PARASITIC AUTHENTICATION

Tim Ebringer
Department of Computer Science and Software Engineering
The University of Melbourne
tde@cs.mu.oz.au

Yuliang Zheng
LINKS – Laboratory for Information and Network Security
School of Network Computing
Monash University
yuliang@pscit.monash.edu.au

Peter Thorne
Department of Computer Science and Software Engineering
The University of Melbourne
pgt@cs.mu.oz.au

Abstract This paper focuses on protocols for human/smartcard interaction which allow the user to authorise individual smartcard transactions, whilst not sacrificing useability or security.

In the past, protocols for secure transactions have traded off useability against security whereas the protocols presented here are designed so that they tradeoff security against hardware complexity and always give high useability. Our protocols utilise some of the concepts and assumptions present in *sessional authentication*, but also make improvements to this model.

We do not propose the use of biometrics for authentication. Biometrics are viewed with apprehension by many users since they are irrevocable.

Keywords: authentication, smartcards, useability, privacy, wearable, wireless

1. INTRODUCTION

It is a significant challenge to engineer systems — particularly portable devices — that are both secure and user-friendly. Users desire both traits, but these traits tend to conflict with one another since the addition of any kind of interactive security protocol between the user and the portable device introduces inconvenience for the user.

This paper investigates protocols for human/smartcard interaction which allow the user to authorise individual smartcard transactions, whilst not sacrificing useability or security. The main idea is to transform the useability/security tradeoff into a computational complexity/security tradeoff via the use of sessional authentication and the use of some existing identification protocols.

1.1. SESSIONAL AUTHENTICATION

In a database, transactions are atomic such that either all of the operations in each transaction are reflected properly in the database, or none are. In such situations, it is both feasible and necessary to authenticate every transaction. Conversely, in the situation where a user is entering data at a computer terminal, every keystroke constitutes a transaction, yet it would be absurd to authenticate every keystroke. Thus, the assumption is made that after a user logs in, then it is the same user who is in fact pressing the keys until such time as the user chooses to log out. The authentication lasts for a session, and it is the user who decides the length of the session.

This approach forms the basis a more useable system for authorising smartcard-based transactions. The user initiates a *session* with the smartcard, such that the smartcard will have authorisation to complete transactions for the duration of the session.

We want to sustain security for the session, and hence we introduce a secondary device which will act as the user's authentication proxy for the duration of the session.

1.2. THINGS WE AVOID, AND WHY WE AVOID THEM

The interactive security procedures that have been used to authenticate transactions in the past relied on the user performing some kind of interactive protocol with the smartcard. Typically, these involved keying in a PIN, password, or else used a biometric. We argue that passwords are inappropriate and cumbersome, whilst biometrics inherently possess undesirable side-effects.

Passwords. Neither passwords nor PINs are an ideal solution; not only are they a weak identification measure, they are frequently misused.

Klein showed that, using a dictionary of 62,727 words, he was able to crack 24.2% of passwords in a typical UNIX password file (Klein, 1990). Additionally, in a portable wallet, passwords are cumbersome to enter. Users, their ingenuity often underestimated by developers, would be likely to find a method of subverting the irritating and continual authentication checks.

Kahn argued that during World War I, the Russians suffered some crushing defeats due to their soldiers reverting to a simple cipher system when they found their official cipher too hard to use (Kahn, 1967). The analogous situation with PIN numbers is that users who find a plethora of PINs too difficult to remember often weaken PIN security. For instance, using the same PIN for many different applications, choosing trivial PINs (such as '1111' or '1234'), basing the number on a personal fact like a birthday (14th of November becomes '1411') or else, on the advice of the banks, basing the number on a four letter word (since PINs are commonly four digits long)[1]. Passwords are even more cumbersome to enter into a small device and suffer a similar range of problems.

This kind of security fault, whereby users unwittingly weaken a theoretically secure system, flows directly from poor useability aspects of PIN authentication systems. Formal security standards such as ITSEC "dehumanise" protocols in that they assume protocols are rigorously followed. For example, a protocol might specify that for choosing a pin p, $p \in_R \mathbb{Z} : 0000 \leq p \leq 9999$, whereas a human may not be quite so 'random' in their selection of PIN. Anderson argued that the failure of many real-world cryptosystems can be traced to this kind of flaw (Anderson, 1993).

