Off-line generation of limited-use credit card numbers

Aviel D. Rubin Rebecca N. Wright

AT&T Labs – Research
180 Park Avenue
Florham Park, NJ, 07932 USA
{rubin, rwright}@research.att.com

Abstract. Recently, some credit card companies have introduced limited-use credit card numbers—for example, American Express's single-use card numbers and Visa's gift cards. Such limited-use credit cards limit the exposure of a traditional long-term credit card number, particularly in Internet transactions. These offerings employ an *on-line* solution, in that a credit card holder must interact with the credit card issuer in order to derive a limited-use token. In this paper, we describe a method for cryptographic *off-line* generation of limited-use credit card numbers. This has several advantages over the on-line schemes, and it has applications to calling cards as well. We show that there are several trade-offs between security and maintaining the current infrastructure.

1 Introduction

The proliferation of e-commerce on the Internet has not resulted in a wide diversity of on-line payment mechanisms. While novel schemes such as PayPal [7] have gained in popularity, most business to customer transactions still utilize standard credit card numbers over a Secure Socket Layer (SSL) connection [5]. SSL provides encryption so that data is not revealed in transit, and server authentication so that the merchant identity is confirmed to the customer. (While SSL provides for mutual authentication, most consumers do not have the necessary public key certificates for it and virtually all consumer-oriented Web merchants implement only server authentication.)

Unfortunately, despite the use of SSL, there is no guarantee that the user is not being fooled by a malicious merchant (c.f. [6]) or, at least in earlier versions of SSL, that an outside attacker might not be able to break the encryption [3]. There are several ways SSL can break down even if the encryption mechanism is not broken. Most users do not actually verify the certificate on a secure site. That is, most users simply look for the browser's indication that a page has been encrypted, such as Netscape's blue padlock, rather than actually looking at the certificate itself to verify that the merchant name in the certificate matches their expectations. Many users do not do check even for this encryption indicator. Furthermore, even if users do check certificates, it is relatively easy for just about anyone to obtain one. There are over 50 root certificate authorities' public keys

In *Proceedings of Financial Cryptography 2001* (FC2001)

in a typical Netscape browser, and many more in Internet Explorer. In addition, there are other ways that users can be fooled into thinking they are visiting an intended site when in fact they are at an attacker's site [4].

Besides the risk of exposure of card numbers during transit, there is also the risk of exposure of card numbers while stored at a merchant's site. There have recently been several high profile cases in the news where merchants' sites were broken into and stored credit card numbers were stolen (c.f. [10]). Even in the physical world, credit cards are exposed simply by being used. Fraudulent merchants or employees may sell or use their customers' credit card numbers, and attackers can look for discarded credit card receipts in trash bins.

By having a single credit card number that is reusably and indefinitely used as an authorization token, the traditional credit card system creates substantial risk for the credit card companies, who lose millions or billions of dollars a year due to fraud.[1] The big companies—Visa, Mastercard, and American Express—insulate their customers from risk by shouldering any loss above \$50 themselves; in many cases, even the \$50 charge to the customer is waived. Thus, there is great incentive for credit card companies to implement schemes that make it more difficult for credit card numbers to be compromised. Fraud reduction is also advantageous to customers and merchants because the cost of fraud results in higher transaction costs charged by credit card companies to merchants, which must in turn either be absorbed by the merchant or passed to customers. Additionally, customers whose cards are compromised must deal with the inconvenience of replacing their cards and the potentially devastating and difficult-to-correct effects on their credit ratings.

The Secure Electronic Transactions (SET) protocol was designed to protect credit card numbers from malicious parties, and even from merchants. Unfortunately, SET never took off. There was too much overhead required, and buy-in was needed from too many different parties. Credit cards over SSL, on the other hand, require no additional infrastructure, and are easy for users to understand. It is not surprising that this is currently the standard for business to consumer commerce.

