Parasitic Authentication To Protect Your E-Wallet

How can an e-wallet, a handheld computer that consolidates a user's personal items, stores vital information, and facilitates financial transactions, guarantee security without being cumbersome? Parasitic Authentication offers handheld computers security without reducing convenience.

Tim Ebringer
Peter Thorne
University of Melbourne

Yuliang Zheng
Monash University

The electronic wallet (e-wallet) has received much attention lately. It promises to consolidate many of the personal items carried around by the modern individual: wallet, phone, pager, diary, and keys. In fact, Nokia's *9001 Communicator* already combines the phone, pager, and diary into one unit.

The question arises, however, of how to provide user authentication. Traditional protection mechanisms require users to enter a PIN or password every time they wish to perform a transaction. More sophisticated techniques include using a biometric device, such as a fingerprint scanner, which is integrated into the e-wallet. Both of these options, have disadvantages. Usability problems due to authentication are a significant barrier to the adoption of e-wallets.

In this article, we present some novel uses of existing protocols whereby a concealable, wireless, and portable device can temporarily act as an authentication proxy for the user. The e-wallet then becomes a parasite—feeding off the small device for required authentication and identification information.

E-PURSES AND AN E-ID?

The traditional wallet provides four main functions for the user: It holds identification information such as a driver's license, facilitates two distinct payment systems (cash and credit), and acts as a repository for temporary tokens such as bus tickets.

There is little doubt that cunningly engineered cryptographic protocols can efficiently perform these wallet-type functions, but can they add value? Indeed, e-wallet and smart-card developers face a daunting hurdle in convincing consumers to adopt an electronic purse.

From the consumer's point of view, the credit card is an extremely attractive payment mechanism. In most legal jurisdictions, the onus is on the merchant to prove that a disputed transaction was made, thus placing very little risk with the consumer. Cash is a highly reliable payment system that has worked for centuries. How can the e-wallet compete?

The verification problem

Users expect some mechanism to prevent a thief from using their e-wallet.[1] This has proven quite difficult to achieve. The most obvious solution requires the user to perform some kind of *identification protocol* with the e-wallet before each transaction. Traditionally, the user must divulge some secret such as a password or PIN. More recent innovations have used biometrics.

Passwords and PINs

Neither passwords nor PINs are an ideal solution: Not only are they a weak authentication measure, they are also frequently misused.[2,3] For example, banks sometimes tell their users to use a word as their PIN since an ATM keypad has letters associated with the numbers. We extracted the four-letter words from our `/usr/dict/words` file, encoded them as PINs according to the keypad on a commercial ATM, and eliminated duplicates. From 25,486 words, we extracted 2,207 four-letter words, which reduced to 1,411 different PINs. So our keyspace ended up being about 15 percent of its original size.

0018-9162/00/$10.00 © 2000 IEEE

This kind of security fault, where users unwittingly weaken a theoretically secure system, flows directly from poor usability aspects of PIN authentication systems. Formal security standards such as ITSEC assume protocols are rigorously followed. For example, a protocol for choosing a PIN p might specify that $p \in_R Z$: $0000 \le p \le 9999$, whereas humans might not be quite so random in their PIN selections. Ross Anderson argued that the failure of many real-world cryptosystems stems from this kind of flaw.[4]

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 passwords is that entering them into a small, portable device is awkward. On a dark, 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 e-wallet spluttering in a puddle, the ferry disappearing into the gloam. This failure has nothing to do with the correctness of the protocol, and is primarily an engineering problem.

Biometrics

A more effective method might be to integrate a biometric authentication device, such as a fingerprint scanner, into the e-wallet. The e-wallet would not perform a transaction unless a valid fingerprint is scanned.

Such a scheme raises two security concerns. First, a sophisticated and determined attacker could bypass the fingerprint scanner. Once perfected, knowledge of the bypass technique would spread to hacker communities with dazzling speed. As an example, the Sony Playstation was initially engineered so that games could not be duplicated, yet services are widely available to modify the original hardware so that it accepts pirated games. Also, unlike passwords, it is impossible to use a hash of a biometric for verification purposes because a scanned biometric image will always be slightly different. Second, and perhaps more importantly, users feel uncomfortable storing a biometric—something that makes them unique—on a computer, especially on a portable computer. Many users feel that a thief could steal their "identity" and masquerade as the user—theoretically forever—since there is no way to revoke a biometric.

