Limited-lifetime shared-access in mobile systems

Zygmunt J. Haas [a] and Sanjoy Paul [b]

[a] *AT&T Bell Laboratories, Room 15G-212, 67 Whippany Road, Whippany, NJ 07981, USA*
[b] *AT&T Bell Laboratories, 101 Crawfords Corner, Holmdel, NJ 07733, USA*

Received October 1994

Abstract. In this paper, we propose a simple access protocol to shared information in a mobile environment. The objective of the proposed scheme is to allow a specified set of users access to shared information and protect the confidentiality of the information from users outside this set. The set of users may be updated from time to time. In particular, the duration during which a user is allowed access to the shared information may be restricted. Furthermore, the information itself has limited lifetime and the confidentiality of the information has to be preserved during this lifetime only. The proposed access protocol is in particular suited to the mobile environment, because of the loose binding of the communicating entities.

1. Introduction: The problem

In this paper, we investigate the problem of limited-lifetime confidential access in a mobile environment. The problem and the underlying set of assumptions is as follows:

- The information, referred to here as the *secret*, confidentiality of which needs to be protected, is of limited lifetime. The confidentiality is to be protected during this lifetime only.
- The secret typically consists of a large amount of data.
- A specific set of users, referred to here as the *subscribers*, is allowed to access the secret. Non-subscribers should be prevented from accessing the secret.
- A user becomes a subscriber by remitting an electronic payment, such as charging his credit card account, for example.
- The set of subscribers may change from time to time. In particular, the duration during which a subscriber is allowed access to the secret, referred to here as the *subscription period,* may be restricted. The subscription period is also subscriber- dependent. In other words, the subscriber set is dynamic.
- The number of subscribers is typically very large.
- A subscriber is to be delivered the secret (or a portion of the secret) upon request.

Our environment is that of a mobile system. In such a system, subscribers may be disconnected from the network frequently and for long time durations. The system should not incur expenditures (in transmission resources, for example) for disconnected users. Moreover, there is the underlying assumption that the communication in such a system may be provided, at least in part, by wireless links. Thus the amount of transmitted data should be minimized.

To address the mobile environment, our protocol is based on the connectionless Client-Server model, loosely coupling the subscribers to the secret holder. Because mobile machines are less reliable, the amount of data cached on the mobile is minimized. Finally, the system incurs no transmission costs for disconnected users.

An example of an application with the above requirements is the *electronic newspaper* – the secret, confidentiality of which needs to be protected. In this application, a set of users (the subscribers) are allowed access to the newspaper for predetermined, by the paid subscription fee, length of time. Typically, a newspaper contains a large amount of data; a user is usually interested in a small portion of the information in the newspaper. As old news are of little value, confidentiality of an old newspaper edition need not be preserved.

Because we do not place any restrictions on the storage system nor on the transmission medium, our protocol needs to ensure that no one can easily access the secret without being authorized to do so. In our example application, the newspaper can be stored on a public server and the transmission can be done over the broadcasting wireless medium.

We would like to emphasize that there is no need to provide a strong crypto system, because of the limited lifetime of the secret. It is sufficient that the strength of the system is such that the anticipated time to break the secret is longer than the secret's lifetime. (We refer to this property as weak crypto system.) Alternatively, it suffices that the expected expenses associated with breaking the system are significantly higher than the price of the subscription.

The challenge is, thus, to propose a set of *simple* protocols, capable of supporting the above assumptions. We said simple, as the protocols are to be executed on

CPU cycle-limited and storage-limited mobile machines. We base our protocol on 'not-so-new' scheme of two sequential symmetric crypto systems, which is described in the next section. Section 3 outlines the message exchanges in our protocol, showing how the secret is protected from various attacks. In section 4, we provide some additional thoughts on how the interface routine, a central element in our protocol, can be constructed. Section 5 is a short summary of the paper and the Appendix contains some analytical calculations showing the relative improvement of our protocol as compared with a simple encryption-per-access scheme.

Throughout this document we use $E_k(x)$ to denote information x encrypted with the key k.

