SECURING THE WORLD WIDE WEB:

SMART TOKENS AND THEIR IMPLEMENTATION

Michael F. Jones

Bruce Schneier

ABSTRACT

This paper introduces Smart Tokens, discusses some of their basic properties and outlines their general role in securing software applications on the World-Wide Web. In addition, the completed hardware and software architecture of a current implementation is described. Finally, an electronic commerce application for Smart Tokens involving major US financial institutions is discussed.

1.0 INTRODUCTION

Security concerns impose a severe constraint on a vast array of products and services that can be offered within the context of the World-Wide Web. Electronic commerce on the Web will be enabled by emerging security protocols such as S-HTTP and SSL. Introducing additional choices such as Microsoft's recent release of STT and PCT may have the effect of causing confusion and therefore, delaying implementations. However, it is clear that serious attention to security has now become mainstream. S- HTTP and SSL, which incorporate Public-Key cryptosystems, are just beginning to be implemented by WWW applications developers. Recent announcements by major software vendors indicate that widespread implementation of these standards is likely to occur [10].

Public-Key cryptosystems involve an authority issuing a pair of complimentary encryption keys to each user in the system. One of the keys is intended to be made public, analogous to an e-mail address and is called the Public-Key component. The other key in the pair must be available for use only by the "owner" of the key-pair and is called the Private-Key component. Application software making use of the key- pairs provides users with a rich set of security functions that essentially lifts the current constraints on electronic commerce in the WWW environment. For an introduction to Public-Key cryptography, please refer to [3].

Although Public-Key cryptosystems have many desirable characteristics in securing distributed systems, they typically rely upon the ability of the system to protect the beneficial use of the Private-Key component from all but the intended user. If the Private-Key component can be copied, or is made public, the authenticity of transactions using that Public-Key pair are called into question and therefore, cannot be trusted. In the commercial internetworked environment, software-only solutions for protecting the Private- Key component are inherently vulnerable to attack by viruses and other methods of compromise such as password guessing schemes.

Smart Tokens, the subject of this paper, are hardware devices with associated software that have the ability to perform Private-Key operations without the Private-Key ever being vulnerable to compromise. The WWW security community is in general agreement that Smart Tokens will play a major role in the future of electronic commerce on the Internet. This paper further describes a Smart Token implementation with characteristics that make it suitable for use with general purpose software applications such as word processors, e-mail packages and WWW browsers. Portability and isolation from the hardware layers are important architectural goals. Finally, a major electronic commerce application in which the Smart Token is used as the analog of traditional checkbook functionality is then discussed.

2.0 SMART TOKENS

A "smart token" is an easily portable device that does special-purpose operations for its user, generally identifying the user to some larger computer system. A smart token can look like a PC Card, 3.5" diskette, credit card, pocket calculator, or many other things--the important feature is that it carries some secret information for you, and that it does some internal calculations when you need them performed. A smart token is often designed to be tamper-resistant: It is difficult to take apart. It is protected with a user password, so that even if it is physically stolen, it will be difficult to impersonate their owner.

Just as most pocket calculators are used to do arithmetic, most smart tokens are used to identify their user to some remote computer. If the user's identification checks out, then she is allowed to do something: make a purchase on her credit card account, read her e-mail from a public terminal, board a plane, or log in to a remote computer system, etc.

2.1 THE ROLE OF SMART TOKENS

To see the value of this, consider making a purchase on the Internet. People used to type their credit card numbers directly into a computer, and then send those numbers to the merchant. This is insecure both because those credit card numbers can be easily collected by someone who monitors network traffic, and because the merchant has no way to confirm that the person who typed in the credit card number is the same person who owns the credit card.

Current solutions include encryption, which hides the buyer's personal information during transition, and digital signatures, which confirms the identity of the buyer. Financial models include digital checks, digital credit cards, and digital cash. These solutions protect against network monitoring, but do nothing to stop password guessing, password collection at the buyer terminal, or password compromise. The seller still has to trust that the person who signed the digital payment order has not accidentally disclosed his password or private signature key.

Tamper-resistant tokens are needed to compute the digital signatures for electronic commerce applications; they are the best way to prevent disclosure of the signer's private signature key. If the private signature key is disclosed, then anyone can use it to forge the signer's signature. If significant numbers of private keys are disclosed and are used to forge electronic checks, electronic credit cards, or electronic cash, then these forms of money will not be accepted. In a situation like purchases on the WWW, where other forms of identification can't be used, merchants

must rely on the security of the signer's private signature key.

Enough tamper-resistance is needed to make it economically unattractive for attackers to steal signature cards, extract the private key, and pass bad "checks" with that key before the card is reported stolen and the account changed.

