. In order to sidestep this barrier, we consider the protocol in use today (the Insecure CC Protocol in Chapter 2) from the perspective of a point of sale.

A point of sale may do rudimentary checks on the structure of the Card Information messages that it receives, but it relies on the bank in order to validate the data in these messages. The Insecure CC Protocol, from the perspective of the point of sale, is illustrated in Figure 7.1 and operates as follows:

1. The point of sale issues a Solicitation message over the NFC channel. This message contains no payload of interest.

2. The point of sale receives a Card Information message over the NFC channel. This message consists of the following fields:

    - A 16-digit card number
    - A 4-digit expiration date
    - An 8-digit iCVV

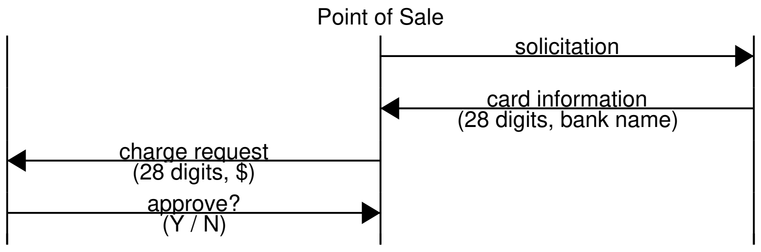
    In practice, this is an opaque* 28-digit numeric value (16 + 4 + 8). This data is accompanied by the bank name.

* There are some constraints on this value. For example, an expiration date of "8321" does not need to be verified by a bank to be declared invalid. Similarly, there are constraints on credit card numbers as well. These have little effect on our approach, so we ignore them for simplicity.

3. After receiving the Card Information message, the point of sale issues a Charge Request message to the bank specified by the bank name field in the Card Information message. The Charge Request message consists of the card's credit card number, expiration date, iCVV, and the price to charge. In practice, this repeats the same 28-digit numeric value received in the Card Information message, accompanied by the price to charge.

4. Later, the point of sale receives an Approval message in response to the Charge Request message. The Approval message consists of the bank's decision on whether to accept the charge, as described in Section 2.2.

Figure 7.1: Insecure CC Protocol, Point of Sale Behavior



A key observation of the point of sale's behavior is that it consists of relaying data specified by the card to a destination *also specified by the card.* While the point of sale may interpret the 28-digit data content of the Card Information message as consisting of three fields describing a credit card, the customer may instead use these fields to encode an arbitrary 28-digit message, provided that the recipient of the Charge Request message decodes it accordingly.

We leverage this observation in the construction of the Unlinkable Wallet Protocol, conforming to existing point of sale behavior by using this 28-digit message as a carrier for the data we wish to transmit. In so doing, we ensure that the point of sale in the Unlinkable Wallet Protocol remains identical to its counterpart in the Insecure CC Protocol.

7.2 Use of Any Credit Card

Few credit cards today support contactless transactions natively. As a result, the majority of credit cards available to customers cannot generate valid iCVVs. The Unlinkable Wallet Protocol should allow the use of any credit card, including those for which no valid iCVVs exist.

This matter is complicated by our goal of using existing point of sale infrastructure: current point of sale devices require an iCVV as part of an NFC Card Information message. To get around this, we can leverage the architecture of credit cards themselves.

Each credit card is a collection of independent interfaces. Common interfaces seen on credit cards today include:

- **Visual**: the data which can be visually read from the card
- **Magstripe**: the data encoded on the magnetic strip
- **NFC**: the data and computation involved in contactless payments
- **Chip**: the data and computation involved in chip payments

These interfaces operate independently: a transaction involves the use of a single interface as negotiated by the customer and the retailer. The bank accepts a valid Charge Request over any interface supported by the credit card. While some cards support a different set of interfaces than others, an important observation is that *every* credit card supports the visual interface.

Making use of NFC communication is desirable for a new credit card protocol, because a smart phone supporting NFC communication enables the customer to send and receive arbitrary messages. However, for a charge to be successful, the bank must receive a Charge Request pertaining to a *non-NFC* interface due to the lack of valid iCVVs associated with non-contactless credit cards.

These two seemingly contradictory requirements suggest the construction of a tunneled protocol: if we place an entity between the point of sale and the bank, this entity may receive an NFC Charge Request from the point of sale, convert it into a Charge Request of another interface (for example, the visual interface), and send this converted Charge Request to the bank.

Chapter 8

Design of the Unlinkable Wallet Protocol

In this chapter, we present our design for the Unlinkable Wallet Protocol, which achieves our three primary goals: supporting unlinkable payments and allowing the use of any credit card, while using existing point of sale infrastructure.

First, we base our design of the Unlinkable Wallet Protocol on the following two assumptions:

- The retailer is not malicious
- The Wallet Application is connected to the Internet at all times

Later, as we refine the protocol, we erode these assumptions to yield a secure and resilient protocol.

The Unlinkable Wallet Protocol is designed solely for use with Electronic Wallets and cannot be used with physical credit cards. Thus, the first step for a customer is to download the Wallet Application and initialize a virtual credit card by registering a physical card into the Wallet Application.

The card registration process is simple. The customer launches the Wallet Application and enters the credit card information for the card to be registered. This information consists of the cardholder name, card number, expiration date, and billing address. The Wallet Application then sends this data in a Registration message to a new principal: the Wallet Server. This Registration message is transmitted securely over the Internet.

The customer may then use any card registered through the Wallet Application to make unlinkable transactions.