Both PINs and passwords suffer from the fact that an attacker may learn the secret by observation.

But perhaps the biggest problem with PINs and especially passwords is that they are awkward to enter into a small, portable device. On a dark and stormy night, a user trying to pay his ferry fare home by tapping a pin into a small device may find himself with his short-circuited smartcard spluttering in a puddle, the ferry disappearing into the gloam and only a three-headed dog for company. The protocol, whilst theoretically sound, failed to anticipate some of the broader human and environmental conditions under which it was to be performed.

Biometrics. A more effective method might be to integrate a biometric authentication device, such as a fingerprint scanner, into the

smartcard. The smartcard would not perform a transaction unless a valid fingerprint was scanned.

A concern with such a scheme is that users may feel uncomfortable storing a biometric (which many regard as something that makes them uniquely 'them') on a computer. Especially on a portable computer which would be a target for theft, users feel that if a thief was to steal their 'identity', then they could masquerade as the user — theoretically forever — since there is no way to revoke a biometric.

2. PARASITIC AUTHENTICATION

We describe a system whereby we give the user the ability to temporarily delegate their responsibility for authorising a transaction to a small and portable secondary device which is carried and concealed by the user.

Provided the smartcard is able to communicate with the secondary device and verify that it is still the *same* device that was used to begin the session, then the smartcard may assume that this constitutes the necessary authorisation to complete a transaction.

We thus rely on the continuing *proximity* of the secondary device to the smartcard as the source of authentication. We use existing identification protocols to ensure that the smartcard is not confused (or spoofed) by secondary devices that do not belong to the owner of the smartcard.

The smartcard can be regarded as a parasite, feeding off the smaller device for authentication (see figure 1 and figure 2).

2.1. BENEFITS

Such a system offers two major benefits:

- **Useability**. The user no longer has to enter a password, PIN or perform any other kind of interactive security procedure with the smartcard in order to authorise each transaction. Whilst the session is sustained, the interactive security is performed on the user's behalf by the secondary device.

- **Security**. The user does not have to store a biometric, and the user is still protected against loss of the smartcard. In order for security to be compromised, the user must lose *both* the smartcard and the secondary device simultaneously.

  If the chances of losing your smartcard are p, then since the secondary device is supposed to be small and concealable (for example, in an earring or button), and provided sensible precautions are taken (such as not gluing the secondary device to the smartcard),

Figure 1 A parasitic eWallet (red glow, in user's hand) feeding off a smaller host device (blue glow, in button and/or earring) for identification

the probability of losing the secondary device should be q, where $q < p$. We argue that $q < p$ because the secondary device need not be operated for the duration of the session (which is likely to be a day), and should be similar to an item of apparel — like an earring — that is not removed for the entire day.

If these precautions are taken, then the chance of losing both the smartcard and the secondary device simultaneously should be less than p^2.

2.2. SECONDARY DEVICE CHARACTERISTICS

We envisage the following requirements for the secondary device:

- **Miniature**. The secondary device must be small enough to be unobtrusive, and can be hidden somewhere in the user's clothing (in an inside pocket, shoe, underwear, earring, etc.).
- **Self-powered**. The authentication device must carry enough power (or be able to draw power from an energising electric field) so that it can function for extended periods of time away from a power source. In recent years, battery technology advanced remarkably, the most apparent evidence being the extended battery life of tiny mobile phones.
- **Disposable**. Loss of the authentication device should not be a major catastrophe. At worst, loss of *both* the smartcard and the secondary device would mean a window of time in which a user's smartcard was vulnerable to abuse. No information that an attacker might find useful beyond the duration of the session — such as biometric information — would be stored on the authentication device. Clearly, it would also be highly advantageous for the secondary device to be cheap.
- **Wireless**. In order to keep the secondary device hidden, and to not inconvenience the user, it must communicate wirelessly.

