Beyond Parasitic Authentication

Lucas C. Ferreira
IC-Unicamp and Webmind Inc.
lucasf@ic.unicamp.br

Ricardo Dahab
IC-Unicamp
rdahab@dcc.unicamp.br

Abstract

Parasitic authentication [Ebringer et al., 2000] is a novel approach for user authentication in electronic wallet systems which combines security and usability. We propose an extension to parasitic authentication that integrates it to payment protocols thus increasing the security of electronic wallets. The additional security comes at the expense of greater complexity in the secondary devices used in the authentication process.

1 Introduction

Applications running in mobile devices, specially electronic wallets, need authentication methods that (i) require the physical presence of the user in the authentication process but are otherwise transparent to him or her; and (ii) do not require the user to type passwords very often. In [Ebringer et al., 2000], the authors discuss these requirements and present *parasitic authentication* as a solution to the problem of authenticating users of electronic wallets. They identify three classes of authentication, which are based on (i) something known to the user, (ii) something possessed by the user, or (iii) something inherent to the user. Passwords are a typical example of the first class: assuming that only the authorized user knows the correct password, the system will authenticate anyone able to produce the right combination of user-name and password. The second class makes use of tokens owned by users. Examples of these are smart cards and synchronized *one-time password* electronic tokens. The third class includes biometric techniques, which measure some physical characteristic in order to identify the user. The three classes are listed in increasing order of security. Biometrics promise the greater level of security and ease of use, but its acceptance present difficulties such as privacy concerns and the impossibility of revocation of the measured characteristic.

Parasitic authentication is based on using tokens that can be hidden in the user's clothes and that answers authentication requests. As such, parasitic authentication lies somewhere in between the use of tokens and biometrics: as with biometrics, the property needed for authentication is supposed to be part of the user; however, the system still presents the possibility of revoking or changing that property. Moreover, using tokens in conjunction with parasitic authentication opens the possibility for even more secure and elaborate authentication protocols.

We argue that, although parasitic authentication is a step towards better identification systems for electronic wallets, its current form is inadequate since users may still have their wallets used by knowledgeable thieves. We propose pushing this paradigm a step further and integrating the authentication token into the payment protocol thus increasing the security of the system while maintaining usability.

Section 2 below sketches the architecture and protocols from [Ebringer et al., 2000]. In Section 3 we present our approachfor increasing security and convenience of parasitic authentication. The last section presents our conclusions.

2 Parasitic Authentication

Parasitic authentication [Ebringer et al., 2000] involves both new architecture and protocols. The new architecture includes a novel device, the *authenticator*, responsible for authenticating the user. The protocols specify how the

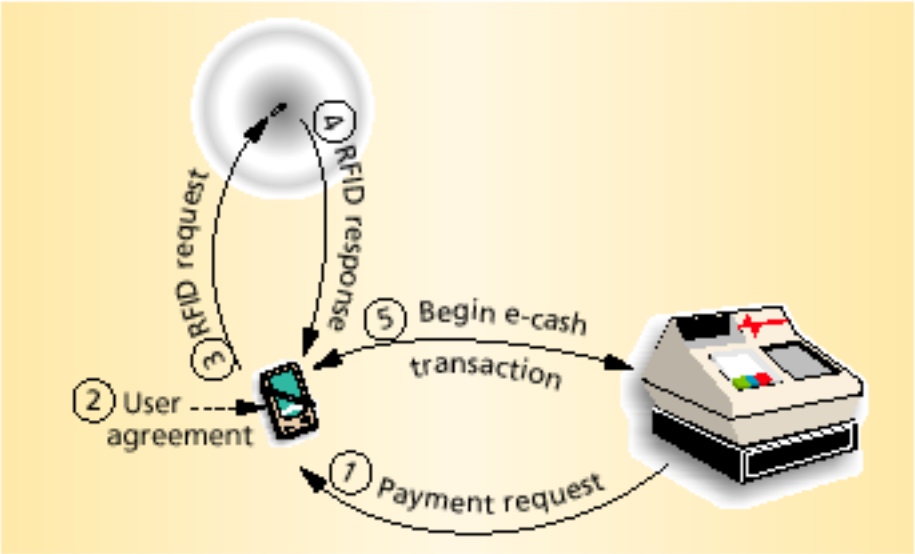

Figure 1: The parasitic authentication architecture, showing the wallet at the center, the transceiver or authenticator at the top and the merchant terminal on the right.

wallet will interact with the authenticator in order to achieve secure user authentication. We will first describe the architecture and then sketch the protocols proposed by Ebringer et al.

2.1 Architecture

The authenticator's operation is based on its proximity to the electronic wallet. More specifically, the wallet does not function away from the authenticator. Thus the user can carry the authenticator in the pocket and be sure that the wallet will not operate in case it is misplaced or stolen.

