The following paper was originally published in the
Proceedings of the 3rd USENIX Workshop on Electronic Commerce
Boston, Massachusetts, August 31–September 3, 1998

A Resilient Access Control Scheme for Secure Electronic Transactions

Jong-Hyeon Lee
University of Cambridge

For more information about USENIX Association contact:

1. Phone: 1 510 528-8649
2. FAX: 1 510 548-5738
3. Email: office@usenix.org
4. WWW URL: http://www.usenix.org/

A Resilient Access Control Scheme for Secure Electronic Transactions

Jong-Hyeon Lee *
Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge CB2 3QG, England
Jong-Hyeon.Lee@cl.cam.ac.uk
http://www.cl.cam.ac.uk/~jhl21

Abstract

There have been many studies of the management of personal secrets such as PIN codes, passwords, etc., in access control mechanisms. The leakage of personal secrets is one of the most significant problems in access control. To reduce such risks, we suggest a way of authenticating customers without transferring explicit customer secrets. Furthermore, we give a secure on-line transaction scheme based on our access control mechanism.

Needham gave an example of Personal Identification Number (PIN) management for banking systems [Nee97] that presented a way to control PIN codes. It inspired us to develop an access control model for electronic transactions which enforces a strict role definition for personal secret generation and maintenance. We extend it to a payment model. Our scheme provides enhanced privacy for customers, non-repudiation of origin for the customer order and payment transactions, and protection from personal secret leakage. Since it does not rely on either public key cryptosystems or auxiliary hardware such as chip cards and readers, its deployment within existing environments could be cost-effective.

*This work is supported in part by the EPSRC, under grant number GR/L95809 on Resilient Security Mechanisms. The views and conclusions in this paper are that of the author alone.

1 Introduction

For user authentication, traditional passwords or PIN codes are still one of the most common methods, although they are notoriously vulnerable to attacks. Even systems with sophisticated security protocols often employ such authentication procedures, since they are easy and familiar to users. For this reason, many studies of personal secret management have been done, like strengthening passwords [ALN97], enhancement of existing password protocols [BM92, BM93], protection for poorly chosen passwords [GLNS93], one-time password schemes [Hal94, Rub96], and so on.

In access control, we have to consider who generates secrets and who maintains them. Some banks generate customers' PIN codes, and send them to customers. Some on-line shops ask customers to generate passwords for their services, and keep the passwords in their database. It is common that personal secrets are known to service providers such as banks or on-line shops, as shared secrets between the service providers and customers. But there is no reason for customers to trust service providers; this is dangerous.

Needham gave a simple PIN code management scheme for banking systems [Nee97], which shows a strict role definition of PIN generation and management. The PIN codes are generated and maintained by customers themselves. This idea can be a basis for building a privacy-enhanced transaction model with strict role definition.

One of the most frequently asked question in se-

curity is protection versus cost. It is a good idea to have alternatives with the same function at different levels of cost. Public key cryptosystems provide high confidentially, authentication, etc., but they require a public key infrastructure. For some applications, it is not a proper solution to adopt a public key cryptosystem because of the cost.

There have been many attempts to use hash functions rather than public key cryptosystems. IBM's KryptoKnight is such an example [MTvHZ92]. From a user's perspective, KryptoKnight provides services and facilities which are very similar to those of Kerberos [NT94] and based on the well-known Needham-Schroeder scheme [NS78], but it uses hash functions and Message Authentication Codes (MACs). We also use only one-way hash functions to build an access control mechanism and a payment mechanism. Since it is based on hash calculations and nonces, the required infrastructure is lighter than that of a public key cryptosystem, to which it can provide a low cost alternative.

We will develop a customer-oriented transaction model in which personal secrets are generated and maintained by customers. It supports customer registration both in a bank and in an online shop, and a transaction procedure between three principals; a customer, a merchant, and a bank.

2 Analysis of Needham's Example

Needham's PIN code management system differs from current ATM systems in the generation and handling of the PIN code. Most ATM systems use encryption to protect customers' PINs. The details vary from one bank to another, but many use variants of a system originally developed by IBM, in which the PIN is derived from the account number by encryption. That is, PINs are generated by banks. This makes it difficult to resolve disputes between banks and their customers [And94].

