Personal Security Environment on Palm PDA

Margus Freudenthal
Cybernetica
Akadeemia tee 21, Tallinn, Estonia
margus@cyber.ee

Sven Heiberg
Cybernetica Tartu Lab
Lai 36, Tartu, Estonia
sven@cyber.ee

Jan Willemson
Cybernetica Tartu Lab
Lai 36, Tartu, Estonia
jan@cyber.ee

Abstract

Digital signature schemes are based on the assumption that the signing key is kept in secret. Ensuring that this assumption holds is one of the most crucial problems for all current digital signature applications. This paper describes the solution developed and prototyped by the authors – using a mobile computing device with a smart card reader for creating digital signatures. We give an overview of several common settings for digital signature applications and problems they have, describing also several frameworks for mobile security applications. A discussion about the choice of devices, design issues, concrete solutions and their security concerns follows. We conclude that although nothing can prevent careless private key handling, careful management is easier and more convenient when using our solution.

1 Introduction

1.1 Digital Signatures

Rapid developments in the area of e-commerce have forced companies and governments to think about using a digital analogue to physical signatures in everyday business. The most widespread method for giving digital signatures uses public key cryptography introduced by Diffie and Hellman [15]. Every user generates a key pair (PK, SK) consisting of a public key PK and a private key SK. The public key is made available for everybody; whereas the private key must be stored in such a way that no one except for the key pair owner is able to access it.

When singing the document X, the signer uses an SK to produce the signature $\sigma = Sig_{SK}(X)$. When the verifier obtains the pair (X, σ), the public key PK must provide the means for establishing, whether σ comprises a valid signature to X calculated with the private key SK.

There are several essential points where the physical and digital signatures differ. The physical signature is directly connected to the signer (via his physiological properties, like hand muscles etc), but to the document only through a common medium.

The digital signature, on the other hand, is most closely connected to the document. The connection between the signer and the signature is indirect, relying on the assumption that only the signer has the private key and is hence able to produce signatures. The connection between a public key and the real person is established via one of several proposed Public Key Infrastructure paradigms (PGP [22], PKIX [3], SPKI [4] etc.)

An usual, hand-written signature is something belonging essentially to us; it cannot be stolen from us. (It can be forged, but usually the forgery can be detected by experts.) Thus our signature is bound to us in a secure way. This is not necessarily the case with digital signatures, which are just a bit strings on some medium. Everybody who can gain access to the person's private key can sign documents on his behalf. This means that one must protect the private key all the time in order to be sure that no one else has access to it.

In this paper we focus on an approach for making digital signatures secure. Our goal is to produce a "What You See Is What You Sign" environment. The problems of keeping the private key secret are discussed also.

1.2 Attacks on Signing Environments

Most of the current digital signature applications are based on the software-only paradigm. This means that all the computations are carried out in the personal computer's software. The user's private key is stored on the computer's hard disk and encrypted using a pass-phrase or secured by other means. There are several problems associated with this scheme. First, it is quite hard to keep the user's private key secret at all times since today's desktop operating systems are not very secure. Every week new security holes are discovered that allow reading data from computers connected to the Internet. Viruses and Trojan horse programs might be installed on computers to send secret information to an attacker via E-mail or other means.

Secondly, even if a computer has a secure operating system and is not connected to any network, there are issues concerning physical security. In a normal office environment, the system administrator or even a janitor can wait until the employee has left the office, open up his computer, take out the hard disk and read it with the help of another computer, bypassing all the security measures. Article [20] contains more information about extracting keys from raw disk image.

Introducing physical security in office environment could be quite costly, e.g. one would have to install a safe for every workstation. One cannot store two computers in one safe because every user should be the only person to access his computer. For the same reason it should be prohibited for the system administrator to touch any workstation or for multiple persons to share some resource (e.g. an application server).

These measures would apply to an environment with high security requirements, affluent resources and considerable risk of physical attacks. Otherwise, if the resources necessary to purchase a safe for every computer are unavailable, the risks must be reduced by some other means.

