Secure Timeliness: On the Cost of Non-Synchronized Clocks

Raphael Yahalom The Hebrew University

ABSTRACT: The establishment of event timeliness in a distributed system by reliance on an assumption of closely synchronized clocks may introduce security flaws. We present a method for determining the message cost associated with the relaxation of such an assumption in protocols. The method is used to examine multi-domain authentication protocols and to establish that for these protocols the additional cost may be insignificant.

Raphael Yahalom was supported in part by grants from the Recanati and Schonbrunn Foundations at the Hebrew University, Jerusalem.

1. Introduction

It is often important that an entity involved in interactions with other remote entities in a distributed computing system be able to establish that certain events have happened relatively *recently*. Such a *timeliness* requirement implies that an entity is able to determine, in terms of its own local clock, the furthest point in the past in which some event at another location has actually occurred. A *secure* timeliness requirement is one in which such determination is possible even when malicious attackers may attempt to lead an entity to wrong timeliness conclusions.

Establishing the timeliness of an event may be important in cases in which the information corresponding to that event is valid only for a limited period, or in which such information should only be acted upon once. In the hostile environment we are considering, attackers may be able to block messages and then send them at arbitrary points in the future, re-play messages which were already received in the past, or in some cases, set certain clocks backwards or forwards.

Consider a **transfer $10000 from** $account_P$ **to** $account_Q$ request submitted by an authorized client from some local node to a bank server. An attacker may attempt to replay that request again at a later point hoping that the funds would be transferred again. He could alternatively block the original message and (possibly much) later, after that request was aborted by the initiator due to a *time-out*, send it to the server (who will then receive that request for the first time and may consider an aborted request as valid).

Solutions to such timeliness challenges can be classified into two categories: those which rely on an assumption that various clocks of different entities are closely synchronized (cf. [5, 16, 11]), and those which do not rely on such an assumption (cf. [14, 7, 15, 18]). Although the Needham-Schroeder protocol [14], which was the first to address related issues, is not based on a synchronized clocks assumption, most current real systems, such as Kerberos [16] or SPX [17], are. Some protocols, such as the CCITT data exchange protocol [4], provide the option to select between these two possibilities. It is generally accepted that closely synchronized clocks may reduce the number of messages required to achieve the specified goals of a distributed exchange [11].

However, an assumption that certain clocks are always closely synchronized may be considered questionable, in particular in widely-distributed, possibly open systems. For example, it is conceivable that clocks at certain sites may maliciously or incompetently be set wrongly, and so, at least temporarily, not be synchronized with other clocks in the system. This may imply potentially serious security flaws in protocols which rely on close clock synchronization for their correctness ([7, 1]).

In this paper we present an approach for computing the number of extra messages that are required in a protocol in which timeliness requirements need to be achieved without an assumption of closely-synchronized clocks. It turns out that in certain cases such extra message cost may be insignificant. It may thus be reasonable in such cases to design a protocol without a synchronized clocks assumption and thus reduce the potential security risks implied if such an assumption is in fact invalid.

In particular we examine a multi-domain authenticated exchange protocol. For that protocol we establish that with a closely synchronized clock assumption $n+3$ messages are required, whereas without that assumption only $n+4$ messages are required for achieving all the protocol goals. In other words, adding a single message to the $n+3$ required anyhow, suffices in order to attain all the secure timeliness goals of the protocol, without assuming close synchronization between any of the clocks in the system.

The rest of this paper is organized as follows. In the next section we introduce a model within which event timeliness, ways of establishing it, and the kinds of risks that may be associated with close clock synchronization assumptions, are described. In section 3 we present a general method for determining the number of extra messages that are associated with the relaxation of the synchronized clock assumption for a protocol. In section 4 we demonstrate the use of that method to examine multi-domain authenticated exchange protocols. Finally in section 5 we present some concluding remarks.

