CERIAS Tech Report 2007-46

Dynamic Virtual Credit Card Numbers

by Ian Molloy, Jiangtao Li, and Ninghui Li

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

Dynamic Virtual Credit Card Numbers

Ian Molloy[1], Jiangtao Li[2], and Ninghui Li[1]

[1] Purdue University
{imolloy,ninghui}@cs.purdue.edu
[2] Intel Corporation
jiangtao.li@intel.com

Abstract. Theft of stored credit card information is an increasing threat to e-commerce. We propose a dynamic virtual credit card number scheme that reduces the damage caused by stolen credit card numbers. A user can use an existing credit card account to generate multiple virtual credit card numbers that are either usable for a single transaction or are tied with a particular merchant. We call the scheme dynamic because the virtual credit card numbers can be generated without online contact with the credit card issuers. These numbers can be processed without changing any of the infrastructure currently in place; the only changes will be at the end points, namely, the card users and the card issuers. We analyze the security requirements for dynamic virtual credit card numbers, discuss the design space, propose a scheme using HMAC, and prove its security under the assumption that HMAC is a PRF.

Key words: e-commerce, credit card theft

1 Introduction

Credit cards are one of the most widely used payment mechanisms for both business-to-consumer and business-to-business commerce today. Credit card transactions account for billions of dollars in transactions daily [18], and these transaction records are often stored in various kinds of databases. Many e-commerce websites store credit card information for user convenience, as users will use these sites multiple times over a period time and would prefer not to enter the credit information for each transaction. Examples of such sites include PayPal, online shopping websites such as Amazon.com, and online travel sites such as Expedia. Online merchants may also keep records of credit card numbers for dealing with charge-backs and other disputes. Credit card processing centers will also store credit card numbers and transactions in an attempt to detect fraud. Anomalies in purchase characteristics such as amounts, retailers, frequencies, and locations can be an indication of fraud. Detecting these anomalies more quickly can be beneficial to both the cardholder and card issuer. Other organizations such as hotels, will store credit card numbers for liability from damages and incidentals.

The extensive databases kept by numerous parties quickly become highly desirable targets for those wishing to steal credit card numbers and commit fraud. There have been several high-profile cases in recent years. For example, in 2001 attackers stole the customer records (including credit card information) of

the online merchant Bibliofind, a subsidiary of Amazon.com [9]. In 2005 attackers broke into credit card processing center CardSystems Solutions Inc. and stole over 40 million credit card numbers [12]. Not all losses are the result of an online attack. Recently, stolen laptops have resulted in the loss of credit card numbers for 243,000 Hotels.com customers [1] and 80,000 Department of Justice employees [19].

In this paper, we propose a dynamic virtual credit card number scheme that reduces the damage caused by theft of stored credit card information. A user can use an existing credit card to generate a "virtual credit card (VCC) number" that is restricted in a number of ways. For example, it may be usable for a single transaction, or be linked with a particular merchant and have a lower credit limit and a shorter expiration date than the actual card. Such a VCC number can be generated using devices carried by the user, e.g., a cell phone or a PDA, without online contact with the card issuing bank. In our scheme, VCC numbers have the same format as normal credit card numbers. Merchants should be able to process a transaction with a VCC number in the same manner they use today; no change to their existing databases and applications is needed. Only the end points, i.e., the cardholders and the card issuers, need to be aware that a VCC is used. We also point out that a card holder can still use the actual card the old fashioned way. Our design aims at facilitating deployment. We have implemented a prototype for generating VCC numbers using Java 2 MicroEdition (J2ME) that runs on MIDP2.0 compliant cell phones. We have tested our MIDlet on Sony Ericsson z520a and Nokia 6102i model phones.

Several credit card issuers (CitiBank, Discover, and MBNA) already offer services similar to the concept of VCC. However, they all require users to install software onto a computer and communicate with the credit card issuer to get a new VCC number, and such numbers cannot be tied with a particular merchant.

The rest of this paper is organized as follows. We review current attempts to secure credit card transactions online in Section 2. In Section 3, we analyze the necessary security properties for a VCC scheme and examine the solution space. We present our approach and discuss real-world considerations in Section 4, and give proofs of security in Section 5. We conclude with 6.

2   Related Work

There have been several attempts to reduce the usefulness of stolen card numbers. One widely adopted solution is security codes, also known as card verification values. A card verification value (CVV)[3] is a three- or four-digit value stored on the magnetic strip, allowing for verification of the card when read. A second code, known as the CVV2, is not stored on the magnetic strip of a card, and is printed on the back of a card instead. Merchants can ask for the CVV2 code during transactions, but are forbidden from storing them in their databases [17]. This has the potential to reduce the usefulness of a stolen credit card number in

---

[3] The card verification value goes by many different names according to different credit card companies

situations where "face-to-face" verification is not possible, such as online, mail order, and telephone transactions. While CVV2 required a change to the card acceptor infrastructure, it is now ubiquitous.

CVV2 doesn't provide a perfect solution. Not all merchants or card issuers require CVV2 to approve a transaction. Flaws in an online processing center or merchant may allow attackers to gain CVV and CVV2 codes that are stored either inadvertently or temporarily while awaiting authorization. Finally, similar checksum codes used in Cartes Bancaires cards has been compromised [7], and other vulnerabilities have been found in ATM cards in the past [2].

