Authentication

Beacon Based Authentication

Azad Jiwa Jennifer Seberry (FIMA) Yuliang Zheng

Centre for Computer Security Research,
University of Wollongong.
Wollongong, NSW 2522, Australia.
Telephone: +61 42 21 4327 Fax: +61 42 21 4329
Email: azad/j.seberry/y.zheng/@uow.edu.au

Abstract.
Reliable authentication of communicating entities is essential for achieving security in a distributed computing environment. The design of such systems as Kerberos, SPX and more recently KryptoKnight and Kuperee, have largely been successful in addressing the problem. The common element with these implementations is the need for a trusted third-party authentication service. This essentially requires a great deal of trust to be invested in the authentication server which adds a level of complexity and reduces system flexibility.
The use of a Beacon to promote trust between communicating parties was first suggested by M. Rabin in "Transactions protected by beacons," *Journal of Computer and System Sciences* , Vol 27, pp 256-267, 1983. In this paper we revive Rabin's ideas which have been largely overlooked in the past decade. In particular we present a novel approach to the authentication problem based on a service called *Beacon* which continuously broadcasts certified nonces. We argue that this approach considerably simplifies the solution to the authentication problem and we illustrate the impact of such a service by "Beaconizing" the well know Needham and Schroeder protocol. The modified protocol would be suitable for deployment at upper layers of the communication stack.

Term Index: Beacon, Authentication, Network Security, Information Security, Security Protocol.

1 Introduction

In the past thirty years the rapid evolution of electronic data communication has been breath taking. This growth and reliance on electronic communication has increased concern about data privacy and integrity. Much effort is currently being devoted to providing security services for a variety of communication environments. Authentication is universally acknowledged as being essential for secure communications in a distributed system.

Informally, authentication is the capability of the recipient of a communication to be able to verify that the message did come from the alleged sender. As a further requirement, it is usual to expect such systems to perform verification over an open insecure network. This requirement constrains the authentication system in the following manner:

- Reliance must not be placed in the physical security of all hosts in the distributed system.
- It should be assumed that packets traveling along the network can be read, modified and inserted with little effort by an adversary.
- Trust cannot be based on the senders' address.
- Trust cannot be based on the senders' operating system security.

Currently, the dominant authentication protocols employed in a client-server environment use a broker to arbitrate between principals who wish to communicate. In such a scheme, a principal must first contact the broker to obtain credentials which can be trusted by its communicating partner. The protocol will typically rely on an exchange of cryptographic messages and the participant's knowledge of some secret. Well known examples of such systems are Kerberos [16, 7, 2], SPX [17], KryptoKnight [8] and Kuperee [6].

In this paper we present a novel approach to the authentication problem for a distributed computer system using a public key cryptographic system and we introduce a new service called *Beacon* which has been inspired by Rabin [12]. A Beacon broadcasts, at regular intervals, certified nonces which are accessible to all hosts within the network. In section 3 we give a brief description of Rabin's use of Beacons. In section 4 the approach taken in this paper is described.

As a means of contrasting the Beacon based approach to authentication with the more established authentication protocols, in section 5 we "beaconize" the very well known Needham and Schroeder public key protocol. We assume that the reader is familiar with the basic philosophy of Needham and Schroeder. For a complete description of the protocol we would direct the reader to [9].

In section 6 we discuss the advantages of the beaconized approach. The paper finally closes in section 7 with some concluding remarks.

2 Conventions

Throughout this paper certain terms are used which may appear to be ambiguous or new to readers. In the following section the use of these terms, in the context of this document, are stated. This should avoid any difficulty caused by their being used differently elsewhere.

2.1 Terminology

A *host* is a computer with a unique address which is connected to a computer network.

The *user* is the human being who is using some services provided by the network. All programs in the system are initiated by users. The main purpose of authentication is to verify the identity of the users within a network.

A *client* is a program which runs on behalf of a user in order to request some service that is available on a remote host. Since the client runs on a user's behalf, it assumes the user identity and privileges. Any action taken by a client is said to have been carried out by the user.

The *server* is a program which provides a service to a client on the network. It is usually installed by the system administrator and runs on behalf of the system. A server remains active in a network for much longer than a client and thus is given an identity. Each server is therefore registered with the system, in a similar manner to a user, and must prove its identity before a client will accept it.