8.1 Basic Unlinkable Wallet Protocol

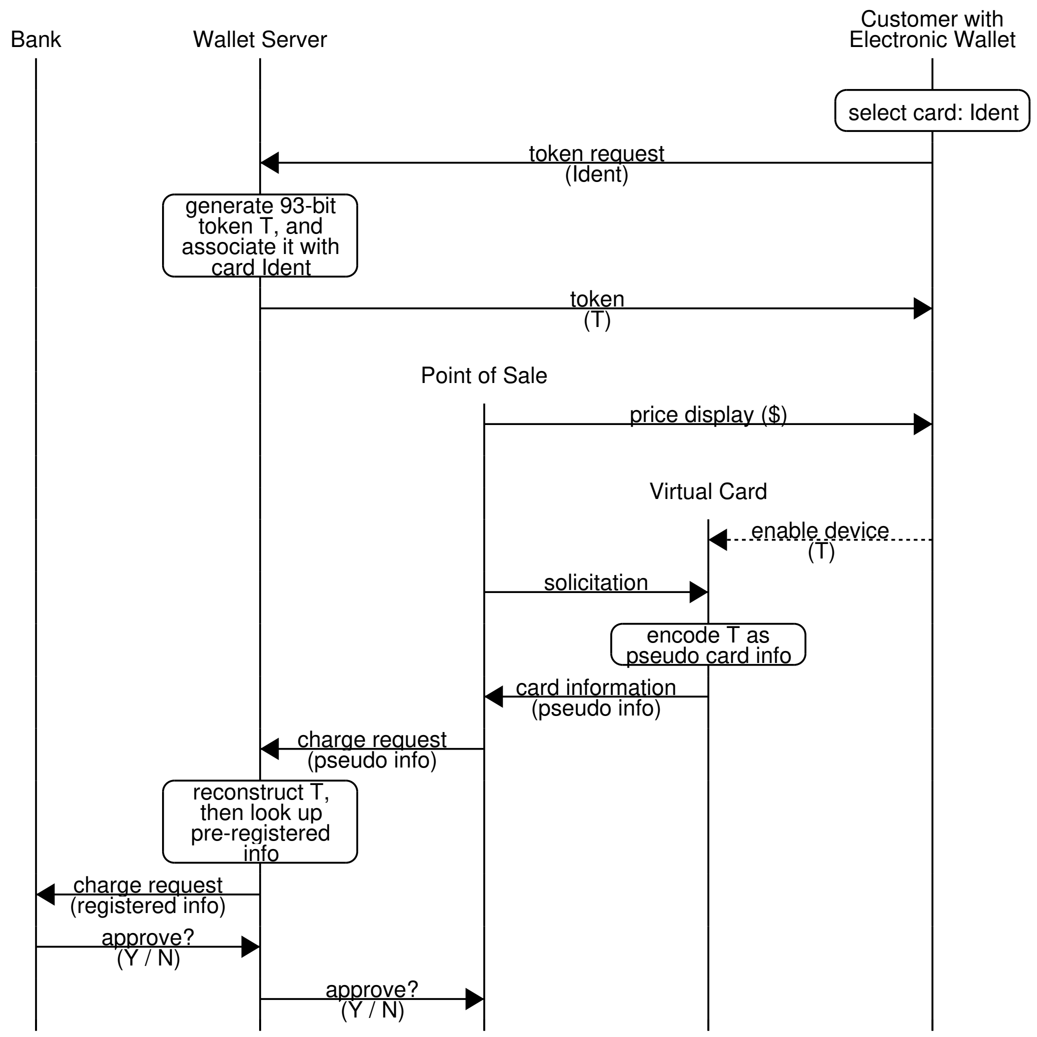
When the Wallet Server receives a Registration message, it stores the card information and associates a unique card identifier, denoted *Ident*, with this record. In the basic protocol, the Wallet Server responds to the Registration message with this identifier *Ident*. The Wallet Application stores this value securely, associating it with the credit card being registered. The protocol, illustrated in Figure 8.1, operates as follows:

1. The customer selects a credit card in the Wallet Application.

2. The Wallet Application sends a Token Request message to the Wallet Server. This message consists of the card identifier *Ident* associated with the selected card, and is sent securely over the Internet. Note that this message should be authenticated to prevent a customer from requesting a token to a different customer's card.

3. The Wallet Server then generates a random 93-bit token T and associates

Figure 8.1: Basic Unlinkable Wallet Protocol


Bank
Wallet Server
Customer with Electronic Wallet
select card: Ident
token request (Ident)
generate 93-bit token T, and associate it with card Ident
token (T)
Point of Sale
price display ($)
Virtual Card
enable device (T)
solicitation
encode T as pseudo card info
card information (pseudo info)
charge request (pseudo info)
reconstruct T, then look up pre-registered info
charge request (registered info)
approve? (Y / N)
approve? (Y / N)

it with the card identified by *Ident*. The Wallet Server responds to the Wallet Application with a Token message containing T.

4. The point of sale displays the price to charge ($) on its screen.

5. The Wallet Application now enables the virtual credit card, initializing it with token T. The virtual credit card begins listening for Solicitation messages.

6. The point of sale sends a Solicitation message to the virtual credit card over the NFC channel.

7. The virtual credit card converts token T into a 28-digit number k (note that 28 digits is sufficient to store a 93 bit value: $log_{10}(2^{93}) \approx 27.995$). It then responds with a Card Information message. This message has the following fields:

   - **Pseudo Card number:** the first 16 digits of k
   - **Pseudo Expiration date:** the subsequent 4 digits of k
   - **Pseudo iCVV:** the remaining 8 digits of k
   - **Pseudo Bank name:** the Wallet Server

8. The point of sale constructs an Charge Request message from the (pseudo) card number, expiration date, iCVV, and the price it wishes to charge. This message is sent to the bank named in the Card Information message. As a result, the Charge Request message is directed to the Wallet Server and *not* an actual bank. Note that from the perspective of the point of sale, the Wallet Server appears to be a bank like any other.

9. The Wallet Server reconstructs k from the Charge Request message and computes the 93-bit token it represents. The Wallet Server then searches

its database for this token, in order to identify the card used in this transaction. If the token is not found, the Wallet Server sends a "declined" Approval message to the point of sale and aborts the protocol. Otherwise, the stored card details are retrieved from the Wallet Server's database. The Wallet Server then invalidates the token and sends a *visual* Charge Request to the card's bank with the following fields:

- Cardholder name
- Card number
- Expiration date
- Billing address

Note that unlike the Card Information message sent by the Wallet Application, this data reflects the actual credit card information, acquired by the Wallet Server during the card registration.

10. The bank receives the *visual* Charge Request from the Wallet Server and processes this transaction as normal. It then responds to the Wallet Server with an Approval message indicating whether the charge has been accepted.

11. The Wallet Server forwards the bank's Approval message to the point of sale.

Note that in this protocol, the Wallet Server has a dual role: to the point of sale, the Wallet Server appears to be a bank, while to the bank, the Wallet Server appears to be a point of sale.

Note too that the Wallet Application needs to have a ready connection to the Internet. Each transaction requires secure communication with the Wallet Server in order to receive the next token T. Should a connection to the Internet be unavailable to the Wallet Application, after using T, the customer is unable to perform subsequent purchases with this virtual credit card until a Token Request exchange can take place.

This protocol is unlinkable, because the information received by the point of sale consists solely of a random value, accompanied by a bank name identifying the Wallet Server. Any credit card can be used in this protocol, because the bank receives *visual* Charge Request messages, and all credit cards support the visual interface. Finally, this protocol uses existing infrastructure in that the behaviors of the point of sale and the bank are unchanged from those of the Insecure CC Protocol.

8.2 Protecting against Malicious Retailers

The Basic Unlinkable Wallet Protocol is vulnerable to attacks by malicious retailers, as described in Section 4.1. In this section, we describe the Secure Unlinkable Wallet Protocol to defend against these attacks while continuing to support unlinkable transactions.

This new protocol is very similar to the Basic Unlinkable Wallet Protocol described in Section 8.1. In the Basic Unlinkable Wallet Protocol, a 93-bit token was used to identify a virtual credit card to the Wallet Server. In the Secure Unlinkable Wallet Protocol, this token consists of only 80 bits, leaving 13 bits with which to bind a price to the token. In binding a price to a given token, we provide the same defense against malicious retailer attacks as the Secure CC Protocol described in Chapter 4.