1.1 Related work

Realizing the security problem in indefinitely reusable credit card numbers, credit card issuers have recently started to introduce limited-use credit card number solutions that can be layered over the existing infrastructure. American Express offers single-use credit cards and Visa offers limited-value gift credit cards. The design and architecture of another solution is presented by Shamir [9]. The main idea is to enable users to shop at existing Web merchant sites without exposing long-term credit card numbers, and without requiring changes to the Web pages. All of the existing solutions require users to have an on-line secure

[1] Although part of the authorization token eventually changes due to the expiration date, this is only infrequently (typically, once every one to three years) and furthermore it is easy to guess the subsequent expiration date from the current one.

interaction with the credit card issuer during or shortly before a purchase, in which a new single-use or limited-use credit card number, which we call a *token*, is obtained. The token is linked to the user's existing account, in that charges made with the token will be charged to the original account. Such tokens provide more security than standard reusable credit cards because even if they are learned by an attacker, they are either of no further use or of limited further use.

In the on-line setting used by these solutions, a card holder who wishes to make a purchase visits the Web site of the card issuer to obtain limited-use tokens. There, the card holder has the option to enter his name and account number, or perhaps a stronger method of authentication is performed, and then the card holder obtains a token to use for his purchase. The card issuer stores the token with the account, along with any restrictions on its use such as dollar amount or merchant name. When the token user shops at a merchant site, the token is entered into a Web form as if it were a traditional credit card number. From the merchant's point of view, the credit card is like any other. The merchant clears the credit card number with the issuing bank. The issuing bank then looks up the account and checks that the token has been stored with the account and is therefore valid.

There are several problems with the on-line setting. When the card holder obtains the token, the connection to the card issuer needs to be secured, typically by SSL, because the traditional long-term credit card number will be communicated over this connection. SSL places a performance burden on the server. Many simultaneous SSL connections could bring a server to its knees, and any solution involving a central SSL server does not scale well. Furthermore, a spoofed credit card company site could collect legitimate credit card numbers from unsuspecting users. In general, it is not a good idea for a site to exist with the sole purpose of collecting credit card numbers from people. It promotes bad habits and creates desirable targets. A simple attack against DNS and a certificate from any root CA is all an attacker needs to run a credit card collection site in this model.

1.2 Our work

In our work, we consider an off-line model, where limited-use tokens (including tokens limited to a single use) can be implemented from traditional credit cards without requiring that a user interact with the credit card issuer as part of every transaction. Like the on-line solutions, our solution is designed so that it can be layered on top of the existing e-commerce infrastructure without any change to the merchant's systems or the user's browser. Off-line protocols have the advantage that the card holder need not interact with the credit card issuer to create limited-use tokens, and in particular, no secure channel to the credit card issuer is required at the time of token generation. This is useful because it removes the reliance on authentication with SSL. In our solution, expenses are billed to the original card without exposure of the card number; token generation is off-line for the credit card holder. Off-line schemes have the advantage that they can be used even for purchases where there may not be user access to a computer network, such as purchases made over the telephone.

Limited-use tokens can be useful for features other than limiting risk. In our work, a token can have various restrictions associated with it: we can limit the number of uses of a token, its validity period, the set of recipients, the amount of money, and even the category of product for which it can be used. For example, a token might only be good for $100 worth of books from either Amazon or Barnes & Nobles during the first week of classes. Tokens might be used to enforce or keep track of a personal budget. A user could create a token with a particular monetary limit that can only be used in restaurants, and thus enforce a limit on how much she spends eating out; different token with a different monetary limits could be created for additional expense categories. A parent could create a token with special restrictions to give to a child in college. There are all sorts of creative gift possibilities with credit card tokens, such as a token good for three days of restaurant and entertainment expenses up to $1000.

In Section 2, we present design requirements for token-based solutions. We present a proposed solution in Section 3, including an intuitive and easy-to-use user interface for the token-derivation application, and we discuss some security issues in Section 4. In Section 5, we discuss the use of limited-use tokens for telephone calling cards. We conclude in Section 6.

2 Requirements

In order to be successfully deployed, a system for generating and using limited-use tokens must satisfy several requirements.

Ease of use The system should not place unreasonable burden on the users. This point cannot be overstated. If a system is bulky or requires users to learn new techniques and to adopt new ways of shopping, then it is likely to fail. For instance, users should only be required to type fairly short strings that consist of alphanumeric characters and are not case-sensitive. The use of more general strings might be viable if they need only be cut and pasted, but even then the strings should not be too long, since strings that "wrap" from one line to the next are not always handled correctly by cutting and pasting.