In Needham's proposal, banks do not know customers' PINs. A PIN code is generated by a customer herself, and is not stored by the bank. From the bank's point of view, this feature liberates it from the responsibility for internal leakage of PINs and assurance of their fair and truly random generation. Banks thus have a defence against an allegation that they negligently permitted the PIN to become known. For customers, they can obtain more privacy by generating and keeping their own PINs for themselves.

Needham's scheme is described under the assumption that a customer has a personal computer and a card writer. Let H be a one-way hash function. To initialise the scheme, the customer writes on the card a random number r and a hash $H(n,b)$ of her name n and date of birth b. She writes $H(n,b)$ and $H(r,pin)$ on a floppy disk, where the pin is chosen by herself. She then takes the floppy to the bank and says "Please connect $H(r,pin)$ to my personal details $H(n,b)$ and my account number 401608 80614874".

When a cash machine accepts her card, it reads $H(n,b)$ and r, works out $H(r,pin)$ where pin is the PIN as entered, and sends the value $H(n,b)$ and $H(r,pin)$ to the bank. Note that the PIN is never sent to the bank and as r is random it cannot be recovered from $H(r,pin)$. The bank looks up $H(r,pin)$ and if it is found, the table yields $H(n,b)$ for checking as well as the account number.

In this example, there are two principals and one intermediate: a customer and a bank are principals and an ATM is an intermediate. A banking transaction is performed between a customer and a bank, and authentication procedures are done by an ATM and a bank. The ATM checks the validity of a card holder by PIN entered, the bank checks a customer's information and her account number by $H(r,pin)$.

The separation of capability between principals is well defined, but there is no such separation between the customer and the ATM. The assumption that the PIN is never sent to the bank even in encrypted form requires the customer trusts the ATM and its operator. The role of the ATM is more important than at present and a false-terminal attack using a corrupted ATM is still possible. However this is inherent in the use of magnetic strip technology.

A replay attack for the pair of hash values $(H(n,b),H(r,pin))$ is possible on a communication line between an ATM and a bank. If an

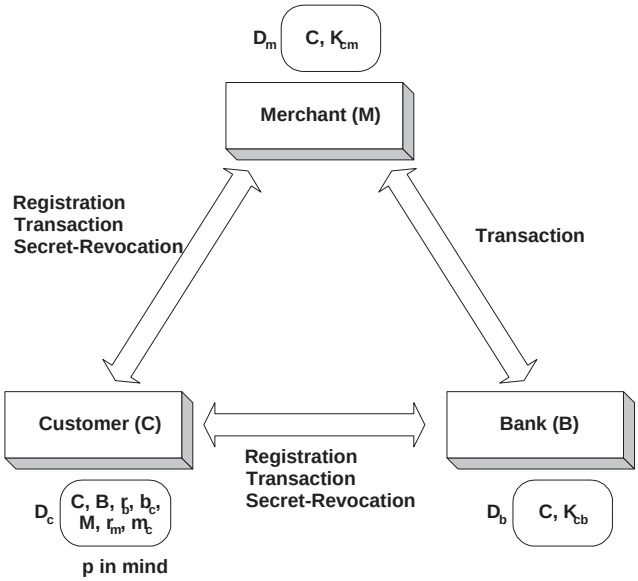

Figure 1: Principals and their interfaces

attacker takes the hash pair from the line, he can replay this pair on the same line and be authorised as a valid customer by a bank. Since the two values are always the same, they can be used at any time by attackers. We could prevent such attacks, of course, by installing a time-varying encryption or authentication mechanism on the line between the bank and ATMs. However, the bank would then join the ATM inside the customer's trust perimeter, and the value of the scheme would be lost. Against replay attacks, we use a nonce for each transaction in our payment scheme.

3 The Resilient On-line Transaction Scheme

We construct an on-line transaction scheme based on the analysis in the previous section. We define three procedures for access control and electronic transactions: a registration procedure, a transaction procedure, and a secret-revocation procedure. The principals in these transactions are a customer C, a merchant M, and a bank B. Merchants include Internet shopping malls, book sellers, on-line travel agents, news providers, etc.

We describe four procedures with brief protocol flows. In the protocol description, the notation $A \Longleftrightarrow B$ stands for protocols between A and B, $A \rightarrow B$ a message from A to B. The symbol $:=$ denotes an assignment. The notation (x, y) is the concatenation of x and y. Figure 1 shows the interfaces between the principals.