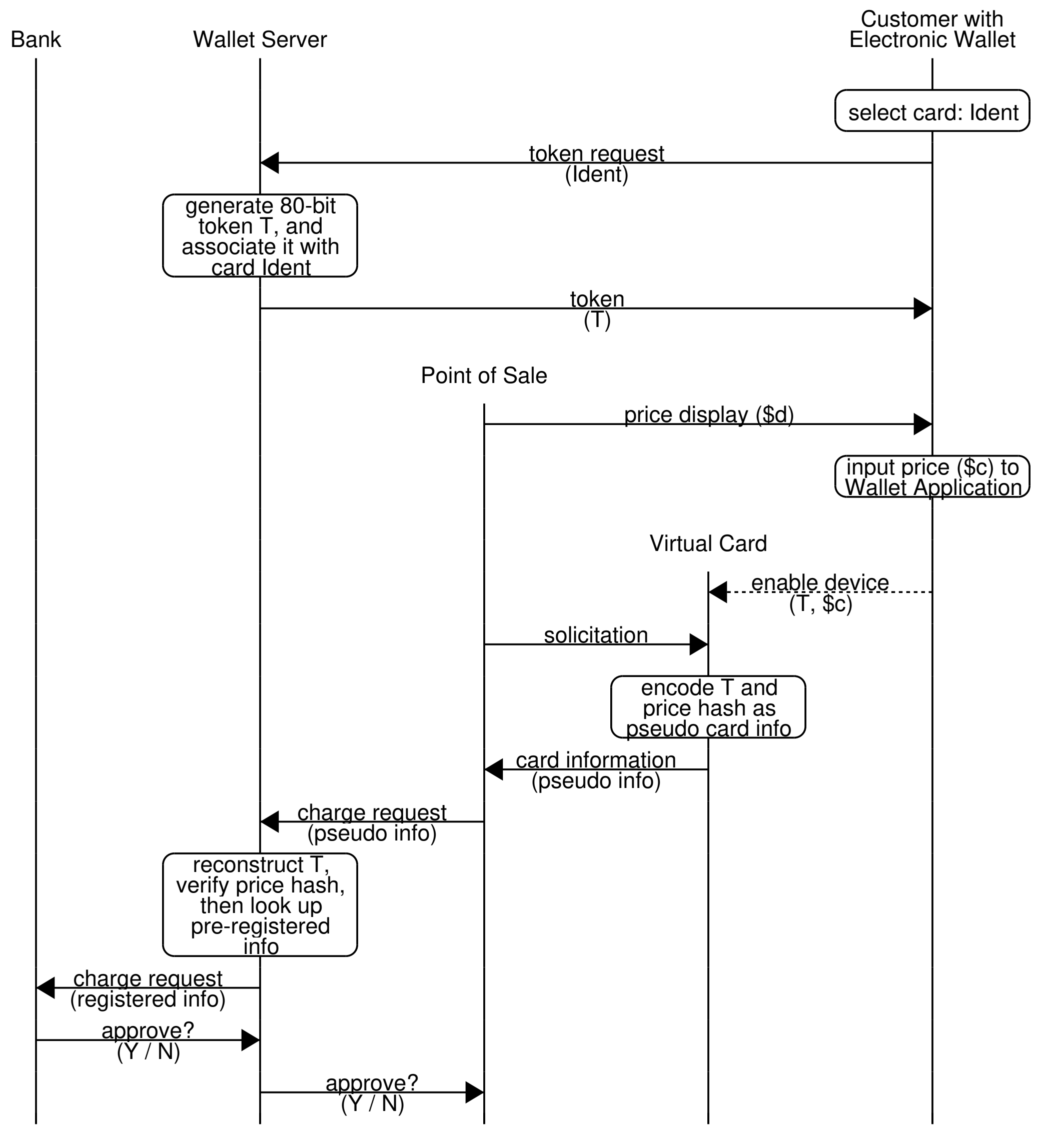
When the Wallet Server receives a Registration message, it stores the card information and associates a card identifier *Ident* with this record. The Wallet Server also generates a secret key *SK* associated with this card. In the Secure Unlinkable Wallet Protocol, the Wallet Server responds to the Registration message with this identifier *Ident* and the key *SK*. The Wallet Application stores these values securely, associating them with the credit card being registered. The protocol, illustrated in Figure 8.2, operates as follows:

1. The customer selects a credit card in the Wallet Application.

2. The Wallet Application sends a Token Request message to the Wallet Server. This message consists of the card identifier *Ident* associated with the selected card, and is sent securely over the Internet. Note that this message should be authenticated to prevent a customer from requesting a token to a different customer's card.

3. The Wallet Server then generates a random 80-bit token T, and associates it with the card identified by *Ident*. The Wallet Server responds to the Wallet Application with a Token message containing T.

Figure 8.2: Secure Unlinkable Wallet Protocol


Bank
Wallet Server
Customer with Electronic Wallet
select card: Ident
token request (Ident)
generate 80-bit token T, and associate it with card Ident
token (T)
Point of Sale
price display ($d)
input price ($c) to Wallet Application
Virtual Card
enable device (T, $c)
solicitation
encode T and price hash as pseudo card info
card information (pseudo info)
charge request (pseudo info)
reconstruct T, verify price hash, then look up pre-registered info
charge request (registered info)
approve? (Y / N)
approve? (Y / N)

4. The point of sale displays the price to charge ($\$d$) on its screen.

5. The customer enters the price to be charged ($\$c$) into the Wallet Appli-

cation.

6. The Wallet Application now enables the virtual credit card, initializing it with token T and price $\$c$. The virtual credit card begins listening for Solicitation messages.

7. The point of sale sends a Solicitation message to the virtual credit card over the NFC channel.

8. The virtual credit card calculates the HMAC of token T combined with price $\$c$, keyed with key SK, and selects the first 13 bits of this hash. It then combines the 80-bit token T with this 13-bit hash to acquire a 93-bit value. Finally, it converts this 93-bit value into a 28-digit number k, and responds with a pseudo Card Information message as before.

9. The point of sale constructs a Charge Request message from the (pseudo) card number, expiration date, iCVV, and the price it wishes to charge. This message is sent to the bank named in the Card Information message. As a result, the Charge Request message is directed to the Wallet Server and *not* an actual bank. Note that from the perspective of the point of sale, the Wallet Server appears to be a bank like any other.

10. The Wallet Server reconstructs k from the Charge Request message and computes the 80-bit token and 13-bit price hash that it represents. The Wallet Server then searches its database for the token to identify the card used in this transaction. If the token is not found, the Wallet Server

sends a “declined” Approval message to the point of sale and aborts the protocol. Otherwise, the secret key SK is retrieved from the Wallet Server’s database, and the Wallet Server calculates its own version of the price hash using the price indicated by the point of sale in the Charge Request message.

If the Wallet Server’s price hash does not match the price hash in the Charge Request message, the Wallet Server sends a “declined” Approval message to the point of sale and aborts the protocol. Otherwise, the stored card information is retrieved from the Wallet Server’s database. The Wallet Server invalidates the token, and sends a *visual* Charge Request to the card’s bank with the following fields:

- Cardholder name
- Card number
- Expiration date
- Billing address

Note that unlike the Card Information message sent by the Wallet Application, this data reflects the actual credit card information, acquired by the Wallet Server during the card registration.

11. The bank receives the *visual* Charge Request from the Wallet Server, and processes this transaction as normal. The bank then responds to the Wallet Server with an Approval message indicating whether the charge has been accepted.

12. The Wallet Server forwards the bank's Approval message to the point of sale.

In the Secure Unlinkable Wallet Protocol, the Card Information messages are bound to the price entered by the customer: a malicious retailer, lacking knowledge of SK, cannot generate a replacement price hash should it charge a different price in the Charge Request message. This protocol is also unlinkable: all 93 bits represented by the (pseudo) Card Information values are indistinguishable from random to the retailer.

However, like in the Basic Unlinkable Wallet Protocol, the Wallet Application needs to have a ready connection to the Internet. Once a token has been used, a subsequent token must be acquired from the Wallet Server in order to make a subsequent purchase.