Interoperability The system should be layerable on top of existing infrastructure. We should be able to deploy it without requiring merchants to change their Web sites. In particular, this means the tokens should preferably be 16 characters long, so that users can enter them into the existing credit card number field on Web forms; they may even be required to be strictly numeric due to type-checking on existing merchant Web forms.

Limited transparency It should be clear to the card holders that they are not sending long-term credit card numbers to the merchant, if that is not the case. For example, if one wishes to design a system that intercepts merchant Web forms and automatically replaces traditional credit card numbers by limited-use tokens, care must be taken to ensure it is done in such as way that the user understands that the card number is not being transmitted over the network to the merchant.

Security A limited-use token should not be usable beyond its intended uses, whether by the user, the merchants, or an attacker. Similarly, it should not be possible for an attacker to successfully generate and use tokens whose expenses are charged to someone else's credit card.

We use these requirements to guide our design process. Each requirement represents an objective that is difficult to quantify, but the more we adhere to the spirit of the requirements, the more likely it is our system will be adopted.

In the off-line model, card holders generate tokens on their own. Since it is not generally reasonable to assume personal computers are safe from hacking by outsiders, we assume that the credit card holder has an auxiliary tamper-resistant computing device that can protect secrets and has a reliable clock. For example, this device might be a PDA, such as a Palm Pilot or Windows CE device, or a PC equipped with a smart card reader or other tamper-resistant hardware. We assume that the owner can control access to the data on it by physical or cryptographic means.

Throughout the rest of this paper, we refer to the entity with whom people have credit card accounts as a *card issuer*.[2] We refer to a traditional credit card number held by a person as their *account number*, and we refer to the person as a *card holder*. The intended user of a limited-use token (who may be either the card holder or another person such as a gift recipient) is referred to as the *token user* or simply the *user*. We refer to the PDA or computer that is used for the off-line computation of tokens as the *device*.

3 Off-line token generation

In this section, we present our proposed system, which consists of two parts. The first part, discussed in Section 3.1, is the token generation application. The second part, discussed in Section 3.2, is the protocol for using generated tokens.

3.1 Token generation

We have already discussed several applications of limited-use tokens. In order to support as many of these applications as possible, our goal is to represent as many restrictions as possible while still meeting our ease-of-use and security requirements. To this end, we propose the following design.

Restrictions In our system, there is a set of possible restrictions that is universal. For example, the monetary restrictions can be $20, $50, $75, $100, $150, $200, $300, $500, $1000, $5000. The categories of expense can be food, books,

[2] For simplicity, we do not in this paper separate out other entities that may be involved in transaction processing such as a merchant acquirer, but rather assume that the merchant talks directly to the card issuer. In order to implement this in the real system, intermediaries would be required to forward the appropriate messages.

travel, entertainment, luxury, clothing, electronics, etc. The validity periods can be one hour, four hours, twelve hours, one day, three days, a week, a month. It might also be desirable to include the identity of a merchant in the restrictions. Since we want to limit the size of the description, we suggest allowing the user only to specify the first few characters of the merchant name.

In the token generation application, all of the possible values are enumerated. Then, the values are laid out in a table, and the plaintext of the token consists of an index into the table. For readability, selected restrictions can be represented as an enumeration of the various restrictions. This is analogous to the way cryptographic algorithms and parameters are listed in SSL ciphersuites. For example, a setting of restrictions on a credit card might be:

one-hundred-dollars–books–one-week–same-store–two-uses

As discussed in Section 3.2, tokens are formed by encrypting the selected restrictions. If we want tokens to look like traditional credit card numbers, tokens must be 16 characters long, possibly restricted to 16 digits since non-numeric characters may break some Web sites that check credit card numbers to make sure they are digits. The symmetric cipher may require that the plaintext token be padded, and we also add a value for timestamping and uniqueness (as discussed in Section 3.2 below). Further, in traditional credit card numbers, the first four digits are typically used to represent a bank code, and the last digit is usually a checksum. Hence, if we wish to stick with 16 character tokens, then we can only use 11 characters to represent the restrictions. This means we can represent somewhere between 10^{11} and 36^{16} combinations of restrictions. While it seems that 10^{11} is more than enough combinations of restrictions for most interesting applications, there are also security issues based on the size of the space of possible tokens. We discuss security issues further in Section 4.

