A Secure Credit Card Protocol over NFC

Oliver Jensen
University of Texas
Austin, Texas
ojensen@cs.utexas.edu

Mohamed Gouda
University of Texas
Austin, Texas
gouda@cs.utexas.edu

Lili Qiu
University of Texas
Austin, Texas
lili@cs.utexas.edu

ABSTRACT

NFC ("Near Field Communication") is a short-range wireless communication channel. The current NFC credit card protocol allows a contactless credit card to communicate wirelessly with a Point-of-Sale in order to perform a purchase. This protocol is vulnerable to four common attacks: eavesdropping, skimming, relay attacks, and compromised Points-of-Sale. The attacker's objective is twofold: stealing sensitive information, and performing unauthorized. We use stepwise refinement to design a secure NFC credit card protocol which defends against all four of these attacks. The resulting protocol does not use heavyweight cryptographic operations, instead using only inexpensive primitives such as pre-computed hashes, indexing, and XOR operations. Moreover, it explores the lower-bound of computation required on the card to mount an effective defense against these four classes of attacks. As such, the energy and computational requirements of the credit card in our protocol are kept to a minimum.

CCS Concepts

•**Networks** → **Protocol correctness;**

Keywords

Credit card payments; secure mobile payments; Near Field Communication (NFC); authentication, privacy;

1. INTRODUCTION

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 the credit card reader. This is advantageous because a Point-of-Sale's ability to read a credit card is no longer contingent on reading the credit card's magnetic strip, notorious for becoming demagnetized or being difficult to read due to dirty or corroded contacts on the reader. As a result, the system becomes more robust with fewer parts needing regular service.

The wireless communication channel used for contactless credit cards is NFC, or "Near Field Communication". NFC is a form of RFID ("Radio Frequency Identification"). Like many RFID protocols, it operates at a frequency of 13.56 Mhz and employs a model where one of the two communicating parties may be *passive*. Passive tags have no power source of their own, and instead draw power via magnetic inductance from the reader while communication occurs. But unlike most RFID protocols wherein passive tags can be read at a distance of up to about 1 meter, the range of NFC is extremely limited: a passive NFC tag's communications can only be read within a range of about 2-4 centimeters. It is a combination of these two properties—the ability to communicate with a device lacking a power source (i.e. a credit card), and the very short range—that makes NFC attractive for credit cards. The former property is desirable for convenience, the latter for security.

NFC is a simple, plain-text communication channel. As such, any mechanisms for message confidentiality, integrity or authentication need to be implemented on top of NFC. As we show in Section 2 and 3, contactless credit cards do little to protect these transmissions. Instead, they rely on the short range of NFC to justify the assumption that malicious nodes will not be within range of a contactless credit card while escaping the victim's notice.

As a result, the contactless credit card protocol is vulnerable to four common attacks which we detail in section 3: eavesdropping, skimming, relay attacks, and data theft due to compromised Points-of-Sale. The objective of these attacks is twofold: stealing sensitive credit card information (such as the credit card number and expiration date), and performing purchases which were not authorized by the credit card holder.

These vulnerabilities are not merely theoretical. Rather, they are being widely exploited in the wild. For example, attacks compromising Points-of-Sale have resulted in the theft of credit card information of tens, if not hundreds, of millions of credit cards, and have earned notoriety in recent months with mainstream news-outlets such as the Wall Street Journal and the New York Times [15] [16] [2] [7] [10], due to the sheer scale of these attacks.

The primary contributions of this paper are as follows:

1. We identify prevalent attacks on the NFC credit card protocol.
2. We design an alternative credit card protocol which defends against all of these attacks.

ICDCN '16, January 04 - 07, 2016, Singapore, Singapore

ISBN 978-1-4503-4032-8/16/01…$15.00

DOI: http://dx.doi.org/10.1145/2833312.2833319

3. We use *stepwise refinement* to construct this secure protocol and prove its correctness in the face of these attacks.

4. Our secure credit card protocol is constructed to explore the lower-bound of required computation to defend against these attacks, avoiding expensive computation on the credit card. We do not use heavyweight cryptographic primitives such as public-key infrastructure. Instead, our protocol uses only inexpensive primitives such as pre-computed hashes, indexing, and XOR operations, while still being provably safe in the face of the enumerated attacks.