8.3 Tolerating Lack of Internet Access

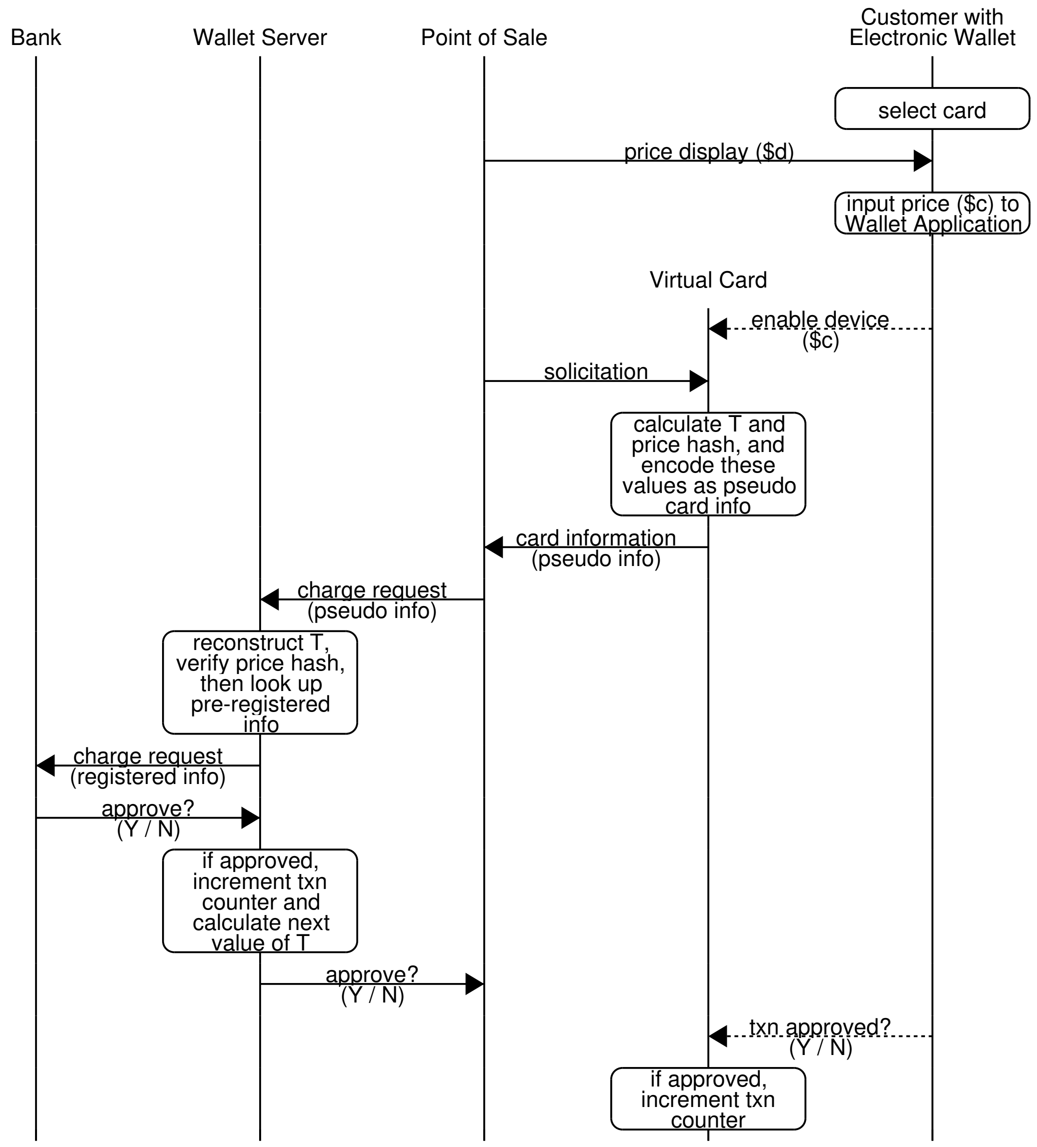
Both the Basic Unlinkable Wallet Protocol described in Section 8.1 and the Secure Unlinkable Wallet Protocol described in Section 8.2 suffer from requiring access to the Internet when a transaction is being made. Each transaction requires the virtual credit card to acquire a new token T from the Wallet Server. As such, temporary lack of Internet access (e.g. paying for parking in an underground garage, or making purchases while in a foreign country) significantly hampers the ability for customers to use their credit cards. This can be alleviated somewhat by requesting multiple tokens at once, but remains a fundamental limitation of the protocols.

In this third protocol (termed simply the Unlinkable Wallet Protocol), we solve this problem by generating the single-use tokens within the Wallet Application. Tokens are generated deterministically and using information available to both the Wallet Server and Wallet Application. As a result, both the Wallet Application and Wallet Server are able to independently generate the same tokens for a given card without needing to communicate (over the Internet). Communication between the Wallet Application and the Wallet Server is no longer necessary besides during card registration (and during the resolution of any synchronization problems).

As before, when the Wallet Server receives a Registration message, it stores the card information and associates a card identifier *Ident* with this record. The Wallet Server also generates a secret key *SK* associated with this card. In the Unlinkable Wallet Protocol, the Wallet Server responds to the Registration message with this identifier *Ident* and the key *SK*. The Wallet Application stores these values securely, associating them with the credit card being registered.

Both the Wallet Application and the Wallet Server initialize a transaction counter for this card and set it to zero. In addition, the Wallet Server calculates the initial token T by calculating the keyed HMAC of the transaction counter (currently at 0) keyed with *SK*. The Wallet Server stores this initial value of T with the credit card record. The protocol, illustrated in Figure 8.3, operates as follows:

Figure 8.3: Unlinkable Wallet Protocol


Bank
Wallet Server
Point of Sale
Customer with Electronic Wallet
select card
price display ($d)
input price ($c) to Wallet Application
Virtual Card
enable device ($c)
solicitation
calculate T and price hash, and encode these values as pseudo card info
card information (pseudo info)
charge request (pseudo info)
reconstruct T, verify price hash, then look up pre-registered info
charge request (registered info)
approve? (Y / N)
if approved, increment txn counter and calculate next value of T
approve? (Y / N)
txn approved? (Y / N)
if approved, increment txn counter

1. The customer selects a credit card in the Wallet Application.

2. The point of sale displays the price to charge ($\$d$) on its screen.

3. The customer enters the price to be charged ($\$c$) into the Wallet Application.

4. The Wallet Application now enables the virtual credit card, initializing it with price $\$c$. The virtual credit card begins listening for Solicitation messages.

5. The point of sale sends a Solicitation message to the virtual credit card over the NFC channel.

6. The virtual credit card calculates token T by calculating the HMAC of the transaction counter keyed with SK and selecting the first 80 bits. The virtual credit card then proceeds as before. It calculates the price hash by combining T with price $\$c$ and calculating the HMAC of this value with key SK, then selecting the first 13 bits. It then combines the 80-bit token T with this 13-bit price hash to acquire a 93-bit value. Finally, it converts this 93-bit value into a 28-digit number k and responds with a pseudo Card Information message as before.

7. The point of sale constructs a Charge Request message from the (pseudo) card number, expiration date, iCVV, and the price it wishes to charge. This message is sent to the bank named in the Card Information message. As a result, the Charge Request message is directed to the Wallet Server and *not* an actual bank. Note that from the perspective of the point of sale, the Wallet Server appears to be a bank like any other.

8. The Wallet Server reconstructs k from the Charge Request message and computes the 80-bit token and 13-bit price hash that it represents. The Wallet Server then searches its database for the token to identify the card used in this transaction. If exactly one token is not found, the Wallet Server sends a "declined" Approval message to the point of sale, and aborts the protocol. Otherwise, the secret key SK is retrieved from the Wallet Server's database, and the Wallet Server calculates its own version of the price hash using the price indicated by the point of sale in the Charge Request message.