2. *The Timeliness of Events*

A hardware platform located at some physical location together with the local processes which it may be executing are considered *an entity*. A distributed system consists of multiple entities and communication links. Entities may communicate with each other by exchanging messages on these communication links.

Attackers may read, or modify, any part of any message flowing on a communication link, and similarly suppress any part of such a message or generate a new one. Attackers may thus also replay old messages.

An entity may share a secret with another one and use it as a key in some pre-agreed encryption scheme.[1] Such an encryption scheme is assumed to be associated with the following properties:[2]

Let $\{msg\}_K$ represent the result of encrypting msg with a key K. Then, with sufficiently high probability

- Only a holder of K may obtain msg from $\{msg\}_K$
- Only a holder of K may have initially generated $\{msg\}_K$.
- A holder of msg and $\{msg\}_K$ can not derive K.
- Any change to any bit in $\{msg\}_K$ may affect randomly all the bits of the decrypted message.

A protocol execution is a sequence of messages exchanged for the aim of achieving some goals. As part of such protocol executions, each entity may perform local *events*—take local actions (e.g. receive a message, send a message, provide cash, etc.).

Each entity may possess a clock which ticks at a certain rate—thus increasing its value. Clocks may be set to a value such that consequent ticks will increment that value. As we discuss below, each pair of clocks in the system may or may not be closely synchronized.

There are circumstances in which it is important to establish at some entity A that an event E_B that occurred at some other entity B has occurred relatively *recently*—that is, it has occurred no longer than Δ ticks ago as measured on A's clock. We refer to that as a requirement of A to establish the *timeliness* of E_B.

In particular, two reasons may motivate such a requirement:

1. The information associated with E_B may be considered by A as *stale* a certain period after it occurred. For example, if E_B is a request by B to A to purchase a specified number and type of shares, B may not wish A to go ahead and perform the transaction if it arrived long after it was sent.[3]
2. The information associated with E_B may be considered by A as *valid only once*. The actions at A which are implied by E_B are non-idempotent. In the shares-purchase example above, A could have received the request rel-

1. For the sake of consistency, throughout this paper we refer only to shared-key encryption schemes. The timeliness issues associated with public-key based protocols are analogous.
2. Whereas these are quite strong properties, we assume that a scheme such as DES provides a reasonable approximation of such properties.
3. Note that in the environment we assume, an attacker could have blocked the original request, and transmitted it to A, say weeks, later.

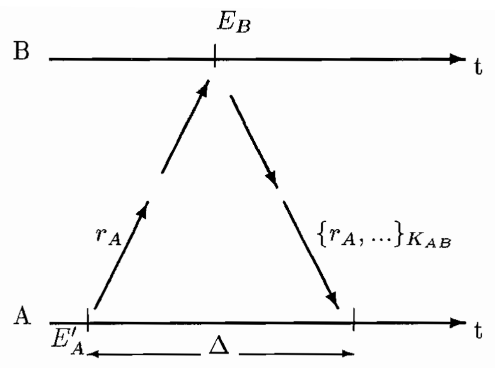

Figure 1. Asynchronous timeliness.

atively shortly after it was generated, but replayed for the second (or nth) time. It is common for A to try and detect such replays by maintaining some bounded state—for example corresponding to messages it received within some time window of length Δ'.

A's ability to establish the timeliness of E_B depends on whether it assumes B's clock to **always** be closely synchronized with its own ([5, 16, 11]) or not ([14, 7, 15, 18]).

Consider first the case in which A does not make such an assumption about clock synchronization. As was first suggested by Needham and Schroeder in [14] and generalized by the author in [18], a necessary condition for A to establish the timeliness of E_B at B is a *message flow* from A to B (which we denote A $\leadsto$ B) which includes a random value r_A (chosen from a sufficiently large space) generated by A, arrives at B before E_B is generated followed by a flow B $\leadsto$ A.[4] That non-synchronized clocks case is demonstrated in Figure 1.