User interface User interface is crucial in any system that involves many users, especially if their level of experience with computers varies widely. We envision a set of pull-down menus or other graphical interface, independent of the particular device, for selecting from a predetermined set of restrictions. The user's device must contain the table of possible restrictions. Every time the user generates a token, the application can present the user with a list of choices, say via a pull-down menu for the restrictions.

In order to allow the largest possible number of interesting settings of restrictions and to reduce visual clutter for the user, some less interesting combinations of restrictions will be disallowed. That is, certain choices early on will restrict the set of choices for other restrictions. For example, if the user selects the number of uses of the token to be one, the system may not allow for any transaction over \$500. It is up to the credit card companies to define the set of possible restrictions. The user chooses which restrictions from the set of possible restrictions to utilize for a particular token when creating the token.

Once the user picks all of the restrictions, the device should display the properties of the token in a manner that the user can confirm that this is what is desired. The user can then approve it, in which case the encryption takes place and the token is displayed, the user can modify it, or the user can discard it.

3.2 The protocol

The main idea behind our protocol is a simple one: the card holder and the credit card issuer already have a relationship. In order to start using limited-use credit card numbers, the card holder must obtain a long-term secret key K from the card issuer. The key K must be stored in the user's device and remembered by the card issuer. Note that the card holder and the card issuer already share the semi-secret traditional credit card number, but we prefer to use a different key because it allows us to make K longer than a traditional credit card number, and because it is reasonable to expect that even if a limited-use system is adopted, it will be gradual and incomplete. That is, card holders will still use their traditional credit cards for some purchases. Given that, it is not wise to use the same card number as the secret shared by a card holder and the card issuer.

To generate a token, the card holder starts by choosing the desired restrictions. Once the restrictions are chosen, the device encrypts using K using an authenticated encryption scheme (discussed more in Section 4.2). Assuming the scheme used is secure, an attacker will not be able to learn the key K and use it to forge new tokens. However, it does introduce a new way that an attacker might be able to make charges to an account number—namely, by guessing a valid limited-use token. This is discussed further in Section 4.3.

We also must address the issue of "replay" attacks. In order to ensure that a limited-use token cannot be replayed for additional uses once its specified restrictions have been met, the card issuer maintains a database of "used up" tokens and checks before verifying a token that it has not already been used up. In order to avoid a database whose entries must be stored and looked at forever, we follow the standard technique of using expiration dates. That is, before encryption, the device adds a timestamp to the restrictions indicating the time of generation. In order to be valid, the token must first be used within a specified time—say, one month. Tokens first received after the one month expiration of their timestamps are rejected as invalid. Thus, the card issuer can remove used up tokens from the database once their expiration period has elapsed. Once seen by the card issuer, a single-use token has to be stored until a month from the timestamp in the token, and can then be removed from the database. A multiple-use token is stored when it is first used, and must then be stored for the larger of one month or the time limit in the token. (For example, a multiple-use token good for a year would need to be stored for the full year so that it would still be recognized as valid on subsequent uses even after one month.)

Our protocol has three parts, which occur sequentially, but need not be in quick sequence. The first part is a transformation (via encryption) from the restrictions and the long-term key K to the token. The second part is the communication of the token and the identifying information via the merchant to the credit card issuer. The third part is the verification of the token by the credit card issuer. Figure 1 shows the execution of the protocol. The card holder interacts with a token-generation application on the device locally, probably first authenticating to the application with the account number or another human-enterable password. Using this application, the user picks from a set of restrictions to specify the type of limited usage desired as described in Section 3.1. Once selected, the restrictions, along with the timestamp for avoiding replays, are encrypted with the long-term key K to form a token. Note that the timestamp also ensures that different tokens generated with the same restrictions are different, provided that sufficient time granularity is used, which ensures that different instances of the same restrictions with the same account number are distinguishable.