   If the Wallet Server's price hash does not match the price hash in the Charge Request message, the Wallet Server sends a "declined" Approval message to the point of sale and aborts the protocol. Otherwise, the stored card information is retrieved from the Wallet Server's database. The Wallet Server then sends a *visual* Charge Request to the card's bank with the following fields:

   - Cardholder name
   - Card number
   - Expiration date
   - Billing address

   Note that unlike the Card Information message sent by the Wallet Application, this data reflects the actual credit card information acquired by the Wallet Server during the card registration.

9. The bank receives the *visual* Charge Request from the Wallet Server and processes this transaction as normal. The bank then responds to the Wallet Server with an Approval message indicating whether the charge has been accepted.

10. The Wallet Server examines the Approval message from the bank. If the charge was accepted, it increments the transaction counter associated with the virtual credit card and recalculates the next expected value of T. The Wallet Server then forwards the bank's Approval message to the point of sale.

11. The Wallet Application prompts the customer as to whether the charge was accepted by the bank. Note that the Approval message is not sent to the Wallet Application or virtual credit card as part of the protocol, and so the Wallet Application must rely on the customer to enter this data. If the charge was approved, then the virtual credit card increments its transaction counter.

The Unlinkable Wallet Protocol is identical to the Secure Unlinkable Protocol, with the exception that the tokens T are generated independently in both the Wallet Application and the Wallet Server. As a result of this change, there are two potential failure cases which must be considered.

First, since the protocol relies on the customer manually maintaining the transaction counter within a virtual credit card, we must account for potential customer error. If a customer fails to increment the transaction counter on a successful purchase (or incorrectly increments it on a failed purchase), then the Wallet Server and the virtual credit card will not agree on subsequent values of T. The transaction counter can be re-synchronized by way of an authenticated message exchange with the Wallet Server, sent securely over the Internet. Customer error in maintaining a virtual credit card's internal transaction counter thus results in an inoperative virtual credit card until such a time as the Wallet Application regains temporary access to the Internet.

Second, since the Wallet Server can no longer exercise full control over token generation, we need to consider the possibility of token collisions. A token collision occurs when two different virtual credit cards calculate the same token T around the same time. This occurs when $H(SK_1,\ txn\_ctr_1) = H(SK_2,\ txn\_ctr_2)$. In this case, both virtual cards become inoperative until *at least one* of the corresponding Wallet Applications regains access to the Internet and negotiates a new transaction counter*. Note however that the probability of such a collision is negligibly small and will be examined below in Chapter 9.

* We refer to the value as a "transaction counter" because we increment it with every transaction successful, but there is no requirement that it reflect an accurate number of transactions. It need only be a value which (a) changes on every successful transaction, and (b) changes in such a way that both the Wallet Application and Wallet Server can independently agree on the new value. As such, resolution to a collision consists simply of incrementing the transaction counter of whichever card is first to reconnect to the Internet, then synchronizing this value with the Wallet Application.

Chapter 9

Analysis of the Unlinkable Wallet Protocol

The Unlinkable Wallet Protocol has several tangible benefits, as described in the previous chapter. However, collision mitigation strategies are inconvenient, so we consider the expected frequency of such events. In this chapter, we analyze the features of the Unlinkable Wallet Protocol which make it attractive, as well as the statistical frequency of collisions.

9.1 Security Analysis of the Protocol

The Unlinkable Wallet Protocol is secure against malicious outsiders. Simply by using an Electronic Wallet, the customer is protected from skimming and relay attacks "for free", because the phone must be unlocked and ready to transmit in order to respond to a Solicitation message. The Unlinkable Wallet Protocol provides protection against eavesdroppers and compromised points of sale, because each message consists of two single-use tokens which lose all value after the transaction takes place.

Due to the inclusion of the price validation hash (the last 13 bits of the 93-bit message), the Unlinkable Wallet Protocol defends against malicious retailers by binding the charge token to a price confirmed by the customer.

Furthermore, the Unlinkable Wallet Protocol provides the customer with the privacy property of *unlinkability* from the retailer. In this protocol, retailers lose the ability to use credit card numbers as a method to correlate purchases and track customer purchasing profiles through purchase records.

In [20] and [21], we considered unlinkability an explicit non-goal: for deployment, the protocol required extensive cooperation from retailers who currently enjoy this property. The Unlinkable Wallet Protocol requires no such cooperation from retailers or the payment industry, which means that acquiescing to retailers' wishes is not necessary.

Note that the information correlating an individual credit card to an individual purchase is not lost: it is now learned by the Wallet Server instead. Indeed, the Wallet Server finds itself in a very privileged position: by inserting itself between retailers and banks, it appropriates knowledge from retailers regarding transaction correlation and appropriates knowledge from banks regarding transaction locations*.

Even so, the Unlinkable Wallet Protocol protects customer privacy by separating knowledge of itemized purchases and correlation between purchases. It is potentially valuable for an entity to be able to answer the questions *"where does the customer spend money? What does the customer spend money on?"*

* This appears to be consistent with the operation of Electronic Wallet applications like Android Pay and Apple Wallet: Android Pay has announced partnerships with some banks, which results in your credit card's purchases through Android Pay continuing to earn rewards normally offered by your credit card (e.g. 5 points per dollar spent on gas, 3 points per dollar spent on groceries, etc.). This suggests that, lacking a partnership with Electronic Wallet providers, banks lose information on the nature and location of purchases.

Currently, banks have complete information on where customers spend their money and retailers have complete information on what their customers purchase (at that retailer). The Unlinkable Wallet Protocol breaks part of this information away from both parties: all purchases appear to the bank as being made out to the Wallet Server, and all purchases appear to the retailer as being made by a unique credit card.

9.2 Collision Analysis of the Protocol

Every transaction processed by the Wallet Server service updates a card's token T to a new 80-bit value. If at any time two cards possess the same token, the Wallet Server can no longer process purchases for either card, as a Charge Request message cannot identify which card to charge. While hash functions such as those in the SHA-2 family are collision resistant, this does not mean that collisions cannot occur. Furthermore, truncating the token down to 80 bits greatly increases the probability of collisions.

We wrote a simulator to model Wallet Server behavior in order to ensure that messages contained sufficient information for the protocol to succeed and to monitor for collisions. We seeded it with 10 million credit cards and processed several million transactions. During the simulation, no collisions were observed.

Modeling this behavior mathematically, we note that the Birthday Paradox plays a role in determining the number of credit cards that a database can contain before expecting a token collision: the expected number of cards registered before observing a collision is approximately $\sqrt{\frac{\pi}{2} \cdot 2^{80}} \approx 1.4 \cdot 10^{12}$. However, registering a credit card is by definition an event in which the Wallet Application is connected to the Internet (and thus to the Wallet Server). As such, collisions during registration are not a concern, as a differing key SK or counter value can simply be chosen and synchronized between the Wallet and the Server. Provided that there are fewer than 2^{79} (approximately $6.04 * 10^{24}$) credit cards registered, card registration is computationally easy (expecting no more than two attempts without causing a collision).