2. The *Locker Key* scheme

A brute force solution to the shared access problem presented in the previous section is to establish a secure link between any subscriber and the location of the secret. This can be accomplished by assigning a secret key K_i to the user i (the key being known to the user and to the secret holder), so that when a user requests access to the secret, his identity is verified and the secret or part of the secret is encrypted with K_i and sent to the user. [1] The major problem with this solution is that it is performed on the encryption-per-access basis; i.e., each time a user requests access to the secret, the secret holder invokes the encryption algorithm. This places a relatively large load on the secret holder, especially during peak demand times and can lead to slow system response. The opposite approach, in which "real-time" CPU cycles are "traded" for memory by storing (in the network or on the mobiles) a copy of the encrypted secret per subscriber, is prohibitively expensive in the amount of required storage, since the secret contains large amount of data, only a small portion of which will be accessed by a subscriber.

Our protocol is based on a simple scheme, which we refer to here as the *Locker Key* scheme. The scheme is based on two symmetric cryptographic systems: the *master key*, K_{MK}, and a user specific key, K_i for user i. The idea is to protect the information from users outside the group by encrypting it with the Master Key K_{MK} and making the master key available to the users within the group only. User i is provided with a locker (usually a buffer at the server, but can be anywhere in the network) in which K_{MK} is placed encrypted (locked) with K_i (i.e., $E_{K_i}(K_{MK})$ is stored at the user i buffer). The secret is encrypted with K_{MK} and can be made publicly available. When user i wants to access the secret, he first retrieves the K_{MK} from $E_{K_i}(K_{MK})$ by using K_i (unlocking his locker) and then retrieves the secret by using K_{MK}. The scheme is pictorially depicted in Fig. 1, where k_1 is the user's private key and K_{MK} is the master key.

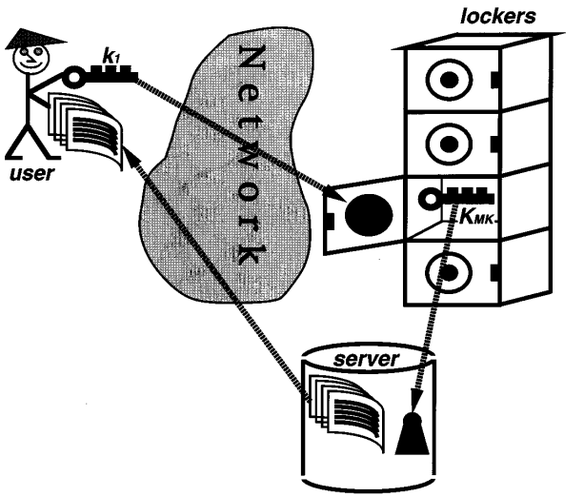

Fig. 1. *The Locker Key* scheme.

3. The *Locker Key* scheme for electronic newspaper access

Throughout this document we use the term "user" to specify either a human operator or an application program. The actual meaning will be clear from the context.

In this section, we show how the basic *Locker Key* scheme can be used to build a protocol that preserves limited-lifetime secret confidentiality in a dynamic subscriber set environment. To facilitate the exposition, we use the electronic newspaper example as the application. The reader is, however, reminded that the protocol presented here can support any application with the requirements outlined in section 1. We first present the basic scheme, and then additional modifications, necessary to support the required level of confidentiality protection.

The basic scheme is as follows: a single copy of the newspaper is stored on the server, encrypted with the master key, K_{MK}. User i that wishes to purchase a current copy of the electronic newspaper, sends his credit card number and user-generated key, K_i to the server. In return, K_{MK} is locked in his locker; i.e., the key $E_{K_i}(K_{MK})$ is placed in the user-i buffer. When user i requests a copy of the newspaper or an article in the newspaper, she "opens her locker," retrieves the key K_{MK}, and decrypts the encrypted portion of the newspaper.

The exchange of information for the proposed scheme is shown in Fig. 2 and starts with user-i subscription to the service by sending his ID (i.e., i), credit card number, his key (K_i), and current date/time to the news-

[1] [1] describes a set of protocols for establishing a secure point-to-point channel, especially suitable for the wireless and mobile environments.

