RZ 3197 (# 93243) 01/03/00
Engineering & Technology 6 pages

Research Report

Cable-Free Electronic Locks

Ceki Gülcü

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

Cable-Free Electronic Locks

Ceki Gülcü

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

We describe a new and flexible approach for managing physical security. Our approach does not require cabling connecting locks to a management center. A simple networking protocol to disseminate access control information is presented. Cryptographic protocols to protect against various types of attacks are also considered.

1 Introduction

Each and every one of us relies on keys to gain access to our homes, offices, and cars, to snap open the lock on our bicycles, or unlatch our postal boxes. Keys are undeniably an integral part of modern society.

Although widespread, traditional keys suffer from a number of shortcomings. First, in the case of a security breach, for example, as a result of the loss or unauthorized duplication of a key or the deprecation of trust assumptions, the lock must be replaced. Second, a key is valid as long as the corresponding lock remains in place. For example, once an access right is given to an individual by handing him/her a key, that right cannot be revoked unless the key is returned or the lock changed. As a consequence of these limitations, traditional keys do not allow one to institute a time-based access policy, for example allowing access only during office hours.

Whereas traditional keys offer only inflexible and expensive management primitives, they are widely available, very reliable, and reasonably cheap. Replacement solutions have been invented and are finding their way to the market. Most notably, electronic locks provide flexible management capabilities.

This paper describes how cost-effective and easily manageable electronic locks can be implemented. We begin by presenting the setting for the problem to be considered. Having fixed the context, the primary innovation, namely the propagation of access control information in cable-free environments, is described. Then, we discuss attacks and implied security assumptions.

Contrary to most current electronic security systems, the system described in this paper does not require fixed network connectivity of any kind, be it wire or radio-based. Consequently, the costs associated with cabling or radio transmission equipment is eliminated. Experience shows that this cost can be one order of magnitude higher than the cost of the lock itself.

Instead of the locks receiving electrical power via a fixed cable, we could imagine that the required energy to operate the lock is delivered either through the user's key or through an embedded battery in the door. Clearly, in such a construction, power consumption must be kept to a strict minimum so that the batteries would last at least a few years. Electronic locks exhibiting the desired physical properties such as tamper resistance, operational reliability and long battery life already exist on the market. The Safixx smartlock from ACOLA GmbH in Villingen-Schwenningen, Germany, is an example of a lock that exhibits these properties. The present document emphasizes the logical aspects of cable-free lock constructions. The physical construction of cable-free locks is beyond the scope of this paper.

Available cable-free products resort to manual key management, which becomes rather involved as the number of users increases. In this document the term "keys" does not designate the traditional metallic key but rather any arbitrary carrier of information, in particular smartcards or IBM's JavaCard.

2 Problem Setting

We suppose the existence of a *Door Domain Access Manager* (DDAM), denoting the entity holding authority over an ensemble of doors. The DDAM plus the set of doors[1] constitute an *administrative domain.* For the sake of simplicity we also assume that the DDAM is composed of a single moral entity. This restriction may be removed by applying threshold signatures [12] and straightforward extensions to the communication protocol presented in this document. A user is granted access through a door only if his/her key exhibits an appropriate token issued by the DDAM. All tokens are minted with an expiration date. The DDAM may choose to revoke the access rights of any user at any time.

Keys are built such that they may hold multiple tokens as well as other types of data. Whether tokens are based on public key cryptography, shared secrets or some other type of security primitive is irrelevant at this stage.

In addition to storage capacity, doors have computational capabilities allowing them to process tokens issued by the DDAM. The relation between the DDAM and the doors is that of master and slave. It is the DDAM's reserved prerogative to determine who may traverse which door and until what time and date. Consequently, the doors within an administrative domain completely trust the DDAM and blindly obey its orders. However, doors do not honor orders given by a DDAM outside their domain. *We assume no direct cabling linking the doors amongst themselves nor with the DDAM.* This mandates an alternative communication mechanism so that the DDAM and locks can still exchange information. Note that we do not preclude the possibility of doors exchanging messages with one another.

[1] In this document, the term *door* and *lock* are interchangeable.

2.1 Suitability Considerations

As users are required to present a valid token to access a door (by construction), the dissemination of positive access rights is not an issue. The DDAM can pass a token to a given user through a (secure) terminal or the token can simply be dropped at a door until the user unwittingly picks it up. Renewing tokens that are about to expire can be accomplished in a similar fashion.

On the other hand, how can one ensure that a negative token (i.e. access revocation) reaches the relevant lock without excessive delay?

The solution we are about to present is particularly appropriate in cases where all users happen to go through certain central doors, such as a building entrance, elevators, or garage barriers. In another favorable scenario—typical of business environments—rapid propagation of information is guaranteed by workers who arrive at work in the morning or by the maintenance/security personnel who frequently go through many doors.