As we saw above, if the security level of the operating system or the physical environment is low, it is unsafe to store keys in computer, especially when they can be used to sign documents that have legal value. The problem is partially solved by smart cards. Smart cards are single-chip computers that have non-volatile memory and are able to perform a limited number of well-defined operations. User can store his key on a smart card that usually has some physical security features. Signing process also takes place inside the smart card, which means that the user's key is never seen outside the card. Smart card is usually protected by password and can easily be carried with user at all times.

Previously it was mentioned that Trojan horses and viruses can be installed on the user's computer to steal the keys. Besides stealing sensitive information, they can also be used to alter some part of the system's behavior. This enables the following attack: instead of stealing the user's private key, the hacker can modify the signing software so that it makes changes to a document before it is signed. As a result the user sees one document, but signs something else. On some occasions this attack is not discovered before the signed fake document is used against the user.

We note that smart cards do not help here because they blindly sign any data that is sent to them and the user has no way of verifying that this data is what he wanted to be signed. Furthermore, since this kind of attack is not excluded, users can falsely repudiate their signatures by saying that since they do not have total control over their signing computer, some documents could have been forged by someone else. With this setting, it is impossible to have digital signatures to be legally binding. Therefore, even when using smart cards, we must still apply previously described physical security measures for computers that are used for signing.

This setting is clearly not feasible in a normal business environment. However, there is no need to secure a normal office PC when the user only has to have a complete control over the device that is used for signing documents. In current days, various mobile devices have quite reasonable computing abilities and displays. In this article we present our solution to secure signing problem: a personal security environment that is based on a mobile computing device.

2 Related Work

The first mention of mobile signing device we are aware of comes from Donald Davies [14]. He noted that running signing software on a normal workstation is insecure and proposed a special-purpose device for signing financial transactions. This device would look like a calculator and would let the user to enter the account numbers, product codes and amounts applicable to a transaction. From these data it would generate a transaction order and sign it.

David Chaum builds a framework for achieving privacy and untraceability by using card computers in digital transactions [13]. The card computer was meant to be a mobile hand-held computing device capable of accepting smart cards and used to connect different digital identities a person might have for different purposes.

European Community's ESPRIT program includes a project [10], which has the goal of implementing off-line digital payment scheme using two kinds of devices: wallets and guardians. Wallets are devices that contain a screen and some sort of keyboard, guardians are smart cards that store financial information. In this scheme, the guardian protects the interests of the money issuer while the wallet protects the interests of the user: payment information is communicated to the guardian through the wallet, which asks the user to confirm that this information is correct. Since the wallet is under the user's control, it prevents fake terminal attacks where the payment terminal shows one amount on screen, but deducts a completely different amount from the user's card.

Arnd Weber notes in [21] that in current electronic commerce schemes customer bears all the liability for everything that is done using his private key. He argues that the user should be liable only if he was using a secure signing system, otherwise his signature should bear no legal value because it could be easily faked. He also proposes a special-purpose "electronic wallet" that would contain a screen and a keypad and could be used for money transfers.

Gobioff et al [17] analyze the problem that smart cards do not have any means of direct communication with the users. They propose schemes to substitute secure input with

secure output and vice versa. However, their schemes are not very applicable for practical use since they require the user to perform some encryption in his head.

Balfanz and Felten [9] observe that normal personal computers should not be trusted to perform cryptographic operations. They implement PKCS#11 [6] compatible library on PC that performs cryptographic operations on Palm PDA. As an added feature, they use the Palm to enter the PIN code and display the decrypted text so that sensitive information never leaves PDA. Despite this, their solution is still not secure because the document is displayed and hashed on an insecure PC. The PDA acts only as a smart card with a secure keyboard.

In their list of risks concerning public key infrastructures, Ellison and Schneier [16] note that PKI assumes that the user is responsible for everything that is performed using that key, even when this key is not kept very securely. They also point out that the verifying computer must be as secure as the signing computer or else the whole system breaks down.

In their article [11] Boneh and Daswani describe personal security environment built on Palm PDA. They use this PSE as an electronic wallet that is capable of producing digital signatures as part of their wallet protocol.

3 Our Solution

3.1 Choice of Platform

When creating a mobile signing device, there are several important design decisions to make. There is a choice between mono-functional and multi-functional devices. A mono-functional device can be made more secure. Firstly, it can be designed from scratch to be physically tamper-resistant. Secondly, we have only one program running on it and this program interacts with the untrusted outer world through a clearly defined application-level protocol, which allows the user to transfer only signed and unsigned documents. Signing key can be kept in a well-protected part of the memory.