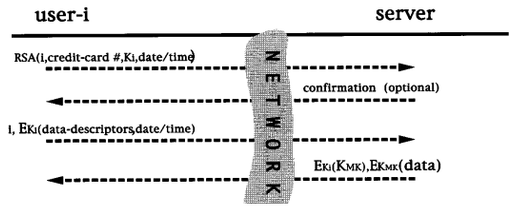

Fig. 2. The *Locker Key* scheme: basic description.

paper server, encrypted with the *public key* of the server.[2] The user ID can be the mobile's IP address, for example. Alternatively, it can be assigned by the server. The credit card number is used for billing purposes and can be the user's account number, his phone number, or any other charge mechanism. K_i is the user- generated, user-specific key that will be used to encrypt the communication between the server and user-*i*. (In general, the key K_i could be user's credit card number. However, to allow more flexibility and to improve the security level of the scheme, the two parameters are separated here.)

The date/time[3] in the subscription message is the date/time on the mobile machine when the subscription message is generated and is used to protect against retransmission attack or packet duplication. It is assumed that the timers of the server and the user are *very* loosely synchronized. (For example, synchronization within 15-30 minutes are sufficient.) Received subscription message will be executed only if there is no current subscription under the same billing account and if the following holds:

$$|Time_{message} - Time_{server}| \leq Limit\,, \qquad (1)$$

where $Time_{message}$ and $Time_{server}$ are the date/time field from the subscription message and the date/time at the server when the subscription message is received, respectively. The *Limit* is the allowable mis-synchronization between the mobile's and the server's timers in addition to the maximum network delay.

If a subscription message is lost, the user times out and resends its subscription message. It the lost message reappears at the server, because of condition (1) only one of the subscription messages is executed and the other is ignored.

The server can, optionally, confirm the subscription. When a user requests newspaper data, he sends to the server his ID (*i*) with the description of the requested data (*data-descriptors*), which can be optionally encrypted with K_i for privacy (i.e., if a user does not want others to know what data he is accessing, the *data-descriptors* may be encrypted). The server responds by sending the current K_{MK} encrypted with K_i and the requested newspaper data, encrypted with K_{MK}. A similar mechanism used in the subscription process to avoid retransmission attack or network packet duplication could also be used in every data request message. However, we note that the consequences of retransmission attacks or packet duplication in data request messages are not as severe as in the subscription case, since in the former the result is only that protected data will be retransmitted, while in the latter a user may be charged several times for the same subscription.

The server may periodically re-encrypt the newspaper with a new master key, K_{MK}, and place this new key in all the eligible lockers. When a user's access permission expires, his locker is not reloaded with the new master key.

We assume that, since the newspaper contains rather a large amount of data, users requests are article-based; i.e., user requests first the index of the articles (which is an article itself), with each future access requesting a single article per request. Thus each newspaper article is individually encrypted and stored on the server, allowing access to one article at a time.

Note the following features of the basic protocol:

- A **single** encrypted newspaper copy is stored at the server.
- The newspaper encryption is done **once** (per master key).
- The encryption can be done "**off-line**" (i.e., not "real-time"), thus eliminating possible server congestion instances when the demand peaks.
- Even if the master key changes periodically, the server does not multicast the new key. As long as the lockers of the eligible users are modified, the users can access the lockers with their respective K_i and retrieve the new master key. Since redistribution of the often-changed key to inactive (e.g., disconnected) users in a mobile environment wastes expensive bandwidth, while our protocol avoids such waste through the use of the *Locker Key* mechanism, our protocol is especially suited for the mobile environment.

The basic scheme, as described here, is susceptible to fraud by malicious subscribers, because they can distribute one or more of the following pieces of information to non-subscribers, breaking the confidentiality of the system.