5. Our protocol ensures that retailers maintain their ability to correlate purchases from the same credit card.

We recognize that altering currently-running systems is difficult, particularly so in the payment industry. As such, we suggest a method of using our proposed protocol while avoiding *any* changes to currently running payment processing systems. We accomplish this by defining an additional system within the bank, a "payment proxy" which can translate our secure protocol's charge requests into those of the current protocol. By avoiding any changes to existing systems, we gain the additional benefit that banks may continue to support Points-of-Sale which use the current (insecure) protocol. In so doing, we hope to ease adoption within an industry that is very resistant to change or modification.

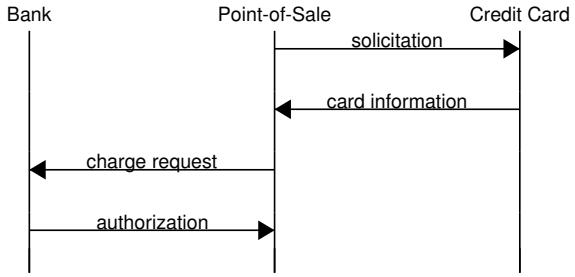
2. NFC AND THE CURRENT CC PROTOCOL

NFC ("Near Field Communication") is a widely-adopted short-range wireless communication channel. It is designed for communication between an active (powered) reader and a passive tag, where the tag is powered by the reader through magnetic inductance. NFC is based on RFID. and has many similar physical characteristics, while ensuring a very limited range of about 2-4 cm around a passive tag.

The physical NFC specification is designed to provide sufficient power to passive tags for expensive operations, including those necessary for public-key cryptography. However, the circuitry and computational ability of tags themselves vary greatly. Some tags, such as the German National ID card, employ traditional public-key infrastructure (PKI). Other tags, such as Mifare Classic tags, are limited to storing, retrieving, incrementing and decrementing data blocks, but utilize stream ciphers and challenge-response mechanisms to protect communications. Current contactless credit cards are capable of only very limited computation, and do not engage in any cryptographic operations.

The protocol for performing contactless credit card transactions over NFC (termed *CC Protocol*) is composed of two query-response pairs. First, the NFC reader (hereafter referred to as a *Point-of-Sale*) solicits the card for its credit card number and expiration date, and the card responds with this information. In its response, the credit card also includes an iCVV, or *integrated Card Verification Value*: a dynamic security token intended to authenticate the message. Once this has completed, the Point-of-Sale sends a charge request to the bank with the information received from the credit card, and then receives an authorization response to accept or reject the charge.

Figure 1: The current CC Protocol



The exchange of messages in the CC Protocol is shown in Figure 1. They are: *solicitation*, *card information*, *charge request* and *authorization*. Note that after the card responds to the Point-of-Sale, its involvement in the transaction is complete. The contents of these messages are as follows:

Solicitation: First, the Point-of-Sale solicits the credit card for its information. The solicitation is composed of a number of messages sent in both directions, identifying the Point-of-Sale type (e.g. `2PAY.SYS.DDF01`) and the credit card type (e.g. `VISA CREDIT`). Since these messages are constant for a given Point-of-Sale and credit card, we abstract the solicitation messages as a single request from the Point-of-Sale to the credit card.

Card Information: The credit card responds to the solicitation by sending back the following card information:

- the credit card number
- the credit card's expiration date
- the iCVV
- the name of the bank that issued the card

The iCVV is an unpredictable 4-byte value freshly generated for every solicitation response, and is subsequently used by the bank to validate the transaction as described below.

Charge Request: The Point-of-Sale issues a charge request to the bank. This request is composed of:

- the credit card number
- the credit card's expiration date
- the iCVV
- the dollar amount to be charged

Authorization: When the bank receives a charge request, it uses the credit card number to look up the account, verifies the expiration date, and then validates the iCVV to authorize the purchase. It will generally also perform some additional checks, such as verifying that the card was not reported lost or stolen, or matching this purchase's location against the known location of the card holder. Finally, it responds with its authorization decision.

When the credit card is manufactured, a secret seed value is shared between the credit card and the bank. This enables the credit card and the bank to both generate the

same iCVV sequence, unpredictable to any party that does not have access to this seed. The iCVVs are simply sequential elements of this sequence[1]: each time the credit card responds to a solicitation, it generates the next iCVV in the sequence and includes it with the card information response.

In order to make an authorization decision, the bank searches through its account database, which is indexed by the credit card number. Once the bank locates the account, it verifies that the received expiration date matches the expiration date on file. In addition, it recalls the iCVV from this credit card's previous charge request and generates the next element in the sequence, then compares the received iCVV to the value it generated.