If E_B is represented by the generation by B of $\{r_A, \ldots\}_{K_{AB}}$, the key is a secret key shared between A and B; A can establish that, with sufficiently high probability, E_B could not have occurred before E'_A and so has occurred not longer than Δ ticks, on its local clock, before it received the corresponding message.

Consider now the case in which A assumes that B's clock is always closely synchronized with its own. In particular, A assumes that whenever a clock reading ts_B is made in B, its own clock value is ts_A and $ts_B - \delta_{AB} \leq ts_A \leq ts_B + \delta_{AB}$. In such a case the flow A $\leadsto$ B is not required, and the flow B $\leadsto$ A simply needs to contain the value ts_B, representing B's clock value when E_B was generated.

4. The notion of a *nonce* introduced by Needham and Schroeder in [14] is in fact more restrictive. A nonce is a value which was never used by A before (such as a counter value). Consequently, such a nonce may only allow A to establish *valid only once* aspect of E_B but not its *staleness* aspect.

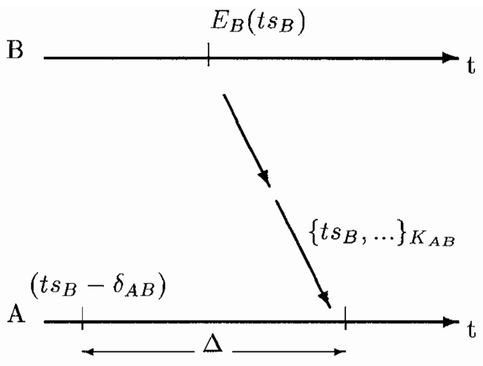

Figure 2. Synchronous timeliness.

That case is represented in Figure 2.

If A received that message at ts'_A it can conclude that E_B has occurred at most $ts'_A - (ts_B - \delta_{AB})$ ticks ago on its local clock.

An assumption about clocks being always closely synchronized may in many cases be considered questionable (e.g. [15, 1, 7]). Not only is the problem of synchronizing clocks across multiple sites in distributed environments considered to be a potentially non-trivial challenge, but in the environment we assume attackers may maliciously set certain clocks backward or forward.[5] Consider for example the attack described by Gong in [7], demonstrated in Figure 3.

Assume that at some point in the past B's clock was temporarily set forward by an attacker to a value greater by δ_B than its previous (correct) value. Consequently, at ts'_B B may have unwittingly generated a message which includes the value ts_B. A should now conclude that the message was generated at B at most $ts'_A - (ts_B - \delta_{AB})$ ticks ago on its local clock, whereas in fact the message was generated $ts'_A - (ts_B - \delta_{AB} - \delta_B)$ ticks ago on its local clock.

We may thus conclude that, ignoring overheads associated with conventional clock synchronization algorithms, certain potential attacks associated with clock values may be eliminated at the cost of additional messages—the additional A $\rightsquigarrow$ B message flow.

In the next sections we demonstrate that in some cases that additional cost may not be significant.

5. For example, an attacker may set the clock in a workstation before a valid user starts her session.

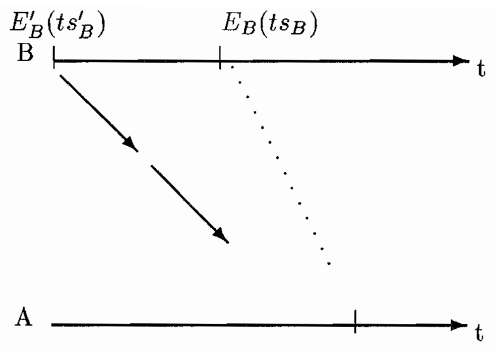

Figure 3. A clock-forwarding attack.

3. *Determining the Additional Cost*

In this section we present a method for determining the additional cost, in terms of extra messages, associated with protocols which do not require an assumption of synchronized clocks. In the next section we demonstrate the approach in the context of authentication protocols.