- K_i
- K_{MK}

[2] Note that we assume that the server's public key is available from a key registry, which is the equivalent to "yellow pages" for public key cryptography and allowing secure transmission of set-up or query data to various servers, such as newspaper, weather, stock-quotes, etc. Public key encryption is used only at this set-up stage for secure transmission of the secret information of user *i* to the server (e.g., the key K_i) and to hide the identity of the user. In all other cases, symmetric cryptography is used.

[3] It is assumed that all the dates and times are adjusted to one standard time, EST, for example.

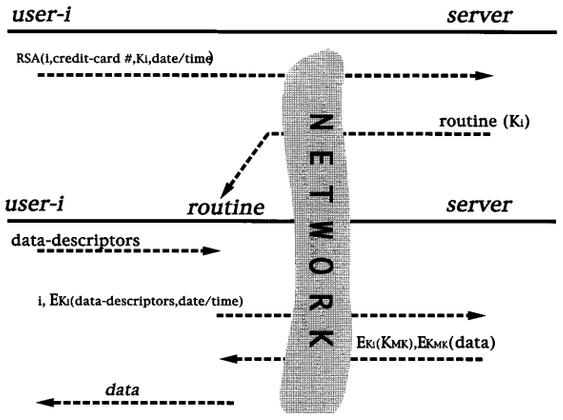

Fig. 3. Use of the interface routine to prevent fraudulent users' behavior.

- the newspaper itself.

Note that since the authorized user i has access to K_i, K_{MK}, and $E_{K_{MK}}(newspaper)$, he can retrieve the newspaper and distribute it to an unauthorized user j. Alternatively, if the newspaper contains a large amount of data, which is impractical or costly to decrypt and distribute in its entirety, user i may choose to hand only K_{MK} to user j. User j can tap a newspaper article when it is send to another user and decrypt the article using his (illegal) knowledge of K_{MK}. Furthermore, user j can now impersonate user i by requesting a newspaper article from the server. Some limited protection against distribution of K_{MK} is provided by changing the master key frequently. In that case, user j, who has obtained an **old** master key from user i, will not be able to read the newspaper, because the newspaper is now encrypted with a **new** master key. This, of course, might not prevent a malicious user from periodically, possibly automatically, retrieving and rebroadcasting the value of K_{MK}.

The periodical change of K_{MK} requires user i to frequently re-distribute the new K_{MK} to user j. Instead, user i can hand to user j his private key, K_i, allowing user j to fully "impersonate" user i (i.e., accessing user-i locker). All the above scenarios compromise the confidentiality of the system.

In the modified version of the basic protocol, the above fraudulent behaviors are prevented by restricting the users from having direct access to the server. This is accomplished by introducing the notion of the *interface routine.* The interface routine is a (relatively short) software program, which is sent as an object code to the user as part of the subscription process (see Fig. 3).[4]

[4] There may be some concern of running a code provided by the service provider on users' machine. Firstly, we point out that this is not a new approach; the Intelligent Agents (e.g., General Magic Telescript Language) are designed to move within the network and be executed on users' machine. Secondly, a limited shell may be created on the users' machine, restricting the operation of such code.

The interface routine provides a means for the user's machine to access the server, in such a way that the value of the K_{MK} is hidden from the user, thus preventing the user from distributing the K_{MK} or from decrypting and distributing the newspaper. The routine can be considered as an extension of the server, remotely executing on the user's hardware. To access the newspaper, user simply executes the interface routine, which fetches $E_{K_i}(K_{MK})$ from the server transparently to the user. The routine should also provide a Graphical User Interface (GUI) to the user. All the operation of the routine is automatic, hiding the (anyway low) complexity of the protocol. It is emphasized that usage of the interface routine does not complicate the access, but actually simplifies the access procedure. (In fact, GUI will be most probably provided to the subscribers).

When user requests the newspaper article, the interface routine retrieves from the server the $E_{K_i}(K_{MK})$ together with the encrypted portion of the requested article. The routine, then, uses the user's K_i to decrypt the master key, which is in turn used to decrypt the newspaper. The cleartext is then handed to the user's program or displayed on his screen (through the GUI, for example). Thus, all communication between the user is with the routine and not with the server directly.