A second approach requires the cardholder to enter an additional username and password for online transactions. Two examples are "Verified by Visa" for Visa credit cards [16] and "MasterCard SecureCode" for MasterCard credit cards [13]. These solutions require changes to the card acceptor infrastructure, which are not yet commonplace. These schemes do not work with telephone or mail-order transactions, hindering usage.

A third solution is to use proxy or virtual transaction numbers instead of the real credit card number [10]. This scheme has been developed by Orbiscom and is in use by MBNA, Citi Bank and Discover [15, 8, 6]. When a cardholder wishes to make a transaction, she contacts her issuing bank and requests a temporary number. The bank generates a new number not currently in use, associates it with the cardholder's current account, and returns the proxy number. The cardholder can then use the proxy number in exactly the same manner as her real credit card number. The card acceptor does not need to know the credit card number given is a proxy number; the translation is performed by the card issuer.

The proxy numbers are an attractive solution for the problem of credit card theft. One disadvantage is that the proxy number must originate from the credit card issuing institution and must be done online. We propose a proxy number that can be generated offline, without any per transaction communication between cardholder and card issuer and no changes to the infrastructure currently in place.

3   Problem Description

3.1   How Credit Cards Work

Because we would like a solution that does not require any changes to the current infrastructure and protocols, we first examine how credit card processing currently works.

Credit Card Number Format A credit card number is a maximum of nineteen digits that can be broken into three pieces: issuer, account number, and checksum. Most numbers (including Visa, MasterCard, and Discover) are sixteen digits while AmericanExpress numbers are fifteen. The first six digits make up the issuing bank and the last single digit is the Luhn check digit. The Luhn code is a one-digit checksum of the credit card number which can be calculated and verified by anyone [20]. This yields a maximum of twelve digits for the account

number. As most credit cards have 16 or 15 digits, the limit of the account number is 9 or 8, respectively. This is a limit we have to take into consideration as we want to generate virtual card numbers that work with the current infrastructure.

Parties in Credit Card Processing Credit card transactions involve several parties. The three of interest are: *cardholder*, *card issuer*, and *merchant* (card acceptor).
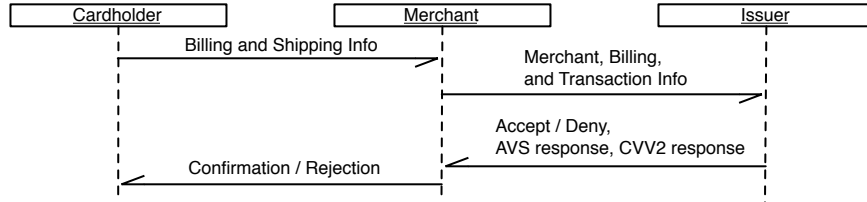

Fig. 1. Traditional Credit Card Processing

Credit Card Processing Parameters Figure 1 shows the card processing steps. The cardholder sends *credit card information* including name, billing address, account number, expiration date, and CVV2 to the merchant. The merchant send this together with *merchant information*, (which is configured when a business gains merchant status, and includes the merchant's bank and account numbers, merchant name and number), and *transaction information* (including the date and time, the amount of the transaction, a merchant-specified order number, and often specific information such as the point of sale device used) to the issuer.

The issuer may return several pieces of information to the merchant (e.g. authorization or rejection, address verification service (AVS) and CVV2 match responses). AVS tells the merchant how well the billing address supplied by the cardholder matches the billing address on record. A rejection notice overrides any decision the merchant may make to accept the transaction, while the treatment of AVS and CVV2 responses are up to the discretion of the merchant [17].

3.2 Security Properties for VCC Schemes

In a VCC scheme, the cardholder is able to generate a VCC number that is bound to a single transaction, or with a single merchant and maximum transaction amount. We require such a scheme to have the following properties.

1. *Complete* - Any cardholder can generate a VCC number from her credit card account number, the transaction information and/or limitation on usage of the VCC, and any other information the cardholder may have.
2. *Sound* - Given the public transaction information and VCC number, the card issuer is able to uniquely identify the associated account.
3. *Account Hiding* - Knowing the public transaction information and the VCC number, no adversary has a non-negligible advantage in recovering the original credit card account number.

This is motivated by the original motivation of having VCC numbers, that is, to hide the actual credit card numbers.

4. *Forgery Resistant* - Knowing an account number and some virtual credit card transaction information associated with the account, no adversary has a non-negligible advantage in forging a valid VCC associated with this account.

   Even with a VCC scheme, the original credit card number may still be stolen, because customers may choose to use the original card number in some transactions and because of card loss. We would like to ensure that a stolen card does not enable one to easily construct valid VCC numbers.

In other words, we are concerned with two types of threats: finding out the cardholder's original account number through transaction information involving VCCs, and the generation of valid VCC numbers if an attacker obtains account numbers. We are not concerned with malicious merchants who attempt to abuse a VCC (such as multiple submissions), as such threats are dealt with by existing dispute resolution procedures and laws.

Note that when VCC numbers are generated online by the card issuer, the sound and complete properties are no longer needed. One solution that would satisfy the other two properties is to randomly generate account numbers until encountering one that has not already been used. Such a solution would not work when we allow VCC numbers to be generated offline.