PARASITIC AUTHENTICATION

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 communicate with the secondary device and 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 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 computer users: *sessional authentication.*

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.

Sessional authentication

In a database, transactions are atomic: Either all the operations in a transaction are reflected properly in the database, or none are. In such situations, it is both feasible and necessary to authenticate every transaction. Conversely, when a user enters data at a computer terminal, every keystroke constitutes a transaction, yet authenticating every keystroke would be absurd—and recursively impossible for passwords. The system assumes that the user who logs in is the same user who presses the keys until she chooses to log out. The authentication lasts for a session, and the user decides the length of the session.

We use this approach as a basis for a more usable system for authorizing e-wallet-based transactions. The user initiates a session with the e-wallet and a secondary device, authorizing the e-wallet to complete transactions during the session. The proximity of the secondary device to the e-wallet sustains the session.

A session could end in several ways. The e-wallet could end the session itself after a certain time period has elapsed. Alternatively, the e-wallet 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 by using a panic button. But most important, the session ends when the e-wallet and secondary device cannot communicate because of physical separation beyond their communication range. Choosing more conservative ways to end the session better protects the user in the event that *both* the e-wallet and the secondary device are lost simultaneously but decreases usability since the user must begin a new session to use the e-wallet.

Expected economic and security benefits

Such a system offers two major benefits: usability and security.

Usability. The user no longer has to enter a password or PIN or perform any other kind of interactive security procedure to authorize each transaction. Once the user initiates the session, the secondary device handles the interactive security procedures on the user's behalf.

Security. The user does not have to store a biometric yet is still protected against loss of the e-wallet. To compromise security, the user must lose *both* the e-wallet and the secondary device simultaneously. If the chances of losing your e-wallet are p, given that the secondary device is small and concealable and provided sensible precautions are taken, the chances of losing the secondary device should be q, where $q < p$. We argue that $q < p$ because the secondary device need not be handled or produced 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 or necklace—that is not removed for the entire day. If these precautions are taken, then the chance of losing both the e-wallet and the secondary device simultaneously should be considerably less.

The idea of parasitic authentication is partly an evolution of ordinary wireless authentication.

Secondary device characteristics

The problem is essentially one of providing a convenient, fast, mobile and secure method of performing *entity identification*. Entity identification techniques can be divided into three main categories, depending on whether the security is based on *something known, something possessed*, or *something inherent*.[5] Identification measures based on *something known*, such as passwords or PINs, are difficult for the user to remember and cumbersome to enter. Measures based on *something inherent*, such as fingerprints, are intrusive and impossible to revoke. This leaves us with protocols based on *something possessed*. In effect, we are *exchanging* something known for something possessed.

The e-wallet becomes a parasite to the transponder—feeding off it for identification. In fact, the authentication device can now be the host to many different parasites. We envision it as having the following physical characteristics.

Miniature. The authentication device must be small enough to be unobtrusive and hidden somewhere in the user's apparel.

Self-powered. The authentication device must carry enough power, or be able to draw power from an energizing electric field, so that it can function for extended periods of time away from a power source. In recent years, battery technology has advanced remarkably, as evidenced by the extended battery life of tiny mobile phones.

Disposable. Loss of the authentication device should not be a major catastrophe. At worst, simultaneous loss of *both* the e-wallet and the authentication device would mean a small window of time in which a user's e-wallet was vulnerable to abuse. No information that an attacker might find extremely useful would be on the authentication device. Clearly, it would also be highly advantageous for the authentication device to be inexpensive.

Wireless. To keep the authentication device hidden, and not inconvenience the user, it must communicate wirelessly.

We have not yet described the authentication device's computational abilities because there are certain engineering trade-offs in its design. As the device's computational power 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. Realistically, 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.