Any communicating network entity, that is client or server, can be referred to as a *principal.*

A *realm* is organized in a hierarchical structure and has a strong analogy to *domains* in Internet. A single authentication server, be it a Beacon or Third-party Authenticator, is responsible for all local principals. This collection of network entities, the authentication server and its clients, are referred to as a realm. A client who wishes to obtain some service from a server within the same realm would use the service to prove its identity.

2.2 Cryptographic Requirements

The Beacon based system relies on a public key (or asymmetric) cryptographic system. Many such systems have been designed and examples of such schemes can be found in [19, 15]. For the purpose of our discussion the system's cryptographic requirements can be simplified and kept generic.

A public key system differs from a secret key (or symmetric) system in that each principal in the distributed system has a pair of keys. The calculation of this private and public key pair, given the initial conditions, is easy. However, it would be infeasible for an adversary who knows a public key to calculate the secret key or the initial conditions. Each principal has a private key which is kept secret and is only known to that principal. The corresponding public key is made available to all other principals. In the remainder of this paper "$\{\}_Z$" means encipher with key Z. The private and public key pair will be denoted as "X" and "Y" respectively and subscripts will be used to identify the associated principal. Thus the Beacon would have a key pair X_{BN},Y_{BN}, the principal, Alice, would have the pair X_A, Y_A while Bob would have X_B, Y_B.

The advantage of a public key system is that it supports secrecy, authentication, and integrity. Communication secrecy is supported by the transformations:

$$M = \{\{M\}_{Y_B}\}_{X_B}$$

That is, suppose Alice wishes to send a secret message, M, to Bob. Then Alice must have access to Bob's public key and encipher the message, thus:

$$C = \{M\}_{Y_B}$$

Alice sends Bob the cryptographic string C. On receipt Bob is able to employ his private key to decipher the message.

$$\{C\}_{X_B} = \{\{M\}_{Y_B}\}_{X_B} = M$$

The encryption and decryption processes are easy using the appropriate keys. However, it would be infeasible for an adversary to decipher C without the private key X_B, ensuring secrecy. Now, since Y_B is publicly known, Bob has no way of being certain of the sender's identity. Thus authenticity has not been assured using this method.

Authentication, using a public key system, is satisfied by the following transformation.

$$M = \{\{M\}_{X_A}\}_{Y_A}$$

Alice is able to "sign" her message to Bob by using her private key, X_A:

$$C = \{M\}_{X_A}$$

Bob is able to verify that the message could have only come from Alice by deciphering the message using Alice's public key, Y_A, thus:

$$\{C\}_{Y_A} = \{\{M\}_{X_A}\}_{Y_A} = M$$

If the message is plain-text, Bob knows that C has in fact not been altered. However, if the message, or any portion of the message, is a random string then it may be difficult for Bob to ascertain that the message has not been altered merely by examining it. For this reason it is more usual for Alice to employ a suitable one-way hashing function (eg. [14, 18]) to produce a Message Digest (MD). Alice would sign the MD with her private key and append it to the message. On receiving the message, Bob is able to reproduce the MD in order to confirm that the message is from Alice and that it has not been altered. Finally, all three can be employed by Alice to communicate securely with Bob thus:

$$C = \{M, \{MD\}_{X_A}\}_{Y_B}$$

3 Rabin's Approach Using Beacons

The use of a Beacon as a security service within a distributed computer system was first suggested by M. Rabin [12]. He defined a Beacon as emitting, at regular intervals, a random integer sealed using a suitable cryptographic signature system. The integer would be selected randomly and uniformly within the range 1 to N, where N is publicly known. With this basic concept he proposed two protocols based on a probabilistic approach, one for the signing of contracts and the second for information disclosure. In the following section Rabin's use of Beacons is illustrated by briefly outlining the contract signing protocol. For a full account, the reader is referred to the original paper.

Consider, Alice and Bob have negotiated a contract over a computer network and are ready to exchange signed copies. It is not possible for them to meet and they wish to use the network to make the exchange electronically. We can assume that the contracts can be signed by a suitable electronic signature scheme. The problem addressed by Rabin was, if Alice signs and sends the contract to Bob, then Alice is committed, but Bob is not. Bob could take the opportunity to look for a better deal, leaving Alice vulnerable. A similar argument can be made about Bob.