Such an approach is targeted at protocols' design and analysis. One of the important steps in such phases is the determination of whether to rely on closely-synchronized clocks. Our results lead us to suggest that for certain protocols the potential risks associated with such a reliance may outweigh the performance gains.

Essentially, we compare the optimal flow of messages in a protocol in which all entities are assumed to possess closely-synchronized clocks, with a derived protocol in which the clocks are not synchronized. It turns out that in various cases the required extra flow A $\rightsquigarrow$ B, or parts of it, can be combined with flows which are inherently part of the protocol. Note that, because of protocol or environment characteristics (e.g. encrypted messages need to be sent via a path of entities which share keys), a flow A $\rightsquigarrow$ B may include more than a single message.

The method consists of the following three steps:

1. Determine which events need to be established as *timely* by which entities. In particular, construct a set S in which each element is a pair (E_j, E_i) of events at entities P_j and P_i respectively. The meaning of such a pair is that at the point when E_i occurs, entity P_i needs to be able to establish that E_j is timely (i.e. that it has occurred within the last Δ ticks, for some Δ).

2. Determine the optimal message flow of the protocol, starting with an initiation entity A, assuming that all clocks are closely synchronized.

3. Let C_{ij} denote the number of messages that need to be added to a protocol's flow so that there is a message flow $P_i \leadsto P_j$ for each pair (E_j, E_i) in S such that the last message of the flow arrives to P_j before E_j. Consistently with flows implied by the protocol and environment specifications, compute the minimal $\sum_{S} C_{ij}$.

That minimal sum over all the pairs in S is the additional message cost associated with the relaxation of the synchronized clock assumption.

4. Multi-Domain Authentication Protocols

In this section we apply the method presented in the previous section in the context of a multi-domain shared-key authentication and data-exchange protocol.

Let a *shared key path* be a sequence of entities such that each entity shares a long term secret with its successor and predecessor. Consider an environment with two entities A and B and in which the shortest shared key path between them, $A, AS_A, AS_2, ...AS_{n-1}, AS_B, B$ consists of n intermediate entities (authentication servers). The identities of the entities on that shortest shared key path can be predetermined, for example based on some global hierarchical structure ([2, 10, 6, 20]). The entities A and B consider all the entities in that path to be trustworthy with respect to performing the algorithm correctly, and, in addition, they both consider AS_B to be trustworthy with respect to the generation of good quality temporary session keys [20].

Informally, the protocol requires that both A and B receive a timely secret session key which is generated by AS_B and that a single secure timely request-response exchange between A and B be performed using that new session key.[6]

The above specifications imply the following message flows:

$A \leadsto AS_B$	AS_B would only generate a key after it receives information from the initiator, A.
$AS_B \leadsto A$	A needs to obtain the session key generated by AS_B. That flow is constrained by secrecy requirements—the generated key should not be obtainable by non-trusted entities (e.g. ones which are not on the trusted shared key path).

6. Variants of this protocol, for example, ones in which an entity other than AS_B generates the session key [19], or in which the session itself has a different structure than a simple request-response, may similarly be considered.

$AS_B \leadsto B$	Similarly, B needs to obtain the session key generated by AS_B.
$A \leadsto B$	B needs to obtain the data-request which A generates. The secrecy and integrity of that request need to be protected by the new session key.
$B \leadsto A$	Similarly, A needs to obtain the data-response which B generates.

The following timeliness requirements are associated with such a protocol:

- By the time it sends its data-request message (event $\textit{send-data}_A$) A needs to have established that a key generation event at AS_B (event $\textit{key-generation}_{AS_B}$) is timely.
- By the time it sends its data-response message (event $\textit{send-res}_B$) B needs to have established that a key generation event at AS_B (event $\textit{key-generation}_{AS_B}$) is timely.
- By the time it sends its data-response message (event $\textit{send-res}_B$) B needs to have established that event $\textit{send-data}_A$ is timely.
- By the time it receives the data-response message (event $\textit{receive-res}_A$) A needs to have established that event $\textit{send-res}_B$ is timely.