Limitations

Attackers can subvert parasitic authentication by using reverse engineering and modifying the e-wallet to ignore the results of the authentication check. There are ways to resist this, such as splitting crucial information between the e-wallet and the transponder (and even the merchant), but our goals were primarily to ensure low power and cost in the transponder, and, above all, useability. All the protocols we suggest are *at least* as secure as passwords. We avoided public-key operations because they are computationally expensive, would raise the cost of the transponder, and would delay its response time.

Alternative techniques for heightening security might include marking the money on the e-wallet so that it is only valid for the session for which it was initialized. Although this does not necessarily prevent the reverse-engineering attack, it at least puts an upper bound on the window of time in which the attack must be performed. This, however, is an aspect of the *payment* protocol, which takes advantage of the sessional nature of parasitic authentication. Because we are only offering a user-friendly framework for approving transactions and saying nothing about the nature of the transactions themselves, we will not speculate about what additional steps the payment protocol might use to increase security.

Deja vu?

The idea of parasitic authentication is partly an evolution of ordinary wireless authentication, which has been part of access-control mechanisms for some time now. There are significant differences, however; We designed parasitic authentication with low power and portability in mind; in most traditional wireless authentication mechanisms, the device doing the authentication is immobile. The service that the parasitic authentication system provides also differs from that expected from access-control systems, whose primary responsibility is to prevent unauthorized access. A secondary consideration is to provide the service level appropriate to the authentication level of the individual. For parasitic authentication, the normal mode of operation is not to challenge the access right, but to confirm authentication in a symbiotic relationship.

Table 1. Summary of protocols used between the e-wallet and transponder, where *H* represents a polynomial time-computable hash function; $\log_2 R[j]$ is the number of bits in the request number; and r is the random number that the e-wallet generates as a challenge.

Number	Computational complexity	Security	Efficiency	Cost
1	None	Provides casual security, approximately equivalent to passwords, but will not thwart a determined attacker	No computation, minimal communication	Less than $1
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 $R[j] + 160$ bits need to be transferred.	About $10
3	Must be able to compute *H*	Secure. Will never run out of authentication data.	Computation: only a single block of a hash function needs to be computed. Communication: $log_2\, r + 160$ 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 item 3, but may need more powerful microcontroller

SYSTEMS

Several systems are suitable for transponders, each offering a different level of computational power. The first is a completely passive transponder that can only return a serial number. The second requires very little computational ability but can store values, and subsequently retrieve and transmit them. The third is a host that, in addition to the capabilities described above, can compute a one-way hash function. The fourth is a transponder that can perform modular arithmetic. Table 1 compares and summarizes all four systems.

Polyrandom collections

Some of the systems we propose both assume and require the existence of one-way functions. Polyrandom collections from long bit strings to short bit strings constitute very good hash functions.[6] But to ease interoperability and implementation, we suggest using the SHA-1 hash function, which maps a string of arbitrary length to a 160-bit string in a one-way manner $\{0,1\}^k \rightarrow \{0,1\}^{160}$.

We assume the existence of a one-way, polynomial time-computable hash function, which we refer to as H.

Schnorr identification protocol

Our final system uses an authentication protocol that allows, in real time, one party (the *verifier)* to know that the identity of another (the *claimant*) is as declared. We have selected Schnorr because of the small computation and communication overhead it introduces, but another authentication scheme such as Fiat-Shamir[7] or Guillou-Quisquater[8] could easily be used, depending on exact requirements.

System 1: A passive, RFID-type transponder

Here we propose using a multibit Radio Frequency Identification (RFID) transponder as the host. Although we are not using anything particularly sophisticated cryptographically, this system offers the equivalent security of passwords, should be inexpensive, and should effectively protect users from their wallets being abused if they misplace them.

If we build reader capabilities into the e-wallet, we can configure it to use a particular transponder. This would involve telling the e-wallet to accept a new transponder—perhaps by entering a password. Although we try to avoid using passwords, users would need to enter this particular password far less frequently than if we used passwords entirely through our system. The e-wallet would subsequently broadcast a query to which the nearby transponder would respond with its ID string. For subsequent wallet use, the e-wallet would broadcast a query and wait for the expected ID string to be returned.