3.1 The registration procedure

Let us assume that a customer has an account with a bank. The registration procedure has two steps: a registration with a bank and a registration with a merchant. First, the customer generates a random number r_b, chooses a secret p, and calculates a hash $C := H(n, b)$ of her name n and date of birth b. If the combination of n and b is not sufficient to identify a customer from others, one may add more detailed information. The customer writes r_b, B, and C on a diskette for private use, say D_c, and writes C and $K_{cb} := H(r_b, p)$ on a different diskette for registration, say D_b. Then she sends D_b to the bank B with her account number a, e.g. by registered mail or personal delivery to a branch of the bank.

When the diskette D_b is received, the bank makes a link between the customer's account and K_{cb}, and sends an acknowledgement with a random b_c back to the customer. The customer stores b_c on her private diskette D_c. This is the registration procedure with the bank.

The registration procedure with the merchant is as follows: The customer generates a random number r_m, stores it on her private diskette D_c, and then sends the merchant a diskette D_m containing C and $K_{cm} := H(r_m, b_c, p)$ in the same way as in the above procedure. Then the merchant registers the customer's information on his database, and sends an acknowledgement with a unique merchant secret m_c back to the customer. m_c is a uniquely issued value for each customer, and will be used for verification of the merchant in transactions by a bank. The customer stores m_c on D_c, calculates $K_{mb} := H(m_c, K_{cb})$, and sends (C, m_c) with the merchant name M to the bank. Then the bank constructs K_{mb} with m_c and K_{cb}, and adds K_{mb} and M to the customer's information.

In this procedure, since we have not assumed secure communication paths between the customer and the merchant/bank, we used physical transfer of shared secrets by diskette. If a secure path is available such as SSL/TLS [DA97] or SSH [Ylö96], we can replace diskette transfer by such a path. As a further alternative, the customer can send K_{cm} to the merchant in a physical transaction between her smartcard and the merchant terminal.

Thus the customer can establish a relationship with a merchant either when she is on the merchant's premises, or when she has a secure link to the merchant, or when the bank is on-line. At the same time, the customer could establish a payment limit for the merchant (though we omit the details).

In some cases like closed user group services, the merchant needs to authenticate the customer's eligibility for the service. During the registration procedure, the merchant can request appropriate information such as a membership, age, etc., for the verification and provide classified services in the transaction procedure up to customers' eligibility.

Protocol

$\boxed{C \Longleftrightarrow B}$

C:	generates r_b and p
	calculates $C := H(n, b)$ and $K_{cb} := H(r_b, p)$
	stores C, B and r_b on diskette D_c
	stores C and K_{cb} on diskette D_b
$C \to B$:	D_b, a
B:	stores C and K_{cb} w.r.t. a
	generates a random number b_c
$B \to C$:	b_c
C:	stores b_c on D_c

$\boxed{C \Longleftrightarrow M}$

C:	generates r_m
	stores M and r_m on diskette D_c
	calculates $K_{cm} := H(r_m, b_c, p)$
	stores C and K_{cm} on diskette D_m
$C \to M$:	D_m
M:	generates account for C with K_{cm}
	generates a random number m_c
$M \to C$:	m_c
C:	stores m_c on D_c
	calculates $K_{mb} := H(K_{cb}, m_c)$
$C \to B$:	C, M, m_c
B:	constructs K_{mb} with m_c and K_{cb}
	adds K_{mb} and M to C's information

3.2 The client software distribution

It is necessary for the customer to trust the client software she uses. This client software can be a plug-in for web browsers, a in-house program, etc., and obtained by download, mail delivery, or whatsoever. For such software, the specification should be public, and anyone can provide it. In theory, the customer could implement the software for herself, so she can trust her code and does not need to consider malicious action or compromise during download. In practice, she has to obtain the software from vendors she can trust. Whoever the code provider is, it should be guaranteed that the code does not contain malicious and erroneous code. The way to achieve proper verification is a controversial and interesting topic, and so is trusted distribution. Neither is a main focus of our paper.