Later, when the token is to be used, it is communicated to the merchant along with identifying information such as the card holder's name and billing address. Such identifying information is already typically requested by merchants for Web and telephone purchases. In the case that the restrictions are not very restrictive (for example, if the merchant is not specified), it may still be desirable to send the limited-use token over an encrypted channel (SSL) to the merchant so eavesdroppers cannot overhear the token and use it before the user. However, note that even if the token itself is overheard, the eavesdropper won't know a priori which purchases are compatible with the restrictions in the token, since the restrictions are encrypted. Even if SSL must be used, a limited-use token provides additional security over a general-use one if the merchant is not known to be trustworthy or if there is concern about whether its databases are properly secured.

Once the merchant receives the token, it need not communicate further with the token user. The merchant must then get verification from the card issuer before fulfilling the user's order. To do this, the merchant passes the token and identifying information to the credit card issuer. The card issuer then attempts to verify that the token is a valid token and has not yet been used to the limits specified. If the merchant need not immediately respond to the user (for example, if the user is purchasing physical goods that will be shipped by mail), the merchant can wait to complete this step, perhaps batching several transactions together.

Once the card issuer receives the token and identifying information, it uses the identifying information to look up the long term key. The card issuer then uses the key to decrypt the token. If the decrypted token is not of the proper form, the card issuer notifies the merchant that the transaction is denied. If it is of the proper form, the issuer checks that the restrictions are met and that the token is not already in the "used-up" database. If the restrictions are not met or if the token is in the used-up database, the transaction is denied. If the restrictions are met and the token is not in the used-up database, the transaction is approved. If approved, the issuer approves the transaction to the merchant, who then fulfills the user's order.

Finally, the card issuer updates its databases: first, if the token is now used up (and not yet expired), whether due to monetary limits or transaction number