Setup. An individual purchases a transponder having the random and unique ID string I, and places it near, or plugs it into, the e-wallet. The individual instructs the e-wallet to accept a new transponder by entering a password. The e-wallet broadcasts a query and receives the string I from the nearby transponder, which it commits to memory. The individual drops the transponder into a shoe and takes the e-wallet shopping.

Operation. The user initiates a secure electronic transaction, for example, with the e-wallet. Before sending order and payment information to the merchant, the e-wallet broadcasts a query to the transponder and verifies that it returns the string I. The SET protocol resumes, and order and payment information go to the merchant, as illustrated in Figure 1.

Now let's say the user accidentally leaves his e-wallet in the shop. The merchant unfortunately takes the e-wallet and tries to crank a few extra SET transactions through it. The e-wallet, before it sends order and payment information, first broadcasts a query. Because the transponder is still in the user's pocket,

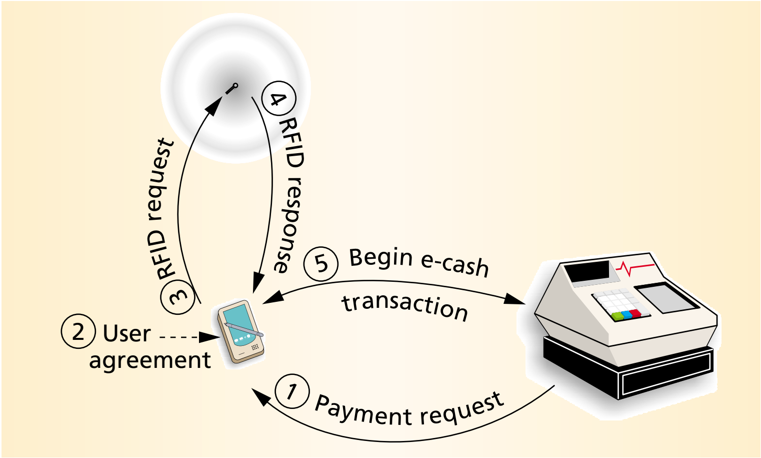

Figure 1. Intended flow of usage and authentication.

the wallet cannot get the return value *I* and therefore aborts the transaction.

Efficiency and security. This passive transponder is ineffective against a more sophisticated adversary. If the merchant from the previous example possessed her own receiver, it could have also picked up the return value from the transponder and spoofed the return signal.

This system provides only casual protection against an attacker, but it is at least as secure as password protection. If a password were entered, the merchant might be able to observe what was entered, either by carefully watching the user or using a hidden camera.

Another way around the transponder check is to hack the software within the e-wallet 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. The difficulty in making this modification on a handheld computer depends to a large extent on the device implementation. Thus, although subverting the transponder in this manner is possible, whether it would become common practice is unknown.

This system offers users the equivalent of password protection, while significantly increasing the usability and efficiency of entering authentication information into their e-wallets. 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 $1.

System 2: A transponder with memory

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

Setup. The user brings the e-wallet within range of the user's transponder and out of range of any other transponders, or else, more securely, plugs the transponder into the e-wallet. The user then enters her PIN into the e-wallet to authorize transfer of authorization responsibilities from herself to the transponder. The e-wallet generates a secret, random key k. The e-wallet generates $H(k, 0)$, $H(k, 1)$, …, $H(k, 9999)$ and transmits these values to the transponder, where they are stored in the array.

Operation. The e-wallet wants authorization to make a transaction. The e-wallet requests $R[j]$ from the transponder. It checks that $R[j]$ equals $H(k, j)$. The *next* time it wants authorization, the e-wallet requests $R[j + 1]$.

Efficiency and security. This system, if implemented carefully, is computationally secure. However, the transponder clearly must not transmit any values more than once. Enforcing this policy, especially in an environment where many e-wallets are requesting values of $R[j]$, could result in the transponder rapidly running out of values. In this case, the transponder needs to recharge with new hash values based on a new key k_{new}. There are several ways to reduce the danger of running out of responses. For example, the e-wallet could initially give the transponder a unique identification string that the e-wallet must transmit before requesting a hash value in order for the transponder to respond, but this would do nothing to stop a denial-of-service attack.