Consequently, the set S contains the following 4 pairs:

$$\begin{array}{l} \{ \ (\textit{key-generation}_{AS_B} \ , \ \textit{send-data}_A \ ) \\ \quad (\textit{key-generation}_{AS_B} \ , \ \textit{send-res}_B \ ) \\ \quad (\textit{send-data}_A \ , \ \textit{send-res}_B \ ) \\ \quad (\textit{send-res}_B \ , \ \textit{receive-res}_A \ ) \ \} \end{array}$$

Let ts denote a timestamp, S_n denote a unique session identifier generated by the initiator A, and $K_{AS_iAS_j}$ denote a secret key shared between entities AS_i and AS_j (in particular K_{AB} is the new session key generated by AS_B for A and B). $\{msg\}_K$ denotes the encryption of msg with a key K, and $AS_i \rightarrow AS_j : msg$, denotes a message msg sent from AS_i to AS_j. The following protocol, which relies on closely synchronized clocks, achieves the specified goals[7] :

7. We assume that shared-key path information is globally available to all principals, so that, for example, each principal can unambiguously determine the next one in the flow below. Such an assumption is justified in cases in which the principals are organized in some global static structure (e.g. a hierarchy) which also reflects their trust relations. That assumption may be relaxed, cf. [20], by incorporating additional path related information into the protocol messages.

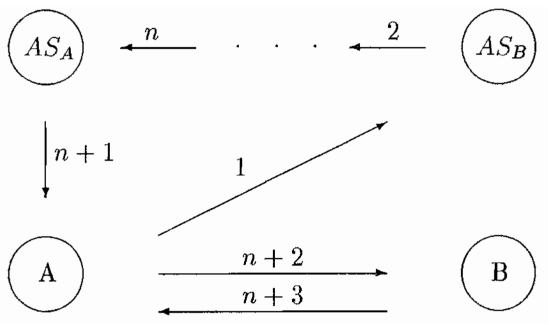

Figure 4. A synchronous authentication protocol.

1.	$A \rightarrow AS_B:$	S_n, AS_B, A, B
2.	$AS_B \rightarrow AS_{n-1}:$	$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_B AS_{n-1}}}$
		$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_B B}}$
3.	$AS_{n-1} \rightarrow AS_{n-2}:$	$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_{n-1} AS_{n-2}}}$
		$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_B B}}$
....		
$n.$	$AS_2 \rightarrow AS_A:$	$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_2 AS_A}}$
		$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_B B}}$
$n+1.$	$AS_A \rightarrow A:$	$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_A A}}$
		$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_B B}}$
$n+2.$	$A \rightarrow B:$	$\{S_n, A, B, ts_A, request\}_{K_{AB}}$
		$\{S_n, AS_B, ts, A, B, K_{AB}\}_{K_{AS_B B}}$
$n+3.$	$B \rightarrow A:$	$\{S_n, B, A, ts_B, response\}_{K_{AB}}$

The message flow of that protocol is described in Figure 4.

The optimality of that protocol follows from the fact that the $AS_B \rightsquigarrow A$ flow, considering its secrecy constraints, cannot contain less than n messages (because of the assumption regarding the shortest shared key path), and the fact that the flow $AS_B \rightsquigarrow B$ was in fact piggybacked on top of the $AS_B \rightsquigarrow A$ and $A \rightsquigarrow B$ flows. The other flows may not be piggybacked and consist of only a single message each.

The fact that timeliness requirements are attained follows from the included timestamp fields and the synchronized clocks assumption.

We now consider the flows that need to be added to that protocol in an environment in which close clock synchronization is not assumed. Consider the timeliness requirements as represented by the pairs of events in set S:

- The first pair implies a flow $A \rightsquigarrow AS_B$. That flow is already included in the protocol, because there is a requirement that A, the initiator, will notify AS_B of the need to generate a session key.