On the other hand, all this device can do is to sign documents. Lots of people already have mobile phones and personal digital assistants (PDAs) and they do not want to carry around yet another device just for signing (except for high-end users who often sign documents with significant monetary value). Most users would want to have signing functionality integrated into their mobile phone or PDA instead of having to pay extra money for specialized signing device. It is sure that a mono-functional device can be made more secure, but in the authors' opinion the efforts to create such a device would not pay off on the today's market. So we chose the multi-functional signing device paradigm for our prototype because it is more user-friendly and easier to implement.

Two most obvious devices that can also incorporate signing function are a mobile phone and a PDA. Both have adequate screens for previewing documents and input devices to enter passwords for unlocking signing keys. Mobile phones have the advantage of a built-in smart card reader and they can also be used to transfer signed documents using SMS or WAP protocol. On the other hand, PDAs have larger screens and better input facilities enabling the users to create documents on the signing device itself. In addition, almost all the PDAs let the users upload their own software and have developer tools and programming information readily available, whereas the mobile phones have their software hardwired into the memory.

We chose Palm PDA [1] as our development platform mainly for following reasons:

- It has the choice of several development tools and its operating system documentation is freely available;
- It has large screen (related to its size). It is also one of the smallest PDAs.
- It is one of the most popular PDAs because of its moderate price and high usability. Its low price also means that user can use his PDA only for signing, thus gaining the increased security of a monofunctional device.

3.2 Design Issues

Our prototype personal security environment for signing electronic documents consists of a Palm III PDA, a smart card reader and a smart card. In the following section some technical details are considered in more depth. The choice of Palm leads us to a very natural question – do we really need something additional to our PDA? It would be very convenient to do all the cryptographic operations in Palm and thus reduce the cost and increase the usability of the overall system. There are several problems though which have lead us to the smart cards.

The Palm is not a suitable device for some cryptographic primitives. The RSA 512 bit key generation takes approximately 4 minutes on its 16MHz Motorola 68000 processor. Signing with this key requires about 7 seconds. If some well-formed public exponent such as 3 is used, then the verification procedure will be relatively fast (1 second). The issues are much worse with the 1024 bit RSA where the key generation takes about 30 minutes.

This may seem as a hindering factor because even longer keys (2048 bits according to [18]) are recommended for today's applications. However, these recommendations assume that the signing key must be secure for the lifetime of the signature (could be 20 years for a long-term lease

contract) and thus at least 512 bits long to gain any security at all. It is possible to improve the situation by using state of the art time-stamping methods [12]. In this case our key needs to be secure only until the time certificate is valid. Since cracking the 512-bit key still requires a major computing effort, it can be considered reasonably secure for that time period. Still, we did not choose the way of time-stamping as it requires a whole new infrastructure not present as a standard today.

There is no crypto-API in PalmOS. Starting with the PalmOS 3.1 there are some cryptographic primitives implemented in PalmOS API (DES, MD4, MD5), but those functions are not suitable for our application. Most of the security software vendors for the Palm have implemented cryptographic primitives on their own. Still, they like to use symmetric primitives, because of the Palm's weak processing power. For purposes of digital signatures asymmetric primitives are essential to be implemented.

There is a version of SSLeay 0.8.1 ported to PalmOS which implements a wide range of cryptographic primitives starting from hash functions (SHA1, MD5) and symmetric ciphers (DES, IDEA) to asymmetric crypto-algorithms such as RSA. Unfortunately this library is no longer being supported (although SSLeay has found its successor in OpenSSL). During the development we tried to port OpenSSL to PalmOS, but had to stop because of the restrictions of the Palm. The main problem was that OpenSSL discontinued support for 16-bit systems. Also, some special features of Palm development tools made the porting of OpenSSL quite a tedious task. The Metrowerks CodeWarrior did not have shared library support suitable for us, GNU tools had several bugs. The interested reader is referred to [2] for further information about developing software for Palm platform using GNU tools.

PalmOS does not have memory protection. Every part of the memory can be read by every process. This is sufficient enough to steal the private keys kept in the Palm. One could write a malicious program which would just send the signing application's database to the PC during data transmission between the Palm and the desktop computer.