3.3 Examining the Solution Space

Our desired security properties impose limitations on the potential solution space. The sound property states that the card issuer *must* be able to recover the account number from the VCC transaction information, whereas the account hiding property states that an attacker *must not* be able to recover the account number. This indicates that the card issuer must know something that the attacker doesn't know. Such a secret can be shared between the cardholder and the card issuer, or known only by the card issuer.

One potential solution is to use the account number to derive a secret key, and to use some keyed-MAC of a message containing the transaction information to generate the VCC number and the CVV2 code for the VCC. There are two problems with this solution. First, this violates the forgery resistant property, because knowing the account number enables one to forge virtual credit card numbers. Second, the account hiding property can also be broken easily, because an attacker can perform an exhaustive search over the space of valid account numbers. Credit card numbers are highly structured and have a small space, making exhaustive search attacks feasible.

One attempt to fix the above solution is to add additional secret information that is currently shared between a card issuer and a cardholder, such as social security number or mother's maiden name, to derive a secret key. Under this design, when an attacker obtains an account number, the attacker would still needs to know additional information to be able to construct a virtual credit card number. However, this design suffers from another weakness. An attacker who

somehow obtains one's account number can use an exhaustive search attack to try to recover other secrets that are used in the process of generating the key. Such a design protects account numbers at the cost of increasing danger of revealing this other sensitive information, which is arguably more sensitive than credit card numbers. We thus choose not to adopt this design.

Another potential approach is to use public key cryptography. This has the advantage of eliminating the need for a shared secret between each cardholder and the card issuer. The card issuer would have a public key, and the cardholders would encrypt their account numbers with the issuer's public key. There are several problems with this approach. This does not satisfy the forgery resistant property, as anyone knowing the account number and the issuer's public key can generate a valid VCC number. Also, most public key cryptography systems produce ciphertexts much larger than the bits allowed by the space of virtual credit card numbers, typically on the order of 160-1024 bits and above. As a VCC number plus the CVV2 code may contain only 12 digits, the ciphertext must be truncated, making decryption infeasible.

We thus decided to use a design where each cardholder shares a secret with the card issuer for each account, and this secret is beyond the long-term secrets (such as an SSN or mother's maiden name) already shared between a card issuer and a cardholder.

4 Our Proposed Scheme

4.1 A Dynamic Virtual Credit Card Scheme

We assume the cardholder already has an account with the card issuer. The card issuer knows the cardholder's name and address, which we shall call the billing information, B. The card issuer will have provided the cardholder with an account number, C. Finally, they negotiate a shared secret, such as a password P. Note that the cardholder may already have a password through web access to their account information. A bank can choose to use this password or a different password for the VCC scheme. The advantage of using one password is ease of use. The disadvantage is that if an attacker gets access to the VCC number, then the attacker can use dictionary attack to try to recover the password.

In the description of our scheme below, we use two functions: H, a function that generates a key from a shared secret, and F, which can be thought as a keyed MAC function. This description is for generating a one-time VCC number, which can be used for a single transaction. We will describe how to generate a usage-limited VCC number in Section 4.2.

Generation The cardholder will:

- Choose an expiration date, E, for the virtual card. This is usually the current month.
- Generate a string for the transaction, $\sigma = E||B||M||T$, where M is merchant information, and T is transaction amount.

- Generate the shared key $K = \mathsf{H}(C\|P)$
- Calculate $V = \mathsf{F}_K(\sigma) \bmod 10^n$, in which n is the length of the account number plus the length of the CVV2 code.
- Divide V into V_1 and V_2. Prepend the card issuer code to V_1 and append a valid Luhn code to get the VCC number. V_2 is the CVV2 code.

Verification To verify that a merchant submitted VCC number is valid for a given transaction, the card issuer will:

- Identify the original account C', using the billing information (name and address) supplied in the AVS.
- Find the password P' associated with the account C', and calculate the shared secret $K' = \mathsf{H}(C'\|P')$.
- Calculate $V' = \mathsf{F}_{K'}(\sigma') \bmod 10^n$, using σ' from the merchant supplied values.
- If the submitted VCC number and CVV2 code match V', then process the transaction as usual, otherwise reject the transaction.

The above scheme is complete, as any cardholder can generate a VCC number. It is sound assuming that an account number can be uniquely identified given the name and address of a cardholder. In section 4.2 we discuss how to relax this restriction. In section 5.4 we show that using any pseudorandom function for F and H will satisfy the account hiding and forgery resistant properties.

We have written a prototype implementation of the card generation process in J2ME that is lightweight, fast, and capable of running on a wide variety of hardware, including cellular phones. In our current implementation, we use $\mathsf{SHA1}$ for the function H and $\mathsf{HMAC\text{-}SHA1}$ for the function F.

4.2   Real World Considerations

Multi-use VCC Numbers In some situations, a cardholder may wish to generate a VCC number that can be used for multiple transactions with one merchant, for example in systems such as PayPal or Amazon.com's 1-Click. In this case, the cardholder may want to set a credit limit lower than the limit of the account. As we would like to use the existing infrastructure, the credit limit chosen by the cardholder must be encoded in the VCC number.