Smart tokens often require a password in order to function. This provides the token some certainty that the person using it is the person who is supposed to be using it. This isn't always necessary--for some applications, entering a password each time the token is used is more trouble than it's worth. In general, if a person can use the token to spend money or access sensitive data, it will have a password. The user enters the password on his keyboard, or directly into the token via a keypad. Even if the computer has been hacked to record passwords, that won't allow anyone to break the system; they still have to get possession of the smart token.

The most common application for a smart token is to convince some larger system of a user's identity, so that the larger system, perhaps with help from the token, will allow the user to do something. Protocols for proof of identity are a well-studied area of cryptography, and several techniques are discussed in [8]. For example, in order to allow a user to log in to a remote system, a computer might require the user to use a "one-time password" stored in the token. Since only the token and the remote computer know what the next password should be, only this token could have given the right password.

Once the user and token have identified themselves to the larger system, then the system and the token can work together to allow the user to do something. For example, a software metering token, after it has identified itself to the software being metered, can authorize another execution of the software and increment its internal counter by one.

2.2 APPLICATIONS OF SMART TOKENS

Restricting access to remote computer systems: A physical "key" token can be used to restrict access to a computer system accessible via Internet or modem. The computer system and the smart token can work through an interactive protocol that verifies each to the other, and can even agree on a session encryption key. This type of system allows a user to log in through an untrustworthy terminal without leaving access to his remote account with the terminal.

A physical "key" for digital signatures: Suppose a user has a private signature key that she uses to authorize contracts of up to $10 million. She may not feel comfortable trusting this key to her personal computer. Even if it's protected with a password, a really capable attacker might install some software to capture her password, and later, her key. A million-dollar contract can't be signed without both smart token and personal computer being involved. The smart token can be kept physically locked up, and will be protected by a password in any event. Similarly, she could use a threshold signature scheme, which might require the agreement of (say) three of five high-level executives in order to sign a major contract. Each of these executives can be given a smart token, and can be required to enter their password to permit the contract to be signed. These protocols are described in [8].

WWW Purchases: In place of a normal VISA card, a user has a smart token. When she wants to buy something, she puts her card in her computer and enters her password. The card then handles the transaction automatically. It should be impossible for anyone to capture enough information from the transaction to perform more transactions. It's even possible for the smart token to keep a transaction log. Other payments systems, called "digital cash" systems, keep the user's transactions anonymous unless she tries to defraud the payment system (i.e., by spending the same electronic dollar twice). These sorts of schemes are discussed in [8].

Software metering: Another nice application for smart tokens is in application software metering. Ideally, a user would be able to load up a single token (perhaps a PC Card) with the licensing information for all her software. Each time she or one of her employees opened an application, that application's meter would be incremented by one. The metering might measure hours or minutes of application time used, maximum number of users, or might even bill specific functions of some applications more heavily than others. This kind of token mimics the little meters that are used in some self-service photocopy shops, where a user is given a meter, which is required to run the copying machines, and which counts the copies used. When the meter is returned, the user is charged for her copies. The physical security of the token is trusted to prevent the user from resetting the counters. The same metering token can also be used for other metering applications: interactive-TV set-top boxes, automobile tolls, and public transportation payments.

Single-copy documents: If a user has a document that needs to be readable, but not copyable, a smart token can act as that document. When someone reads the smart token, they first verify its identity, then read the contents of the token's document. This document may also be digitally signed by some kind of notary. So long as an attacker cannot recover the token's secret information, which it uses to identify itself, the token can't be copied. A single token can conceivably carry many such documents. Variations on this allow the token to "spend" the documents (perhaps they're rail or bus passes), deleting the special identifying information from each document as it is spent. This is discussed in [2]. Note that this doesn't prevent anyone from copying the document from the token--instead, it simply keeps them from claiming that their copy is the original.

Electronic Subscriptions: If a user wishes to purchase a six-month subscription to an electronic newspaper or news service, he buys a newspaper token. When he gets to a terminal (maybe in a hotel room), his token can authorize him to access the latest news from this newspaper. Probably, additional services will be made available for extra charges, such as more extensive photo coverage, or expanded coverage of specific areas of the news--these can also be known and authorized by the smart token.

Secure storage devices: Some smart tokens hold significant amounts of flash-RAM, which can be used to hold secret user data. The token also holds some physically secure memory, which keeps an encryption/decryption key. The user must enter the right password to gain access to the data. (Often, the password and the internally-stored data must both be used to determine the encryption/decryption key.) It is possible to split the flash-RAM into many partitions, and encrypt each with different passwords and keys.

Secure tokens can also be used to implement protocols for electronic auctions, secure voting, anonymous transactions, and others.

2.3 HOW SMART TOKENS WORK

This section is meant as a brief introduction to the operations of smart tokens. To get a better understanding of the algorithms and protocols discussed here, see [2] and [8].