The fact that a user knows her K_i allows her to intercept the communication between the routine and the network, to retrieve $E_{K_i}(K_{MK})$, and to obtain the K_{MK}, bypassing the routine all together. This problem can be eliminated by creating K_i from two components: one supplied by the user, K_i^u, and one supplied by the server, K_i^s. K_i is then computed by the server and the routine, for example as: $K_i = K_i^u \bigoplus K_i^s$. This change is depicted in Fig. 4. The server-supplied K_i^s is hidden in the routine; thus K_i is not known to the user. Now, even if a user intercepts the communication between the server and the routine, she cannot retrieve K_{MK}, because K_i is not known to her. The routine can be sent to the user in the clear, because nobody other than the specific user can provide K_i^u to the routine to compute K_i that is required for the routine to operate.

A dishonest user i still can transmit a copy of his routine to an unauthorized user j, who can use the routine to access the newspaper. Note that user i needs to supply

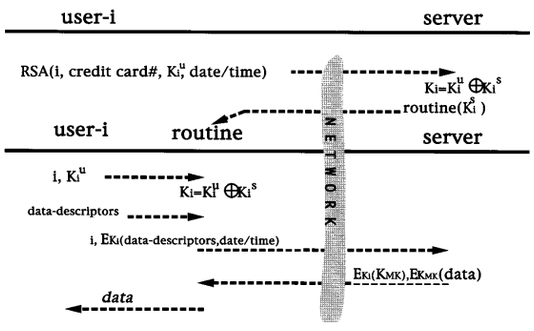

Fig. 4. Modification to protect against user's interception.

user j with his (secret) key K_i^u. User j can then invoke the routine and supply K_i^u to the routine to read the newspaper. We suggest the following two solutions to the problem:

1. Prevention by apprehension: Similarly to [2], the interface routine, when invoked using correct K_i^u, allows access to the original user's billing information (e.g., his credit card number), by *flashing* the billing information of the original (legal) user on the screen, for example. In that case, the original user i may be reluctant to give it away.

2. Charge per-access: In this approach, in addition to the subscription, a user may be charged small fee on the per-access basis. Alternatively, the initial subscription fee allows some limited number of accesses. In this way, the original user that hands his routine to another user will be charged (or effectively charged) for the other user's accesses.

Finally, we note that another form of fraudulent behavior is to redirect the routine output to a file [5] and distribute the file to unauthorized users. Protection against such behavior falls under the category of electronic marking for copyright protection and is beyond the scope of this work. An interested reader is referred to [3] and [2] for more details.

4. The security of the interface routine

The scheme presented in this work relies on the fact that the interface routine is made relatively difficult [6] to tamper with, so that it can hide the values of the keys. Additionally, it should be impossible for anyone to write his own version of the routine, without the forgery being prevented (or at least detected) by the server. (See also [3] and [2].) We present here some thoughts on the design of the interface routine. Interested reader should also examine current literature on software copyright.

One way of ensuring the security of the interface routine is to have "customized" routine structure. The objective is to make automatic retrieval of data from the routine code impossible and manual retrieval much more expensive [7] than the price of a single newspaper's edition. In particular, reverse engineering (i.e., disassembly of the code) should be made considerably more difficult, requiring "manual" processing. Customization of the routine may, for example, involve changing the location of the keys within the routine's code or within the run-time memory, changing the data flow of the routine's execution, adding extra commands, etc. Furthermore, the routine should be often re-distributed; for example, with every new newspaper edition. Thus, an attempt to disassemble the routine code will allow the offender to gain illegal access only to that edition and assuming that the cost involved in disassemblying the code is much higher than the price of access to a **single** edition, an attempt to illegaly access the routine will not pay off.