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

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

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

System 3: A transponder that can compute a one-way hash function

In this model, we give the transponder slightly more computational power by letting it perform the hash algorithm 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.[9]

Setup. The user brings the e-wallet within range of her own transponder and out of range of any other transponders or else temporarily connects the e-wallet to the transponder. The user enters her password into the e-wallet to authorize transfer of authorization responsibilities from herself to the transponder. The e-wallet generates a random key k, which it sends to the transponder.

Operation. The e-wallet wants authorization to make a transaction. The e-wallet generates a random number r, transmits it to the transponder, and also generates $H(k, r)$. The transponder computes $H(k, r)$ and transmits this back to the host. The e-wallet 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.

Efficiency and security. Unlike the passive system, this system never runs out of replies, although the complexity of performing the hash probably requires a 4- or 8-bit microcontroller and an on-board power supply. A commercial off the shelf technology such as Bluetooth[10] can accomplish efficient and low-power wireless communication.

When using this secret prefix system, as long as the key is padded to the block size of the hash function being used, the first block of the hash can be precomputed and the chaining variables remembered so that you have a kind of initialization vector.[11]

The **e-wallet** must securely transmit the key to the transponder. A simple way to enforce this security is to require that the transponder be physically connected to the e-wallet to transfer the key.

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

System 4: A transponder that can perform modular arithmetic

Modular arithmetic provides a third tier of security. With this kind of computational power, we can use cryptographic identification protocols such as Schnorr, Fiat-Shamir, or Guillou-Quisquater. We chose Schnorr for our example because of its low communications overhead and because it minimizes computation for the transponder.

Traditionally, Schnorr identification requires a trusted authority (TA). This is so that Alice can identify herself to Bob, even if Bob has never met Alice before, because Bob trusts the TA. With our system, the e-wallet would not have gone through the setup process with a particular transponder, so who takes on the role of the TA?

Since the user owns, trusts, and uses the e-wallet and transponder, the e-wallet might as well function as its own TA. In this case, the e-wallet can choose and subsequently forget the secret that the transponder will use to identify itself as well as precomputing the accompanying modular exponentiations. 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.

We use the e-wallet to precompute the modular exponentiations because we assume the e-wallet has 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, it can use its idle time to precompute its own values.

For parasitic authentication, the normal mode of operation is not to challenge the access right, but to confirm authentication in a symbiotic relationship.

A potential weakness with parasitic authentication in general is that if attackers who get hold of the wallet can guess the password used to delegate authority to the transponder, they can replace the original transponder with one of their own. If the TA is truly a trusted third party (such as the user's bank), as Schnorr originally intended, then this danger would be reduced because the replacement transponder used by the attacker would have to be signed by the user's bank.

Again, the downside is that the transponder needs to be significantly more complex so that it can generate random numbers, as well as perform modular exponentiations. There is also an added inconvenience if the user loses his transponder because the bank must sign the replacement.

Setup. The TA generates or specifies two large primes p and q such that $q|\,(p-1)$. The TA also generates $\alpha \in Z_\mathrm{p}$ with order q; a security parameter $t <$ 40; a secure signature scheme with a secret signing algorithm sig_TA; a public verification algorithm ver_{TA}; and a secure hash algorithm. The Digital Signature Standard can be used for the signature algorithm and SHA-1 for the hash. The transponder sends its ID number a (where $0 \leq a \leq q-1$) to the TA. The TA computes $v = \alpha^{-a} \bmod p$ and $s = sig_{TA}(v)$; 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 $\gamma_i = \alpha^{ki}$ mod p. The TA gives s, K, and Γ to the transponder. The e-wallet and transponder are now ready for use.

Operation. The e-wallet wants authorization to make a transaction. It broadcasts a query to the transponder for identification information. The transponder sends the e-wallet s and γ_i (where $i = 1$ following the *setup* process). The e-wallet verifies $ver_{TA}(s) = true$. The e-wallet chooses a random number r, $1 \leq r \leq 2^t$ and transmits it to the transponder. The transponder computes $y = k_i + ar \bmod q$ and sends y to the e-wallet, incrementing i. The e-wallet verifies that $\gamma \equiv \alpha^y v^r$ mod p. If the congruence holds, the e-wallet goes ahead with the transaction.