Passwords: A token can deal with passwords in two basic ways. The simplest is to check the user's password against an internally stored value, and authorize the user's request if the password matches that value. (The actual password isn't usually stored. Instead, some value based on the password is stored, so that the password isn't revealed even if an attacker manages to read the token's internal memory.) The second way, more complex but more secure, uses the password as a decryption key, to decrypt some internal set of values, which are then used to authorize the user's request. In this case, the token itself has no way to determine whether the password was correct--the larger system being connected to must do that. When secret data is stored in the token, it is common for it to be encrypted under a key derived from the password.

Identification Protocols: A token can identify itself to another system in many different ways. The most common method is for the system and the token to share some secret data. The outside system sends a "challenge" (a random string of bits) to the token, and the token must calculate the proper "response," based on the secret data. Other systems allow a token to identify itself, using a public and private key. The token must hold the private key, the outside system must only know the public key. For a good introduction to this kind of system, see [8].

Digital Signatures and Message Authentication Codes: A token may authorize a transaction if the token digitally signs a timestamped request for the transaction to take place. This involves a secret key kept in the token, and a public key known by the system. See [2] and [8] for discussions of this kind of algorithm and system. More generally, a digital signature may be used anytime a block of data needs to be verified as having come from this token, and the systems that will do the verification don't share any secret data with this token.

3.0 CRYPTOGRAPHIC API STANDARDS

In order to integrate cryptographic functionality with "off the shelf" commercial software, there have been many recent efforts to develop a modular Cryptographic Application Program Interface (CAPI). Of these, there are three proposals that are proving to be widely accepted. They are: the GSS-API (Internet Engineering Task Force)[5], the GCS-API (X/Open)[9], and Cryptoki (RSA)[5].

Although it is beyond the scope of this section to describe these three CAPIs, it is important to note that they differ significantly in the degree of cryptographic knowledge required on the part of the application developer for implementation. The GSS-API requires the least knowledge of the underlying cryptography and Cryptoki requires the most understanding. In addition, Cryptoki is the only one of the three CAPIs that was written primarily for smart cards and tokens. It includes an abstract token interface that is intended to be the only layer in a software architecture that requires change in order to implement a wide variety of Smart Tokens. A useful analysis of CAPIs can be found in [7].

Since Cryptoki requires more knowledge of the underlying cryptography, it will be helpful to some application developers for an additional higher level API to be provided along with the Smart Token software development tools.

4.0 AN EXISTING SMART TOKEN IMPLEMENTATION

4.1 GENERAL DESCRIPTION

The currently implemented Smart Token combines high-density flash memory and data security functions in a ubiquitous PC Card Type I package. It is compatible with PC Card release 1.x and 2.x memory card specifications. The design provides the computer user with removable, secure non-volatile memory plus data and communications security support in a single package.

This Smart Token implementation is available with storage capacities from 1 to 24 Megabytes. In normal operation, the Smart Token's memory is compatible with host computers having PC Card adapter slots for additional memory or removable media. The Smart Token uses flash memory with 64-Kbyte block erase capability and supports both word-wide and byte-wide transfer modes.

Security features provide memory access control as well as support for data security functions such as secure remote log in, Public-Key encryption, digital signatures, etc. The security feature is provided by the FIPS PUB 140-1 Level 3 compliant Cryptographic Support Processor (CSP) embedded in the card. The CSP is an integrated circuit based on ISO standard smart card technology which is recognized internationally as a secure vessel for key and password storage. The Smart Token's CSP provides secure computing functions such as random number generation, key encoding, and key comparisons, while the private key information never leaves the secure silicon. Multiple passwords can be stored on the chip.

Passwords stored in the CSP can be changed, but not read, by host resident software. The Smart Token is shipped with default memory passwords installed in the CSP. The operating environment and software resident in the CSP prevents access to the secure storage, but allows certain defined operations using the secure data. The CSP can detect physical security violation, attempts such as probing the chip, de-soldering the chip, and electronic probing involving single stepping the clock.

Access to the CSP is through host-resident software and Smart Token drivers described below. The electrical interface between the host and the Smart Token is compatible with PC Card memory-only release 1.x and 2.x. Data transfer between the host and the CSP interface is controlled by the card interface ASIC which supports the ISO 7816-3 standard. Host resident software supports the RSA Cryptoki standard.

The presence of the CSP allows host resident software to execute secure data interchange such as remote log in, data communications, digital signatures, information "metering" and electronic funds transfer (EFT), using standards including DES, DSS, PKCS and RSA. The Smart Token is all solid state, requires no batteries and is robust compared to storage media such as floppy disks.