3.3 The transaction procedure

The transaction procedure is as follows: When a customer wants to make a transaction, she establishes a connection to the merchant by invoking the client software. This software asks the customer to type the secret p, and establishes

a connection[1] to the merchant. Then the merchant generates a transaction identifier t and sends it back to the customer. This value is a nonce for preventing from replay attacks, and it consists of transaction time, date, and a serial number. After reading the random number r_m from the diskette D_c, the software calculates $H(t, K_{cm})$, and sends $(C, H(t, K_{cm}))$ to the merchant. It should be guaranteed that the client program does not send the customer secret p to the merchant.

The merchant looks up his database with searching key C, and checks validity of $H(t, K_{cm})$ for customer authentication. If it is valid, the merchant calculates $H(t, m_c, K_{cm})$, and sends it to the customer with an acknowledgement. After accepting the acknowledgement, the customer checks the hash value with m_c for merchant authentication.

When the customer makes an order, the merchant requests her to pay, and sends a payment serial number y and payment amount u. The customer generates a nonce n_c, calculates $T_{cmb} := H(t, y, u, K_{mb}, K_{cb})$, and confirms payment to the merchant by sending $(C, B, a, u, y, n_c, T_{cmb})$. The merchant asks the bank B to clear the amount, by sending $(M, C, a, u, t, y, n_c, T_{cmb})$. If T_{cmb} is correct, the bank transfers the money to the merchant with an acknowledgement[2] including $H(n_c, b_c, K_{cb})$. The value $H(n_c, b_c, K_{cb})$ is used by the merchant to generate an acknowledgement ticket for the customer. The bank has to store T_{cmb}'s that it has received for a pre-defined period in order to detect replays. In current credit card systems, the merchant must request that the bank clear transactions within a specified period from the transaction date - such as 21 days in UEPS/VISA COPAC [And92] - otherwise payment will not be made. The storage of T_{cmb} does not require much resource. After receiving the acknowledgement from the bank, the merchant calculates $T_{bmc} := H(t, u, K_{cm}, H(n_c, b_c, K_{cb}))$ and sends this value as an acknowledgement to the customer. The customer checks this value, which can be used as a proof of the purchase order and of a proof of payment in case of dispute.

We may consider a false merchant who wants to cause problems for the real merchant by taking orders and not delivering. Since T_{cmb} contains K_{mb}, he cannot gain monetarily. Furthermore, he has to send T_{cmb} containing K_{cm} back to the customer, so that the customer can check the eligibility of the merchant. Such a false merchant cannot step into the transaction.

Protocol

$C \to M$:	service invocation via client software
M:	generates a transaction id t
$M \to C$:	t
C:	calculates $H(t, K_{cm})$
$C \to M$:	$C, H(t, K_{cm})$
M:	checks $H(t, K_{cm})$
	calculates $H(t, m_c, K_{cm})$
$M \to C$:	$H(t, m_c, K_{cm})$
$C \to M$:	makes an order
$M \to C$:	y, u
C:	generates a nonce n_c
	calculates $T_{cmb} := H(t, y, u, K_{mb}, K_{cb})$
$C \to M$:	$C, B, a, u, y, n_c, T_{cmb}$
$M \to B$:	$M, C, a, u, t, y, n_c, T_{cmb}$
B:	checks T_{cmb}
	calculates $H(n_c, b_c, K_{cb})$
$B \to M$:	$H(n_c, b_c, K_{cb})$
	transfers the money of amount u
M:	calculates $T_{bmc} := H(t, u, K_{cm}, H(n_c, b_c, K_{cb}))$
$M \to C$:	T_{bmc}
C:	checks T_{bmc}

3.4 The secret-revocation procedure

When a customer wants to change her secret, she has to send previously used information with a new secret. We suggest a way of revocation and update of secrets: the customer sends a nonce n_b with C to the bank, then the bank sends back $T_{bc} := H(n_b, K_{cb})$. The customer checks T_{bc}, if it is valid, she requests

[1] The secure connection is not necessarily required, since we have the secrets K_{cb} and K_{mb}. If we have a secure link between the customer and the merchant, the transaction id t also can be used as a shared secret.