2.3. COMPARISONS

The idea of Parasitic Authentication is partly an evolution of ordinary wireless authentication which has been employed in access-control mechanisms. However, there are significant differences: Parasitic Authentication has been designed with low power and portability in mind; in most traditional wireless authentication mechanisms, the device doing the authentication is immobile.

The service which the Parasitic Authentication system provides also differs from that expected from access control systems. The primary responsibility of access-control systems is to prevent unauthorised access. A secondary consideration is to provide the service level appropriate to the authentication level of the individual. In the case of Parasitic Authentication, the normal mode of operation is not to challenge the access right, but to confirm authentication in a symbiotic relationship.

2.4. ENDING THE SESSION

There are several ways for a session to end; the smartcard could end the session itself after a certain time period has elapsed; alternatively, if the smartcard was being used as an eWallet, the smartcard might end the session after a certain amount of money was spent, thereby capping the possible losses. Another option might be for the user to end the session themselves by using a 'panic' button. Choosing more conservative ways to end the session better protects the user in the event that *both* the eWallet and the secondary device were lost simultaneously, but decreases useability (since the user must begin a new session to resume use of the eWallet).

2.5. TRADEOFFS

There are certain engineering tradeoffs in the design of the authentication device which we will discuss in section 3. As the computational power of the device increases, it can perform increasingly sophisticated identification routines. However, it also becomes more complex, needs more power, and correspondingly becomes less portable and user-friendly (see Figure 3).

Realistically, however, we reach a limit where the authentication device has enough computational power to satisfy the initial design goals, beyond which additional complexity does not offer significantly increased security.

3. SYSTEMS

We offer several systems which are suitable for transponders of possessing certain levels of computational power. The first is for a transponder that is completely passive and can only return a 'serial number'.

The second is for a transponder that has extremely little computational ability, but can store a number of values, and subsequently retrieve and transmit them.

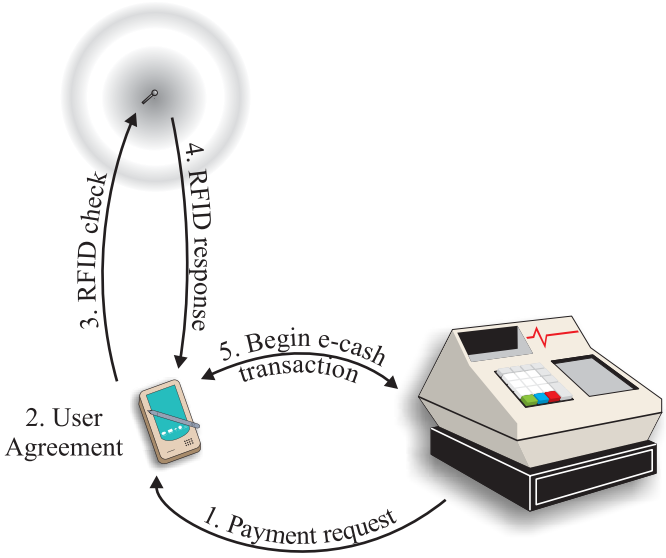

Figure 2 Flow of intended usage

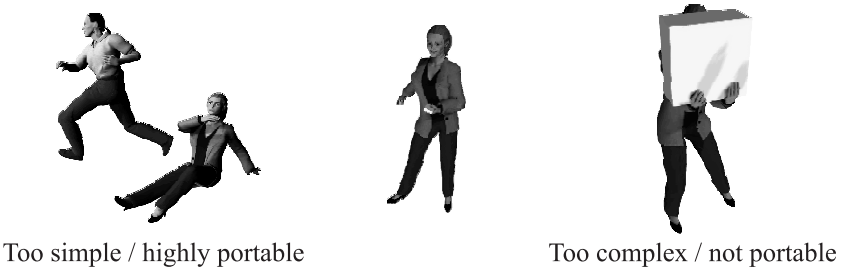

Figure 3 Tradeoffs in the design of the transponder

The third is for a host that, as well as the capabilities above, can compute a one-way hash function and the last is for a transponder that can perform modular arithmetic.