A credit card transaction is then modeled by selecting a single token and replacing it. In such a transaction, the Birthday Paradox is not relevant: intuitively, updating the token for card number 83 cannot cause the tokens for cards 12 and 41 to collide with each other. As a result, given a database of n Account Hashes, the expected number of transactions before a collision occurs is $\frac{2^{80}}{n}$. Using the 10 million cards from our simulation, this implies that the expected number of transactions before a collision occurs is $\frac{2^{80}}{10{,}000{,}000} \approx 1.2 \cdot 10^{17}$, or approximately 120 million billion transactions. It is thus hardly surprising that no collisions were observed in simulation.

As such, the inconvenience of rendering both cards inoperative until at least one of them reconnects to the Internet is not particularly onerous.

Chapter 10

Summary and Future Work

In this chapter, we summarize the main contributions of this dissertation, and describe a number of future research directions suggested by this work.

10.1 Summary

The contactless credit card protocol in use today neglects even basic protections. In Chapter 2, we examine the protocol in detail and show that it is vulnerable to four categories of outsider attacks: eavesdropping, skimming attacks, relay attacks, and attacks facilitated by compromised points of sale. The current contactless credit card protocol is likewise vulnerable to attacks perpetrated by malicious retailers as discussed in Chapter 4, providing no protection to customers from retailers wishing to take advantage of them. In addition, the contactless credit card protocol poses significant privacy risks to customers as discussed in Chapter 6, allowing retailers to correlate purchases made using the same credit card.

In response, we design a family of three replacement contactless credit card protocols to guard against the deficiencies in the current protocol:

1. The Externally Secure CC Protocol
2. The Secure CC Protocol
3. The Unlinkable Wallet Protocol

The Externally Secure CC Protocol guards against the outsider attacks listed above. The Secure CC Protocol guards against both outsiders and malicious retailers. The Unlinkable Wallet Protocol guards against both outsiders and malicious retailers while protecting customer privacy from curious retailers and banks.

The Externally Secure CC Protocol is described in Chapter 3. We take particular note of the fact that in the existing contactless credit card protocol, no cryptographic operations are in use. We hypothesize that this is to keep manufacturing costs of these cards down so that they can be inexpensively replaced on suspicion of theft or misuse. As such, the Externally Secure CC Protocol seeks not to increase computational requirements on the credit card. We make use of the existing iCVV generation algorithm as our source of entropy, and eschew using any other expensive operations (such as hash functions).

We design the Externally Secure CC Protocol to incorporate a challenge-response structure and base the response-generation function around an abstract function H. We determine that while a cryptographic hash function would suffice, such functions are comparatively expensive and provide properties unneeded for this application.

Thus, we begin by determining what properties are required of function H to prevent the outsider attacks described. We then decompose H into simpler sub-functions F and G, and map the determined properties of H onto corresponding properties of F and G. We provide inexpensive implementations of functions F and G using only indexing and `XOR`, and show that these implementations satisfy the necessary properties of F and G. In so doing, we provide a concrete implementation of function H which uses minimal computation and prove that its use as the response-generation function in our challenge-response protocol is sufficient to defend against outsider attacks.

The Secure CC Protocol is described in Chapter 4. We begin by exploring possibilities for retailers to behave maliciously and describe two attacks which a retailer may carry out: an Over-charge attack, and a Transparent Bridge attack. Both attacks take advantage of the fact that a credit card's response to a solicitation (in both the existing contactless credit card protocol, and the Externally Secure CC Protocol) may be used to issue an arbitrary charge to the credit card.

The Over-charge attack is quite simple: the retailer displays the correct price to the customer, but then issues a Charge Request accompanied by an arbitrarily higher price to the bank. Should the fraud be detected on the victim customer's monthly statement, the customer may declare the charge as fraudulent to his bank and request a charge-back. However, small increases are unlikely to be noticed.

The Transparent Bridge attack is more insidious: the malicious retailer's point of sale and a malicious accomplice's specialized device work together to relay data out-of-band in real time. When a victim customer attempts to make a (comparatively inexpensive) purchase from the malicious retailer, the malicious accomplice is also ready to make a (comparatively expensive) purchase at another (victim) retailer. The malicious accomplice forwards any communication it receives from the victim point of sale to the malicious retailer, who in turn forwards these messages to the victim credit card. Likewise, any responses sent by the victim credit card are forwarded to the malicious accomplice's device, which sends them to the victim point of sale. In so doing, the victim customer is actually paying for the malicious accomplice's (expensive) purchase, while the malicious retailer is losing the (inexpensive) purchase. This is particularly dangerous because the malicious parties leave no traces: should the fraud be detected and declared as such to the victim's bank, the victim retailer is left facing the charge-back. In either case, one of the two victims is left facing the (fraudulent) bill.

We note that both of these attacks are predicated on the customer's lack of participation in the protocol (beyond simply allowing it to occur), and involve modifying or replacing a charge after it has been confirmed by the customer. To defend against these attacks, we leverage the use of Electronic Wallets. Electronic Wallets are applications like Android Pay or Apple Wallet, which allow a smart phone to emulate a contactless credit card. In particular, we design the Secure CC Protocol to allow the customer to confirm the purchase price on a device under the customer's control: we augment the Solicitation message to include the price to be charged, and the customer's credit card then binds this price to the charge token.

If the customer uses an Electronic Wallet, this application (running on the customer's device) requests that the customer confirm the purchase price. The resulting charge token is then bound to this price, and cannot be used to complete a more expensive purchase. As such, while the Secure CC Protocol does not outright prevent the Transparent Bridge attack, it renders the attack mostly harmless.

If the customer does not use an Electronic Wallet and is instead using a physical credit card, there is no interface on which the customer may confirm a price, and so the card needs to assume confirmation for any price. As such, users of physical credit cards do not gain protection from malicious retailers. Nonetheless, they remain protected from malicious outsiders when using this protocol.

In Chapter 5 we discuss the use of Electronic Wallets in more detail. An Electronic Wallet is a collection of (virtual) credit cards, emulated by a smart phone. This model provides a significant advantage over physical credit cards: the wallet may virtualize an unlimited number of credit cards precluding a bulky wallet, virtualized cards do not respond to solicitations while the phone is locked and unattended, and the wallet provides a rich interface with which the customer may communicate with a (virtual) credit card.

In Chapter 6, we discuss the properties known as *linkability* and *un-*

linkability. All contactless credit card protocols discussed thus far (including the protocol in use today) are linkable: retailers are able to identify multiple purchases made using the same credit card. This ability is valuable to retailers, because it allows them to construct purchasing profiles on their customers. These profiles can then be used for more effective marketing, or can be sold to interested third parties.

Purchasing habits can reveal sensitive information about a customer, and thus some customers may wish to prevent retailers from constructing such profiles on them. Currently, the only way in which to do so is to eschew the use of credit cards entirely. We consider how to approach constructing an unlinkable payment protocol for customers concerned about their privacy.

To construct an unlinkable protocol, we must transmit identifying information from the credit card to the bank, by way of a carrier (the retailer) who may not learn the contents of the message. As such, employing cryptography is a natural approach. However, symmetric cryptography cannot yield an unlinkable protocol: the card must identify to the bank which customer's key to use to decrypt the message, and so the retailer may simply identify cards based on this key identifier. By contrast, asymmetric cryptography provides sufficient primitives to construct an unlinkable protocol, since all cards can encrypt their messages using the same key (the bank's public key).