4.2 SOFTWARE ARCHITECTURE

This section provides an overview of the software architecture of the PC Card Smart Token. Applications are presented in a Microsoft Windows 3.1 environment, with PC Card software support provided by SystemSoft card and socket services. Some of the architectures presented are considered to be building blocks for higher level functionality, and are designed in a way which will promote future advanced application development. There are six major software components which interface with various levels of DOS, Windows, and PC Card architectures. They are:

Smart Token resident software for encryption and protection services provided by the CSP

a DOS driver interface to the CSP, with block driver hooks to support transparent file encryption.

a DOS card service vendor specific driver to support file system initialization.

a Windows DLL which implements a CSP API protocol layer through DPMI services and the CSP driver.

a Windows GUI user application for administration of card protection and encryption.

a Cryptoki support library which implements the Cryptoki API with CSP services

The first component is Motorola 6805 instruction code which is resident in the CSP memory space of the Smart Token card. The other components are loadable or installable drivers and applications that execute on the host machine. A configuration for these components is illustrated in Figure 1. In this scheme, the CSP and encryption support driver is a DOS TSR application which is loaded at boot time and executes in real or virtual 86 processor modes. The card and socket service components are contributed by SystemSoft and provide PC Card 2.1 compliant support for Smart Token card interfacing. A Smart Token specific DOS driver is provided which implements a card service vendor specific call to reinitialize file system components when required by Smart Token protocols. The flash translation layer (FTL) is a SystemSoft character driver which emulates a random access block device on a flash platform. In Windows, several components allow for high level application and end-user card management. The CSP protocol DLL implements an API for performing CSP operations through the CSP driver. The Cryptoki support library implements a similar interface packaged into a Cryptoki API set. Cryptoki applications may link with the library to receive Cryptoki support on the Smart Token platform. The Windows File Manager is the standard Windows 3.1 file system management application, and represents any number of high level file components running in Windows protected mode. Lastly, the Smart Token Card Manager is a Windows protected mode GUI administration utility.