It is possible that a card may generate an iCVV without communicating it to the bank. For example, a charge request may become corrupted in transit, or a Point-of-Sale may experience a network failure. As a result, a credit card's iCVV may have advanced further in the sequence than the bank expects. To handle this situation, the bank may generate several iCVVs in the sequence for comparison to the received value.

If a match is found, the bank considers the iCVV to be valid. It updates its state into the pseudorandom sequence to reflect the received iCVV, and continues with any other checks to be performed before authorizing the charge. If no match is found, the bank considers the iCVV to be invalid and declines the charge.

3. ATTACKS ON THE CC PROTOCOL

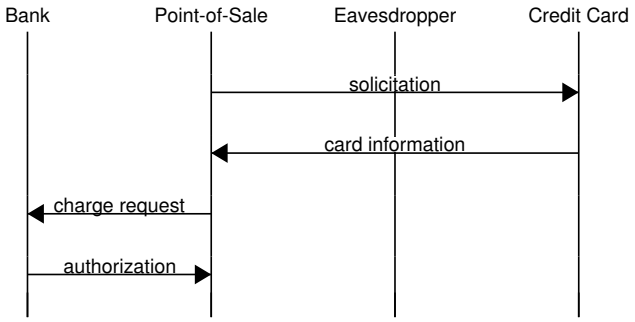
The current CC protocol has several aspects which render it dangerous. Sensitive data is transmitted in the clear, can be re-used by malicious parties, is learned by potentially insecure systems. Neither the Point-of-Sale nor the credit card authenticate to each other. Furthermore, proximity is used as an indicator of intent, requiring a card holder to maintain constant vigilance on the surroundings of their credit cards. These aspects invite a number of security attacks on the protocol.

In this section, we describe four types of security attacks on the current CC protocol that have previously been published: *eavesdropping*, *skimming*, *relay attacks*, and attacks facilitated by *compromised Points-of-Sale*. We highlight the practicality of these attacks. For example, eavesdropping, skimming, and relay attacks can be easily performed with no more than a phone with NFC capabilities, or a small number of inexpensive off-the-shelf components. Also, Points-of-Sale have been so widely exploited and compromised that within the last several years it has been reported on by most mainstream news organizations. Indeed, 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.

3.1 Eavesdropping

The goal of an eavesdropper is to gain the victim's credit card information such as the 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 assumed to be secure.) An outline of this attack is shown in Figure 2.

Figure 2: Eavesdropping



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 [8], we modified a MIFARE NFC tag to 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 3, we show our NFC eavesdropping antenna next to a credit card for scale. The resulting antenna is paper thin, flexible, approximately 3cm in diameter and adhesive on one side. As such, it can easily be concealed within range of a Point-of-Sale.

Figure 3: Eavesdropping Antenna (credit card for scale)

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. While our setup cannot decode messages in real time, our construction is to our knowledge the smallest and most easily concealed NFC eavesdropping antenna yet.

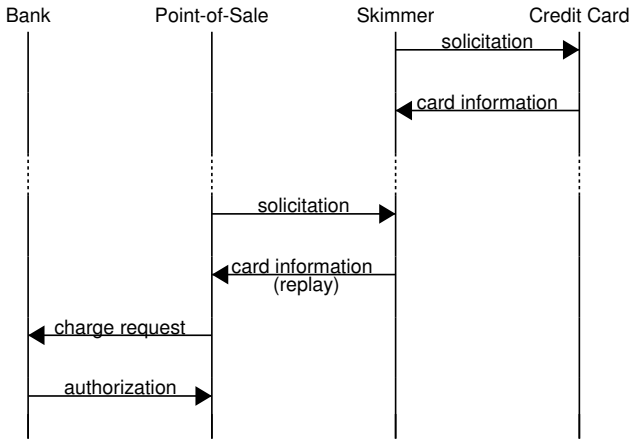
[1]The precise implementation of the iCVV is not a published standard, and indeed may vary between card manufacturers. We assume that it is simply a 4-byte pseudorandom value, as output by a seeded pseudorandom number generator on the credit card.

In the current implementation of the 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.)

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 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 the iCVV. An outline of this attack is shown in Figure 4.

Figure 4: Skimming



This attack can be performed using a smart-phone with NFC capabilities. An Android application called *NFCProxy*[2] 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.