- The second pair implies a flow $B \rightsquigarrow AS_B$. The shortest such additional flow (1 message) can be achieved by the two messages $A \rightarrow B$ followed by $B \rightarrow AS_B$, replacing the $A \rightarrow AS_B$ message. Note that these 2 messages satisfy both the flows implied by the first 2 pairs in $\mathcal{S}$.

- The third pair implies a flow $B \rightsquigarrow A$. That flow is already included by the message $B \rightarrow AS_B$ (added above) followed by the flow $AS_B \rightsquigarrow A$ (required for delivering the key to A).

- The fourth pair implies a flow $A \rightsquigarrow B$. Again, that flow is already included: A is required to transfer the data-request to B.

Thus, the minimal $\sum_{S} C_{ij}$ is equal to 1 (0+1+0+0). That corresponds to the fact that it costs only a single message to relax the assumption of closely synchronized clocks for such a multi-domain authentication protocol. A cost of $n+4$ messages instead of $n+3$ messages seems to potentially be insignificant.

The derived protocol is the following one. r_A and r'_A are random numbers, from a sufficiently large space, generated by A, and r_B a random number generated by B.

1.	$A \rightarrow B:$	S_n, r_A, A, B
2.	$B \rightarrow AS_B:$	S_n, r_A, r_B, A, B
3.	$AS_B \rightarrow AS_{n-1}:$	$\{S_n, r_A, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_B AS_{n-1}}}$
		$\{S_n, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_B B}}$
4.	$AS_{n-1} \rightarrow AS_{n-2}:$	$\{S_n, r_A, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_{n-1} AS_{n-2}}}$
		$\{S_n, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_B B}}$
....		
$n+1.$	$AS_2 \rightarrow AS_A:$	$\{S_n, r_A, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_2 AS_A}}$
		$\{S_n, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_B B}}$
$n+2.$	$AS_A \rightarrow A:$	$\{S_n, r_A, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_A A}}$
		$\{S_n, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_B B}}$
$n+3.$	$A \rightarrow B:$	$\{S_n, r'_A, r_B, A, B, request\}_{K_{AB}}$
		$\{S_n, r_B, AS_B, A, B, K_{AB}\}_{K_{AS_B B}}$
$n+4.$	$B \rightarrow A:$	$\{S_n, r'_A, B, A, response\}_{K_{AB}}$

The message flow of that protocol is described in Figure 5.

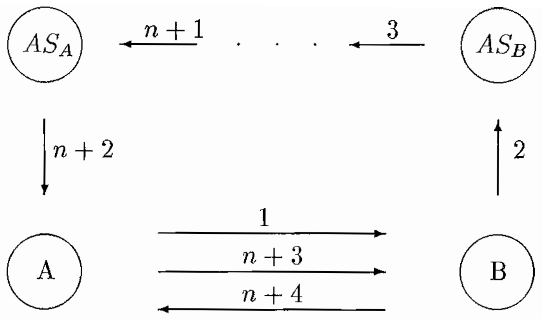

Figure 5. An asynchronous authentication protocol.

4.1. Subsequent Authenticated Exchanges

We now consider cases in which A and B may be involved in exchanges based on a session key K_{AB} which they obtained earlier and which each of them considers as still valid.[8]

Again, consider each such exchange as consisting of a request from A and a response from B. We first refer to the case in which A needs to establish the timeliness of the response, but B does not wish to establish the timeliness of the request. That corresponds to a scenario in which the nature of the request is such that B may consider it valid, and act upon it, even if it may be old, and B also considers such an action to be idempotent—the question of whether that request has already been processed in the past is considered irrelevant. For example, a data query request from a client to a server may have such characteristics.