In Rabin's protocol we can assume the following initial conditions:

- Each of the participants has an asymmetric cryptographic key pair which can be used for signing messages and verifying signatures. The Beacon has the private and public key pair X_{BN},Y_{BN}, Alice's keys are X_A,Y_A and Bob's are X_B,Y_B. The public keys, Y_{BN}, Y_A and Y_B are widely known and are available to the participants. Each participant's private key has been kept secret and is known only to its owner.
- The Beacon broadcasts a token, T every Δ seconds and the next broadcast will be at time $t + \Delta$. The T has the following form:

  $$T = \{t, i\}_{X_{BN}}$$

  where t is the time at which the token was emitted and i is a randomly selected integer between 1 and N. The value of N is publicly known. The Beacon seals the token using its private key X_{BN}.

- Alice and Bob have agreed to the contract name, C, such that:

  $$C = h(contract)$$

  where a suitable hash function h is applied to the text of the contract and the obtained result is used to denote the contract.

- Alice and Bob have agreed and signed preliminary agreements. We will refer to these agreements signed by Alice and Bob as P_A and P_B respectively. The text of P_A is as follows:
  {*If Bob can produce (C,T) signed by Alice and T signed by the Beacon for some token T, then I, Alice will be committed to the contract as of the time t mentioned in the token T.* $\}_{X_A}$

  Bob signs a similar contract with the name Alice and Bob exchanged. Since this agreement does not bind either party to the contract, it reasonable to assume that this initial exchange will pass off without incident.

The protocol for the exchange of contracts requires that both parties follow a timed sequence. The exchange is aborted if one party fails to transmit the specified message within the allocated time. The exchange takes place in the time between Beacon broadcasts. This time Δ is divided in six equal segment. The protocol is as follows:

1.	$t \geq time \leq t + \frac{1}{6}\Delta$	Alice $\Rightarrow$ Bob: P_A
2.		Bob $\Rightarrow$ Alice: P_B
3.	$t + \frac{1}{6}\Delta \geq time \leq t + \frac{1}{3}\Delta$	Alice $\Rightarrow$ Bob: i_A
4.	$t + \frac{1}{3}\Delta \geq time \leq t + \frac{1}{2}\Delta$	Bob $\Rightarrow$ Alice: i_B
5.		$i = i_A + i_B mod N$
6.	$t + \frac{1}{2}\Delta \geq time \leq t + \frac{2}{3}\Delta$	Alice $\Rightarrow$ Bob: $\{C, i, t + \Delta\}_{X_A}$
7.		Bob $\Rightarrow$ Alice: $\{C, i, t + \Delta\}_{X_B}$

Within the first segment (steps 1 and 2) the two parties are required to exchange the preliminary contract. In the next two segments both Alice and Bob exchange random numbers which each has generated independently, that is i_A and i_B. Each party is then able to calculate i. Finally, (steps 6 and 7) the two parties sign and exchange the messages which could bind them to the contract. These messages contain the contract name, C, and possibly the next token. If neither party has cheated and the next token emitted by the Beacon contains i, then both parties are committed to the contract at the same time. If the next emitted token does not contain i, then neither party is committed and steps 3 to 7 have to be repeated. Rabin showed that in such an exchange there is a probability of $1/N$ that one of the parties could cheat successfully.

4 Beacons

Rabin's novel ideas on the use of beacons have been largely overlooked by the research community during the past decade. We revive these ideas by transferring them to the authentication problem. The following is a more detailed description of the concept, feasibility and implementation of a Beacon.

A Beacon, within the context of this paper, is a service which is provided by a secure host in a computer network. The Beacon broadcasts, at regular intervals, a nonce encapsulated within a certified token. The emitted token would be accessible to all hosts on the network and each host maintains a short list of fresh tokens. The additional load caused by this service would be small as each host is only required to listen for a short and relatively infrequent message.

4.1 Token

The token has the following form:

$$N_i, time, life, \{MD\}_{X_{BN}}$$

where:

N_i → is a freshly generated nonce.
$time$ → is the time at which the token was emitted.
$life$ → is the time after which the token will not be valid.
MD → is the message digest.
X_{BN} → is the Beacon's secret encryption key which is used to certify the token.

Each host which receives the token is able to verify its validity by decrypting the MD using the Beacon's public key. The MD ensures that the token has not been tampered with. Since the token is signed with the Beacon's secret key it is reasonable to assume the token originated from the Beacon. Each host is able to maintain, on behalf of its principals, a short list of currently valid tokens. Thus these tokens are available to all principals to use in the authentication process.