Since we have a limited message space, we cannot allow all possible limits. Our design is to use one digit ℓ (we call this the VCC type) to encode whether this is a one-use card number, and if not, what is the credit limit. For example, $\ell = 0$ means single transaction, $\ell = 1$ means a limit of \$50, $\ell = 2$ means a limit of \$100, and so on. Using one digit, we can accommodate 9 different credit limit values. The VCC type digit can be in the VCC number (or the CVV2 code, if almost all merchants use it). Note that this digit must be appropriately encrypted, so that the credit limit cannot be learned by an attacker who gets the VCC number. To accommodate this, we change the design so that the VCC type digit is not generated from $V = \mathsf{F}_K(\sigma) \bmod 10^n$. Instead, we use bits in $\mathsf{F}_K(\sigma)$ that have not been used in generating V to randomly select a permutation π over $\mathbb{Z}_{10}$, and use $\pi(\ell)$ as the VCC type digit.

Collisions Between Actual and Virtual Credit Numbers The sets of possible actual and virtual credit card numbers do not need to be disjoint, under the condition that the VCC scheme is used only when AVS information is provided to the card issuer. If a merchant does not provide AVS information, we must assume we are given an actual card number. If AVS information is provided, then we assume we can uniquely identify the cardholder's account information. There are now two possibilities: either the card number and CVV match the real card, or they do not. If they do not, we process the card as a VCC number. If they do match, then the card number given was either actual, or $C \equiv V$ for the given transaction; neither case violates the soundness or completeness properties, and a second account cannot be incorrectly charged. The attack in which an adversary provides the AVS information of someone else's account and tries to generate a valid VCC number is no easier than the attack of guessing someone else's credit card number and using it, and can be handled by current dispute resolution procedures.

Non-Unique Name-Address Pairs Our scheme relies on the assumption that each name-address pair uniquely identify an account number. When this is not possible, then the probability that the VCC number generated using a second account also matches is about $1/10^n$, where n is the number of digits used in the VCC scheme. There are several approaches to enable us to relax this assumption. One approach is to reject a VCC when a collision occurs, in which case the client generates another VCC, taking a sequence number as an additional input. The probability that a collision occurs after a few rounds is extremely small. Another approach is to change the scheme so that the CVV2 code of the actual credit card is used to as the CVV2 code for the VCC. The bank thus only needs to ensure that name, address, and the CVV2 code together uniquely identify an account. This allows one name-address pair to have multiple accounts.

5 Security

We now present formal definitions of security for a virtual credit card scheme and prove our proposed scheme is secure.

5.1 Security Model

We use the following notations. We say that $\mu(k)$ is a negligible function, if for every polynomial $p(k)$ and for all sufficiently large k, $\mu(k) < 1/p(k)$. We say $\nu(k)$ is overwhelming if $1 - \nu(k)$ is negligible. If S is a probability space, then the probability assignment $x \leftarrow S$ means that an element x is chosen at random according to S. If S is a finite set, then $x \leftarrow S$ denotes that x is chosen uniformly from S. Let A be an algorithm, we use $y \leftarrow A(x)$ to denote that y is obtained by running A on input x. In the case that A is deterministic, then y is unique; if A is probabilistic, then y is a random variable. Let p be a predicate and $A_1, A_2, \ldots, A_n$ be n algorithms then $\Pr\left[\{x_i \leftarrow A_i(y_i)\}_{1 \leq i \leq n} : p(x_1, \cdots, x_n)\right]$

denotes the probability that $p(x_1, \cdots, x_n)$ will be true after running sequentially algorithms $A_1, \ldots, A_n$ on inputs $y_1, \ldots, y_n$.

We next describe our security model for VCC schemes. Let $C \in \{0,1\}^{\ell_c}$ be the original credit card number and $V \in \{0,1\}^{\ell_c}$ be the virtual credit card number, where ℓ_c is the bit-length of the credit card number. Let $A \in \{0,1\}^{\ell_a}$ be the account information, $B \in \{0,1\}^{\ell_b}$ be the customer billing information, $T \in \{0,1\}^{\ell_t}$ be the transaction information, and $S \in \{0,1\}^{\ell_s}$ be the secret that is known to the cardholder and the bank; where ℓ_a is the (maximum) length of the account information, ℓ_b is the length of the billing information, ℓ_t is the length of the transaction information, and ℓ_s is the length of the secret. The secret S has two parts: the first part is the original credit card C, and the second part is a password $P \in \{0,1\}^{\ell_p}$, where ℓ_p is the length of the password and $\ell_s = \ell_c + \ell_p$.

There are three deterministic algorithms in the VCC scheme: Identify, VirGen, and Verify. The algorithm Identify : $\{0,1\}^{\ell_b} \times \{0,1\}^{\ell_c} \to \{0,1\}^{\ell_a}$ is the personal account identification algorithm, i.e., given B and V, Identify(B, V) outputs an account information A. The algorithm VirGen : $\{0,1\}^{\ell_t} \times \{0,1\}^{\ell_s} \to \{0,1\}^{\ell_c}$ is the virtual credit card generation algorithm, i.e., given T and S, VirGen(T, S) outputs a virtual credit card V. The algorithm Verify : $\{0,1\}^{\ell_t} \times \{0,1\}^{\ell_s} \times \{0,1\}^{\ell_c} \to \{\mathsf{true}, \mathsf{false}\}$ is the virtual credit card verification algorithm, i.e., given T, S, V, Verify(T, S, V) outputs either true or false.