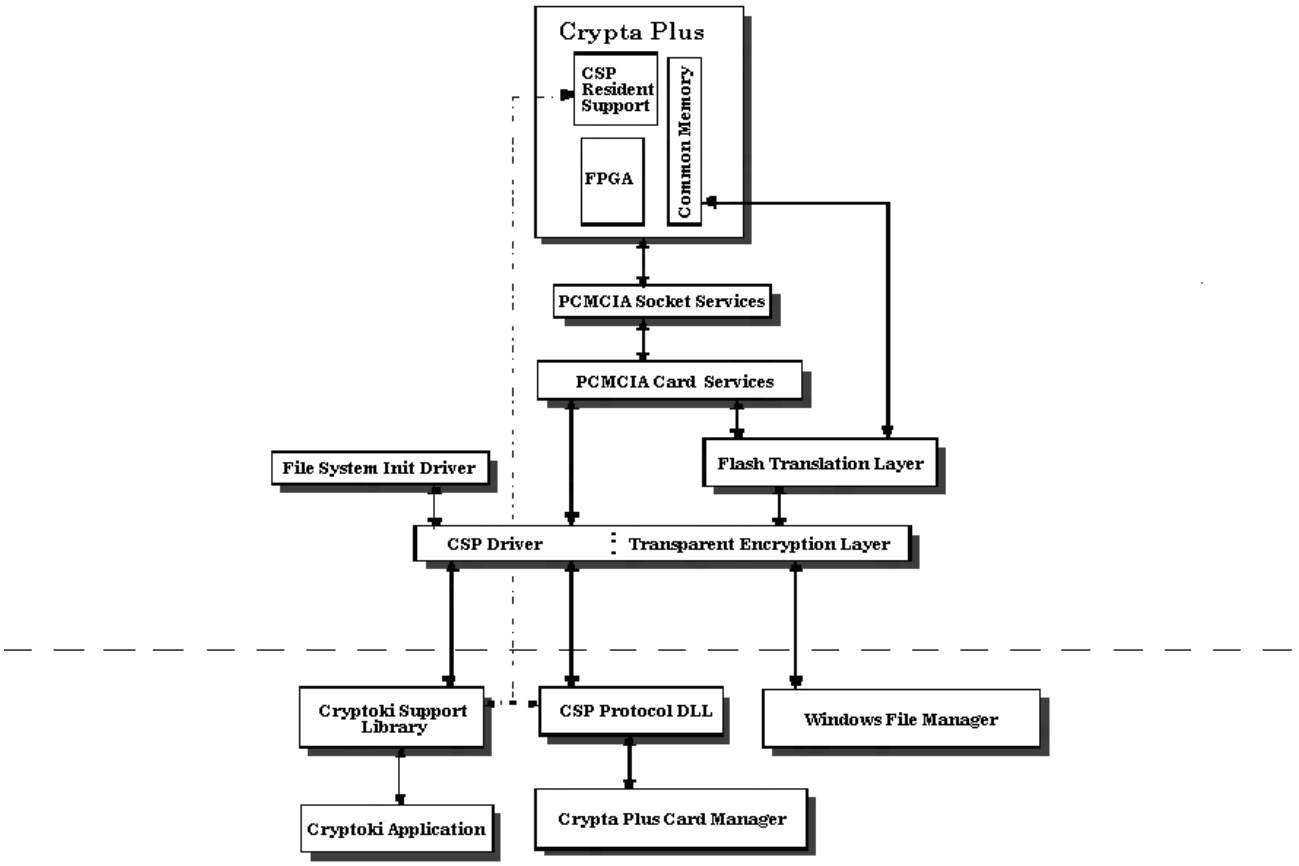

Figure 1

Socket Services: Socket Services provides an interface between Card Services and the host socket hardware. It is intended to allow software layers above it to be independent of the socket hardware implementation, aiding software portability.

Card Services: Card Services coordinates access to cards, sockets, and system resources.

CSP Access Support: This driver contains an application service function for high level pass through communication to the Smart Token CSP. The driver acts as a PC Card, Card Services client, and registers for callbacks on card insertion and removal. When a Smart Token card is inserted, the driver handles the required CSP initialization functionality of requesting a memory window to the Smart Token card and mapping the window to the CSP control register space. It also resets the CSP controller and looks for the proper response sequence. The driver is implemented as a DOS TSR application, loaded prior to Windows.

FTL Transparent Encryption Support: Support for transparent file encryption to the Smart Token flash is implemented on top of SystemSoft's flash translation layer (FTL). The driver traps file system requests targeted to FTL and, if the request is directed to an unlocked Smart Token card, may automatically encrypt or decrypt the request data with the embedded Smart Token key. Encryption is implemented with DES cipher block chaining on a sector basis. Transparent encryption is enabled through high level applications by setting the Smart Token encryption state using the CSP resident API.

Card Service Vendor Specific driver: A DOS character device driver is implemented to provide special vendor specific services required by the Smart Token card. Following a card lock or unlock, high level file system access to the Smart Token flash area changes. However, without notification of the change, the file system will not reinitialize itself to detect the new state. This driver monitors card service registrations for memory clients, and saves card service callback addresses. Following a lock or unlock of the card, a vendor specific card service API is provided which allows for issuing a card service REMOVE or INSERT callback to registered clients in order to force data structure reinitialization.

Cryptoki support: The Smart Token Cryptoki application library provides public entry points for Cryptoki defined functions. This library implements general Cryptoki layer support for token, session, and object management. The library makes use of the RSA BSAFE library for key generation and the support of Cryptoki defined cryptographic functionality. In the current implementation, the Smart Token card supports two predefined global objects which are created with the C_InitToken function. These objects are a public and private RSA key pair. All global objects (predefined or user defined) are stored in an encrypted format on the flash memory of the Smart Token card. Access to this memory is protected by the locking mechanism inherent to the Smart Token hardware design. When a Cryptoki session is started with an application, the Smart Token card must be unlocked, and the global objects loaded into memory. The software logic for unlocking the Smart Token card is implemented in the CSP driver and Smart Token CSP resident support. Once the card is unlocked, an application has access to global Cryptoki objects through the flash file system. Before objects can be loaded, they must be decrypted using a CSP resident secret key. When a Cryptoki session is closed, all global objects (including any new defined objects) are stored on the flash and encrypted with the CSP secret key. The Smart Token card is then relocked until another session is started.

5.0 WWW IMPLEMENTATION - FSTC ELECTRONIC CHECK PROJECT

The Financial Services Technology Consortium (FSTC) is a collaboration of major banks, technology companies and laboratories that was formed to address the critical need for viable means of conducting electronic commerce on public networks such as the Internet. Currently, over sixty organizations are members of the consortium. A secure payment system and deposit gathering mechanism for the banks is considered to be an essential enabling component in the commercialization of these networks.

The Electronic Check Project was developed by the FSTC to provide a secure, all electronic payment system modeled after the familiar paper check. It is an integration of a traditional form of payment within the existing financial services infrastructure and the rapidly growing electronic networks. A detailed description of the project, the functional flows and its objectives can be found in [4]. On September, 21, 1995, a live demo of the Electronic Check Project took place at the Bank of America in San Francisco. Participants in the demo included Bank of America, Bank of Boston, Bank of Montreal, Bank One, Chemical Bank, BBN, IBM, Sun Microsystems, Telequip and Bellcore.

The demo was conducted over the Internet using the World Wide Web. It included the purchase and payment by electronic check of a "Teddy Bear" for the Vice President of the United States, Al Gore, from PC Gifts and Flowers. One of the more remarkable aspects of the demo was that the check actually cleared electronically through the Automated Clearing House of the US banking system. Telequip's PC Card Smart Token implemented as described in Section 4, performed the role of the Electronic Checkbook, generating and signing the first electronic check through the US banking system.