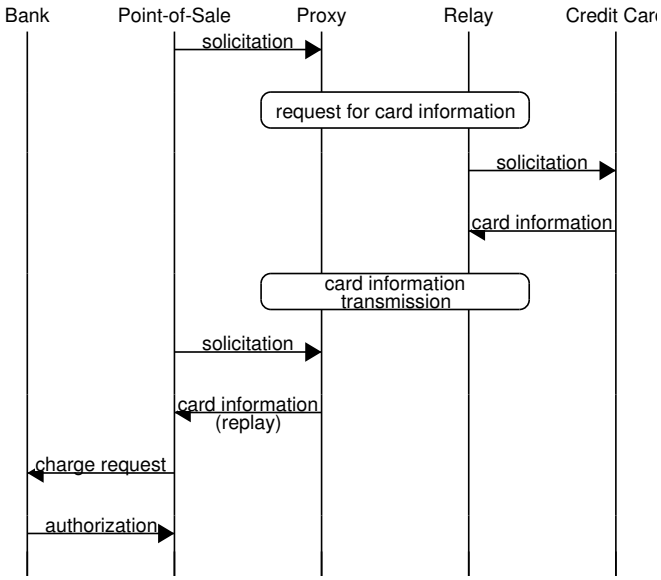
With *NFCProxy* running, the skimmer brings their phone briefly within range of an NFC credit card to acquire the credit card information and the iCVV. When the skimmer wishes to perform the illegitimate purchase on behalf of the victim, the skimmer moves their phone within range of a Point-of-Sale as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

In the current implementation of the 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.)

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. Multiple devices may be used to relay skimmed credit card information across a separate channel, effectively breaking down the assumption of proximity built into NFC. This attack can also be performed using the *NFCProxy* Android application, described in Section 3.2. An outline of this attack is shown in Figure 5.

Figure 5: Relay Attack



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 their 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 it 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 current implementation of the 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.

3.4 Compromised Point-of-Sale

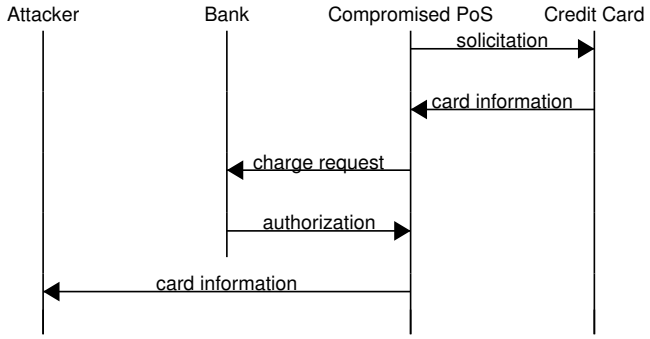
Any protocol in which the Point-of-Sale learns information capable of permitting multiple charges is vulnerable to Compromised Point-of-Sale attacks. We use this term to refer to any attacks which involve the Point-of-Sale or merchant performing (possibly unintentional) actions leading to credit card theft[3]. For example, a Point-of-Sale might be

[2] *NFCProxy*[6], presented at Defcon 20, can be downloaded here: http://sourceforge.net/projects/nfcproxy/

[3] In the Compromised Point-of-Sale model 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, as

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 6.

Figure 6: Compromised Point-of-Sale



Such occurrences are far from a theoretical threat: these attacks have become alarmingly commonplace, to the point that they have been covered by mainstream news sources including the New York Times and the Wall Street Journal.

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[15].

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[16].

Other recent victims of Compromised Point-of-Sale attacks 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)[2], 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[7]), 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 chards[10]).

3.5 Mitigations

The potential pitfalls of using NFC for confidential or sensitive transmissions have been surveyed in the past, and some proposals to mitigate the dangers have been discussed.

For example, Haselsteine et al.[5] survey the security landscape with respect to NFC, enumerating and discussing a number of channel-level attacks. They conclude that NFC could be made secure by initiating conversations with a protocol such as Diffie-Hellman key exchange[1] to establish a shared secret. All subsequent transmissions would be transmitted encrypted symetrically using this shared secret as a key. Such a key-exchange protocol is computationally expensive, but this cost is shared over the entirety of the subsequent transmissions. While such a defense is very effective at preventing eavesdropping, it cannot provide an answer to skimming, relay attacks or attacks facilitated by compromised Points-of-Sale.