As shall be explained below, if control information does not reach the destination node, the source will not receive an acknowledgment. The source can thus detect failure. It follows that in the case that there is no privileged central door and the target door is infrequently visited, the operator of the DDAM is alerted and he/she can always walk to the door in question. Critical doors requiring immediate information propagation can be connected to the DDAM with a dedicated cable.

3 Network Model

To make certain that information is propagated in a timely fashion, users' (physical) keys carry messages to and fro between locks and the DDAM. Moreover, locks act as a temporary repository for messages designated for other locks. *The replacement of cabling by mobile users is the primary innovation of this paper.* We model doors as nodes in a network and users as channels linking these nodes. A temporary channel between door i and door j is established when a user travels through them. To improve connectivity, we require intermediary locks and user keys to act as repositories. In this way, a user Alice may pick a message from door i and leave it at door j. Then, user Bob, who happens to go through door j, carries it until its final destination, say door k, Fig. 1.

More precisely, both keys and doors contain a snapshot view of all current pending traffic, that is,

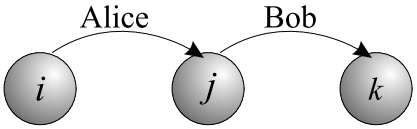

Figure 1: Network model.

all sent but not yet acknowledged packets. This is true for any destination within the administrative domain. When a key is inserted in a door, both key and door update their respective view of the network. Naturally, this update need not be interactive as it may possibly be performed off-line. This way the user shall not be delayed waiting to synchronize the key and the lock before going through a door. Off-line updates mandate that each entity possess its own power source.

3.1 The Networking Protocol

Our protocol is basically simplified TCP/IP [11] with slight optimizations tailored for our unusual setting. The abstract problem can be stated as the building of a transmission control protocol (e.g. TCP) on top of a high loss, varying topology LAN.

First, we assume that each node of the network, that is a key, a door, or the DDAM, can be addressed individually. Hereafter, the term node may refer to a door, a key, or the DDAM. This can be accomplished by using the addressing scheme of any networking layer protocol, in particular the Internet Protocol [7].

As we mentioned above, each node is responsible for propagating packets, even if it is not the intended recipient, until the packet is acknowledged, at which time the packet is erased from the node's memory.

Like in TCP [8], each packet requiring acknowledgment is sent with a monotonically increasing sequence number. TCP has to deal with packet fragmentation caused by the IP [7] layer. In our homogeneous setting, packets are not liable to be fragmented. Hence, our sequence numbers correspond to packets and not the number of bytes sent. This practice is unlike TCP but has the advantage of being both simpler and more space-efficient.

Once a message has been received, the recipient issues an acknowledgment back to the sender. The acknowledgment consists of the usual addressing information and at least two other fields: a cumulative sequence number, denoted n, and a bit field of length w bits.

The actual value of n indicates that packets

bearing a lower sequence number (inclusive) have been correctly received without gaps. The bit field provides information about out-of-sequence packets. The bit field loosely corresponds to a *sliding window.*

As in TCP, packets outside of the sliding window, that is packets bearing a sequence number outside the range $n+1$ to $n+w$, are dropped. On the other hand, if an incoming packet completes to a sequence with no gaps, then n is incremented and an acknowledgment is sent back to the source. Each packet received out of sequence but within the sliding window is tallied by a corresponding bit in the bit field. It is not dropped as would be the case in TCP.

After the acknowledgment propagates back the sender of the packet, the source can *selectively* retransmit packets that have been lost. Packets that have correctly arrived, albeit out of sequence, need not be retransmitted. Also, packets that have been selectively acknowledged are dropped by intermediary nodes regardless of whether the source has received the acknowledgment, immediately freeing precious memory space at each intermediary node.

It is important to note that acknowledgments presented here have an absolute ordering sequence much like cumulative acknowledgments. Thus, they may be compared with one another. *Consequently, only the newest acknowledgment need be stored by the intermediate nodes because the newest acknowledgment overrides all previous ones.*

3.2 Window Size

It is reasonable to assume that in most cases the frequency of revocation messages sent to a specific door is low. Thus, very small window sizes (say 2) may be ample to satisfy even the needs of large organizations.

However, for exceptional cases, the window size can be increased at will. This can be accomplished by the DDAM by sending out a broadcast message instructing all doors to adjust the window size for a given source/destination pair. Broadcast messages bear a fixed destination address recognized by all.

3.3 Clocking information

The DDAM could keep time on behalf of all other entities in the administrative domain. The DDAM sends timing updates as frequently as it can (i.e. each time it has contact with a user or door). As with acknowledgments, only the latest clock information need be stored by intermediate nodes, again saving precious memory space. In this way, there is no need for doors to have individual clocks.