In our examples, we will assume that the smartcard is functioning as an electronic wallet (eWallet). This is a most appropriate application for demonstrating our protocols. The CAFE survey found that eWallet users demand protection against theft in their eWallets (et al, 1994), and the Nikkei surveys showed that users found the security built into eWallets to be cumbersome and not user-friendly (nikkei, 1998; nikkei, 1999).

3.1. FOUNDATIONS

Poly-random collections. Some of the systems we propose both assume and require the existence of one-way functions. Poly-random collections from long bit-strings to short bit-strings constitute very good hash functions (Goldreich et al., 1986), but in order to ease interoperability and implementation, we suggest using the SHA-1 hash function which maps an string of arbitrary length to a 160-bit string $\{0,1\}^k \rightarrow \{0,1\}^{160}$.

We will assume the existence of a one-way, polynomial time computable hash function which we will refer to as $\mathcal{H}$.

Schnorr identification protocol. The most computationally intense system we offer here utilises an identification protocol. We have selected Schnorr because of the small computation and communication overhead it introduces, but it could easily be substituted for a similar scheme such as Fiat-Shamir (Fiat and Shamir, 1987) or Guillou-Quisquater (Guillou and Quisquater, 1988), depending on exact requirements.

3.2. SYSTEM 1: A PASSIVE, RFID-TYPE TRANSPONDER

Here we propose using a multi-bit transponder as the host. Although not sophisticated cryptographically, this system should offer the equivalent security of passwords, be inexpensive and should effectively protect the user from their wallet being abused if they misplace it.

Transponder technology. Transponders were originally electronic circuits that were attached to some item whose position or presence was to be determined. The transponder functioned by replying to an interrogation request received from an interrogator, either by returning some data from the transponder, such as an identity code or the value

of a measurement, or by returning the original properties of the signal received from the interrogator with virtually zero time delay, thereby allowing ranging measurements based on time of flight. As the interrogation signal is generally very powerful, and the returned signal is relatively weak, the returned signal would be swamped in the presence of the interrogation signal.

The function of the transponder was therefore to separate the returned signal from the interrogation signal so that both could be detected simultaneously without one swamping the other.

RFID stands for *radio frequency identification*. It is a widely varied collection of technologies for various applications, ranging from the high speed reading of railway containers to applications in retail. It can be regarded as a potential successor to the barcoding technologies in use today. RFID is based around radio or electromagnetic propagation, providing the capability to read a hidden tag.

Commonly available tags have an operating frequency in the range from 60 kHz to 5.8 GHz depending on application.

Three common types of technologies being implemented are:

- Magnetic based RFID technologies
- EAS based technologies
- Electric field based RFID technologies

However, the properties we require from RFID for our first system are that the transponder be *passive* (have no power source of its own) and *multi-bit*. Multi-bit transponders differ from familiar single-bit anti-shoplifting transponders since they can indicate not only their presence, but also a serial number.

It so happens that the electric field transponders have the characteristics we require: the invention of the backscatter modulation principle at Lawrence Livermore Laboratories in the 1960s and the skills of semiconductor designers to shrink all features into cheap integrated circuits, has meant that electric field type tags in a read only mode can be made extremely cheaply, most probably for less than 10 US cents in high volume. Such a tag would be passive, have no onboard tuned circuits, be read only, consist of a single integrated circuit and a simple antenna, would operate at any of a range of frequencies, be temperature insensitive, and would broadcast a large data value when illuminated by a reader's energising field. In such a system the reader is complex because it provides the frequency stability, the energy of the system, and the receiver selectivity to receive the weak return communications, but the

tags are very cheap. This is ideal for the situations where there is one reader and many interchangeable tags.

Session initialisation and operation. If we build reader capabilities into the eWallet, then we can initiate a session with the eWallet so that it uses a particular transponder. This would involve telling the eWallet to accept a new transponder (perhaps by entering a password[2]). The eWallet would subsequently broadcast a query to which the nearby transponder would respond with its serial number.

Subsequent use of the wallet would involve the eWallet broadcasting a query and waiting for the expected serial number to be returned.

For this first system, we flesh out the protocol with an example of its intended usage. Since usage is intended to be identical for all four systems, we outline only the basic protocol in the description of the subsequent systems.