4.2 Network Synchronization

There is reliance within a Beacon based system that each principal has access to a stable clock and that these clocks are to some extent synchronized. Since the life of a token can be relatively long, say an hour, differences of a few seconds between the hosts can be tolerated. In this section we examine the Internet's Network Time Protocol (NTP) to show that in fact it is feasible to have much closer synchronization between communication hosts.

Any attempt to synchronize communicating entities requires access to an accurate standard. Since 1972 the time standard for the world has been based on International Atomic Time which is currently maintained to an accuracy of a few parts in 10^{12} [1]. Many countries operate standard time and frequency broadcast stations which collectively cover most areas of the world.

The network time protocol (NTP) is an Internet standard protocol [13] which is used to maintain a network of time servers, accessible over normal Internet paths. Even though transmission delays over Internet can vary widely, due to fluctuations in traffic loads and dynamic message routing, NTP acts to provide global synchronization. NTP is built on Internet's User Datagram Protocol (UDP) [11] which provides a connectionless transport mechanism.

The NTP system consists of a network of primary and an estimated total of over 2000 secondary time servers. Primary time servers are directly synchronized by reference source, usually a timecode receiver, or a calibrated atomic clock. Secondary time servers are synchronized by either a primary server or another secondary time servers. Due to the wide dispersal of these servers, access is available using some thousands of routes over hundreds of networks, making the system very reliable.

In a typical configuration used at the University of Illinois and the University of Delaware, the institutions operate three campus servers. These servers are synchronized using two primary servers and each other. The three campus servers in turn provide synchronization for department servers which then deliver time to remaining hosts. In such a configuration, several hundred synchronization milliseconds-seconds would not be uncommon.

4.3 Creating a Beacon

As stated above, a Beacon is a service which, at regular intervals, emits a token which can be authenticated. The emitted token must be accessible to all hosts on the network and each host is required to maintain a short list of fresh tokens. Since the broadcast is short and relatively infrequent, implementation is quite feasible in either software or hardware. Algorithm 1 shows the functionality.

```
Algorithm 1 Beacon()
        1. $t = clock + \Delta$
        2. $N_i = G()$
        3. $MD_i = h(t, life, N_i)$
        4. $T = t, life, N_i, \{MD_i\}_{X_{BN}}$
        5. while $(clock < t)$ wait
        6. broadcast($T$)
        7. goto(1)

end
```

The algorithm begins by setting, t, the time for the next broadcast. Next, the token is constructed prior to broadcasting (steps 2 to 4). The cryptographically strong pseudo-random generator, $G()$, is used to create a nonce N_i. The final component required to create the token is the message digest (MD). The one-way hash function, h, is employed to compress the bit string created by the concatenation of the broadcast time, t, the token life, l and the nonce N_i. The output, which is of a fixed length, is used as the MD. The token, T consists of the MD, MD_i, signed with the Beacon's private key and the other three fields. The token is broadcast at time t (steps 5 and 6). The algorithm is then repeated (step 7).

4.4 One-time Token

It is generally accepted that the beneficial features of a public key cryptosystem are bought at the expense of speed. At present it is not feasible to use a public key system for bulk encryption. In practice, however, it is quite desirable to create a hybrid system in which a public key system is used for authentication and distribution of a session key. The session key would then be used by the two principals to communicate securely using a symmetric key system.

With a Beacon based system, a "one-time token" can be used to simplify the process. By one-time token it is meant that a token emitted by the Beacon can be used only once to obtain service from a particular server; much like an admission ticket to a theatre. Once the token has been presented, it is marked and will not be accepted by that server on any subsequent occasion. The process of marking tokens is much easier for the server than maintaining a database of prior requests. The use of one-time tokens eliminates the possibility of a replay attack and thus simplifies the process.

To illustrate the process consider a very simple case. Alice wishes to communicate securely with Bob. In this case Bob can be thought of as being the server and Alice the client. Assume, for simplicity, that Alice and Bob communicated yesterday and they are both certain that each knows the other's public key. Such an occurrence is not uncommon in a distributed system since most principals communicate within a small group and a cache is commonly used to store commonly used keys. The process has two steps:

1. Alice ⇒ Bob: Alice,N_i, $\{K_{A,B}\}_{Y_B}$, $\{MD\}_{X_A}$
   Alice initiates the exchange by sending Bob a message which contains her name, the nonce N_i, and a session key $K_{A,B}$. The session key is created by Alice, and will be used with a symmetric cryptographic system to secure subsequent messages. Since the session key is the secret in the message, it is the only part that is enciphered with Bob's public key, Y_B. The nonce, N_i, is selected at random by Alice from the list of active tokens and ensures message freshness. The message integrity is protected using a MD which is signed by Alice.