[2] If we are to consider adopting this scheme into existing systems, it is during this money transfer stage that we would overlay our mechanisms. There is usually an intermediate credit card company and the merchant's bank between the customer's bank and the merchant. Transaction clearing is done via this path. There are two typical authentication systems for clearing: one uses signature mechanisms between banks and credit card companies, the other uses secret key mechanisms between them. In both cases, such an infrastructure is already used for banks and credit card companies, but merchants and customers are not in their network. If they extend the infrastructure to all customers and merchants, the cost will be enormous and such extension may not be straightforward. If our scheme is adopted within their infrastructure, no additional investment is needed.

to update her shared information by sending $(C, H(n_b, K_{cb}, b_c) \oplus K'_{cb}, H(K'_{cb}, n_b))$, where K'_{cb} is the new secret. If the customer wants to change p as well, she can do it in the same procedure, but does not have to. When $H(n_b, K_{cb}, b_c)$ is correct, the bank changes the customer's information. Similarly, the customer can change her information in a merchant by using $T_{mc} := H(n_m, K_{cm})$ and $(C, H(n_m, K_{cm}, m_c) \oplus K'_{cm}, H(K'_{cm}, n_m))$, where n_m is a nonce and K'_{cm} is the new secret. If the customer cannot remember her previous information, the information cannot be revoked on-line, and the revocation should be done in off-line communication with both the bank and the merchant.

Protocol

$\boxed{C \Longleftrightarrow B}$

$C \to B$:	C, n_b
$B \to C$:	$T_{bc} := H(n_b, K_{cb})$
C:	checks T_{bc}
	generates the new secret K'_{cb}
$C \to B$:	$C, H(n_b, K_{cb}, b_c) \oplus K'_{cb}, H(K'_{cb}, n_b)$
B:	checks $H(n_b, K_{cb}, b_c)$ and $H(K'_{cb}, n_b)$
	updates C's secret
$B \to C$:	ack

$\boxed{C \Longleftrightarrow M}$

$C \to M$:	C, n_m
$M \to C$:	$T_{mc} := H(n_m, K_{cm})$
C:	checks T_{mc}
	generates the new secret K'_{cm}
$C \to M$:	$C, H(n_m, K_{cm}, m_c) \oplus K'_{cm}, H(K'_{cm}, n_m)$
M:	checks $H(n_m, K_{cm}, m_c)$ and $H(K'_{cm}, n_m)$
	updates C's secret
$M \to C$:	ack

4 Discussion

In general, PIN codes are generated by banks, and passwords for services are maintained by service providers though they are generated by service users. Personal secrets like PIN codes are not protected by separation of duty, and this creates a security gap. We showed how to transfer responsibility for secrets to customers in a practical system. Customers generate and maintain their personal secret to access a service. As a result, the service provider is less exposed to the maintenance of customers' secret data. There are no critical customer secrets on the service provider's side.

To minimise communication costs, off-line transactions are preferred in some cases. To make transactions off-line, one has to be able to postpone procedures between the merchant and the bank for some time, and the proof of purchase has to be issued to the customer when the purchase is made. After receiving T_{cmb} from the customer, the merchant can issue $H(t, u, K_{cm}, T_{cmb})$ as the proof of purchase, and the transaction between the customer and the merchant is done. In this case, the merchant does not have a guarantee from the bank, since there is no communication with the bank. As in current off-line transactions using a cheque and cheque guarantee card, we can set a certain credit limit for such transactions and guarantee the merchant his money. The merchant cannot request a different amount of money to bank from the amount in T_{cmb}, since T_{cmb} contains K_{cb}, and both the customer and the bank will have a hash quantum containing the amount u. The customer is also protected.

Existing electronic banking systems keep a file of merchant-customer tuples for authorised transactions. Thus the storage of T_{cmb} will not be a problem in practice. Indeed, limiting the transaction beneficiaries is an important control of fraud, and desirable in the case of off-line merchant transactions. In effect, our scheme allows new merchants to be registered quickly, but only provided the customer, the merchant, and the bank are all on-line at once.

As an alternative of hash functions for our scheme, we could use Message Authentication Codes (MACs). During the registration procedure, principals share secrets K_{cb} between the customer and the bank, and K_{cm} between the customer and the merchant. When we adopt a MAC for the scheme, these secrets can be used as keys for the MAC without further effort. By the nature of MACs, hashed values can be seen only by key sharers. But it does not help reduce the complexity of the scheme.