Specifically, we would like to address the issue of customizing the routine's execution flow. **Random** interleaving data and executable code may prevent automatical disassembly of the code. Changing this interleaving pattern frequently (i.e., with every routine's edition) will require the offender to understand the flow of the code and manually decode it each time. This approach may be particularly effective, if the hidden keys can be translated into meaningful machine language instructions. Then, the key location is randomly moved within the machine code of the routine. Furthermore, indirect reference to the location of a key will require the intruder to closely follow the code execution flow to determine this location. The key to security of the routine is in the "randomness" of the routine structure. In other words, the number of possible routine structures should be so large, that it would be prohibitively expensive for an intruder to collect all possible versions of the routine. For example, this can be achieved by dividing the code into a large number of small modules and randomly shuffling the modules and the locations of the keys.

Figures 5 and 6 show two "different-looking" routines performing the same task. Two keys are hidden in different locations that are indirectly accessed. In this example, the indirection is done based on the content of another memory location (**OFF**), but can also be implemented using the content of a register. Thus, it should be quite difficult to design a program to determine the

```
        MOV  AX,  I(OFF)   // indirectly jump through
                           // indexing with the
                           // content of location OFF
        MOV  DS,  AX
        MOV  SUM, 0
        CMP  SUM, 100
        JNA  NOT-DONE
OFF:    MOV  AL,  SUM      // the value of this location
                           // determines the key offset
             .
             .
             .
NOT-DONE:
             .
             .
             .
        MOV  DS,  AX
        MOV  ES,  AX
        CMP  SUM, 73       // offset with OFF points here.
        MOV  SI,  AX       // THE KEY STARTS HERE
             .             // AND MAY GO ON
             .             // SPANNING SEVERAL
             .             // "DUMMY" INSTRUCTIONS
             .
```

Fig. 5. Hidding a key in the interface routine: version 1.

[5] As bit-map, for example.
[6] In the weak security sense.
[7] Expensive in time, required resources, etc.

```
            MOV  AX,  I(OFF)  // indirectly jump through
                              // indexing with the
                              // content of location OFF
            MOV  DS,  AX
            MOV  SUM, 0
            MOV  AL,  SUM
            CMP  SUM, 100
            JNA  NOT-DONE
                 .
                 .
                 .
NOT-DONE:
                 .
                 .
                 .
OFF:        CMP  SUM, 45      // the value of this location
                              // determines the key offset
                 .
                 .
                 .
            MOV  DS,  AX      // offset with OFF points here.
            MOV  ES,  AX      // THE KEY STARTS HERE
            CMP  SUM, 73      // AND MAY GO ON
            MOV  SI,  AX      // SPANNING SEVERAL
                 .            // "DUMMY" INSTRUCTIONS
                 .
                 .
                 .
```

Fig. 6. Hidding a key in the interface routine: version 2.

location of the key. The key in the two examples translate into a sequence of legitimate machine instructions. This approach can be used to hide the same key in different locations or two different keys in two different locations.

5. Summary

We have proposed a simple protocol that provides confidential access to a limited-lifetime information in a mobile environment. An illustrative example of an application demonstrating our underlying assumptions is the electronic newspaper. In this application, a large number of users subscribe to the service, which allow them to access the large amount of data that the newspaper contains. The set of subscribers, as well as their subscription period change dynamically. Because the confidentiality of the information needs to be preserved during limited lifetime only and because of the relatively low cost of subscription, the protocol needs to provide only, what we call, weak-security level; i.e., an attempt to break it requires longer than the limited-lifetime or is more expensive than the subscription cost.

Our protocol is based on a simple scheme, which we refer to here as the *Locker Key* scheme. In this scheme, the secret is encrypted by a master key and stored once on a central server. The master key, whose size is significantly smaller than the size of the secret, is then multiply encrypted by a subscriber-specific key.

Performance analysis of the scheme indicates that, with reasonable parameter values, the gain in access time is reduced two to three orders of magnitude. The server load is also considerably reduced. We envision that, with the proliferation of mobile multi-media services, this scheme may be of particular interest, finding applicability for various applications with similar requirements as the ones presented here.

Acknowledgement

We would like to thank Michael Reiter and Thomas Woo for commenting on an earlier version of the paper.