To prevent skimming and relay attacks, only an authorized Point-of-Sale should learn useful information from a credit card. Furthermore, to prevent attacks facilitated by compromised Points-of-Sale, a Point-of-Sale should not learn information that may be used more than once. As such, public-key encryption with certificates[14] would be an attractive (if computationally expensive) avenue to explore. However, such a protocol would be hamstrung in the case of a compromised key without a method for key revocation. As passive NFC tags lack a real-time clock and cannot reliably receive broadcast revocation lists, key revocation or even expiration on credit cards becomes problematic.

4. A SECURE CC PROTOCOL

In this section, we propose a Secure CC Protocol that guards against all four classes of attacks described in Section 3. Our protocol avoids the use of any computationally expensive operations such as signature verification or even hashing. In our construction, we restrict a credit card to performing only basic arithmetic, indexing, XOR, and similarly inexpensive operations. The motivation for this restriction is to ensure that the adoption of our protocol does not significantly increase the cost of manufacturing credit cards. As such, we explore the lower-bound of computation that needs to be executed on the credit card while defending against the four classes of attacks.

Our protocol is designed 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.
3. 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 propose concrete implementations of functions F and G.
5. Finally, we prove 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 our Secure CC Protocol and prove that it is not vulnerable to any of the four classes of attacks described in Section 3.

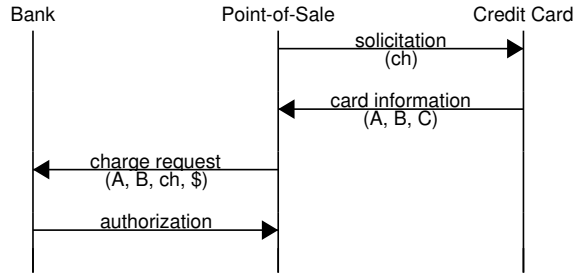
4.1 The Protocol

Our Secure CC Protocol uses the same four messages as the original (insecure) CC Protocol. However the contents of these messages have become more involved. We incorporate a challenge-response, which serves not only to validate the entity responding to the solicitation, but also to protect the

it could trivially charge an arbitrarily large amount to the card.

credit card from malicious card readers. An outline of this protocol is seen in Figure 7.

Figure 7: The Secure CC Protocol



The four messages used in the Secure CC Protocol are described in some detail here.

Solicitation (ch): The Point-of-Sale sends a solicitation message much like in the current 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 (A, B, C): The solicitation response consists of three components, A, B and C. They are as follows:

- ***A = UUID,*** a Universally Unique Identifier[11], is used to identify the card without revealing the card's information to eavesdroppers or any other party. Our use of a UUID, which requires a significant source of entropy, does not violate our computational constraints by the fact that it is fixed. As such, this value is computed by the credit card manufacturer and stored on the card as a constant.
- ***B = 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.
- ***C = bank name*** is transmitted in the clear, as before, so that the Point-of-Sale may route its charge request to the proper entity.

In so doing, A provides identification, B provides authentication, and C provides routing information. In addition, A can be used by retailers in order to recognize credit cards, allowing retailers to continue the popular practice of correlating purchases[4].

[4]The widespread rejection of *ApplePay* has shown that retailers can ultimately wield veto power against protocols which they do not find agreeable. While there are privacy concerns regarding the ability of retailers to correlate purchases from credit cards, we consider it important not to break this feature for this reason.

Charge Request ($A, B, ch, \$$): The Point-of-Sale issues a charge request to the credit card's issuing bank. In this protocol, the Point-of-Sale does not learn the card's private information, but simply forwards the card identification (A) and authentication (B) to the bank specified by C. The Point-of-Sale also includes the challenge ch so that the bank can verify that B is a valid response to ch from card A, as well as the dollar amount to be charged.

Authorization: The bank maintains an index of *UUID* into its account database. When the bank receives a charge request, it identifies the matching record as specified by A, looking up $info_{bank}$ and $iCVV_{bank}$. It then calculates $B_{bank} = H(info_{bank}, ch, iCVV_{bank})$ and verifies that $B = B_{bank}$. This step is equivalent to verifying the credit card information and iCVV in the current CC Protocol, resulting in an authorization decision to accept or reject the charge request.

4.2 Defending Against Attackers

We pit this protocol against the attacks described in Section 3, and identify the two properties **H1** and **H2** needed from function H in order for these attacks to be thwarted.