This scheme provides non-repudiation of both ordering and payment. A merchant cannot make a false transaction without a customer's partici-

pation, because he does not have the customer's secret such as K_{mb} and K_{cb}. Since a customer has to send a merchant the hash T_{cmb} of a payment id and her private information registered in the bank, the customer cannot deny that the transaction was done by her. For both principals, a merchant and a customer, they cannot forge a transaction by denial for the transaction.

One of the underlying ideas is a customer-oriented transaction; a customer plays a central role in setting up relations to a bank and merchants and making transactions, and keeps her secrets against other principals. Collusion between a bank and a merchant is not so helpful in this model.

5 Summary and Future Work

We provided a scalable and resilient access control scheme based only on hash functions, presented applications to electronic payment, and gave a discussion of the scheme. Since this scheme does not share the customer's secret with either the bank or the merchant, we can develop privacy-enhanced and customer-oriented transactions. As the scheme does not deploy a large public key infrastructure, the implementation cost can be lower than conventional schemes such as SET [VM97]. Our scheme can be applied to the existing payment infrastructure with a minimum of change.

We consider applications of our scheme. Paper-based payments using a cheque are still popular, but expensive for banks to process. An on-line electronic cheque can be a system that transfers funds from the payer's bank account to the payee's bank account at the time the transaction takes place. Our scheme is also applicable here, by replacing the merchant with the payee with suitable modifications. One can also consider sophisticated off-line features such as deferred payment, bouncing cheques, etc. Electronic cheque transactions are a topic of future work.

Acknowledgements

I am grateful to Ross J. Anderson, Václav Matyáš Jr. and the anonymous referees for their helpful comments and advice.

References

[ALN97] M. Abadi, T. M. A. Lomas, and R. M. Needham. Strengthening passwords. SRC Technical Note 1997-033, Digital Systems Research Center, Palo Alto, CA, September 1997.

[And92] R. J. Anderson. UEPS - a second generation electronic wallet. In ESORICS '92, volume 648 of LNCS, pages 411–418. Springer-Verlag, 1992.

[And94] R. J. Anderson. Why cryptosystems fail. Communications of the ACM, 37(11):32–40, 1994.

[BM92] S. M. Bellovin and M. Merritt. Encrypted key exchange: password-based protocols secure against dictionary attacks. In the 1992 IEEE Symposium on Security and Privacy, pages 72–84, Oakland, CA, 1992.

[BM93] S. M. Bellovin and M. Merritt. Augmented encrypted key exchange: a password-based protocols secure against dictionary attacks and password file compromise. In the First ACM Conference on Computer and Communications Security, pages 244–250. ACM SIGSAC, November 1993.

[DA97] T. Dierks and C. Allen. The TLS protocol version 1.0. Internet-DRAFT, Internet Engineering Task Force, November 1997. <ftp://ftp.ietf.org/internet-drafts/draft-ietf-tls-protocol-05.txt>.

[GLNS93] L. Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting poorly chosen secrets from

guessing attacks. IEEE Journal on Selected Areas in Communications, 11(5):648–656, June 1993.

[Hal94] N. M. Haller. The S/Key one-time password system. In ISOC Symposium on Network and Distributed System Security, pages 151–157, San Diego, CA, February 1994. see also IETF RFC 1704, 1760, and 1938.

[MTvHZ92] R. Molva, G. Tsudik, E. van Herreweghen, and S. Zatti. Kryptoknight authenication and key distribution system. In ESORICS '92, volume 648 of LNCS, pages 155–174. Springer-Verlag, 1992.

[Nee97] R. M. Needham. The changing environment for security protocols. IEEE Network, pages 12–15, May/June 1997.

[NS78] R. M. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. Communications of the ACM, 21(12):993–999, 1978.

[NT94] B. C. Neuman and T. Ts'o. An authentication service for computer networks. IEEE Communications Magazine, 32(9):33–38, September 1994.

[Rub96] A. D. Rubin. Independent one-time passwords. USENIX Computing Systems, 9(1):15–27, 1996.

[VM97] VISA and MasterCard. SET Secure Electronic Transaction Specification Formal Protocol Definition 1.0, May 1997.

[Ylö96] T. Ylönen. SSH – secure login connection over thr internet. In the 6th USENIX UNIX Security Symposium, pages 37–42, San Jose, CA, June 1996.