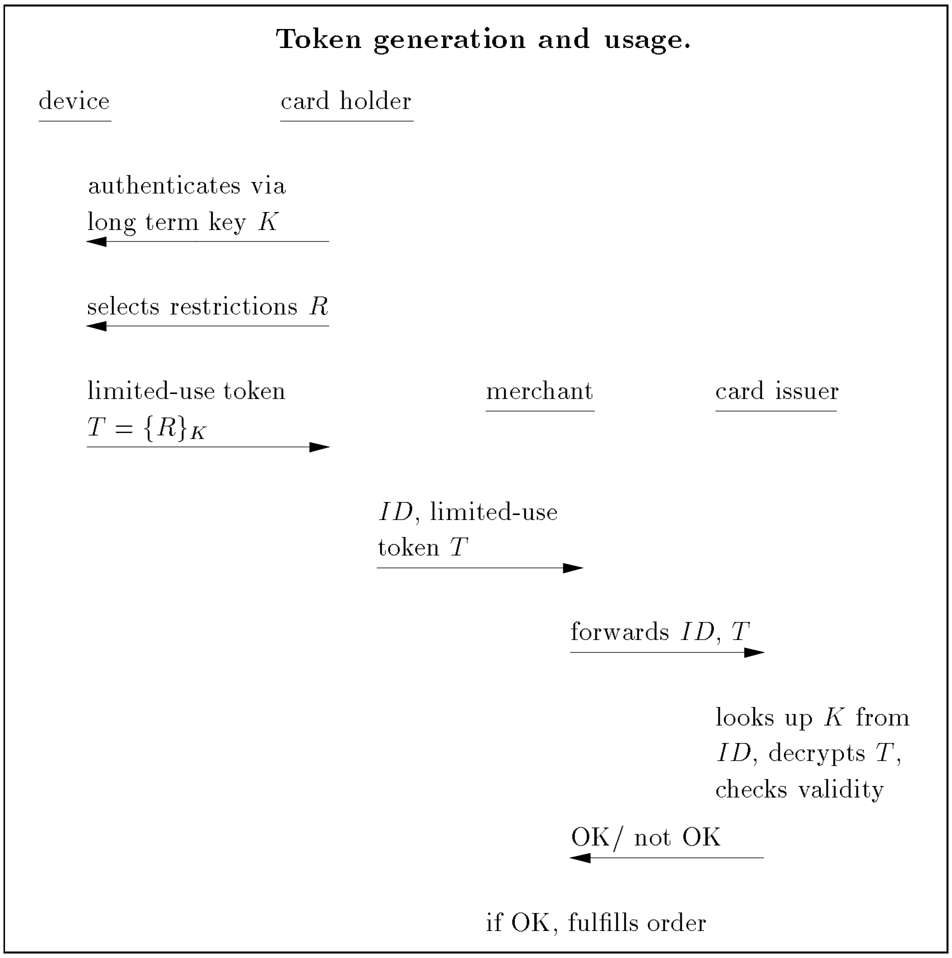

Fig. 1. The card holder authenticates to the device and selects the set R of restrictions. The device uses a key derived from the credit card number to encrypt R and produce a token T. The card holder then transmits T to the merchant, along with some identifying information ID. The merchant forwards these to the card issuer, who uses ID to look up the account, retrieve the card holder's long term key K, and decrypts the token. If the restrictions are met, the transaction is processed; otherwise, it is denied.

limits, it is added to the used-up database. Additionally, if the token is a multiple-use token, the issuer looks for it in a database of current multiple-use tokens, adds it if it is not already there (because this is the first use), and accounts for the current use (e.g. subtracting the monetary amount and decrementing the transaction count). When the remaining amount or the transaction count reaches zero, or the token expires, the token is removed from the current multiple-use database.

4 Discussion

In this section, we discuss some security aspects of our proposed system.

4.1 Token length

It would be desirable for security reasons to have tokens be longer than traditional credit card numbers. For example, 128-bit tokens (38 digits) would allow the use of AES, together with a MAC, as the encryption function. However, our interoperability and ease-of-use requirements suggest that our tokens must be at most 16 digits, or as little as 11 digits if part of the credit card number must be fixed.

There are several problems that arise if we restrict our tokens to be a specific small number of digits, which we address in the following subsections. Most encryption functions have a fixed, longer, block size, and it is not immediately clear how to apply them in this case. Additionally, a small token makes the tokens more susceptible to guessing attacks and causes collisions between tokens created by different users with different restrictions.

One possibility for increasing the size of the token space without requiring changes to merchants' Web sites is to divide a longer token into several parts that can be used in various parts of the name and address fields. While we think this solution violates our ease-of-use requirement and possibly our interoperability requirement, it may be necessary to maintain security.

4.2 Encryption of the tokens

In our protocol, it is important that the tokens are authenticated so that the credit card company knows that they were generated by a valid card holder and that they have not been modified. The resulting token is used both to protect the authenticity of the token and to convey the information in the token privately to the card issuer, in as little space as possible. Authenticated encryption schemes provide encryption and some authentication properties. Decryption returns either the plaintext or an indication that the ciphertext is not a valid ciphertext for the key that was used. Several methods involving encryption and MACs have been proposed; many are analyzed by Bellare and Namprempre [1].

In our protocol, the output of the encryption is limited to the token space. The credit card company server must be able to decrypt the token, so truncating the encrypted token does not work. Furthermore, the token size will not typically fall on the block boundary of most symmetric ciphers. Fortunately, Black and Rogaway [2] describe several ciphers for arbitrary finite domains. In fact, their motivating example is generation of unpredictable credit card numbers, closely related to our application.

4.3 Guessing attacks

A potential problem with our proposal is that an attacker might be able to guess a valid token and use it to make purchases. Since we include identifying information in a transaction (i.e. the user's name and billing address), the attacker must guess a token that works both for a particular name and address (ID) and for the purchase the attacker is making. That is, the attacker must find an ID and token such that the token is the encryption under that card holder's long-term key of a set restrictions compatible with the current purchase. Note that it is possible that two different settings of restrictions for different users with different account numbers may encrypt to the same token. These potential collisions can make an attacker's job easier because a given token will have more valid uses.