Session initialisation.

1 A transponder which has the random and unique serial number S is placed near (or plugged into) the eWallet.

2 The eWallet is told to accept a new transponder by entering a password.

3 The eWallet broadcasts a query and receives the value S from the nearby transponder, which it commits to memory.

4 The transponder is dropped into an inside pocket and the eWallet is taken shopping.

Operation.

1 The user initiates a SET transaction with their eWallet. When, as part of the SET protocol, the eWallet is expected to send order and payment information to the merchant, it first broadcasts a query to the transponder and verifies that it returns the value S. The SET protocol is then resumed and order and payment information are sent to the merchant.

2 The user then forgetfully leaves their eWallet in the shop.

3 The merchant (who is really an evil witch), gleefully takes the eWallet and tries to crank a few 'extra' SET transactions through it.

4 The eWallet, before it sends order and payment information, first broadcasts a query. Since the transponder is still in the user's pocket and out of range, the eWallet cannot get the return value S and therefore aborts the transaction.

5 The witch puts a curse on the user instead.

Efficiency and security of a passive transponder. This system of using a passive transponder is ineffective against a more sophisticated adversary. If the witch in the previous example had her own receiver, then it could have also picked up the return value from the transponder and the return signal of the could easily be spoofed.

Thus, this system provides protection against a casual attacker. But note that it is at least as good as using a password. If a password were entered, the witch might be able to observe what was entered, either by carefully watching the user, using a hidden camera or else referring to her crystal ball.

Another way around the transponder check is to hack the software within the eWallet itself. Modifying the software to ignore the results of an authentication check, while beyond the abilities of a casual computer user, is easy for a specialist with the right tools. However, how difficult it is to make this modification on a handheld computer depends to a large extent on the implementation of the device. Thus, although it is possible to subvert the transponder in this manner, it is unknown whether it will become common practice.

This system offers the user the equivalent of password protection, whilst significantly increasing the useability and efficiency of entering authentication information into their eWallet. It is not trivial for a thief to force the wallet to perform a transaction in the absence of the user (and the transponder).

Moreover, the transponder is certainly disposable — such a device should cost less than U.S. $1.

3.3. SYSTEM 2: A TRANSPONDER WITH MEMORY

Assume that the transponder can store an array of hash values.

As an example, assume that the transponder can store 10,000 160-bit hash values in an array $R[0], \cdots, R[9999]$ and can retrieve and transmit any of these values on demand.

Session initialisation.

1 The eWallet is brought within range of the user's transponder (and out of range of any other transponders) or else, more securely, the transponder is plugged into the eWallet. The user enters their PIN into the eWallet to authorise transfer of authorisation responsibilities from themselves to the transponder.

2 The eWallet generates a secret, random key k.

3 The eWallet generates $\mathcal{H}(k,0), \mathcal{H}(k,1), \cdots, \mathcal{H}(k,9999)$ and transmits these values to the transponder where they are stored in the array.

Operation. The eWallet wants authorisation to make a transaction.

1 The eWallet requests $R[j]$ from the transponder. It checks that $R[j] = \mathcal{H}(k,j)$.

2 The *next* time it wants authorisation, it requests $R[j+1]$.

3.4. EFFICIENCY AND SECURITY OF A TRANSPONDER WITH MEMORY

This system, if implemented carefully, is computationally secure. However, the transponder clearly must not transmit any values more than once. Enforcing this, especially when in an environment where many eWallets are requesting values of $R[j]$ could result in the transponder rapidly running out of values. In this case, the transponder needs to be 'recharged' with new hash values based on a new key k_{new}. There are a number of ways to reduce the danger of 'running out' of responses — the transponder could be given a value which the eWallet must use to precede any request for a hash value, but this would do nothing to stop a denial-of-service attack.

Race conditions might develop if several eWallets are up to the same $R[j+1]$ value, and are all requesting authorisation. This might result in the other eWallets accidentally and repeatedly using up the hash values stored in foreign transponders.