The virtual credit card scheme has the following phases:

- Customer-Bank initialization: In this phase, the customer first sends the billing information B to the bank. The bank then creates the original credit card number C and the account information A for the customer. The bank and customer jointly choose the password P and set the secret $S = C||P$.
- Customer-Merchant interaction: In this phase, the customer and merchant jointly determine the transaction information T. The customer then computes $V = \mathsf{VirGen}(T, S)$, and sends V and B to the merchant.
- Merchant-Bank interaction: In this phase, the merchant sends T, B, and V to the bank. The bank uses Identify(B, V) to identify the customer account A, then obtains the shared secret S based on A, and finally computes Verify(T, S, V). If the output of the Verify algorithm is false, the bank rejects the transaction.

5.2 Security Properties

The virtual credit card must satisfy the sound property, the complete property, the security against forgery property, and the security against account recovery property:

The *sound property* can be stated as:

$$\Pr \left[\begin{array}{c} B \leftarrow \{0,1\}^{\ell_b}, T \leftarrow \{0,1\}^{\ell_t}, S \leftarrow \{0,1\}^{\ell_s}, V \leftarrow \mathsf{VirGen}(T, S), \\ x \leftarrow \mathsf{Identify}(B, V) : x = \bot \end{array} \right] = 0$$

where $\bot$ is a symbol that represents empty output. In other words, given the billing information and the virtual credit card number, we can always identity the corresponding account number.

The *complete property* can be stated as:

$$\Pr\left[A \leftarrow \{0,1\}^{\ell_a}, T \leftarrow \{0,1\}^{\ell_t}, S \leftarrow \{0,1\}^{\ell_s}, x \leftarrow \mathsf{VirGen}(T,S) : x = \bot\right] = 0$$

In other words, we can always generate a VCC number given the transaction information and the secret.

The *secure against forgery* can be stated as follows. We consider forgery under adaptive chosen-message attacks. Our scheme is secure against forgery if an adversary cannot win the following game between a challenger and the adversary:

1. Setup. The challenger runs a setup algorithm to output an account information A and a secret S. The challenger sends the account information to the adversary.
2. Queries. Proceeding adaptively, the adversary requests VCC numbers for at most q messages (transactions) of her choice $T_1, \ldots, T_q \in \{0,1\}^{\ell_t}$. The challenger responds to each query with a virtual credit card number $V_i = \mathsf{VirGen}(T_i, S)$.
3. Outputs. Eventually, the adversary outputs a pair (T, V) and wins the game if T is not any of $T_1, \ldots, T_q$ and $\mathsf{Verify}(T, S, V) = \mathsf{true}$.

Since the space for the virtual credit card is rather small, the adversary can do random guessing. Our security definition (in the following equation) states that the adversary cannot do better than random guessing:

$$\Pr\left[\begin{array}{c} S \leftarrow \{0,1\}^{\ell_s}, (T,V) \leftarrow \mathcal{A}(T_1, V_1, \ldots, T_q, V_q) : \\ \mathsf{Verify}(T, S, V) = \mathsf{true} \end{array}\right] \le 2^{-\ell_c} + \mu(t).$$

where $\mu(t)$ is a negligible function in time t.

The *secure against account recovery property* can be stated as follows. We consider account recovery under chosen-message attacks. Our scheme is secure against account forgery if an adversary cannot win the following game between a challenger and the adversary:

1. Setup. The challenger runs a setup algorithm to output an account information A and a secret S, which comprises of a credit card number C and a password P. The challenger sends the account information to the adversary.
2. Queries. Proceeding adaptively, the adversary requests VCC numbers for at most q messages (transactions) of her choice $T_1, \ldots, T_q \in \{0,1\}^{\ell_t}$. The challenger responds to each query with a VCC number $V_i = \mathsf{VirGen}(T_i, S)$.
3. Outputs. Eventually, the adversary outputs C, the original credit card number.

Since the space for the credit card number is rather small, the adversary can do random guessing. Our security definition (in the following) states that the adversary cannot do better than random guessing:

$$\Pr\left[\begin{array}{c} C \leftarrow \{0,1\}^{\ell_c}, P \leftarrow \{0,1\}^{\ell_p}, S = C||P, \\ C' \leftarrow \mathcal{A}(T_1, V_1, \ldots, T_q, V_q) : C' = C \end{array}\right] \le 2^{-\ell_c} + \mu(t)$$

where $\mu(t)$ is a negligible function in time t.

5.3 Our Abstracted Scheme

We now give an abstract of our scheme that is presented in Section 4, then prove our scheme is secure in the next subsection. Let $H : \{0,1\}^{\ell_s} \rightarrow \{0,1\}^{\ell_k}$ be a collision-free hash function, and $F : \{0,1\}^{\ell_k} \times \{0,1\}^{\ell_t} \rightarrow \{0,1\}^{\ell_k}$ be a family of functions that can be modeled as a pseudorandom function (PRF), where ℓ_s is the length of the secret, ℓ_t is the length of the transaction information, and ℓ_k is the key length of F. Here we also use ℓ_k to denote the output length of the pseudorandom function F and the hash function H.