4.2.1 Eavesdropping

When eavesdropping on a transaction, the eavesdropper learns ch, A, B and C. The challenge ch, the card identifier A and the bank name C are all public information, of no value to the eavesdropper. Only $B = H(info, ch, iCVV)$ 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 $H(info, ch, iCVV)$ is indistinguishable from random.

If function H satisfies property **H1**, then the eavesdropper (ignorant of $iCVV$) cannot distinguish B from random, and as such can gain no useful information.

4.2.2 Skimming

When skimming a credit card, the skimmer provides a challenge ch_{skim} to the card. In return, the skimmer learns A, $B_{skim} = H(info, ch_{skim}, iCVV)$ and C. 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 $H(info, ch, iCVV)$, ch and ch' such that $ch \neq ch'$, one cannot infer $H(info, ch', iCVV)$.

If function H satisfies property **H2**, then the skimmer cannot use the value of B_{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.

4.2.3 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 A, $B_{relay} = H(info, ch_{relay}, iCVV)$ and C, and 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 B_{relay} in order to construct a response to the Point-of-Sale's challenge ch_{pos}, and as a result, cannot perform purchases on behalf of the credit card.

4.2.4 Compromised Point-of-Sale

A compromised point of sale will result in an attacker learning a valid (A, B, C, ch) tuple, which the Point-of-Sale requires in order to perform a charge request. 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 B cannot be reused since the $iCVV$ used in its construction is no longer valid after the charge has occurred.

4.3 Requirements of Function H

In summary, in order to defend against the four classes of attacks described in Section 3, we require that $iCVV$ be a pseudorandom value[5], and that function H upholds the following two properties:

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

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

5. IMPLEMENTATION OF THE SECURE CC PROTOCOL

We find it convenient to define function H as a composition of functions F and G as follows:
$H(info, ch, iCVV) = F(x, iCVV)$ where $x = G(info, ch)$

5.1 Properties of Functions F and G

We require that functino F satisfies the following two properties:

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 $F(x, y)$.

We also require that function G satisfies the following property:

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

[5]The current methods of $iCVV$ generation are not publicly disclosed. Indeed, any bank may use any arbitrary generation function they would like, the only requirement being that it be "unpredictable" by parties other than the bank. Our proposed secure CC protocol imposes the requirement that the $iCVV$ be pseudorandom, since the prediction of any bit of the $iCVV$ could leak sensitive information. We do not believe this requirement to be particularly onerous.

5.2 Theorems

We 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**.

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

PROOF. Let $x = G(info, ch)$ for any $info$, ch. Then $H(info, ch, iCVV) = F(x, iCVV)$. Thus if $y = iCVV$ is indistinguishable from random, then by property **F1**, $F(x, y) = H(info, ch, iCVV)$ 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 $x = G(info, ch)$ and $x' = G(info, ch')$ for any $info$. By property **F2**, evaluating $H(info, ch', iCVV)$ requires knowledge of $x' = G(info, ch')$. However, by property **G1**, one cannot infer $G(info, ch')$ without knowledge of $info$. Further, by property **H1**, no bits of $info$ are leaked and thus $info$ remains secret to all parties except the credit card and the bank. Thus, given $H(info, ch, iCVV)$, ch and ch' such that $ch \neq ch'$, one cannot infer $H(info, ch', iCVV)$. □

5.3 Implementation

We propose implementations for functions G, F and H in Figure 8. Note that function G is defined as the function which returns those bits of a keyed hash $h_k(info)$ for which the corresponding bit of ch was set to 1. For convenience we will refer to the output of $h_k(info)$ as the constant khi. Also note that function F is defined as `XOR`.

Figure 8: Proposed implementation of F, G and H

```
function G(info, ch):
    const khi = hash(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)
```

The function h_k is a strong cryptographic hash function keyed with the bank's secret key k. As $info$ does not change over the lifetime of the credit card, $h_k(info)$ 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.

In the current CC Protocol, $info$ is composed of 96 bits, and $iCVV$ is composed of 32 bits. If we maintain these field-lengths in the Secure CC Protocol, and use a keyed hash 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 $h_k(info)$ and ch have the same number of bits, and that the number of bits in $iCVV$ is equal to the number of *1-bits* in ch.

5.4 Verification

In this section we show that our implementation of function F satisfies properties **F1** and **F2**, and that our implementation of function H satisfies properties **H1**.