4 Cryptographic Aspects

The cryptographic representation of a token can have important effects on security management. We shall first discuss the security implications of a secret-key-based architecture. Next, the implications of a public key architecture are presented.

4.1 Shared Secret

Tokens can be based on a shared secret between the DDAM and the locks. Let us assume that the DDAM and lock l_j share the secret s_j. To grant user u_i access through l_j, the DDAM would authenticate the access string a_{ij}, where

$$a_{ij} = \text{“Let } u_i \text{ through } l_j \text{ until } 1/1/2001''. \quad (1)$$

The authentication could be based on any secure-keyed message authentication function, such as HMAC [5]. We have $t_{ij} = \text{HMAC}(s_j, a_{ij}) \| a_{ij}$, where t_{ij} is the access token and “$\|$” denotes string concatenation. Note that tokens could have other representations based on different types of authentication. User u_i would be granted access through l_j by presenting the token t_{ij}, until January 1, 2001.

As in any digital representation of information, a token may be easily duplicated. In the shared secret setting, if a token can be “seen” by a rogue lock or any other unauthorized party, then the token has in effect been stolen. Indeed, the attacker could present the stolen token to a corresponding lock and obtain illegitimate access.

To mitigate the menace, security must be augmented by other means. For example, we may impose a universally unique serial number to be burned into each physical key and tokens to be issued for a particular serial number. This may provide enough security to repel unsophisticated attackers. However, a determined adversary may print his/her own key with fake serial numbers.

For high-security applications, it is crucial that the physical key allows only selective access to information. When a key is inserted in a lock, it should be able to verify whether the user has a specific token allowing passage through that lock. However, it should not be able to read the tokens relevant to other locks, possibly in other administrative domains.

Even if a key allows selective access to information, lock l_x can pretend to be lock l_y and steal the token (for l_y).

The token duplication attack constitutes a potent threat, particularly if electronic locks become truly ubiquitous. Universal deployment means that keys will be used in mutually untrusting domains.

We can protect against this attack by requiring the lock to identify itself to the key. Only then will the key reveal the token for the identified lock.

If this identification is based on a shared key between the doors and the users, the number of shared keys will explode for even medium-sized organizations—because each lock will have to share a key with each user that goes through it. Lock-user shared keys may be suitable for small settings, nevertheless.

Alternatively, Kryptoknight [1, 3] or Kerberos [4, 6] type third party authentication techniques could be employed. In such a scheme, all users and locks would share a key with the DDAM. As the DDAM shares a key with all nodes, it can convince any node of the identity of another. Let s_i represent the long-term secret that user u_i shares with the DDAM, and s_j the key that lock l_j shares with the DDAM.

The DDAM would randomly choose an encryption key k. The access string a_{ij} would become

$$a_{ij} = \text{“Let through } l_j \text{ user knowing } k\text{, until 1/1/2001”} \tag{2}$$

The DDAM would issue the the ticket T_{ij} defined as

$$T_{ij} = \underbrace{\left\{\text{HMAC}(s_j, a_{ij}) \| a_{ij}\right\}_{s_j}}_{t_{ij}} \| \{k\}_{s_i}\,, \tag{3}$$

where $\{m\}_x$ denotes symmetric encryption of message m using key x and a_{ij} is given by (2).

User u_i would strip the encryption $\{k\}_{s_i}$ from the ticket T_{ij} to retrieve the token t_{ij}. It would then obtain k from $\{k\}_{s_i}$ by decrypting it with its long-term shared key with the DDAM, that is s_i.

To pass through door l_j, the user would present t_{ij}. The lock l_j would retrieve the key k contained in a_{ij} by decrypting t_{ij} with its long-term shared secret key, s_j. The door would then challenge the user for the knowledge of key k. If the response checks with the issued challenge, then the user would be granted access.

Note that the DDAM has to precompute T_{ij} for given message a_{ij} and given nodes u_i and l_j. The gist of this paper is the assumption that there is no direct channel (a connecting cable) to the DDAM. Consequently, the Kerberos setting is unsuitable for situations where either the message being authenticated or the intervening parties are not known in advance. This is an important limitation if the application requires that disruption attacks be prevented (see Section 4.3).

The verbosity of the access string a_{ij} should indicate to the reader that the protocol just described is not optimized. It has not been analyzed for security flaws either. It is intended for illustration purposes only.

4.2 Public Key Setting

An alternative setting would have the DDAM and all users within the domain hold a public/secret key pair. Each lock within an administrative domain would be installed with the DDAM's public key. The DDAM would then certify the public keys of all users within its administrative domain.