2. Bob ⇒ Alice: $\{N_i\}_{K_{A,B}}$
   Bob, having received the request for communication, can confirm that the message did come from Alice and that it has not been altered. The freshness of the message is guaranteed by the use of a nonce, N_i, which was recently broadcast by the Beacon. Since the nonce can be used only once there is of course a finite probability that the nonce chosen by the Alice from the active list has already been presented to Bob by someone else. In such a case the request would be rejected and Alice would have to re-apply with another nonce. The probability of such a collision occurring is dependent on factors such as network load, token frequency and token life. In practical applications the additional load caused by this effect should be minimal.

   Having received a session key which he can trust, Bob completes the protocol by authenticating himself to Alice. He enciphers the nonce with a session key and sends it to Alice. Since only Bob could have obtained session key, the message proves Bob's identity.

5 Beaconizing the Needham and Schroeder Protocol

The Needham and Schroeder (NS) protocol [9] is arguably one of the best known authentication and key distribution protocols. It has been the basis of a number of systems which use the nonce to prove freshness. In 1981 Denning and Sacco [5] pointed out a weakness in the NS protocol and suggested the use of time-stamped certificates to guard against a replay attack. Since that time authentication protocols have been divided into two groups, one preferring the use of nonces and the other preferring time-stamps.

In this section we briefly outline the NS protocol using asymmetric keys. Next we will describe the weakness pointed out by Denning and Sacco and their solution to the problem. For a complete description the reader is directed to the original papers [9] and [5]. We will then modify the NS protocol to take advantage of a Beacon. We will show that the modified protocol simplifies the solution to the authentication problem and has advantages over both the NS protocol and modified protocol suggested by Denning and Sacco.

5.1 Needham and Schroeder Protocol

The NS protocol requires a trusted authentication server (AS) to establish trust between two principals wishing to communicate. Each principal within a realm which is dominated by a particular AS, is required to register his or her public key with that AS. To establish trust between principals, the AS must have the trust of all principals within its realm, to maintain and distribute these keys reliably.

The NS protocol can be divided in to distinct sections. The following illustrates the two protocol sections.

Public Key Distribution Protocol

Consider the situation where Alice wishes to communicate with Bob but is not certain of his public key. Thus she must apply to the AS to obtain Bob's public key. The steps required are as follows:

1. Alice ⇒ AS: Alice,Bob
   Alice sends a message to the AS requesting the public key. The requesting message is in clear text and is the names of both principals.
2. AS ⇒ Alice: $\{\text{Bob}, Y_B\}_{X_{AS}}$
   The AS responds with a message containing the requested public key and is signed with the AS's private key. The message contains the name of the key's owner which allows Alice to verify that the reply contains the correct key.

This exchange does not by itself provide any assurance that the request was initiated by Alice nor of the freshness of the AS's reply.

Connection Protocol

Assuming that both Alice and Bob are able to obtain any required keys from the AS, the following are the steps required for them to authenticate each other in order to establish a conversation. Alice is the initiator.

1. Alice ⇒ Bob: $\{N_A,\text{Alice}\}_{Y_B}$
   Alice is able initiate the authentication process by sending Bob a message which contains a nonce, N_A, and her identity. The message is enciphered with Bob's public key, Y_B, which means only Bob will be able to access N_A.
2. Bob ⇒ Alice: $\{N_A, N_B\}_{Y_A}$
   On receiving the message, Bob obtains the nonce N_A. However, Bob cannot be certain of freshness nor of the identity of the actual sender. To verify identity and guard against a replay attack, Bob generates a nonce, N_B, and sends it to Alice. Bob also proves his identity to Alice by including N_A in the reply. The message to Alice is encrypted with Alice's public key.
3. Alice ⇒ Bob: $\{N_B\}_{Y_B}$
   As a final step in this authentication process, Alice proves her identity to Bob by returning N_B.

5.2 Denning and Sacco's Modification

In [5] Denning and Sacco analyzed the protocol and pointed out that it is only secure while there hasn't been a key compromise. The solution suggested by Denning and Sacco uses time stamped certificates. The form of these certificates is as follows:

$$\{P, Y_P, T\}_{X_{AS}}$$

where:

P → is the principal's identification.
Y_P → is the public key belonging to principal P.
T → is the time at which the certificate was issued.
X_{AS} → is the AS's private key which is used to sign the certificate.