The set S includes only one pair:

$\{\ (send\text{-}res_B, receive\text{-}res_A)\ \}$

Assuming close clock synchronization the optimal protocol is the following:

1. $A \rightarrow B : \{A, B, request\}_{K_{AB}}$
2. $B \rightarrow A : \{B, A, ts_B, response\}_{K_{AB}}$

Relaxing the close clock synchronization assumption and considering the set S we conclude that a flow $A \rightsquigarrow B$ is required before B generates its response. However such a flow is already part of the protocol. Consequently the asynchronous version of the protocol requires no additional messages:

8. Kehne, Schonwalder, and Langendorfer [9] and Neuman and Stubblebine [15] have considered subsequent exchanges in which only A stores the session key as well as a ticket generated by B. That ticket includes that session key (along with some of its parameters) and A forwards it to B when required. Our analysis applies to such variations as well.

1. $A \rightarrow B : \{A, B, r''_A, request\}_{K_{AB}}$
2. $B \rightarrow A : \{B, A, r''_A, response\}_{K_{AB}}$

We now consider the case in which in addition to A establishing the timeliness of the response, B is required to establish the timeliness of the request. That for example corresponds to a case in which the request from A represents a shares purchase order.

The set $\mathcal{S}$ now contains two pairs:

$$\{ \ (\, send\text{-}req_A \, , \ send\text{-}res_B \,) \\ (\, send\text{-}res_B \, , \ receive\text{-}res_A \,) \ \}$$

The closely synchronized clocks version of the protocol still contains two messages:

1. $A \rightarrow B : \{A, B, ts_A, request\}_{K_{AB}}$
2. $B \rightarrow A : \{B, A, ts_B, response\}_{K_{AB}}$

The additional pair in the set $\mathcal{S}$ implies a $B \rightsquigarrow A$ flow which precedes the request from A to B. Such a flow needs to be triggered by the initiator A and so an additional $A \rightsquigarrow B$ flow is required. Consequently 2 additional messages are required for the version of the protocol in which clocks are not closely synchronized, and we obtain the following protocol:

1. $A \rightarrow B : A, B$
2. $B \rightarrow A : B, A, r''_B$
3. $A \rightarrow B : \{A, B, r''_A, r''_B, request\}_{K_{AB}}$
4. $B \rightarrow A : \{B, A, r''_A, response\}_{K_{AB}}$

5. *Conclusions*

A few researchers have pointed out the potential risks associated with protocols like Kerberos [16] which rely on a closely synchronized clocks assumption for establishing timeliness (cf. [7, 1]). Kehne et al. [9] and Neuman and Stubblebine [15] have recently proposed protocols which would achieve the goals of Kerberos but would not require a synchronized clock assumption.[9]

9. The Kehne-Schonwalder-Langendorfer protocol in [9] is sub-optimal as demonstrated in [18]. The Neuman-Stubblebine protocol, based on a protocol presented by the author in [18], is similar to a single domain special case version of the protocol presented in section 4 and is indeed optimal.

The contribution of this paper is to present a systematic method for determining the message cost associated with a relaxation of the synchronized clocks assumption and to demonstrate that there are important cases in which such additional cost may be considered insignificant. Even in the cases in which such additional cost is significant the trade-off between potentially enhanced security and degraded performance needs to be carefully evaluated.

Further work needs to proceed in various directions:

- Applying the method to other important types of protocols.
- Considering cases in which only certain clocks are assumed to be closely synchronized while others are not.
- Extending protocol analysis methods, for example those of Burrows, Abadi, and Needham [3], Gong, Needham and Yahalom [8], Meadows [12], Millen, Clark, and Freedman [13], and others, to capture notions such as clock synchronization and timeliness in the sense we have presented them here.
- Examining systematically general security-performance trade-offs in distributed protocols.

Acknowledgements

I thank the anonymous referees for helpful comments and suggestions.