Assuming that the hash function being used is SHA-1, then if the transponder has an array of size M, then the amount of data that needs to be transferred during setup is $160 \cdot M$ bits. During a transaction, only $\log_2 M + 160$ bits need to be transferred.

Finally, performing a hash in hardware is made slightly simpler because the data that is being hashed is known to be less than 448 bits. Thus SHA-1 will treat this as a single 'block' of data. The hardware will not have to cope with producing hashes for data of arbitrary length.

3.5. SYSTEM 3: A TRANSPONDER WHICH CAN PERFORM $\mathcal{H}$

In this model, we give the transponder slightly more computational power by giving it the capacity to perform $\mathcal{H}$. We use the *secret prefix* technique developed by the Internet Security and Privacy Working Group (SPWG) for use in the Simple Network Management Protocol (Galvin et al., 1991).

Session initialisation.

1 The eWallet is brought within range of the user's transponder (and out of range of any other transponders) or else physically connected. The user enters their PIN into the eWallet to authorise transfer of authorisation responsibilities from themselves to the transponder.

2 The eWallet generates a random key k, which it sends to the transponder.

Operation. The eWallet wants authorisation to make a transaction.

1 The eWallet generates a random number r, transmits it to the transponder and also generates $\mathcal{H}(k, r)$.

2 The transponder computes $\mathcal{H}(k, r)$ and transmits this back to the host.

3 The eWallet checks that the received value of the hash agrees with the hash that it calculated itself, and proceeds with the transaction if this is true.

3.6. EFFICIENCY AND SECURITY OF AN ACTIVE TRANSPONDER

Unlike the passive system, this system will never 'run out' of replies, though the complexity of performing the hash will probably require a 4- or 8-bit microcontroller and an on-board power supply. Efficient and low-power wireless communication might be accomplished with a COTS technology such as Bluetooth (Bluetooth, 1999).

When using this *secret prefix* system, then so long as the key is padded to the block size of the hash function being used, then the first block of the hash can be precomputed and the chaining variables remembered so that you have a kind of initialisation vector (Tsudik, 1992).

Since the key needs to be transmitted to the transponder, this must be performed in a secure fashion. A simple way to enforce this is to

require that the transponder be physically connected to the eWallet in order to transfer the key.

The amount of data that needs to be transferred is quite minimal at $\log_2 k + 160$ bits.

3.7. SYSTEM 4: A TRANSPONDER THAT CAN PERFORM MODULAR ARITHMETIC

Modular arithmetic allows a third tier of security. Once we have this kind of computational power, cryptographic authentication protocols such as Schnorr, Fiat-Shamir or Guillou-Quisquater become available for use. We chose to use Schnorr in our example because of its low communications overhead and because it keeps computation to a minimum for the transponder.

Variations on using Schnorr identification. Traditionally, Schnorr identification requires a trusted authority (TA). This is so that Alice might identify herself to Bob, even if Bob has never met Alice before (but Bob trusts the TA). The corresponding analogy with our system is that the eWallet would never have gone through the setup process with a particular transponder, so a question is: who takes on the role of the TA?

Since the eWallet and transponder are both owned and trusted by the user, then the eWallet might as well be its own TA. In this case, the eWallet can also choose (and subsequently 'forget') the secret that the transponder will use to identify itself as well as precomputing the accompanying modular exponentiations (which is what is done in the system below). In this case, the only arithmetic that the transponder must perform in response to an identification request is a modular addition and a modular multiplication. This is a very modest amount of computation.

In the system below, we use the eWallet to precompute the modular exponentiations, because we assume that the eWallet to have more computational power available. However, this once again leaves the transponder with the unattractive prospect of 'running out' of replies. If the transponder has the silicon and power to perform modular exponentiation, then it can use its 'idle time' to precompute its own values.

A potential weakness with *parasitic authentication* in general is that if an attacker who gets hold of the wallet can guess the password used to delegate authority to the transponder, then they can replace the original transponder with one of their own. If the TA was really a trusted third

party (such as the user's bank) as Schnorr originally intended, then this danger would be reduced, since an attacker would have to get hold of a transponder that was *signed* by the user's bank.