The Denning and Sacco modified protocol combines the authentication and key distribution into a single process. That is, if the principals are able to obtain public keys reliably and message freshness can be guaranteed, then the communicating principals are able to use the features of their public key cryptographic system to authenticate messages. The steps of the modified protocol are as follows:

1. Alice ⇒ AS: Alice,Bob
   As before, Alice sends a request for two certificates, one containing Bob's public key and the other containing Alice's public key.
2. AS ⇒ Alice: C_A, C_B
   The AS responds with a message containing two signed certificates. C_A contains Alice's public key and C_B contains Bob's.

3. Alice ⇒ Bob: C_A, C_B
   Alice initiates the conversation with Bob by sending the certificates. Since the certificates are signed by the AS and contain a time-stamp to prove freshness, Bob is able to trust them.

In [5] Denning and Sacco point out that in order for Alice to obtain the certificates and deliver them to Bob, the certificates must have a lifetime. By this it is meant that the certificates must be valid for a duration of time. The length of the certificate lifetime would depend on factors such as the synchronization discrepancy between hosts and communication delays. During this period the protocol is vulnerable to a replay attack. Thus if the certificate lifetime is kept short, the protocol reduces the likelihood of a replay attack, but does not eliminate it.

Another feature of the modified protocol is that principals are no longer able to cache commonly used keys. Since the certificate lifetime must be kept short to minimize the risk of a replay attack, the AS must initiate all conversations.

5.3 A Beacon Based Approach

We now introduce a Beacon to the distributed system and modify the NS protocol to take advantage of the new service. As in the unmodified NS protocol, the beaconized protocol can be divided into two sections. The first enables a principal to obtain another's public key. The second, the connection protocol, is used by a principal to initiate a conversation. We end this section by modifying the connection protocol to include the distribution of a symmetric session key. Once again we will use the over worked principals, Alice and Bob, to demonstrate the protocol features.

Public Key Distribution Protocol
The following are the steps required for Alice to obtain Bob's public key.

1. Alice ⇒ AS: Alice,Bob
   Alice sends a message to the AS stating her name is Alice and requesting Bob's public key. The message is in plain-text and only contains the two identities.
2. AS ⇒ Alice: Bob, $Y_B, N_i, \{MD\}_{X_{AS}}$
   Since the reply to Alice contains no secret information, the message is not enciphered. As in the NS protocol the message contains the requested public key and the name of the key's owner. This ensures that the request made by Alice has not been altered. The nonce, N_i, is picked randomly by the Beacon from the list of active tokens and is used to guarantee that this message is not a replay. The message integrity is ensured by the MD which is signed by the AS.

Since the Beacon based system uses the concept of a "one-time token", if N_i has previously been presented to Alice, then it would have been marked and consequently the AS's reply would be rejected. In such circumstances, Alice would have to reinitated the request. The probability of such a collision occurring, in a practical application, is quite low.

Connection Protocol

Assuming that both Alice and Bob are able to obtain the required keys, the following is the step required for Alice to initiate a conversation.

1. Alice ⇒ Bob: Alice,N_j, $\{MD\}_{X_A}$
   Alice initiates the exchange by sending Bob a message which contains her name and the nonce, N_j. The nonce is selected at random by Alice from the list of active tokens and is used to ensures message freshness. The message integrity is protected using a MD which is signed by Alice. Since the message contains no secret information it is not encrypted with Bob's public key.

If N_j has previously been presented to Bob, then it would have been marked and the request for connection would be rejected. In such an event Bob would reply with an error message and Alice would select another nonce and reinitiate the request.

Distribution of a Symmetric key.

At present it is not feasible to use a public key system for bulk encryption. Thus, it is quite desirable to create a hybrid system in which a public key system is used for authentication and distribution of a session key. The session key would then be used by the two principals to communicate securely using a symmetric key system. We now modify the protocol to allow the two principals, Alice and Bob, to share a session key.

1. Alice ⇒ Bob: Alice,N_k, $\{K_{A,B}\}_{Y_B}$, $\{MD\}_{X_A}$
   Alice would initiate such an exchange by sending Bob a message containing her name, the nonce N_k, and the session key, $K_{A,B}$. Once again N_k is used to ensure freshness. Since the session key is the only secret in the message, it is the only part that is enciphered with Bob's public key, Y_B.The message integrity is protected using a MD which is signed by Alice.