- Identify(B, V): This algorithm takes $B \in \{0,1\}^{\ell_b}$ and $V \in \{0,1\}^{\ell_c}$ as input and outputs the account information A. Here, we assume the bank keeps a database of the clients' information, including billing information and account information. Given the billing information B, the bank can lookup the database to identify the corresponding account information A.
  Note that in this abstraction, V is not used in the algorithm Identify since we assume there is only one bank. In practice, V is used to identify the bank who issues the credit card, and then B is used to identify the account in that bank.
- VirGen(T, S): This algorithm takes the transaction information T and the secret S as input, and outputs the VCC number V. It performs the following steps:
  1. Computes $K = H(S)$.
  2. Sets V to be the last ℓ_c bits of $F_K(T)$[4].
- Verify(T, S, V): This algorithm takes the transaction information T, the secret S, and the virtual credit card number V as input, and outputs either true or false. It performs the following steps:
  1. Computes $V' = $ VirGen(T, S).
  2. If $V = V'$ outputs true, otherwise outputs false.

In our proposed scheme, H is SHA1 and F is an HMAC or a CBC-MAC scheme. HMAC was originally shown to be a PRF under the assumption that the underlying hash function was collision resistant [4], and later if the underlying hash function was a PRF [3]. It should be noted that recent work has shown HMAC to not be a PRF when instantiated with certain hash functions, such as MD4, MD5, SHA0 and SHA1 [11]. Differentials for MD4, MD5, SHA0 and SHA1 discovered by Biham et al. and Wang et al. were used to produce a distinguisher of HMAC and a random function when instantiated with these hash functions. While they were able to produce a distinguisher for HMAC-SHA1, they were only able to do so when SHA1 was reduced to 43 rounds, and a probability of $2^{-73.4}$ (more than the general attack of 2^{-80}) and a data complexity of $2^{154.9}$ (more than the general attack of 2^{80}) [14]. We do not feel their results adversely affect our work.

---

[4] Note that in our proposed scheme, $V = F_K(T) \bmod 10^n$ where n is the number of digits in the virtual credit card. In our security model, since we assume V to be a value of length ℓ_c, we set $V = F_K(T) \bmod 2^{\ell_c}$. This simplication will not affect our security proof.

5.4 Security Proofs

In this subsection, we formally prove that our virtual credit card scheme is secure under the definition in Section 5.1. The sound property of our scheme is guaranteed if the bank maintains a proper customer database. Our scheme is also complete as the algorithm VirGen always returns an output. We now focus on the secure against forgery property and the secure against account recovery property.

Theorem 1. *Our virtual credit card scheme is secure against forgery.*

Proof. Let $F : \{0,1\}^{\ell_k} \times \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_k}$ be a pseudorandom function, we now build a family of functions $F' : \{0,1\}^{\ell_k} \times \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_c}$ as follows: Given $K \in \{0,1\}^{\ell_k}$ and $T \in \{0,1\}^{\ell_t}$,

$$F'(K,T) = F(K,T) \bmod 2^{\ell_c}.$$

It is clear that $V = F'(K,T) = F'(H(S),T)$. In the next two claims, we first show that if F is a pseudorandom function, then F' is a pseudorandom function as well. We then show that if F' is a pseudorandom function, then our scheme is secure against forgery.

Claim. If $F : \{0,1\}^{\ell_k} \times \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_k}$ is a pseudorandom function, then $F' : \{0,1\}^{\ell_k} \times \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_c}$, which is defined above, is also a pseudorandom function.

Proof. To prove this claim, we first review the definition of pseudorandom functions. Pseudorandomness of the function family F measures the ability of a distinguisher to tell whether its given oracle is a random instance of F or a random function of $\{0,1\}^{\ell_t}$ to $\{0,1\}^{\ell_k}$. For a distinguisher $\mathcal{A}$, let

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A}) = \Pr\left[f \leftarrow F : \mathcal{A}^f = 1\right] - \Pr\left[f \leftarrow \mathrm{Rand}^{\ell_t \to \ell_k} : \mathcal{A}^f = 1\right].$$

For any integer $q, t \geq 0$, let

$$\mathbf{Adv}_F^{\mathrm{prf}}(q,t) = \max\left\{\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{A})\right\},$$

where the maximum is over all distinguishers $\mathcal{A}$ that make at most q oracle queries and use at most t running time. Intuitively, if F is a pseudorandom function, then F', the last ℓ_c bits of F, should also be a pseudorandom function. We prove this by showing that

$$\mathbf{Adv}_{F'}^{\mathrm{prf}}(q,t) \ \leq \ \mathbf{Adv}_F^{\mathrm{prf}}(q,t).$$

Let $\mathcal{A}$ be a distinguisher that is given an oracle for a function $f' : \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_c}$. Assume $\mathcal{A}$ invoked at most q queries and ran at most t time. We can design a distinguisher $\mathcal{B}$ for F versus $\mathrm{Rand}^{\ell_t \to \ell_k}$ such that

$$\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{B}) \ = \ \mathbf{Adv}_{F'}^{\mathrm{prf}}(\mathcal{A}).$$

Given a distinguisher $\mathcal{A}$, we can build a distinguisher $\mathcal{B}$ as follows. Recall that, given an oracle for a function $f : \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_k}$, $\mathcal{B}$ must determine whether f is chosen randomly from F or from $\mathrm{Rand}^{\ell_t \to \ell_k}$.

1. When $\mathcal{A}$ asks its oracle for query T_i, for $i \in \{1, \ldots, q\}$, $\mathcal{B}$ queries its own oracle using T_i and obtains $f(T_i)$. $\mathcal{B}$ computes $f_i'(T_i) = f(T_i) \bmod 2^{\ell_c}$ and return $f_i'(T_i)$ back to A.
2. If $\mathcal{A}$ outputs 1 then $\mathcal{B}$ returns 1, otherwise $\mathcal{B}$ returns 0.