Figure 1 shows the elements and the communication pattern for the original parasitic authentication protocols:

Electronic Wallet It is responsible for storing all user-related payment information and executing the payment protocol on behalf of the user. It relies on the proximity of the authenticator to identify the user. The wallet should remain blocked if it can not communicate with the correct authenticator.

Authenticator Also called the transceiver, it is responsible for authenticating the user to the wallet, prior to allowing it to engage in payment transactions. Its complexity can vary from a simple password brodcasting device to a complex processor performing modular arithmetic and Schnorr authentication [Schnorr, 1990].

Merchant terminal It acts on behalf of the merchant, interacting with the wallet in order to perform a transaction.

2.2 Protocols

The protocols used in parasitic authentication vary with the complexity and computational power of the transceiver. We first present the basic protocol and then show four different versions for different classes of transceivers. [Ebringer et al., 2000] introduce parasitic authentication as follows: "With parasitic authentication, users can temporarily delegate their responsibility for authorizing a transaction to a small, portable secondary device, carried and concealed by the user. As long as the e-wallet can verify that it is indeed the *same* secondary device that was involved in the setup process, the e-wallet acts as if it has the necessary authorization to complete a transaction. Thus, the *continued proximity* of the secondary device to the e-wallet is the source of authentication. Existing identification protocols ensure that the secondary devices that do not belong to the owner of the e-wallet cannot confuse or spoof the e-wallet. This is akin to a concept intimately familiar to most users: *sessional authentication*."

2.2.1 Basic protocol

When the user and the merchant decide to use an electronic payment system to settle a purchase, the user will power on its wallet and the merchant will instruct its terminal to initiate a payment transaction. Before the wallet begins the transaction, it must get authorization from the user, which is done with passwords in conventional systems. Parasitic authentication-based systems will require the password just once in a session, with the authenticator providing authorization throughout the session[1]. The main steps of a transaction using parasitic authentication are (as shown in Figure 1):

1. The merchant terminal sends a payment request to the user's wallet.
2. The user agrees with the content (description, price etc.) of the transaction.
3. The wallet seeks authorization. If there is no open authorization session, the wallet asks the user to open one by inputing a password. In any case, the wallet will contact the authenticator and wait for the correct answer.
4. The authenticator answers the request.
5. If the answer is correct, the wallet starts the payment transaction with the merchant terminal.

2.2.2 Variations of the basic protocol

Depending on the complexity and computational power of the secondary device, different protocols may be used for the continued authentication. Four different systems of increasing complexity were proposed, from a simple passive transponder that only broadcasts its serial number to an authenticator that can perform modular arithmetic.

Passive RFID transponder. A RFID (Radio Frequency Identification) transponder is only able to broadcast its identity. Accordingly, in step 3 of the protocol the wallet broadcasts a query which is answered by the transponder in step 4.

Transponder with memory. Here the wallet initializes the transponder by storing several hash values in its memory. In step 3 of the basic protocol the wallet requests one of the hashes from the transponder that must answer with the correct value in step 4.

Transponder that computes hash functions. In this case the electronic wallet initializes the transponder by storing a secret key value k in its memory. This key will be input to a hash function $h(k, x)$ used by both the wallet and the transponder (the reader is referred to [Menezes et al., 1997] for more information on cryptographic hash functions). Then in the third step of the protocol, the wallet generates a random number r and transmits it to the transponder. The transponder answers with $h(k, r)$. The wallet computes $h(k, r)$ independently thus checking the answer.

Transponder that can perform modular arithmetic. Here steps 3 and 4 of the protocol, which atually perform the authentication, correspond to the steps of Schnorr's authentication scheme [Schnorr, 1990], not explained here. Schnorr's scheme is believed to be computationally secure and requires only a few modular arithmetic operations.

2.3 Limitations

In their paper, Ebringer et al. argue that it is possible to subvert parasitic authentication by reverse engineering the electronic wallet, eliminating user authentication operations from its program. In other words, with the original parasitic authentication scheme, as the authenticator is only used to unlock the wallet, it may be possible to use well-known hacking techniques to force the wallet to execute a payment without authenticating the user. Ebringer et al. mention these attacks but argue that cryptographic techniques to prevent them would make the authenticator too expensive and slow.

We argue that preventing these attacks may well be worth it, since subverting the authenticator and the wallet software may cause severe damage to the rightful owner of the wallet/authenticator pair. One solution for this limitation is to use the authenticator as an active player in the payment protocol, thus making impossible the execution of the protocol without it. We discuss this solution in the next section.

[1] A single session may include several payment transactions with various different merchants.