Implementation problems together with security reasons forced us to choose smart cards as cryptographic devices for our system. We have implemented a library compatible with the PKCS#11 standard. Currently we support message digesting, signing and verification with smart cards.

3.3 PKCS #11 library

Our PKCS#11 library currently has support for Gemplus, Setec and Schlumberger smart cards .

The hardware token needs a smart card reader to be attached to the Palm. We created the reader as part of our project. It connects to the Palm's serial port. Since the user usually needs to transfer documents to and from his Palm before and after signing, we have made our reader a pass-through device so that there is no need to take the Palm out of the reader for transferring files. The only restriction is that when the card is in the reader, Palm communicates exclusively with the smart card and the pass-through connection is disabled.

Currently there are no commercial products that enable Palm programmers to make use of smart cards. Our solution opens a whole new area of applications for the Palm platform. We have already described the secure signing environment implemented using this platform. The smart card reader with Palm is a good platform for other applications as well. One possible example is a police officer using our terminal to check one's electronic ID-card.

3.4 Signing Application

We have implemented a sample application for our library, which is able to sign records of PalmOS MemoPad application. Our software accesses the MemoPad's database and gives the user the ability to view the documents kept there. If the user has made a request for signing a currently active document, then the PIN for smart card is required. If the PIN is entered correctly the document is sent to the token for hashing, otherwise the access to the token is denied. After the hashing is complete, the hash value will be signed using the private key of the current user. The signature will be attached to the document and the signed document is kept in proprietary format.[1] We include signers' public key in the signature and this can be used to check the validity of the given signature. There is an easily portable application written in GNU-C that can be used to verify the signed documents on the PC.

4 Security Concerns

First of all, it should be noted that having a secure environment for signing is not enough. A verification of signatures must also take place in a secure environment. Otherwise, instead of trying to attack a highly secure system, the attacker can just modify the software that does the verification to display results that are favorable to him.

Although our solution is considerably more secure than a conventional PC-based signing, there are still some security concerns if we analyze our system from a conservative point of view. First of all, it is necessary for the user to carry his PDA with him all the time. If he leaves it out of sight for a couple of moments, there is a chance that somebody might

[1] Currently our software is for demonstration purposes only. Our aim was to create a PKCS#11 library and show its usability by building a lightweight application. See also the note about PKCS#7 signatures in the section 5.

have modified his signing program so that it signs false documents. Furthermore, a PDA's hardware may be modified so that false documents are signed even when signing software is compiled from clean, verified sources and installed from a trusted PC. For this reason, one must be careful when acquiring the signing device. Especially accepting a PDA with preinstalled software from a system administrator is the source of potential problems. Ideally, each employee should buy his device directly from a shop where he can pick one at random from a large crate full of identical PDAs. Of course, one must still trust the manufacturer, but the risk of getting a PDA modified for a particular user is highly reduced.

The physical security of the smart card reader must be considered as well as it can be exchanged for modified reader too. Although encryption can be used to secure the connection between the smart card and the PDA, there are still problems with key distribution, because current smart cards do not support any key exchange protocols.

Additionally, it must be noted that the user should be suspicious about his smart card, too. Although smart card is a reasonably safe place to store keys, there is no reason to assume that the card that you are about to use for digital signature is your own.

Imagine the following attack: you will go to sauna and leave your clothes in a locker. The attacker opens your locker and replaces your card with fake one that looks exactly like yours. The next time you start signing some document, the fake card captures your PIN code and sends it to the attacker. The attacker now has both your card and your PIN and can start signing documents on your behalf.

To fight this attack, you can let your card create a key pair and store the public key in your computer. You can even get a certificate on that key from some certification authority so that you won't have to store it very securely. Then you can verify the authenticity of your card by letting it encrypt random nonce. If you can verify the result using card's public key, you can safely send your PIN to card. Some cards like Cyberflex Access and Cryptoflex even have special commands "Internal authenticate" and "External authenticate" that are used respectively for authenticating card to terminal and authenticating terminal to card. The second command is dualistic to the first one: the card sends nonce to terminal and verifies the result with an internally stored public key. This enables to authenticate the user with cryptographic methods instead of just sending the PIN over unprotected channel. However, to make use of this feature, one must have some secure place for storing one's private authentication key. But if we had a secure place for storing keys, we wouldn't need smart cards in the first place.