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. The steps are 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 because the user trusts both devices. Attackers might extract the secret a from the e-wallet, but if they can do that, they might as well just hack the e-wallet so that it ignores invalid responses from the transponder. Eavesdroppers on the communications between the transponder and the e-wallet gain 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 compact than the passive device. However, the Schnorr protocol leaves us with several implementation options.

The communication overhead is larger than in the previous systems. The values that need to be transmitted during the operation phase are γ, r, and y.

Although the e-wallet holds much promise in the marketplace, key issues such as usability have hindered its adoption. The devices we have proposed, along with some novel uses of existing protocols, address some crucial usability and security issues related to using a portable e-wallet. Additional research focuses on sharing the secret stored by the transponder among multiple wireless cooperative devices using secret-sharing schemes, partial digital signatures, and proactive update of the shares. This allows us to build a degree of loss-tolerance into the system while retaining user privacy. This ongoing work is the subject of a patent. ✷

Acknowledgments

This work was partly sponsored by a grant from the Australian Research Council.

References

1. J.-P. Boly et al., "The ESPRIT Project Café," *Proc. European Symposium on Research in Computer Security '94, Springer-Verlag*, New York, 1994, pp. 217-230.
2. D. Kahn, *The Codebreakers*, MacMillan Publishing Co., New York, 1967.
3. D.V. Klein, "Foiling the Cracker: A Survey of, and Implications to, Password Security," *Proc. Security Workshop*, Assoc, Berkeley, Calif., 1990, pp. 5-14.
4. R. Anderson, "Why Cryptosystems Fail," *Proc. ACM 1st Conf. Computer and Comm. Security*, ACM Press, New York, 1993.
5. A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, Boca Raton, FL, 1997.
6. O. Goldreich, S. Goldwasser, and S. Micali, "How to Construct Random Functions," *J. ACM*, Oct. 1986, pp. 792-807.
7. A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems," *Proc. Crypto 86, Lecture Notes in Computer Science 263*, *Advances in Cryptology,* Springer-Verlag, New York, 1987, pp. 186-194.
8. L.C. Guillou and J.-J. Quisquater, "A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory," *Proc. Eurocrypt 88*, *Lecture Notes in Computer Science 330*, *Advances in Cryptology*, Springer-Verlag, New York, 1988, pp. 123-128.
9. J. Galvin, K. McCloghrie, and J. Davin, "Secure Management of SNMP Networks," *Proc. IFIP Integrated Network Management Symp*., Elsevier Science Pub., Amsterdam, 1991.
10. *Bluetooth Specification V1.0 A*, 1999; http://www.bluetooth.com.
11. G. Tsudik, "Message Authentication with One-Way Hash Functions," *Proc. Infocom 92, 11th Conf. Computer Communications*, IEEE Press, Los Alamitos, Calif., 1992, pp. 2055-2059.

***Tim Ebringer** is a PhD student at the University of Melbourne. His research interests include cryptographic algorithms and protocols, engineering of practical authentication, and security systems. Ebringer received a BE in computer engineering from the University of Melbourne. Contact him at tde@cs.mu. oz.au.*

***Peter Thorne** is a reader in computer science at the University of Melbourne and a former head of the Department of Computer Science and Software Engineering. His research interests include computer forensics, computer security and the early history of computing. Thorne has a PhD in computation from the University of Melbourne. Contact him at pgt@cs.mu.oz.au.*

***Yuliang Zheng** is a reader and founding director of the Laboratory for Information and Network Security at Monash University. He cofounded the PKC annual international conference series dedicated to the theory and practice in public key cryptography. His research focuses on efficient data security techniques and their applications in network computing and e-commerce. Zheng received a PhD in electronic and electrical engineering from Yokohama National University, Japan. He is a member of ACM and IACR and a senior member of the IEEE. Contact him at Yuliang.Zheng@infotech.monash.edu.au.*