We show that

$$
\begin{aligned}
\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{B}) &= \Pr\left[f \leftarrow F : \mathcal{B}^f = 1\right] - \Pr\left[f \leftarrow \mathrm{Rand}^{\ell_t \to \ell_k} : \mathcal{B}^f = 1\right] \\
&= \Pr\left[\begin{array}{c} f' \leftarrow F' : \\ \mathcal{A}^{f'} = 1 \end{array}\right] - \Pr\left[\begin{array}{c} f \leftarrow \mathrm{Rand}^{\ell_t \to \ell_c}, f'(T) = f(T) \bmod 2^{\ell_c} : \\ \mathcal{A}^{f'} = 1 \end{array}\right] \\
&= \Pr\left[f' \leftarrow F' : \mathcal{A}^{f'} = 1\right] - \Pr\left[f' \leftarrow \mathrm{Rand}^{\ell_t \to \ell_c} : \mathcal{A}^{f'} = 1\right] \\
&= \mathbf{Adv}_{F'}^{\mathrm{prf}}(\mathcal{A})
\end{aligned}
$$

In the above equations, it is clear that if a function f is randomly chosen from $\mathrm{Rand}^{\ell_t \to \ell_k}$, then f', which outputs the last ℓ_c bits of f, is a random function from $\mathrm{Rand}^{\ell_t \to \ell_c}$. We finish the proof by showing:

$$
\mathbf{Adv}_{F'}^{\mathrm{prf}}(q, t) = \max\left\{\mathbf{Adv}_{F'}^{\mathrm{prf}}(\mathcal{A})\right\} \leq \max\left\{\mathbf{Adv}_F^{\mathrm{prf}}(\mathcal{B})\right\} = \mathbf{Adv}_F^{\mathrm{prf}}(q, t)
$$

Claim. If $F' : \{0,1\}^{\ell_k} \times \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_c}$ is a pseudorandom function, then our scheme is secure against forgery.

Proof. In this proof, we show that if there exists a forger who can forge a virtual credit card, then we can build a distinguisher to distinguish F' from random functions. Our proof is similar to the proof in [5]. More formally, let

$$
\begin{aligned}
\mathbf{Adv}_{F'}^{\mathrm{vc}}(q, t) &= \max\left\{\Pr\left[\begin{array}{c} S \leftarrow \{0,1\}^{\ell_s}, (T, V) \leftarrow \mathcal{A}(T_1, V_1, \ldots, T_q, V_q) : \\ \mathsf{Verify}(T, S, V) = \mathsf{true} \end{array}\right]\right\} \\
&= \max\left\{\Pr\left[\begin{array}{c} S \leftarrow \{0,1\}^{\ell_s}, (T, V) \leftarrow \mathcal{A}(T_1, V_1, \ldots, T_q, V_q) : \\ F'(H(S), T) = V \end{array}\right]\right\}
\end{aligned}
$$

We want to show that

$$
\mathbf{Adv}_{F'}^{\mathrm{vc}}(q, t) \leq \mathbf{Adv}_{F'}^{\mathrm{prf}}(q, t) + 2^{-\ell_c}.
$$

Let $\mathcal{A}$ be a forger who tries to forge a virtual credit card. Assume $\mathcal{A}$ invoked at most q queries and ran at most t time. We can design a distinguisher $\mathcal{B}$ for F' versus $\mathrm{Rand}^{\ell_t \to \ell_c}$ such that

$$
\mathbf{Adv}_{F'}^{\mathrm{prf}}(\mathcal{B}) \geq \mathbf{Adv}_{F'}^{\mathrm{vc}}(\mathcal{A}) - 2^{-\ell_c}.
$$

Given a forger $\mathcal{A}$, we can build a distinguisher $\mathcal{B}$ as follows. Recall that, given an oracle for a function $f' : \{0,1\}^{\ell_t} \to \{0,1\}^{\ell_c}$, $\mathcal{B}$ must determine whether f' is chosen randomly from F' or from $\mathrm{Rand}^{\ell_t \to \ell_c}$.

1. When $\mathcal{A}$ asks its oracle for query T_i, for $i \in \{1, \ldots, q\}$, $\mathcal{B}$ answers with $V_i = f'(T_i)$.
2. $\mathcal{A}$ outputs a (T, V) pair such that $T \notin \{T_1, \ldots, T_q\}$.
3. If $V = f'(T)$ then return 1 else return 0.

It is easy to see that

$$\Pr\left[f' \leftarrow F' : \mathcal{B} = 1\right] = \mathbf{Adv}_{F'}^{\mathrm{vc}}(\mathcal{A})$$

$$\Pr\left[f' \leftarrow \mathrm{Rand}^{\ell_t \to \ell_c} : \mathcal{B} = 1\right] \geq 2^{-\ell_c}$$

Subtract the above two equations, we obtain $\mathbf{Adv}_{F'}^{\mathrm{prf}}(\mathcal{B}) \geq \mathbf{Adv}_{F'}^{\mathrm{vc}}(\mathcal{A}) - 2^{-\ell_c}$. The above reduction shows that the probability of a successful forgery is less than the probability of distinguishing F' from a random function plus $2^{-\ell_c}$. Therefore, our scheme is secure against forgery if F' is a pseudorandom function.