The access token for user u_i through lock l_j would be the DDAM's signature on the message a_{ij}.[2]

Whenever a user wishes to pass through a door, it would present the token to the lock, which would verify DDAM's signature. To prevent impersonation, the door would require the user to prove knowledge of the secret key corresponding to the user's (DDAM certified) public key.

This proof can be based on any challenge response type protocol, such as Schnorr [10], Feige-Fiat-Shamir [2] or Guillou-Quisquater [9].[3]

The solution just presented separates the user's credentials from his/her public key. Thus, Alice's credentials may be transported to the relevant door without Alice having to intervene. To benefit from new credentials, Alice simply has to prove her identity. Alice does not have to obtain a new secret from the DDAM nor change her public key. The new credentials are transparent to Alice.

On the other hand, if we had tied each of Alice's credentials to a corresponding secret held by Alice, then Alice would have to communicate to the DDAM on a secure channel each time her credentials have changed.

For example, in the Guillou-Quisquater [9] protocol, the DDAM would send Alice her new credentials, denoted by the string J, and a corresponding secret

[2] As defined in Equation (1).

[3] This list is not exhaustive.

B, such that

$$JB^v \equiv 1 \pmod{n}, \quad (4)$$

where n is a public modulus and v is a public exponent. Only the DDAM knows the inverse of $v \bmod \phi(n)$ and hence efficiently computes B, verifying (4).

To allow the access rights granted by J, the relevant lock would challenge Alice with a random number and check whether Alice knows the secret corresponding to J, which is B. As usual, Alice does not reveal B as a result of the challenge response protocol. Note that the secret B has to be sent by the DDAM to Alice on a secure channel.

4.3 False Acknowledgment Injections

If an attacker is able to inject an acknowledgment message bearing a high sequence number, all intermediary hops will drop packets bearing a lower sequence number. Although this *disruption attack* will eventually be detected, it will momentarily prevent all communications between the DDAM and the locks.

In another attack variation, assuming that newer messages have higher priority, an attacker may flood locks by sending fake packets bearing high sequence numbers. The intermediate nodes will regard such packets as being newer and drop lower-numbered but genuine messages. This too will disrupt communications.

High-security applications, especially those slated for universal deployment, mandate that all messages, including acknowledgments, be authenticated. Thus, all nodes, that is all doors, users, and the DDAM, would have a public key certified by the DDAM.

The Kerberos-type shared secret setting does not allow messages to be authenticated for two dynamically chosen parties. We can prevent disruption attacks only in the public key setting.

5 Conclusion

We have presented a new and flexible architecture for managing physical security using electronic keys. Our architecture does not require a permanent channel between the locks and a key management center. The elimination of cabling significantly reduces the cost of installation.

The cable-freeness property raises the question of timely propagation of access control information. We solve this problem by having users act as transmission channels and locks as message repositories.

We have presented a customized networking protocol similar to TCP that achieves selective retransmission of lost packets while still retaining absolute ordering for acknowledgments.

Security implications of shared-key versus public-key-based architecture have been discussed. Owing to the key explosion problem, a shared-key architecture is suitable for lower-security applications or for small-scale deployment. High-security applications or universal deployment both mandate public key cryptography.

References

[1] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva, and M. Yung. The KryptoKnight family of light-weight protocols for authentication and key distribution. *IEEE Transactions on Networking in 1995*, 1995.

[2] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology: the Journal of the International Association for Cryptologic Research*, 1(2):77–94, 1988.

[3] P. Janson, G. Tsudik, and M. Yung. Scalability and flexibility in authentication services: The kryptoknight approach. In *IEEE INFOCOM'97*, Tokyo, Japan, April 1997.

[4] J. Kohl and C. Neuman. RFC 1510: The Kerberos Network Authentication Service (V5), September 1993. Status: PROPOSED STANDARD.

[5] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication.* Internet Engineering Task Force, February 5 1997. RFC 2104.

[6] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications Magazine*, 32(9):33–38, September 1994.

[7] J. Postel. *Internet Protocol.* Internet Engineering Task Force, September 1 1981. RFC 791.

[8] J. Postel. *Transmission Control Protocol.* Internet Engineering Task Force, September 1 1981. RFC 793.

[9] Jean-Jacques Quisquater and Louis Guillou. A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. In Christoph G. Günther, editor, *Advances in Cryptology—EUROCRYPT 88*, volume 330 of *Lecture Notes in Computer Science*, pages 123–128. Springer-Verlag, 25–27 May 1988.

[10] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[11] W. Richard Stevens. *TCP/IP Illustrated, Vol. I.* Addison Wesley, 1994.

[12] Douglas R. Stinson. *Cryptography - Theory and Practice.* CRC Press, 1996.