Again the downside is that transponder needs to be significantly more complex: it must be able to generate random numbers, as well as perform modular exponentiations. There is also added inconvenience to the user if they lose their transponder, since a replacement must be signed by the bank.

Session initialisation.

1 The TA generates and/or specifies two large primes p and q such that $q|(p-1)$; $\alpha \in \mathbb{Z}_p^*$ with order q; a security parameter $t \approx 40$; a secure signature scheme with a secret signing algorithm sig_{TA} and a public verification algorithm ver_{TA} and a secure hash algorithm[3].

2 The transponder sends its ID number $a; 0 \leq a \leq q-1$ to the TA.

3 The TA computes $v = \alpha^{-a} \bmod p$, $s = sig_{\mathrm{TA}}(v)$, chooses several random values $K = \{k_1, k_2, \ldots, k_n\}$ where $0 \leq k_i \leq q-1$, and computes $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ such that $\gamma_i = \alpha^{k_i} \bmod p$. The TA gives s, K and Γ to the transponder. The eWallet and transponder are now ready for use.

Operation. The eWallet wants authorisation to make a transaction.

1 The eWallet broadcasts a query to the transponder for identification information.

2 The transponder sends the eWallet s and γ_i (where $i = 1$ following the *setup* process).

3 The eWallet verifies $ver_{\mathrm{TA}}(s) =$ true and that the s is the correct s used to initiate the session.

4 The eWallet chooses a random number r, $1 \leq r \leq 2^t$ and transmits it to the transponder.

5 The transponder computes $y = k_i + ar \bmod q$ and sends y to the eWallet (and increments i).

6 The eWallet verifies that $\gamma \equiv \alpha^y v^r \bmod p$. If the congruence holds, the eWallet goes ahead with the transaction.

Note that we have used the TA to precompute values of γ_i so that the transponder does not have to perform a modular exponentiation.

Efficiency and security of transponders that can perform modular arithmetic. The steps are very similar to issuing a certificate in the Schnorr identification protocol, but we don't require that the transponder keep its ID number a secret from the trusted authority. The TA could theoretically impersonate the transponder, but this makes no sense in our context since the user trusts both devices. An attacker might extract the secret a from the eWallet, but if they can do that, then the might as well just hack the eWallet so that it ignores invalid responses from the transponder.

An eavesdropper on the communications between the transponder and eWallet gains nothing that will help them impersonate the transponder.

This more complicated transponder — needing a power supply, wireless communication capabilities, memory and some computational capabilities — will be more expensive (and less disposable) than the passive device. However, as discussed above, the Schnorr protocol leaves us with several implementation options.

The communication overhead is larger than the previous systems. The values that need to be transmitted[4] during the operation phase are γ, r and y. If we use DSS for the signature, as suggested, then $\gamma + r + y = 512 + 40 + 160 = 712$ bits.

3.8. DISCUSSION

All of these protocols are vulnerable in the sense that, if an attacker were to reverse-engineer the eWallet, she could force the eWallet to ignore the result of the identification check and make the transaction anyway.

The merchant could be included in the protocol. Then, we could share a secret between the eWallet and the transponder such that the merchant could verify the signature. However, this assumes that the merchant is honest, which is not a wise assumption to make. Even if the merchant were trusted, then this type of protocol would still involve public-key computations, which are presently computationally expensive. Even our final protocol only needed to compute a modular addition and multiplication — the rest could be precomputed.

In order to keep the transponder cheap, low power and small, one of our design criteria was to avoid the use of expensive cryptographic operations (remembering that computing a modular exponentiation is $O(n^2)$ in hardware (Shand and Vuillemin, 1993)), thus, we have not offered a protocol involving the merchant.

It is worth remembering that reverse-engineering the eWallet is a technically demanding task which far exceeds in complexity of the 'observe

password then swipe the eWallet' attack. The eWallet is still vulnerable, but it is much more secure than if passwords were used, and, perhaps even more importantly, it is extremely user-friendly with its authentication.

4. CONCLUSION

Although the eWallet holds much promise in the marketplace, key issues such as useability have hindered its adoption. We have proposed a device and specified some novel uses of existing protocols which should address some crucial useability and security issues related to using a portable eWallet.