3 A Step Beyond

To overcome the limitations of parasitic authentication stated above, we have to consider the authenticator, as well as the wallet, as entities in payment protocols. We will base our discussion on the payment protocol model of [Ferreira and Dahab, 1998], whose security is based on public key encryption and digital signatures. In this scenario, the best possibility is to delegate signature operations to the authenticator, as those signatures will guarantee user authentication. The main concern here is which communication protocol the wallet and the authenticator will use to communicate. As the computing capacity of the authenticator increases, this protocol can be made better. Based on research being conducted at several places (see [Orlando and Paar, 2000]or [Leung et al., 2000]), we assume that the authenticator will at least be able to execute signature operations using an Elliptic Curve Cryptosystem with 160 bit keys (for more information on Elliptic Curve Cryptosystems, the reader is referred to the introductory chapter of [Hernández, 2000]). The next sections present possible protocols for authenticators with increasing computing power.

3.1 Signature-only authenticator

In this scenario, the authenticator is only able to execute signature operations based on a private key present in its private storage. The authenticator will then sign all 160 bit strings it receives and broadcast the signature back. In this case, the main security problems are denial of service and signing spurious bit strings.

To achieve denial of service, an adversary would keep sending strings to the authenticator, overloading it and making it unavailable to answer legitimate requests. Those attacks are unlikely in the environment parasitic authentication would be used, namely inside a store, as they can hardly be directed to a single customer and would be very likely detected, allowing the store owner to take any required action (technical or legal) to counter it. Another possibility is congestion caused by a large number of customer authenticators in a department store. In this environment, many authenticators would try to answer to each wallet request, causing not only authenticator overloading if many wallets are in use but also congestion of the radio frequency used. The solutions for those problems are limiting the transmission capacity of these equipments to a minimum and setting many possible frequencies for authenticator-wallet communication.

To avoid adversaries being able to use strings signed by someone else's authenticator, we can change the protocol to require two signatures on the payment order. The first signature is provided by the authenticator and the second is provided by the wallet, using a password protected private key. With this change, only someone that can fool the authenticator and the corresponding wallet can fake payment orders. This is certainly much more difficult than just reverse engineering lost or stolen wallets, as there is also need to have the authenticator sign the payment order. For this class of authenticators, the protocol steps are:

1. The wallet prepares a hash[2] of the payment order.
2. The wallet transmits the hash to the authenticator and begins computing its own signature of the hash.
3. Upon reception of the hash, the authenticator will sign it and air-wave the resulting signature.
4. The wallet verifies the signature and appends it to the already signed payment order and sends the result to the merchant's terminal.

After these steps, the merchant terminal can contact the bank to verify the validity and settle the payment order as in the original protocols for electronic payments.

3.2 A little more than just signing

If the authenticator is able to sign strings and perform some more logical operations, we may increase the security of the proposed protocol by forcing the authenticator to require a password to perform its signature. The password could be inserted in the device storage in the same way as its private key. That enhancement prevents overloading

[2] A cryptographic hash (in the correct format and padded to 160 bits) may be used to decrease the size of the information to be signed.

the authenticator if there are multiple wallets trying to execute the protocol near it. An adversary still can record the password by eavesdropping the transmission from the wallet to the authenticator and attempt to cause denial of service by overloading the authenticator or by filling up the radio frequency used for transmission. The adversary may also try to sign its own strings, but this signature has no use unless it can get the corresponding signature from the wallet or compose a fake message from the hash, which is computationally infeasible with a good cryptographic hash function.

If the protection mechanisms and protocol described in the previous section are used, this solution may be adequate for most environments. Another enhancement would be adding non-volatile writable memory to the authenticator to allow for the use of one-time passwords, possibly the scheme described in the next section. The complexity and security of the one-time password scheme to be used would be dependent of the computing power of the device.

3.3 Fully featured authenticator

A fully featured authenticator would have the memory and processing power to decrypt and sign 160 bit strings, execute some other logical operations and also contain non-volatile writable memory. With such an authenticator, it is possible to construct a secure protocol to guarantee secure authentication and to ensure that only authorized wallets can request signatures from the authenticator.

When a signature is needed, the wallet will send the authenticator an encrypted password and a hash value to be signed. The password is made up of two random numbers: the first is the current password and the second is the password to be used in the next request. The authenticator only executes its program if this wallet authentication succeeds. Those numbers can be 80 bit long, so the two parts of the password make up a single encryption block. This prevents replay attacks, where an adversary would eavesdrop the encrypted new password and resend it with fake data to be signed.