However, as described in Chapter 7, we espouse two additional goals when constructing an unlinkable contactless credit card protocol: being able to use any credit card and making use of existing point of sale infrastructure. The latter goal is particularly important, as requiring modification to points of sale would require retailer cooperation, and one cannot expect retailer cooperation when removing a feature valuable to retailers. Furthermore, using existing infrastructure alleviates significant barriers to adoption, in that one doesn't need to replace all points of sale across all retailers to use the new protocol.

Existing point of sale infrastructure expects a Card Information message including a 16-digit credit card number, a 4-digit expiration date, and an 8-digit iCVV. This yields 28 numeric digits, or just over 93 bits, of message-space in which a new protocol may operate. Asymmetric cryptography requires block sizes of thousands of bits in order to be secure, and thus is not an appropriate choice for an unlinkable contactless credit card protocol.

The Unlinkable Wallet Protocol is described in Chapter 8. This protocol is designed for Electronic Wallet applications rather than physical credit cards and can be used by privacy conscious customers to virtualize any credit card. We take advantage of the fact that the Card Information message indicates the destination of the subsequent Charge Request message, constructing a tunneled protocol. We construct the protocol in three phases, reminiscent of the stepwise refinement process we used in Chapter 3.

The first iteration of this protocol (described in Section 8.1) begins with two strong assumptions: that retailers are not malicious, and that the customer's Wallet Application has access to the Internet at all times. Before each transaction, the Wallet Application and the Wallet Server agree on a 93-bit token. When the (virtual) credit card sends a Card Information message, it encodes this 93-bit token into a 28-digit value and splits it into pseudo-values for a credit card number, expiration date and iCVV. These pseudo-values are then sent in the Card Information message, along with the Bank Name field indicating the Wallet Server. The Wallet Server decodes the pseudo values to reconstruct the 93-bit token, and looks up which of its registered cards is associated with this token. Upon finding it, the Wallet Server then issues a *visual* Charge Request to the issuing bank (consisting of the information one would normally type into an online form to make a purchase on the Internet), relaying the bank's Approval message back to the point of sale. Since the pseudo card information consists of randomized data, this protocol is unlinkable

The second iteration of this protocol (described in Section 8.2) erodes the assumption that retailers are non-malicious. In this version, the agreed upon token consists of only 80 bits, leaving 13 bits left over. We use these remaining 13 bits to contain a "price hash", consisting of an HMAC over the 80-bit token concatenated with the price, keyed with a key known only to the (virtual) card and the Wallet Server. This price hash binds the token to a particular price, employing a defense much like in the Secure CC Protocol described in Chapter 4.

The third and final iteration of this protocol (described in Section 8.3) erodes the assumption that the (virtual) card is connected to the Internet at all times. Instead of the Wallet Application receiving a token from the Wallet Server, both the Application and the Server generate this token independently. The Wallet Server and the Wallet Application both maintain a successful transaction counter. To generate a token for use in a transaction, both parties calculate the HMAC of this counter, keyed with the same key used to calculate the price hash.

Since tokens are now generated rather than chosen, some care must be taken to deal with collisions and synchronization issues. Thus, it is necessary to construct a collision mitigation strategy. While the Unlinkable Wallet Protocol does suffer from potential collisions, a collision merely renders two individual virtual credit cards inoperative until one of the respective Wallet Applications regains access to the Internet. This may be inconvenient, but does not pose a significant problem, and cannot result in mis-routed charges. We analyze the expected frequency of such collisions in Chapter 9, working both mathematically and in simulation. The expected frequency of such collisions is very low, to the point where they are expected not to occur in practice. No collisions were observed in simulation.

10.2 Future Work

The work described in this dissertation suggests the following avenues that merit further research:

1. Protocols relying on an iCVV (i.e. the Insecure CC Protocol currently in use, the Externally Secure CC Protocol, and the Secure CC Protocol) may suffer de-synchronization. This happens when Card Information messages are generated and sent by the card, but resulting Charge Re-

quest messages not reach the bank (as a result of persistent network failures at the point of sale or as a result of repeated skimming attempts). When the card's internal iCVV has progressed past the number of failures anticipated by the bank, contactless functionality of the card becomes inoperative. The construction of a re-synchronization protocol for iCVV states would alleviate this issue.

2. The Unlinkable Wallet Protocol involves fast lookup on a table of registered credit card records by the Wallet Server, via the corresponding single-use token. This table contains every credit card registered with the Wallet Server, and may therefore contain millions of rows. Our description of the protocol suggests pre-computing the next expected token for each card, and using this value as an index over the table. This value is replaced on each transaction, and as a consequence results in frequent index rebuilding, which may result in a performance bottleneck at scale. The construction of a better data-structure for use on the Wallet Server to represent registered credit cards, allowing fast record lookup given a frequently changing lookup key, would provide a significant performance gain and allow the system to scale more effectively.

3. While unlinkable transactions are a significant gain to customer privacy, they present drawbacks in specific scenarios. For example, unlinkable transactions hamper any system in which customers identify themselves via a previously used credit card. This is commonly seen in retail returns

and ticket collection kiosks, where the customer presents the credit card used to make the purchase. The Unlinkable Wallet Protocol renders this form of identification impossible by design. The construction of a protocol in which the Wallet Application can allow the customer to choose, on a per-purchase basis, whether the transaction should be linkable or unlinkable provides the flexibility needed to work around this issue.

The family of protocols presented in this dissertation represents a strong step above the status quo. We consider protection from malicious outsiders to be an urgent priority. Protecting customers from malicious retailers is less urgent, but is nonetheless important given the severity of attacks which malicious retailers may perpetrate. Whether to adopt a protocol providing linkability or unlinkability should be a customer choice: while we do not presume to dictate the best move for the industry in this regard, we place the decision of a customer's privacy in the customer's hands rather than in those of the retailers.

Bibliography

[1] Ross Anderson. Position statement in RFID S&P panel: RFID and the middleman. In *International Conference on Financial Cryptography and Data Security*, pages 46–49. Springer, 2007.

[2] Tim WC Brown, Thomas Diakos, and Johann A Briffa. Evaluating the eavesdropping range of varying magnetic field strengths in nfc standards. In *Antennas and Propagation (EuCAP), 2013 7th European Conference on*, pages 3525–3528. IEEE, 2013.

[3] Wei-Dar Chen, GP Hancke, KE Mayes, Y Lien, and JH Chiu. Using 3G network components to enable NFC mobile transactions and authentication. In *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, volume 1, pages 441–448. IEEE, 2010.

[4] Vedat Coskun, Busra Ozdenizci, and Kerem Ok. A survey on near field communication (NFC) technology. *Wireless Personal Communications*, 71(3):2259–2294, 2013.

[5] W. Diffie and M.E. Hellman. New directions in cryptography. *Information Theory, IEEE Transactions on*, 22(6):644–654, Nov 1976.

[6] Saar Drimer and Steven J. Murdoch. Keep your enemies close: Distance bounding against smartcard relay attacks. In *Proceedings of 16th*

USENIX Security Symposium on USENIX Security Symposium, SS'07, pages 7:1–7:16, Berkeley, CA, USA, 2007. USENIX Association.

[7] ECMA. *Near Field Communication - Interface and Protocol*, 3 edition, June 2013.

[8] Natheniel Popper Elizabeth Harris, Nicole Perlroth. Neiman marcus data breach worse than first said. `http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html`. Accessed: 2015-05-10.

[9] Natheniel Popper Elizabeth Harris, Nicole Perlroth. Neiman Marcus data breach worse than first said. `http://www.nytimes.com/2014/01/24/business/neiman-marcus-breach-affected-1-1-million-cards.html`. Accessed: 2014-11-10.