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

The output of $G(info, ch)$ is composed of a number of bits of $khi = h_k(info)$ selected by the challenge ch. If $ch_1 \neq ch_2$, one cannot infer $G(khi, ch_2)$ from $G(khi, ch_1)$ without knowledge of khi, because the results are composed of different bits of khi selected by the challenges ch_1 and ch_2. If a reasonable keyed hash function h_k is used, these bits are indistinguishable from random to any party without knowledge of *info* and the bank's secret key k. These bits are 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**.

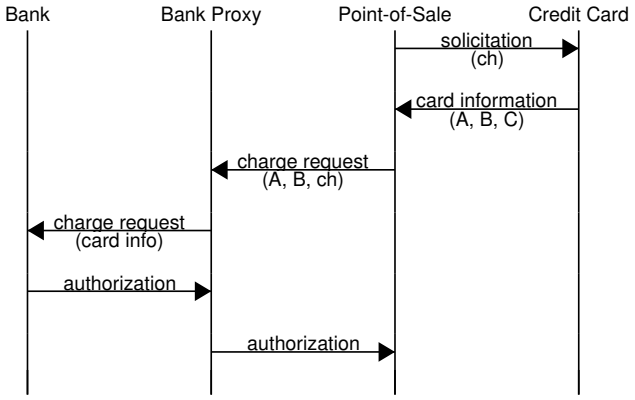
Note that while *info* is considered secret, it cannot be used directly in function G. This is because while most of the data in *info* is unpredictable, many of the bits are not random, and could thus lead to information leakage. For example, in a typical credit card number, the first six digits are taken from the *Issuer Identification Number*, a public value assigned to the issuing bank. 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. Similarly, the attacker knows that the byte representing the expiration month is between `0x01` and `0x12`, and can also attempt to guess the expiration year from a very small number of possibilities. Using the keyed hash $h_k(info)$ solves these problems by rendering each bit of the secret value independently indistinguishable from random.

Using a keyed hash has another benefit: by selecting a hash with the desired number of output bits (or by simply using the hash output modulo some selected value) the challenge-space can be set arbitrarily. If we assume a 32-bit $iCVV$, by using a 96-bit hash (and thus a 96-bit challenge) our challenge-space is 96 *choose* 32, equal to roughly 85 bits of entropy. Keeping the same length $iCVV$, by using a 512-bit hash (and thus a 512-bit challenge), our challenge space is now 512 *choose* 32, equal to roughly 169 bits of entropy.

6. PROXIED CC PROTOCOL

We acknowledge that changing the architecture of a production system is difficult, particularly so within the payment industry. As such, we propose a similar but alternative protocol which avoids modifying currently running payment processing systems, by separating the verification of B from the processing of the payment. This can be done by deploying a proxy, capable of verifying B and translating charge requests into those of the current CC Protocol's format. As a result, the NFC portion of the protocol enjoys the protections afforded by our Secure CC Protocol, while the payment processing systems continue to use the current CC Protocol. This protocol is illustrated in Figure 9.

Figure 9: Separated Secure CC Protocol



This proxied protocol requires the creation of a new entity, the "bank proxy", which is capable of looking up credit card information based on *UUID*, verifying $H(info, ch, iCVV)$, and translating charge requests of our Secure CC Protocol into charge requests as expected by the current CC Protocol. Due to this translation, the payment processing service which accepts or rejects payments can remain implemented exactly as is with no modifications whatsoever. The protocol as experienced by the Point-of-Sale and the credit card is identical to our Secure CC Protocol, and thus enjoys the same protections from the four classes of attacks described in Section 3.

7. RELATED WORK

Madlmayr, Langer, Kantner, and Scharinger provide an analysis of the state of security in NFC communications [13]. They emphasize that the assets to be protected include not only the privacy of the user and the information being transferred, but also the continued operability of the device and the functionality of the host controller. Enumerating a number of reasonable use cases and possible attacks, and conclude that there are some security and privacy issues for which technical solutions exist. However, in the case of skimming or relay attacks, it is not the channel which is under attack but the authenticity of one of the participants, and so such an approach falls short of protecting contactless credit card payments as they exist today.

Francis, Hancke, Mayes, and Markantonakis describe the implementation of a peer-to-peer NFC relay attack using off-the-shelf equipment [4]. Emphasizing the lack of need of secure program memory or code signing, they construct a proof-of-concept NFC relay using four phones and proxying the data via Bluetooth. Francis et al. propose using location information as an NFC relay attack countermeasure, since location information is often available in a mobile setting (whether from GPS or cell tower information). There are potential issues surrounding using location information, such as those arising when no external communication is possible, or those arising from GPS spoofing. In addition, such approaches are not applicable in a setting involving passive tags such as credit cards: lacking a power source, passive NFC tags do not have ready access to their location.