5.4 Attacks on the Modified Protocol

The attacks that can be launched against the beaconized protocol can be broken up in to six categories. In the following section the effects of these are discussed in turn.

- In a *masquerade* attack, an adversary attempts to impersonate one of the principals in the system. Since the principal's secret key is used to prove its identity, for such an attack to succeed an adversary would require knowledge of such a key. If a principal's secret key were to be compromised, it is possible that an adversary could masquerade as that principal while the problem was undetected and before a new key was distributed. However, unlike the case of the NS protocol, an adversary is unable to block the distribution of new keys.

- By *eavesdropping* (or monitoring) network traffic, an adversary hopes to gain some advantage or learn some secret. Since the public key protocol does not require the transmission of any secret information, such an attack cannot succeed. In the case of the hybrid system, the session key is enciphered. Thus an adversary would require knowledge of the deciphering key.
- The goal of a *replay* attack is to gain some advantage or secret knowledge by retransmitting a message which was intercepted earlier. There are three distinct areas in which a replay attack could be attempted. They are:
  1. The token transmitted by the Beacon. The purpose of this message is to broadcast a unique token to all hosts on the network. Since the token has a finite life, if an expired token were retransmitted, the message would simply be discarded. Replaying the message before the token expires gains no advantage for the adversary as the duplicate token would be detected and thus discarded.
  2. The second attempt could be made against the public key distribution portion of the protocol. The protocol consists of two message. The effect of replaying these would be:
     - The protocol is initiated by a principal request another principal's public key. The request is in plain-text and is directed to the AS. Since the information is public and service is freely available, the adversary can gain nothing new.
     - The reply from the AS contains the requested public key. Since the message is unique; in that it contains the name of the recipient, a one-time nonce and is signed by the sender, a replayed message would be detected.
  3. The final message that could be replayed is the request for connection. This message is also unique, thus a replayed message would be detected and the attack would fail.
- A *modification* attack is an attempt to change the contents of packets as they travel across the computer network. For such an attack to succeed the change must be undetected. Such an attack would be futile because the recipient of a message is always able to detect any changes.
- An attempt to *delay* authentication messages would cause the token to expire and prevent the principal from completing the authentication process. Such an attack would have the same results as "denial of service".
- A *denial of service* attack could be launched by an adversary who is able to hinder communications in some manner. Detection and countering of such an attack is best dealt with by other means, such as statistical monitoring of the network.

6 Discussion on Advantages

At present there are two dominant approaches to guaranteeing message freshness within authentication schemes; the method favored by Needham and Schroeder requires that principals generate and exchange nonces. The second method which has been suggested by Denning and Sacco makes use of a timestamp within certificates that are fabricated by the authentication server. In this section we discuss the five main advantages gained by the use of a beaconized approach. We will compare our approach with these well known protocols.

- The Needham and Schroeder (NS) protocol has been the basis for a number of systems. The approach taken by them requires principals, wishing to establish a conversation, to engage in a three step message exchange. The purpose of this exchange is to ascertain the identity of the other principal and to guard against a replay attack. The distribution of public keys is provided by a trusted authentication server, AS. In the worst case, where both principals require the services of the AS to obtain public keys, the protocol requires a total of seven messages to be exchanged.
  In Denning and Sacco's modified protocol (DS) principals are required to obtain certificates from the authentication server. One benefit of this approach is that the number of messages required to initiate a conversation is reduced to three.
  The beaconized protocol allows principals with cached keys to initiate a conversion with a single message. The procedure for a principal to acquire the public key of a communicating partner requires two steps.

- In [5], Denning and Sacco pointed out that the NS protocol is vulnerable to a replay attack in the event that a principal's key is compromised. To over come this difficulty, they suggested the use of time-stamped certificates. The resulting protocol requires a principal who wishes to initiate a conversation, to contact the AS to obtain two certificates; one for each principal. Each certificate has a finite life and contains a public key. During the period between the certificate being issued and expiring, the DS protocol is also is vulnerable to a replay attack. Kerberos [16, 7, 2] is the best known implementation of timestamp certificates and uses the symmetric key version of the DS protocol [5]. In practice the duration of an authenticator within Kerberos is typically five minutes [3]. The DS protocol does not eliminate the possibility of a replay attack, but reduces it.
  The beaconize protocol's use of one-time token eliminates the possibility of a replay attack without the increased steps that are suggested within the NS protocol.