It could be argued that adding a smart card reader does not necessarily improve the security of our system. From a conservative point of view this is true: although the key is not stored on the user's PDA, the user must still keep it in a safe place together with the smart card and the smart card reader. Also, introducing the smart card adds complexity to the system and creates several new subsystems (smart card and reader) that can be attacked. For further discussion about risks associated with adding smart cards to systems, see article [19].

However, for practical purposes this is not necessarily so. Consider two cases: in one case hacker manages to acquire the victim's PDA that contains the signing keys in the PDAs memory. In the other case, he gets the PDA, the smart card reader and the smart card. In the first case, all he has to do is to have the PDA dump all its memory contents to a PC and the extract keys. He probably will not have to use the statistical approach described in [20] because it is very likely that some popular signing program is used that stores its keys in some known location. Then he can perform a brute-force attack on the user's PIN which is used to encrypt key and he can successfully fake signed documents from the privacy of his home. For popular signing applications this attack is probably pre-packaged in some user-friendly exploit program.

In the second case, the hacker has two options: he can tamper with the signing program so that it modifies the documents that are sent for signing. In that case he has to think of a way to make the user believe that the signature is correct and he also must have some way to retrieve this signature when it has been created (which probably means that he will have to steal this PDA once again). The second option is to try to extract the key from the user's smart card using appropriate equipment. Admittedly, all these attacks can be performed by intelligence agencies and really determined individuals. On the other hand, they could also threaten the user with a gun and make him sign whatever they want to with or without a secure signing environment. This means that technical security measures can only be used to protect assets that are less valuable than the cost of a physical attack. In that case, our system can be considered secure because mounting a technical attack (like repeatedly stealing user's PDA or reverse-engineering the smart card) requires a lot of resources. If one limits the things that can be done with his PDA and a smart card – like setting an upper limit on the amount of money that can be affected by any signed document – he is quite safe because the costs of attacking his device can exceed benefits that can arise from a successful attack.

5 Future Directions

Our work with creating a personal security environment will not stop after the first prototype. Our main plans for future are the following:

- Support for standard formats like PKCS#7 [7] for sig-

natures, PKCS#10 [5] for certificate request messages and X.509v3 [8] for user's certificates. Support for these formats is especially important for verification because we would like to be able to securely verify any signed documents in standard formats.

- Support for cryptographic authentication of the smart card.
- Better support for multiple identities. The identities are associated with different private keys and represent different roles of a person. For example, one may sign documents as an employee of company X or as a member of some club. We would like each identity to have a separate security policy that determines the certification authorities that are considered to be "trusted".
- Currently we have an ASCII text viewer for PalmOS, but one would like to have viewers for other document formats also, e.g. XML and HTML. On the other hand, involving more complex document formats causes a serious security problem as one bit string can have several visual interpretations. E.g. a HTML-document can be viewed both with an ASCII-editor as a source code and with a web browser as a formatted text. One of the several interpretations can have more visual information than the other. A malicious individual can include comments like

  ```
  <!-- I give my house for free to
  John Smith -->
  ```

  into a HTML-document. If one signs the document having seen it only with the web browser, John Smith can later claim in the court that the signer had the possibility to see the source code and its content is actually what he meant at the moment of signing. An attack of this kind implies the need for including document type definitions into the documents as well. This technique is standard for XML document format. Unfortunately some of very widespread document formats (e.g. Microsoft Word's .doc-files) have non-public definitions and hence can not be used in security-critical applications.

6 Conclusions

The current digital signature implementations, where one's private key is kept in an untrustworthy PC, have proven not to be suitable for today's applications such as e-commerce. Today's digital signature must have equal legal value with the old-fashioned hand-written signature. Thus one's private key must be kept secure by all means.

Attention has to be paid to the signing environment as well. There is no use in securing one's private key if the signing environment can be easily compromised by some Trojan horse. In such a case, malicious programs can sign anything on user's behalf. In this article we have proposed a personal security environment consisting of a Palm III PDA, a smart card and a smart card reader. This environment makes it possible to convince one in the authenticity of the document to be signed. The user's private key is kept in the smart card and document signing and verification takes place in the device that can be kept in a secure place.