Notes

1. We extracted the four-letter words from our `/usr/dict/words` file, encoded them as PINs according to the keypad on an ATM near the department of one of the authors, and eliminated duplicates. From 25486 words, we extracted 2207 four-letter words which reduced to 1411 different PIN numbers. So our keyspace was reduced to about 15% of its original size. Due to Digital UNIX being a polite OS, we missed some favourite choices of four letter words, but most people's vocabularies are significantly smaller than `/usr/dict/words` anyway, so the *actual utilised* keyspace for four letter words is almost certainly smaller than our estimate.

2. Although we are trying to avoid use of passwords, this particular password will have to be entered much less frequently than if we used passwords right the way through our system.

3. DSS can be used for the signature algorithm and SHA-1 for the hash.

4. If, as suggested in section 3.7, a trusted 3rd party is used as the TA, a certificate will also have to be transferred.

References

Anderson, R. (1993). Why cryptosystems fail. In *ACM 1st Conference — Computer and Communications Security*.

Bluetooth (1999). *Bluetooth Specification V1.0 A*. Available at `http://www.bluetooth.com`.

et al, J.-P. B. (1994). The esprit project cafe. In *ESORICS '94 Proceedings*, pages 217–230. Springer-Verlag.

Fiat, A. and Shamir, A. (1987). How to prove yourself: Practical solutions to identificationand signature problems. In Odlyzko, A., editor, *Advances in Cryptology, Proc. of Crypto '86(Lecture Notes in Computer Science 263)*, pages 186–194. Springer-Verlag. SantaBarbara, California, U. S. A., August 11–15.

Galvin, J., McCloghrie, K., and Davin, J. (1991). Secure management of SNMP networks. In *Proceedings of IFIP Integrated Network Management Symposium*.

Goldreich, O., Goldwasser, S., and Micali, S. (1986). How to construct random functions. *Journal of ACM*, 33(4):792–807.

No.	Computational complexity	Security	Efficiency	Cost
1	nil	Provides casual security, approximately equivalent to passwords, but will not thwart a determined attacker	No computation, very minimal communication	$<$\$1 U.S.
2	store, retrieve and transmit data	Secure if used correctly, but vulnerable to denial-of-service attacks. May also 'run-out' of authentication replies at inconvenient times if not recharged	Computation: only store and retrieve required. Communication: only $\log_2 M + 160$ bits need to be transferred.	$\approx$\$10
3	Must be able to compute $\mathcal{H}$	Secure. Will never 'run out' of authentication data.	Computation: only a single block of a hash function need be computed. Communication: $\log_2 k + 160$ bits must be transferred.	Needs memory, power supply and microcontroller
4	Modular arithmetic	Secure and offers flexibility with how the security is implemented.	Computation: Needs modular exponentiation (slow), but this can be precomputed elsewhere. The computation the transponder needs on the fly is quite modest. Communication: 712 bits must be transferred.	As in 3, but may need more powerful microcontroller

Table 1 Summary of protocols used between the eWallet and transponder

Guillou, L. and Quisquater, J.-J. (1988). A practical zero-knowledge protocol fitted tosecurity microprocessor minimizing bothtransmission and memory. In Günther, C. G., editor, *Advances in Cryptology, Proc. of Eurocrypt '88(Lecture Notes in Computer Science 330)*, pages 123–128. Springer-Verlag. Davos, Switzerland.

Kahn, D. (1967). *The Codebreakers*. MacMillan Publishing Co., New York.

Klein, D. V. (1990). 'foiling the cracker': A survery of, and implications to, password security. In *Proceedings of the USENIX UNIX Security Workshop*, pages 5–14.

nikkei (1998). *NIKKEI Digital Money Systems*.

nikkei (1999). *NIKKEI Digital Money Systems*.

Shand, M. and Vuillemin, J. (1993). Fast implementations of RSA cryptography. In *Proceedings 11th Symposium on Computer Arithmetic*, pages 252–259.

Tsudik, G. (1992). Message authentication with one-way hash functions. In *IEEE Infocom*.