In the demo and subsequent pilot program, Electronic Check makes use of a PC Card Smart Token in the form of an Electronic Checkbook which can be used within the context of the World-Wide Web. A Web browser in conjunction with an Electronic Check application has been integrated with the implementation described in Section 4. Two additional software layers are provided between the Electronic Check client application and the Cryptoki API in order to provide a higher level interface as discussed in Section 3 and to fulfill specific functional requirements of the Electronic Check initiative. The overall goal of this architecture is to make maximum use of existing standards and lower the risks associated with lower level interfaces to cryptographic devices.

5.1 FUNCTIONAL FLOWS

Unlike some of the newer stored value proposals for electronic commerce such as Mondex, Electronic Check is based on the familiar paper check model. Email is substituted for paper delivery by the postal service and digital signatures on the Electronic Check message replace the hand written signatures on paper checks. Since the functional flows are essentially the same as in the paper check model, the system is easy to understand. It is anticipated that rapid adoption of the Electronic Check will take place due to ease of integration and significant cost savings. Support for payment instruments like certified checks, cashiers checks, credit card charge slips and additional features such as future dating, limit checks and multi- currency payments can be accommodated.

Several scenarios for functional flows are described below:

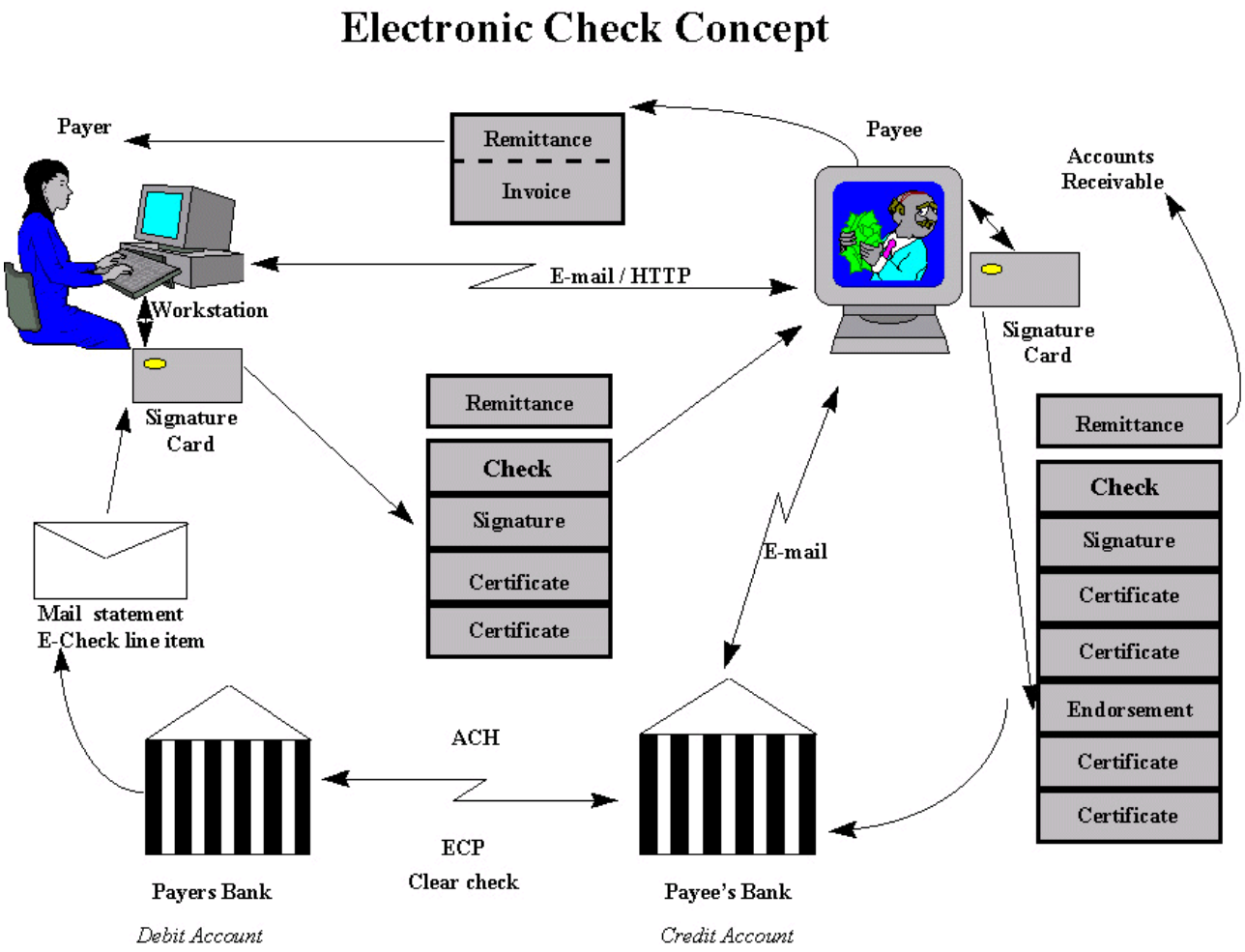

Figure 2

Figure 2 above depicts the typical Electronic Check flow. The payer receives an invoice from payee, generates a Electronic Check and sends it to the payee via email. The payee then emails the received payment to his bank and settles the transaction with payer's bank