[10] EMVCo. LLC. *EMV Contactless Specifications for Payment Systems - Contactless Communication Protocol*, March 2016.

[11] Hasoo Eun, Hoonjung Lee, and Heekuck Oh. Conditional privacy preserving security protocol for NFC applications. *Consumer Electronics, IEEE Transactions on*, 59(1):153–160, 2013.

[12] Luka Finžgar and Mira Trebar. Use of NFC and QR code identification in an electronic ticket system for public transport. In *Software, Telecommunications and Computer Networks (SoftCOM), 2011 19th International Conference on*, pages 1–6. IEEE, 2011.

[13] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. On the security issues of NFC enabled mobile phones. *International Journal of Internet Technology and Secured Transactions*, 2(3-4):336–356, 2010.

[14] Lishoy Francis, Gerhard Hancke, Keith Mayes, and Konstantinos Markantonakis. Practical NFC peer-to-peer relay attack using mobile phones. In *Radio Frequency Identification: Security and Privacy Issues*, pages 35–49. Springer, 2010.

[15] Gerhard P Hancke. A practical relay attack on iso 14443 proximity cards. *Technical report, University of Cambridge Computer Laboratory*, 59:382–385, 2005.

[16] Gerhard P Hancke and Markus G Kuhn. An rfid distance bounding protocol. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 67–73. IEEE, 2005.

[17] Ernst Haselsteiner and Klemens Breitfuß. Security in near field communication (NFC). In *Workshop on RFID security*, pages 12–14, 2006.

[18] Blackwing HQ. NFCProxy. `http://sourceforge.net/projects/nfcproxy/`, 2012.

[19] ISO 14443. *Identification cards - Contactless integrated circuit(s) cards - Proximity cards*, 2000.

[20] Oliver Jensen, Mohamed Gouda, and Lili Qiu. A secure credit card protocol over NFC. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, page 32. ACM, 2016.

[21] Oliver Jensen, Tyler O'Meara, and Mohamed Gouda. Securing NFC credit card payments against malicious retailers. In *International Conference on Networked Systems*, pages 214–228. Springer, 2016.

[22] Clare Kennedy. Millions of card numbers likely stolen during Supervalu data breach, security expert says. `http://www.bizjournals.com/twincities/news/2014/08/18/supervalu-millions-card-numbers-likely-stolen.html?page=all`. Accessed: 2014-11-10.

[23] Ziv Kfir and Avishai Wool. Picking virtual pockets using relay attacks on contactless smartcard. In *Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005. First International Conference on*, pages 47–58. IEEE, 2005.

[24] Henning Kortvedt and S Mjolsnes. Eavesdropping near field communication. In *The Norwegian Information Security Conference (NISK)*, 2009.

[25] Henning Siitonen Kortvedt. Securing near field communication. Master's thesis, Norwegian University of Science and Technology, Norway, 2009.

[26] Hugo Krawczyk, Ran Canetti, and Mihir Bellare. Hmac: Keyed-hashing for message authentication. 1997.

[27] Brian Krebs. P.F. Chang's breach likely began in Sept. 2013. `http://krebsonsecurity.com/2014/06/p-f-changs-breach-likely-began-in-sept-2013/`. Accessed: 2014-11-10.

[28] Paul J Leach, Michael Mealling, and Rich Salz. RFC 4122: A universally unique identifier (UUID) URN namespace. 2005.

[29] Eddie Lee. NFC hacking: The easy way. In *Defcon hacking conference*, volume 20, 2012.

[30] Renaud Lifchitz-BT. Hacking the NFC credit cards for fun and debit. 2012.

[31] Gerald Madlmayr, Josef Langer, Christian Kantner, and Josef Scharinger. NFC devices: Security and privacy. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 642–647. IEEE, 2008.

[32] Konstantinos Markantonakis, Lishoy Francis, Gerhard Hancke, and Keith Mayes. Practical relay attack on contactless transactions by using NFC mobile phones. *Radio Frequency Identification System Security: RFIDsec*, 12:21, 2012.

[33] MasterCard. *PayPass - Mag Stripe - Security Architecture*, November 2007.

[34] Rajalakshmi Nandakumar, Krishna Kant Chintalapudi, Venkat Padmanabhan, and Ramarathnam Venkatesan. Dhwani: secure peer-to-peer

acoustic NFC. In *ACM SIGCOMM Computer Communication Review*, volume 43, pages 63–74. ACM, 2013.

[35] Chris M Roberts. Radio frequency identification (RFID). *Computers & security*, 25(1):18–26, 2006.

[36] Sara Germano Robin Sidel, Danny Yadron. Target hit by credit-card breach. `http://online.wsj.com/articles/SB10001424052702304773104579266743230242538`. Accessed: 2014-11-10.

[37] Michael Roland. Software card emulation in NFC-enabled mobile phones: great advantage or security nightmare. In *Fourth International Workshop on Security and Privacy in Spontaneous Interaction and Mobile Phone Use*, pages 1–6, 2012.

[38] Michael Roland and Josef Langer. Cloning credit cards: A combined pre-play and downgrade attack on EMV contactless. In *WOOT*, 2013.

[39] Michael Roland, Josef Langer, and Josef Scharinger. Practical attack scenarios on secure element-enabled mobile devices. In *Near Field Communication (NFC), 2012 4th International Workshop on*, pages 19–24. IEEE, 2012.

[40] Michael Roland, Josef Langer, and Josef Scharinger. Relay attacks on secure element-enabled mobile devices. In *IFIP International Information Security Conference*, pages 1–12. Springer, 2012.

[41] Michael Roland, Josef Langer, and Josef Scharinger. Applying relay attacks to Google Wallet. In *Near Field Communication (NFC), 2013 5th International Workshop on*, pages 1–6. IEEE, 2013.

[42] Robin Sidel. Home Depot's 56 million card breach bigger than Target's. `http://online.wsj.com/articles/home-depot-breach-bigger-than-targets-1411073571`. Accessed: 2014-11-10.

[43] Sandeep Tamrakar, Jan-Erik Ekberg, and N Asokan. Identity verification schemes for public transport ticketing with NFC phones. In *Proceedings of the sixth ACM workshop on Scalable trusted computing*, pages 37–48. ACM, 2011.

[44] Visa. *Visa Contactless Payment Specification*, Janary 2016.

[45] Ron Weinstein. RFID: a technical overview and its application to the enterprise. *IT professional*, 7(3):27–33, 2005.

Vita

Oliver Christopher Jensen was born in Oxshot, England in 1987, and moved to Geneva, Switzerland before beginning school. After graduating from the International School of Geneva in 2005 with an International Baccalaureate, he attended Colgate University in Hamilton, New York. While at Colgate University, he worked on protocol research with Prof. Vijay Ramachandran, earning High Honors in Computer Science for his senior thesis on traceroute data integrity and route concealment. He graduated in 2009 with a Bachelors of Arts in Mathematics and Computer Science.

Joining the University of Texas at Austin Computer Science Ph.D. program in 2010, Oliver focused his research on subjects relating to security and privacy. Discovering a love for teaching, he taught an upper-division class on security and privacy in networked systems between 2014 and 2017. Oliver was awarded a Masters of Computer Science with a minor in Electrical Engineering in December of 2013, and a Ph.D. of Computer Science in May of 2017.

Email address: *ojensen@utexas.edu*

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.