Security still was, is and will remain a very delicate question. If someone else's signature is badly needed on the document it will usually be given in one way or another. It is only a matter of the price to be paid. In principle, when one has enough time, money or other resources, our system can be compromised as well. Still the attacker's life is made very hard and our system is secure when compared to the old signing environments. If good care is taken of the environment and time-stamping is used in addition to signing, then the system can be put in everyday use without the fear of one's signature being abused by those who must not have access to it. For applications where very valuable signatures are given, additional security measures should be considered.

References

[1] Palm home page: `http://www.palm.com/`.

[2] PalmOS GCC development tools: `http://www.palmos.com/dev/tech/tools/gcc/`.

[3] Public-Key Infrastructure (X.509) (pkix) Charter: `http://www.ietf.org/html.charters/pkix-charter.html`.

[4] Simple Public Key Infrastructure (spki) Charter: `http://www.ietf.org/html.charters/spki-charter.html`.

[5] *PKCS #10 – Certification Request Syntax Standard*, 1993. Available online at `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-10/index.html`.

[6] *PKCS #11 – Cryptographic Token Interface Standard*, 1993. Available online at `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html`.

[7] *PKCS #7 – Cryptographic Message Syntax Standard*, 1993. Available online at `http://www.rsasecurity.com/rsalabs/pkcs/pkcs-7/index.html`.

[8] *ITU-T Recommendation X.509: The Directory: Authentication Framework*, 1997.

[9] D. Balfanz and E. Felten. Hand-held computers can be better smart cards. In *Proceedings of USENIX Security '99*, August 1999.

[10] J. P. Boly, A. Bosselaers, R. Cramer, R. Michelsen, S. Mjolsnes, F. Muller, T. Pedersen, B. Pfitzmann, B. S. P. de Rooij, M. Schunter, L. Valle, and M. Waidner. The ESPRIT project CAFE - high security digital payment systems.

In D. Gollmann, editor, *Proceedings ESORICS'94*, volume 875 of *LNCS*, pages 217–230. Springer-Verlag, 1994.

[11] D. Boneh and N. Daswani. Experimenting with electronic commerce on the PalmPilot. In *Proceedings of Financial Cryptography '99*, volume 1648 of *LNCS*, pages 1–16. Springer-Verlag, 1999.

[12] A. Buldas, P. Laud, H. Lipmaa, and J. Villemson. Time-stamping with binary linking schemes. In *Advances in Cryptology – CRYPTO'98*, volume 1462 of *LNCS*, pages 486–501, Santa Barbara, 1998. Springer-Verlag.

[13] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[14] D. W. Davies. Use of the 'signature token' to create a negotiable document. In *Advances in Cryptology: Proceedings of CRYPTO'83*, pages 377 – 382, New York, USA, 1984. Plenym Publishing.

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

[16] C. Ellison and B. Schneier. Ten risks of PKI: What you're not being told about public-key infrastructure. *Computer Security Journal*, 16(1):1–7, 2000.

[17] H. Gobioff, S. Smith, J. D. Tygar, and B. Yee. Smart cards in hostile environments. In *Proceedings of The Second USENIX Workshop on Electronic Commerce*, Oakland, CA, 1996.

[18] A. K. Lenstra and E. R. Verheul. Selecting cryptography key sizes. Available online at `http://www.cryptosavvy.com/cryptosizes.pdf`, 2000.

[19] B. Schneier and A. Shostack. Breaking up is hard to do: Modeling security threats for smart cards. Available online at `http://www.counterpane.com/smart-card-threats.html`, 1999.

[20] N. van Someren and A. Shamir. Playing hide and seek with stored keys. Available online at `http://www.ncipher.com/products/rscs/downloads/whitepapers/keyhide2.pdf`, 1998.

[21] A. Weber. *Distribution of risks in Implementations of Digital Signatures*. Univ. Freiburg i. Brsg., 1997. Available online at `http://www.semper.org/info/111FR023.ps.gz`.

[22] P. R. Zimmerman. *The Official PGP User's Guide*. MIT Press, 1995.