The Cash and Transfer Scenario

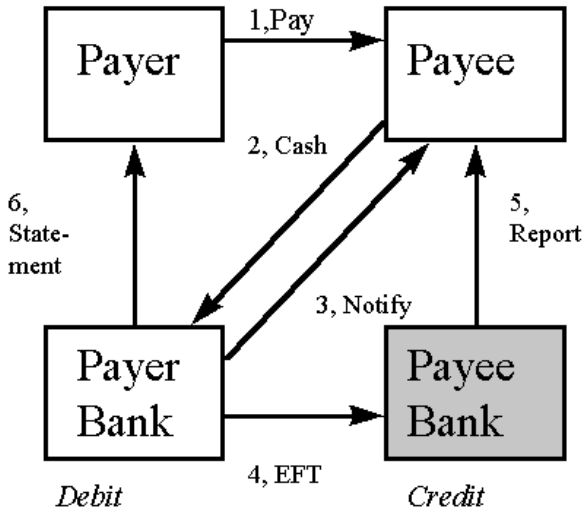

Figure 3

In Figure 3 above, the payer receives a bill/invoice from payee, issues an Electronic Check, and sends it to the payee. The payee presents it directly to the payer's bank to be paid to the payee's account at his bank.

The Lockbox Scenario

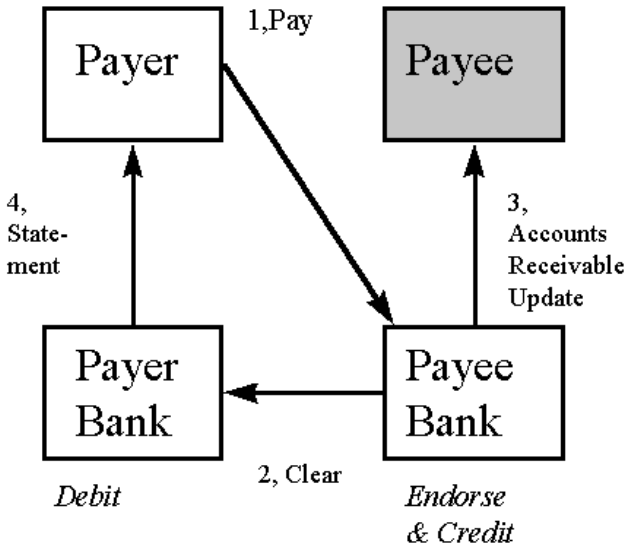

Figure 4

In Figure 3 above, the Payer receives a bill/invoice from payee, issues an Electronic Check, and sends it to the payee's bank, either directly or via a lockbox. The Payee's Bank then sends accounts receivable information to the payee and clears the payment with the payer's bank. In this scenario, there may be no payee endorsement.

The Funds Transfer Scenario

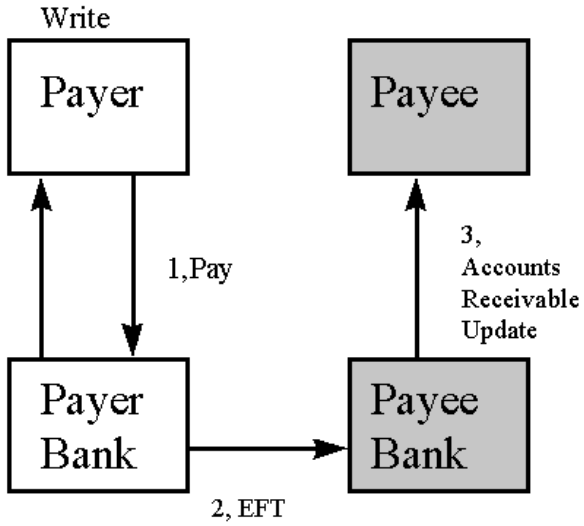

Figure 5

In Figure 5 above, the payer receives a bill/invoice from his bank, (assuming electronic bill presentment allows for capture of the payee's bills by the payer's bank), issues an Electronic Check, an sends it to his bank. The payer's bank, in turn, transfers funds to the payee's account at the payee's bank.

5.2 KEY DESIGN OBJECTIVES

Parameterized Electronic Payments Instrument: By specifying parameters in the Electronic Check message, the check can be transformed into various instruments such as a traveler's check, credit card slip, cashier's check, etc.