In this scenario, the payment protocol does not need to be changed, as only the right wallet is able to request signatures from the authenticator. An adversary is able to listen to the encrypted password and the hash to be signed by the authenticator, but is unable to decrypt the password or construct a valid message from the hash. So, even if the adversary is able to listen to the messages exchanged by the wallet and the authenticator, it is not able to build a message that would fool the bank, leading it to make an unauthorized payment. It is also possible to alter the payment protocol and require the wallet to sign the message, although we feel this is not needed. The protocol steps for this scenario are:

1. The wallet computes a two part password comprised of a new random number and the last password used.
2. The wallet encrypts the password and constructs a hash of the payment order in the right format and length.
3. The wallet builds the message appending the hash to the encrypted password and sends it to the authenticator.
4. The authenticator receives the message, decrypts the password and verifies its validity.
5. If the password is valid, the authenticator signs the hash sent by the wallet and broadcasts its signature.
6. The wallet receives the signature, appends it to the payment order and proceeds with the usual protocol.

4 Communication technologies

Parasitic authentication based systems could be used with many data transmission technologies, specially most low-power wireless technologies. Those techniques are usually named *personal area network* (PAN) technologies. A well known technology for PAN-based computing is the Bluetooth [Bluetooth, 2000] framework, that will soon be available in PDAs or cell phones. Bluetooth is a low power wireless networking system that has been designed to allow the deployment of small area networks that would span all devices carried by a user and also to communicate with access points to broader networks located near the user. Another transmission system of interest with similar characteristics is PicoRadio [Rabaey et al., 2000].

Both techniques presented above allow eavesdropping attacks, as they send data using radio signals that may be received by anyone with a receiver within a short distance of the transmitting device. To counter this type of attack, a technique called *near-field intrabody communication* [Zimmerman, 1996] may be used. This technique uses the human body as transmission medium and transmits messages as very low power electromagnetic waves through the user's body. If this technique is used, the eavesdropper would have to touch the user in order to be able to receive the messages transmitted by the authenticator and the electronic wallet, thus being detected by the owner of the wallet. This technique would also avoid frequency congestion and other problems that arise with the use of wireless communications, and would make the wallet usable only if it is touched by the user that carries the correct authenticator, making the system more secure.

5 Conclusions

Parasitic authentication is a novel authentication mechanism which makes electronic wallets more usable and less intrusive. We presented an extension to the original parasitic authentication framework which improves its security. Our approach requires more computing power for the authenticator, but we believe this computing power will be available soon, if not already available. The proposed solution involves the use of a public key infrastructure by the authenticators, but it does not complicate the deployment of such infrastructure if compared with the usual wallet-based electronic commerce. Our proposal only makes necessary to embed public keys in the authenticator, but this may be the same key that would be used in a wallet in current solutions. In the case where we propose the use of two keys, one for the authenticator and one for the wallet, it is necessary the generation of two different keys where current solutions require only one. We believe this is feasible and is a small additional cost to pay for the increase in security. Parasitic authentication is a step towards more secure and user-friendly e-commerce applications and we believe the extensions presented here are an improvement in both aspects.

References

[Bluetooth, 2000] Bluetooth (2000). Bluetooth web site. http://www.bluetooth.com.

[Ebringer et al., 2000] Ebringer, T., Thorne, P., and Zheng, Y. (2000). Parasitic authentication to protect your e-wallet. *IEEE Computer*, 33(10):54–60.

[Ferreira and Dahab, 1998] Ferreira, L. C. and Dahab, R. (1998). Formal analysis of a model for electronic payment systems. In *I Workshop Brasileiro de Métodos Formais*, pages 141–145.

[Hernández, 2000] Hernández, J. C. L. (2000). *Implementação Eficiente Em Software de Criptossistemas de Curvas Elípticas*. PhD thesis, Instituto de Computação - Unicamp. Most text in English.

[Leung et al., 2000] Leung, K. H., Ma, K. W., Wong, W. K., and Leong, P. W. (2000). FPGA implementation of a microcoded elliptic curve cryptographic processor. In *IEEE FCCM'2000*.

[Menezes et al., 1997] Menezes, A. J., Oorschot, P. C., and Vanstone, S. A. (1997). *Handbook of Apllied Cryptography*. CRC Press. Available online at http://www.cacr.uwaterloo.ca/hac.

[Orlando and Paar, 2000] Orlando, G. and Paar, C. (2000). A high-performance reconfigurable elliptic curve processor for $GF(2^m)$. In *CHES'2000*, Worcester MA, USA.

[Rabaey et al., 2000] Rabaey, J. M., Ammer, M. J., Jr., J. L. D. S., Patel, D., and Roundy, S. (2000). PicoRadio supports ad hoc ultra-low power wireless networking. *IEEE Computer*, 33(7):42–48.

[Schnorr, 1990] Schnorr, C. P. (1990). Efficient indentification and signatures for smart cards. In Brassard, G., editor, *Advances in Cryptology: Proceedings of CRYPTO'89 (LNCS 435)*, pages 239–252. Springer Verlag.

[Zimmerman, 1996] Zimmerman, T. G. (1996). Personal area networks: Near-field intrabody communication. *IBM Systems Journal*, 35(3,4):609–617.