References

1. S. M. Bellovin and M. Merritt, “Limitations of the Kerberos Authentication System,” *Computer Communication Review*, Vol. 20, No. 5, October 1990, pp. 119–132.
2. A. Birrell, B. Lampson, R. Needham, M. Schroeder: “A Global Authentication Service Without Global Trust,” *Proceedings of the IEEE Conference on Security and Privacy*, 1986, pp. 223–230.
3. M. Burrows, M. Abadi, and R. Needham, “A Logic of Authentication,” in *Proceedings of the 12th ACM Symposium on Operating Systems Principles*, Litchfield Park, Arizona, December 1989, pp.1-13. Published as *ACM Operating System Review*, Vol. 23, No. 5, December 1989. A fuller version was published as DEC Systems Research Center Report No. 39, Palo Alto, California, February 1989.

4. CCITT Draft Recommendation X.509 "The Directory-Authentication Framework," version 7, Gloucester, November 1987.

5. D. E. Denning and G. M. Sacco, "Timestamps in Key Distribution Protocols," *Comm of the ACM*, Vol. 24, No. 8, August 1981, pp. 533–536.

6. V. D. Gligor, S.-W. Luan, J. N. Pato: "On Inter-realm Authentication in Large Distributed Systems," in *Proceedings of the IEEE Conference on Security and Privacy 1992*, pp. 2–17.

7. L. Gong, "A Security Risk of Depending on Synchronized Clocks," *Operating Systems Review*, Vol. 26, No. 1, 1992, pp. 49–53.

8. L. Gong, R. Needham, and R. Yahalom, "Reasoning about Belief in Cryptographic Protocols," in *Proceedings of the 1990 IEEE Symposium on Security and Privacy*, Oakland, California, May 1990, pp. 234–248.

9. A. Kehne, J. Schonwalder, H. Langendorfer, "A Nonce-Based Protocol for Multiple Authentications," *Operating Systems Review*, Vol. 26, No. 4, October 1992, pp. 84-89.

10. B. Lampson, M. Abadi, M. Burrows, E. Wobber: "Authentication in Distributed Systems: Theory and Practice," *The 13th ACM Symposium on Operating Systems Principles*, October 1991.

11. B. Liskov, "Practical Uses of Synchronized Clocks in Distributed Systems," in *Proceedings of the 1991 ACM Symposium on Principles of Distributed Computing*, Montreal, Quebec, Canada, August 1991, pp. 1–9.

12. C. Meadows, "Applying Formal Methods to the Analysis of Key Management Protocols," *Journal of Computer Security*, 1, 1992, pp. 5–53.

13. J. K. Millen, S. C. Clark, and S. B. Freedman, "The Interrogator: Protocol Security Analysis," *IEEE Trans on Software Eng* SE-13, 1987, pp. 274–288.

14. R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, Vol. 21, No. 12, December 1978, pp. 993–999.

15. B. C. Neuman and S. G. Stubblebine, "A Note on the Use of Timestamps as Nonces," *Operating Systems Review*, Vol. 27, No. 2, April 1993, pp. 10–14.

16. J. Steiner, C. Neuman, and J. I. Schiller, "Kerberos: An Authentication Service for Open Network Systems," *Proc. Winter Usenix* Dallas, 1988.

17. J. J. Tardo and K. Alagappan "SPX: Global Authentication Using Public Key Certificates," *Proc. of the IEEE Symposium on Security and Privacy*, Oakland, California, 1991.

18. R. Yahalom, "Optimality of Asynchronous 2-Party Secure Data-Exchange Protocols," *Journal of Computer Security*, Vol. 2, No. 2, 1993, pp. 191–209.

19. R. Yahalom, "Optimality of Multi-Domain Protocols," *Proc. of the 1st ACM Conference on Computer and Communication Security*, Fairfax, Virginia, November 1993, pp. 38–48.

20. R. Yahalom, B. Klein, Th. Beth: "Trust Relationships in Secure Systems—A Distributed Authentication Perspective," *Proceedings of the IEEE Conference on Research in Security and Privacy*, 1993, pp. 150–164.