Using the most limited set of possible restrictions will make guessing harder, as there will be fewer tokens compatible with a given purchase. For example, one might implement the system to always require tokens to contain the merchant name, to have relatively short expiration dates, and to have only a small number of uses. In addition to making guessing attacks harder, this also reduces the usefulness of overheard tokens that an attacker might learn and try to reuse.

Even if there were exactly one valid token for a particular purchase, a token space of size 10^{11} appears dangerously small, in that it takes only a few days on a current computer to search the entire space, and an expected time of half that to find a valid token. Note that the use of traditional long-term credit cards results in a search space of size at most 10^{16} (less if some parts of the number are fixed), and once a card number is guessed, it can be reused for additional purchases until it is detected and revoked. Simply allowing non-case-sensitive alphanumeric credit-card length tokens gives us a search space of $36^{11} \approx 10^{17}$— already a small improvement over current credit cards in the time required to find a token valid for a particular purchase. Furthermore, in our case, subsequent purchases will usually require a different token, so a new guessing attack will be required.

More importantly, note that it is not sufficient for an attacker to simply guess a token by enumerating it. In order to learn whether or not the token is valid (without either already knowing the long term key or breaking the encryption another way), the attacker must actually attempt to use the token. This exposes the attacker to possible detection, as well as increasing the amount of resources an attacker must use to perform the attack. If the attacker fixes a specific user ID and attempts different tokens with it, the corresponding user's account will be suspended after a fairly small number of attempts, at least temporarily, and the attack will be very unlikely to succeed. While this is an inconvenience to the user, it is preferable to the alternative of allowing the attacker to succeed. If the attacker instead tries different IDs, he is less likely to succeed because now he also has to either guess valid IDs or find them through other means. Also, whether using the same or different IDs, if the attacker tries his attempts with the same (possibly collaborating) merchant, the issuer can temporarily suspend the merchant, and also can ask for the merchant's cooperation in tracing the attacker through information such as IP addresses from the merchant's Web logs. If the attacker tries to subvert detection by using multiple merchants, he must spend more effort setting up his purchases (and finding desirable purchases will be harder). Although multiple-merchant attacks are harder to isolate from legitimate purchase attempts than single-merchant attacks, they still may be preventable and even traceable with the cooperation of Internet Service Providers using various infrastructure-level tools that respond to denial-of-service attacks (c.f. [8]). If most attacks will be detected before they succeed, and particularly if the attacker will sometimes be identified and punished, attackers will be less likely to even mount these attacks.

One might expect that if the use of limited-use tokens becomes common, merchants might gradually change their Web forms to allow for non-numeric and/or longer credit card numbers, which would allow the use of a larger token space, providing more security against guessing attacks. Note that on-line solutions have the advantage that a token can be validated only for a very short period of time, so guessing attacks are less likely to succeed there, and they may be the best solution unless, or until, the infrastructure changes. But with even minor infrastructural changes such as allowing non-numeric or longer strings in the credit card number field, the off-line setting offers sufficient security, and provides other advantages for the user.

4.4 Collisions

A collision occurs when two different settings of restrictions encrypt to the same token, either with the same key or with different keys. Clearly, collisions are more likely to occur with a smaller token space. Provided the number of possible settings of restrictions is smaller than the token space and the encryption function is truly a permutation, there will not be collisions of tokens for the same user (i.e. with the same key). Since timestamps are also used as part of the input to the encryption function to create the token, it will be necessary for users to change keys from time to time.

If desired, collisions between different users could be avoided entirely by using sufficiently large tokens, divided into part that is fixed and unique to each user and part generated by encryption as we have described. However, it should not be necessary to avoid such collisions, since the card issuer always looks up the appropriate long-term key from the ID, and so will not be confused by them.

4.5 Anonymity

As presented, our system is not anonymous, since the user's name and address is sent with every transaction. While we think anonymity is an important and worthwhile goal, it is orthogonal to protection of the credit card number. We chose to focus only on protection of the credit card number, rather than to cloud the issue by also designing our system to be an anonymous payment system. If desired, our system could easily be modified to provide anonymity to the user from the merchant, for example by encrypting the identifying information for the card issuer using a card issuer's public key stored in the user's device. Note that this would require merchant forms to be able to accept the resulting encryptions in the user name and address fields or elsewhere. Further note that in the case of physical purchases, some valid shipping address is required, so full anonymity may be compromised in any case.