Open Integration with Accounting Systems: Commercial accounting systems will have the ability to interface with Electronic Check modules

through standard API's

Open Integration with Existing Inter-Bank Payments Mechanisms: Trusted gateways will allow connectivity between the public networks and the secured financial networks.

Authentication of Electronic Checks: Checks and checkbooks at any point in settlement cycle through the use of public key certificates.

Fraud Prevention and Confidentiality: Smart Token technology will help eliminate most of the losses due to forgery, alteration, duplication and fraudulent deposits.

5.3 PROJECT PLAN

The demonstration phase of the project was completed in September of 1995. A limited commercial pilot is expected to commence in 1996 and be in place for approximately six months. After evaluation and subsequent modifications have been implemented, a more extensive pilot will be followed by a full production version of the system.

6.0 CONCLUSION

Smart Tokens have the potential to enable a revolutionary expansion in products and services that can be offered on internetworked systems. The essential elements needed to bring this expansion to fruition are just beginning to appear in the marketplace. APIs have now evolved to the point at which mainstream commercial software applications can be architected to include Smart Token capability as a standard feature. Implementations in conjunction with projects initiated by major financial institutions, such as the one described in this paper, offer a starting point and a glimpse of a whole new industry that will underpin the future of electronic commerce on the Web.

7.0 ACKNOWLEDGMENTS

The authors would like to thank John Kelsey, who assisted with the theoretical sections on smart tokens, and Chris Carlisle, who assisted with the section on the current implementation GLOSSARY OF CRYPTOGRAPHY TERMS

Digital Cash - An anonymous electronic payment system, where users withdraw electronic "coins" from their bank, and spend them with other users, without ever having to reveal their identity. Should the user try to spend the same "coin" more than once, his identity would be revealed.

Digital Signature - A digital signature is a way of marking a digital document (like a computer file), so that only a person who knew some private key value could have marked this document this way. There is a public key value that can be used to verify that this document was properly signed, which is published somehow. Digital signatures are commonly done using systems such as RSA, DSA, and El Gamal. A good introduction to this is [3].

Encryption/Decryption - Encryption scrambles a message so that it can't be read without a key, which is known to the intended recipients of the message. Decryption unscrambles the scrambled message, so that it can again be read. Encryption is a generally good way to keep private data (such as a premium television channel) away from unauthorized users.

One-time password - Many computer systems require a password to allow a user to log in. Unfortunately, if a user is logging in over a modem or the Internet, her password can be seen by someone eavesdropping on the line. To defeat this, there are systems that use a different password each time a user wants to log in. The new passwords are generated by some cryptographic scheme, so that even when an eavesdropper catches a user's password, he can't use it to log in to the system. Generally, a token carried around by the user either generates the one-time passwords, or stores them for the user.

Public/Private Key - A public key system has two keys--a private key, known only to an authorized user or system, and used to digitally sign or decrypt documents, and a public key, used to verify digital signatures or to encrypt messages to the owner of the public/private key pair.

Tamper Resistant - A tamper-resistant device is difficult for someone to take apart and change its operation, or recover secret information in it. It's probably not possible to design a computer device that's absolutely tamper resistant, just as it's probably not possible to design a safe that can't be drilled through. It is possible, however, to make tampering with a computer device (such as a smart token) so time- consuming, difficult and expensive, that it's not worth the trouble to try.

Threshold Scheme - A method for splitting up a secret into n "shares," so that it takes k of those n shares to recover the secret. For example, a 2-of-5 threshold scheme creates five shares of a secret, and any two of the five shares together can recover the secret, but a single share can't recover it. See [3] for more information on this kind of scheme REFERENCES

[1] R.J. Anderson, "Why Cryptosystems Fail," Communications of the ACM, v. 37, n. 11, Nov 1994, pp. 32-40.

[2] D.W. Davies and W.L. Price, Security for Computer Networks, John Wiley & Sons, 1989.

[3] P. Fahn, Answers to Frequently Asked Questions About Today's Cryptography, Version 2.0, RSA Laboratories, 1993

[4] FSTC, "Electronic Check Proposal: Public Document" Financial Services Technology Consortium, 1995

[5] B. Kaliski, PKCS #11, Cryptoki, RSA Laboratories, 1995

[6] J. Linn, "Generic Security Service Application Programming Interface," RFC 1508, Nov 1993.

[7] National Security Agency, "Security Service API: Cryptographic API Recommendation," NSA Cross Organization CAPI Team, 12 Jun 1995.

[8] B. Schneier, Applied Cryptography, Second Edition, John Wiley & Sons, 1996.

[9] X/Open, "X/Open Preliminary Specification: Generic Cryptographic Service API," draft 3, Mar 1995.

[10] M. Zurko, WWW Security Standards Forecast: Partly Cloudy, IEEE Cipher #7, 1995