Theorem 2. *Our virtual credit card scheme is secure against account recovery.*

Proof. Recall that the algorithm VirGen takes $T \in \{0,1\}^{\ell_t}$ and $S \in \{0,1\}^{\ell_s}$ as input, computes $K = H(S)$ and $V = F'_K(T)$, and outputs V. Let $S = C \| P$, where C is the original credit card and P is a password. We want to show that no adversary can compute C with a probability more than random guessing. Note that the adversary can query the oracle multiple times with T_i and obtains the corresponding V_i, so that she can try to learn K then learn C. What we prove next is that, even if the adversary learns K, the adversary cannot guess C correctly, i.e.,

$$\Pr\left[C \leftarrow \{0,1\}^{\ell_c}, P \leftarrow \{0,1\}^{\ell_p}, K = H(C\|P), C' \leftarrow \mathcal{A}(K) : C' = C\right] \leq 2^{-\ell_c} + \mu(t)$$

for any polynomial-time adversary $\mathcal{A}$, where $\mu(t)$ is a negligible function in t. This is quite obvious given H is a one-way hash function, that is,

$$\Pr\left[M \leftarrow \{0,1\}^*, K = H(M), M' \leftarrow \mathcal{A}(K) : M' = M\right] \leq \mu(t)$$

Given $K = H(C\|P)$, the adversary can either do a random guessing, i.e., pick $C' \leftarrow \{0,1\}^{\ell_c}$ or find the pre-image of K. Therefore, the overall success probability for the adversary is bounded by $2^{-\ell_c} + \mu(t)$.

6   Closing Remarks

Theft of stored credit card information is an increasing threat to e-commerce. We propose the concept of dynamic virtual credit card (VCC) numbers to mitigate this threat. Dynamic VCC numbers can be generated by credit card holders without online contact with the issuing bank. Using VCC numbers requires no change to the merchant's credit card processing infrastructure and reduces the damage caused by stolen credit card numbers. We have identified the security requirements for VCC schemes and proposed a scheme that offers flexibility as well as security. We have also discussed how to address issues related to real-world deployment of the scheme, and proved that our scheme is secure under commonly used cryptographic assumptions. We believe this is a viable solution to the problem of credit card information theft.

References

1. Hotels.com credit-cad numbers stolen. *CNN Money*, June 2 2006. `http://money.cnn.com/2006/06/02/news/companies/hotels.com_theft/index.htm`.
2. R. Anderson. Why cryptosystems fail. *Communications of the ACM*, 37(11):32–40, November 1994.
3. M. Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. Cryptology ePrint Archive, Report 2006/043, 2006. `http://eprint.iacr.org/`.
4. M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. *Advances in Cryptology — CRYPTO'96*, 1996.
5. M. Bellare, J. Kilian, and P. Rogaway. The security of the cipher block chaining message authentication code. *Advances in Cryptology — CRYPTO'94*, 1994.
6. Citigroup. Citi identify theft solutions: Virtual account numbers. `http://www.citibank.com/us/cards/cardserv/advice/van.htm`.
7. S. Dennis. French banks hacked, March 2000. `http://www.computeruser.com/newstoday/00/03/11/news4.html`.
8. Discover Bank. Discover card: Secure online account numbers. `http://www2.discovercard.com/deskshop?icmpgn=200512_dc_wp_hk_drpdwn_sec_nap_soa_2`.
9. J. Evers. Amazon unit loses credit card data to hackers. *InfoWorld*, March 6 2001. `http://www.infoworld.com/articles/hn/xml/01/03/06/010306hnbiblio.html?0306alert`.
10. D. C. Franklin and D. Rosen. Electronic online commerce card with transaction-proxy number for online transactions. Patent 5883810.
11. J. Kim, A. Biryukov, B. Preneel, and S. Hong. On the security of HMAC and NMAC based on HAVAL, MD4, MD5, SHA-0 and SHA-1. Cryptology ePrint Archive, Report 2006/187, 2006. `http://eprint.iacr.org/`.
12. J. Krim and M. Barbaro. 40 Million Credit Card Numbers Hacked. *Washington Post*, page A01, June 18 2005.
13. MasterCard. Mastercard securecode. `http://www.mastercard.com/securecode/`.
14. B. Preneel and P. C. van Oorschot. MDx-MAC and building fast MACs from hash functions. *Lecture Notes in Computer Science*, 963:1–14, 1995.
15. D. Transactions. Discover redoubles its commitment to single-use card numbers. `http://www.orbiscom.com/news9.php`.
16. Visa International Service Association. Visa security progam: Verified by visa. `https://usa.visa.com/personal/security/vbv/index.html`.
17. Visa International Service Association. Rules for visa merchants - card acceptance and chargeback management guidelines. Technical report, Visa International Service Association, 2005.
18. Visa International Service Association. Visanet fact sheets, 2006. `http://www.corporate.visa.com/md/fs/corporate/visanet.jsp`.
19. T. Weiss. Laptop with credit card info for 80,000 DOJ workers stolen. *ComputerWorld*, March 31 2005. `http://www.computerworld.com/governmenttopics/government/legalissues/story/0,10801,102146,00.html`.
20. J. Ziegler. Everything you ever wanted to know about CC's. `http://euro.ecom.cmu.edu/resources/elibrary/everycc.htm`.