Appendix: Gain in access time by batch encryption

We calculate the reduction in access time experienced by users in a system based on batch encryption (S_1), such as the Locker Key scheme presented here, as opposed to a system in which the encryption is performed in real time (S_2). We assume that there is a single encryption server and that the arrivals of requests at the electronic newspapers server are Poisson distributed with parameter λ. The access time in each one of the systems S_1 and S_2 consists of a number of components. System S_1 includes:

- retrieving the encrypted master key – $r(LC)$,
- retrieving the pre-computed ciphertext of the requested article – $r(CT)$,
- transmitting the two items back to the user – $t(LC + CT)$.

System $S2$ includes:

- retrieve the requested data – $r(A)$,
- encrypt the articles – $e(A)$,
- transmit the ciphertext $t(CT)$.

Assuming zero load, the access time x_1 for the system S_1 and x_2 for the system S_2 are

$$x_1 = r(LC) + r(CT) + t(LC + CT),$$

$$x_2 = r(A) + e(A) + t(CT). \tag{2}$$

Note that, in general, the master key, LC, is of fixed length, while the lengths of the cipher text, CT, and the cleartext article, A, are random variables (r.v.). Thus, x_1 and x_2 are also r.v. with some distribution. Each of the processes $r(\cdot)$ and $t(\cdot)$ can be modeled as a queue. Consequently, each one of the systems S_1 and S_2 is composed of a series of three queues in tandem. Here, we will postulate the following (quite realistic) assumptions:

1. One queue is considerably more congested than others, creating a bottleneck.
2. The time to perform the retrieval, the transmission, and the encryption operation is linear with the amount of data.

3. The time to retrieve the encrypted master key, is considerably shorter than the time to retrieve the article ciphertext; i.e., $LC \ll CT \Longrightarrow r(LC) \ll r(CT)$
4. The size of the ciphertext is proportional to and longer than the cleartext size.
5. The time to encrypt a piece of data is considerably longer than the time to retrieve and/or time to transmit the same piece of data.

Furthermore, we assume that the bandwidth of the retrieval operation is considerably larger, compared to the network transmission bandwidth. This last assumption may not be valid in some systems (e.g., Gigabit networks). However, it appears a reasonable assumption in the conventional networks.

Based on the above assumptions, we approximate the system S_1 and S_2 as single $M|G|1$ queue systems with access times given by

$$x_1 \approx t(LC + CT) \approx t(CT), \quad x_2 \approx e(A)\,. \tag{3}$$

Furthermore, because of the assumptions (4), (2) and (5), we can assert that $\overline{x_2} = k \cdot \overline{x_1}\ (k > 1)$.

Using the Pollaczek–Khinchin formula ([4], page 191), the waiting times in the queue S_1 and S_2, respectively, are

$$W_{S_1} = \frac{\lambda}{2}\frac{\overline{x_1}^2(1 + C_{x_1}^2)}{(1 - \rho_1)}$$

and

$$W_{S_2} = \frac{\lambda}{2}\frac{\overline{x_2}^2(1 + C_{x_2}^2)}{(1 - \rho_2)}\,, \tag{4}$$

where ρ_i is the utilization of queue i, and C_{x_i} is the coefficient of variation of the process x_i, $(i = 1, 2)$. (Coefficient of variation of a random process x is defined as $C_x = \frac{\sqrt{\sigma_x}}{\overline{x}}$.) Thus,

$$\frac{W_{S_2}}{W_{S_1}} = \frac{k(1 + C_{x_2}^2)(k - \rho_1)}{(1 + C_{x_1}^2)(1 - \rho_2)}\,. \tag{5}$$

Because of the assumption (4) above, the distribution of the ciphertext length has the same shape as the distribution of the cleartext length (though with different parameter). Furthermore, because of the assumption (2) above, the times to transmit the ciphertext and encrypt the cleartext are proportional to the size of the cipher and cleartext, respectively. Consequently, one concludes that the distributions of r.v. $t(CT)$ and $e(A)$ have the same shape and that $C_{x_1} = C_{x_2}$. Therefore,

$$\frac{W_{S_2}}{W_{S_1}} = \frac{k(k - \rho_1)}{(1 - \rho_2)}\,. \tag{6}$$

Thus the mean waiting time for the batch system is reduced by a factor on the order of k^2 compared to the mean waiting time for the real-time system.