Lee released the Android application *NFCProxy* which implements skimming and relay attacks on NFC credit card transactions, and provides some analysis of these attacks [12]. The primary focus of this work is to demonstrate how easy it is for the general public (having little-to-no knowledge of NFC) to perform these attacks, and does not broach the topic of countermeasures. While *NFCProxy* limits the communication link between relay and proxy to a direct WiFi connection, any communication channel available to the mobile device could suffice. Though primarily focused on making the attacks simple to perform, Lee envisions extensions to the work encompassing pluggable modules such as protocol fuzzing and man-in-the-middle attacks.

Haselsteine, and Breitfuß provide an analysis of the classes of attackers which may target NFC communications, exploring various different attacks and attack mitigations [5]. They conclude that man-in-the-middle attacks on an *intentional transaction* are not feasible over NFC. That is to say, if a reader is attempting to communicate with a card, a third party cannot subvert or modify the messages en-route. This is due to in part to the infeasibility of preventing a reader from receiving the victim's transmissions. They approach the problem of security from the channel rather than the protocol, and determine that the appropriate action to take would be to perform a key-exchange protocol such as Diffie-Helmann in order to create a shared secret key, then use this shared secret to establish a secure channel. Such an approach defends against eavesdropping, but provides no protection against skimming, relay attacks, or attacks facilitated by a compromised Point-of-Sale.

Kortvedt explores the problem of eavesdropping on NFC communications, and suggests a symmetric encryption solution with a strong mutual authentication protocol [9]. The author suggests "Over-the-Air Programming" (OTA) as a potential solution for key management, as well as the inclusion of SRAM modules for random number generation. While applicable for the NFC protocol in general, such a solution does not lend itself well to simple contactless credit card systems. Furthermore, such a scheme is useful only against eavesdropping, and would not be applicable in preventing skimming, relay attacks or attacks facilitated by a compromised Point-of-Sale.

Eun, Lee, and Oh explore the issue of privacy in the face of NFC eavesdroppers [3]. They suggest the creation of an NFC-SEC protocol, complete with key-exchange and public key cryptography. Their approach has a wider scope than ours, 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). Unlinkability may not be desirable for retailers, who currently enjoy the ability to correlate purchases from the same credit card. Eun et al. focus primarily on payments via NFC in general, and thus are not constrained by the physical properties of passive credit cards. They focus instead on communication between active devices only, and are thus free to use arbitrarily complex computation.

8. ACKNOWLEDGMENTS

Research of Mohamed Gouda is supported in part by the NSF award #1430114.

9. REFERENCES

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

[2] N. P. 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.

[3] H. Eun, H. Lee, and H. Oh. Conditional privacy preserving security protocol for nfc applications. *Consumer Electronics, IEEE Transactions on*, 59(1):153–160, 2013.

[4] L. Francis, G. Hancke, K. Mayes, and K. Markantonakis. Practical nfc peer-to-peer relay attack using mobile phones. In *Radio Frequency Identification: Security and Privacy Issues*, pages 35–49. Springer, 2010.

[5] E. Haselsteiner and K. Breitfuß. Security in near field communication (nfc). In *Workshop on RFID security*, pages 12–14, 2006.

[6] B. HQ. Nfcproxy. http://sourceforge.net/projects/nfcproxy/, 2012.

[7] C. 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.

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

[9] H. S. Kortvedt. Securing near field communication. Master's thesis, Norwegian University of Science and Technology, 2009.

[10] B. 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.

[11] P. J. Leach, M. Mealling, and R. Salz. A universally unique identifier (uuid) urn namespace. RFC 4122, RFC Editor, July 2005. http://www.rfc-editor.org/rfc/rfc4122.txt.

[12] E. Lee. Nfc hacking: The easy way, 2012.

[13] G. Madlmayr, J. Langer, C. Kantner, and J. Scharinger. Nfc devices: Security and privacy. In *Availability, Reliability and Security, 2008. ARES 08. Third International Conference on*, pages 642–647. IEEE, 2008.

[14] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, Feb. 1978.

[15] S. G. Robin Sidel, Danny Yadron. Target hit by credit-card breach. http://online.wsj.com/articles/SB10001424052702304773104579266743230242538. Accessed: 2014-11-10.

[16] R. 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.