5 Calling cards

The off-line protocol presented here can be modified for use with telephone calling cards as well. It is often a problem that "shoulder surfers" see people entering a calling card number into a public phone; the surfers then use the calling card numbers themselves, or worse, sell them to a number of people with instructions to make a single (usually lengthy and international) call with it in a specified time window. The security of a calling card account lies exclusively in the knowledge of the calling card number. If someone sees this number, that person can, until detected, make virtually unlimited calls that are charged to the account holder. This can go undetected until the end of the billing cycle. Since many people now pay their phone bills automatically each month, rather than in response to an itemized bill, they may not notice unusual activity in their accounts until it reaches drastic proportions.

Limited-use tokens are also useful in this situation. In this case, restrictions involve time of day, area code called, number of minutes, number of calls, which numbers can be called, and so forth. For example, a parent might provide a child with a calling card number (token) that only allows calls to home. To create a limited-use token, the user enters the calling card number into the device, and then picks a set of restrictions. The device then outputs the new limited-use calling card number which is an encrypted token containing those restrictions. The encryption key is derived from the calling card number. When a user places a call with a token, the system asks for some identifying information, such as the user's home phone number and zip code, in addition to the calling card number. This can be accomplished by having a different toll-free access phone number for calls using limited-use tokens. When a user enters the token, the system uses the identifying information to look up the user's account number, derive a key, and then decrypt the token to check the restrictions.

As with credit cards, the use of temporary limited-use tokens in calling cards allows a user greater flexibility to manage risk and set parameters on a single long-term account than is achieved by always reusing the traditional account number.

6 Conclusions

We have presented a protocol for generating and using restricted credit card or calling card numbers. At some cost in security, these numbers can be of the same format as the traditional ones, allowing for easy layering of the protocol on the existing e-commerce infrastructure. In our system, users can generate limited-use tokens in an off-line manner, without requiring any interaction with the credit card company. We discussed the advantages of this scheme over currently deployed ones, and also discussed the security issues that may limit the deployment of such schemes without some infrastructural changes.

While the protocol that we present in this paper is not ideal from a security standpoint, it provides greater security than standard reusable credit cards and represents a practical solution that can be accomplished under a strict yet realistic set of assumptions about the current Web commerce infrastructure.

Acknowledgements

We thank Yoshi Kohno, Rick Johnson, and Barbara Fox for helpful comments. We also thank the anonymous and non-anonymous reviewers for their helpful comments.

References

1. Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notations and analysis of the generic composition paradigm. In *Advances in Cryptology – Asiacrypt '00, LNCS vol. 1976*. Springer-Verlag, 2000.
2. John Black and Phillip Rogaway. Ciphers with arbitrary finite domains. In *Proceedings of RSA Security 2002 Cryptographer's Track*. Springer-Verlag, 2002.
3. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on RSA encryption standard PKCS #1. In *Advances in Cryptology – CRYPTO'98, LNCS vol. 1462*, pages 1–12. Springer-Verlag, 1998.
4. Edward W. Felten, Dirk Balfanz, Drew Dean, and Dan S. Wallach. Web spoofing: An Internet con game. In *Proc. 20th National Information Systems Security Conference*, 1997.
5. Kipp E. B. Hickman and Taher Elgamal. The SSL protocol. *Internet draft draft-hickman-netscape-ssl-01.txt*, 1995.
6. D. Kormann and A. Rubin. Risks of the Passport single signon protocol. *In Proceedings of 9th International World Wide Web Conference*, May 2000.
7. PayPal. `http://www.paypal.com`, 2000.
8. Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for ip traceback. In *Proceedings of the 2000 ACM SIGCOMM Conference*, pages 295–306, 2000.
9. Adi Shamir. SecureClick: A Web payment system with disposable credit card numbers. In *these proceedings*.
10. Bob Tedeschi. Technology: Real-time challenges, in Cyberspace and on the ground. *The New York Times on the Web*, January 1, 2001.