The mean system time (T_S) is defined as the sum of mean waiting time (W_x) and the mean service time ($\overline{x}$); i.e.,

$$T_{S_1} = W_{S_1} + \overline{x_1}$$

and

$$T_{S_2} = W_{S_2} + \overline{x_2} = W_{S_2} + k\overline{x_1}\,. \tag{9}$$

Therefore,

$$\frac{T_{S_2}}{T_{S_1}} = \frac{W_{S_2} + k\overline{x_1}}{W_{S_1} + \overline{x_1}} \geq \frac{W_{S_2}}{W_{S_1}} = \frac{k(k - \rho_1)}{(1 - \rho_2)}\,. \tag{8}$$

Now, for example, let us assume that the average article is on the order of 5000 bytes, the (software-based) encryption speed is on the order of 100Kbps, the size of the ciphertext approximately equals the corresponding cleartext, the transmission links are T1 lines (≈1.5Mbps), and the memory access throughput is 10 Mbps (i.e., $r(CT) \ll t(CT)$). Further, assume that the S_2 system is working with utilization 0.5. Thus, $k = 15$ and the access time of the system S_2 is reduced by a factor of 435 – to about 0.02% of the access time of system S_1!

We conclude that there is a substantial gain in performance (improvement of access time) by encrypting the newspaper once off-line for all users as opposed to encrypting the newspaper for each user in real-time upon arrival of his request.

References

[1] M.J. Beller, L.F. Chang and Y. Yacobi, Privacy and authentication on a portable communications system, IEEE J. Select. Areas Comm. 11 (1993) 821–829.

[2] A. Choudhury, N. Maxemchuk, S. Paul and H. Schulzrinne, Copyright protection for electronic publishing over computer networks, IEEE Network (May 1995).

[3] J.T. Brassil, S. Low, N.F. Maxemchuk and L. O'Gorman, Electronic marking and identification techniques to discourage document copying, *Infocom '94*, Toronto, Canada, June 14–16, 1994, pp. 1278-1287.

[4] L. Kleinrock, *Queueing Systems: Volume 1: Theory* (Wiley, 1975).

Zygmunt Haas (ACM 94, IEEE S'84–M'88–SM'90) received his B.Sc. in EE from Technion in 1979 and M.Sc. in EE from Tel-Aviv University in 1985, both with "Summa Cum Laude." From 1979 till 1985 he worked for the Government of Israel. In 1988, he earned his Ph.D. from Stanford University researching fast packet-switched networks, and subsequently joined AT&T Bell Laboratories in NJ, where he is currently a Member of Technical Staff in the Wireless Architecture area.

Dr. Haas is an author of numerous technical papers and holds many patents in the field of mobile networks, wireless communication, high-speed networking, and optical switching. He served as a guest editor for two IEEE JSAC issues ('Gbps Protocols' and 'Mobile Computing Networks') and is on the editorial board of several other journals. His interest include: mobile and wireless communication, personal communication networks, and high-speed communication and protocols.

E-mail: haas@acm.org

Sanjoy Paul (ACM S 1989, ACM 92, Member IEEE 92) obtained his B. Tech (Hons.) from Indian Institute of Technology, Kharagpur, in electronics and electrical communications engineering (EECE) in 1985, followed by M.S and Ph.D in electrical engineering from the University of Maryland, College Park in 1988 and 1992 respectively. He worked as an Assistant Systems Engineer in India for CMC Limited during 1985-1986. He joined Distributed Systems Research Department at AT&T Bell Labs, Murray Hill, in 1992. Currently, he is with the Wireless Networking Research Department at Bell Labs, Holmdel. His technical interests include design, analysis, validation, verification, conformance testing and performance evaluation of communication protocols; multicasting security and privacy; and wireless and mobile networking.
E-mail: sanjoy@research.att.com