- Another feature of the DS protocol is that principals are no longer able to cache commonly used keys. The beaconized approach, like the NS protocol, allows principals to cache commonly used keys which results in the reduction of network traffic and load on the authentication server.

- Within the DS protocol the certificate lifetime must be kept short to minimize the risk of a replay attack, the AS must initiate most (if not all) attempts to communicate. Consequently, a greater amount of trust is invested

in the AS and the realm now has a single pòint of failure.
The beaconized approach reduces the role of the AS to simply being a distributor of public keys which was suggested in the original NS protocol.

- In response to the criticism made in [5] Needham and Schroeder suggested a modification to their original protocol [10]. The revised NS protocol guards against the redistribution of a compromised key by requiring the principals to include a nonce in their communications with the authentication server. The purpose of this nonce is to reassure each principal that the message received from the authentication server is fresh. This revision adds a small amount of load to principals engaged in initiating a conversation.
  In contrast the beaconized approach does not require each principal to have the ability to generate nonces. It would be fair to say that this advantage is offset within the overall system by the additional load caused by the introduction of a Beacon. However, within a practical environment there are likely to be principals of varying abilities and it would be preferable if the generation of good quality nonces was provided by a single service.

7 Concluding Remarks

In this paper we have attempted to revive Rabin's ideas regarding beacons and have shown that the use of Beacons can simplify authentication in a distributed system.

The fundamental difference between our approach and that more traditionally taken is the use of a Beacon to deliver a "one-time token" to each host. This simplifies the authentication process by taking advantage of the features of a public key cryptographic system. In contrast to the modification proposed by Denning and Sacco, the beaconized approach eliminates the possibility of a replay attack, allows principals to cache commonly used keys and preserves the role of the AS as a distributor of public keys.

8 Acknowledgment

We would like to thank Thomas Hardjono and Anish Mathuria for their interest and support. We would also like to thank the anonymous referees for their helpful comments and pointing out the use of Beacons in [4].

This work has been supported in part by the Australian Research Council (ARC) under the reference number A49232172 and the University of Wollongong Computer Security Technical and Social Issues research program. The second author has received additional funding from the ARC under the reference numbers A49130102 and A49131885.

References

1. D. W. Allan, J. E. Grey, and H. E. Machlan. The national bureau of standards atomic time scale: generation, stability, accuracy and accessibility. In *Time and Frequency Theory and Fundamentals*, pages 205–231, 1974.
2. E. Balkovich, S. R. Lerman, and R. P. Parmelee. Computers in higher education: The Athena experience. *Communications of the ACM*, 28:1214–1224, 1985.
3. S. M. Bellovin and M. Merritt. Limitations of the kerberos authentication system. *Computer Communications Review*, 20(5):119–132, 1990.
4. Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot election. In *Proceedings of the STOC'94*, pages 544–553, Montreal, Quebec, Canada, May 1994.
5. D. E. Denning and G. M. Sacco. Time-stamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, Aug 1981.
6. Thomas Hardjono, Yuliang Zheng, and Jennifer Seberry. Kuperee: An approach to authentication using public keys. In M. Medina and N. Borenstein, editors, *Proceedings of the ULPAA'94 International Conference on Upper Layer Protocols, Architectures and Applications*, pages 61–72, Barcelona, June 1994.
7. J. T. Kohl. The evolution of the kerberos authentication service. In *Proceeding of the Spring 1991 European Conference, Tromsø, Norway*, 1983.
8. Refik Molva, Gene Tsudik, Els Van Herreweghen, and Stefano Zatti. KryptoKnight Authentication and Key Distribution System. In Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, editors, *Computer Security - ESORICS 92*, number 648 in Lecture Notes in Computer Science, pages 155–174. Springer-Verlag, 1992.
9. R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.
10. R. M. Needham and M. D. Schroeder. Authentication revisited. *ACM Operating Systems Review*, 21(1):7, January 1987.
11. J. Postel. User datagram protocol. *Request for Comments (RFC) 768*, 1980.
12. M. O. Rabin. Transactions protected by beacons. *Journal of Computer and System Sciences*, 27:256–267, 1983.
13. Network Working Group Report. Network time protocol specification and implementation. *Request for Comments (RFC) 1119*, 1989.
14. R. Rivest. The MD5 message digest algorithm. Request for Comments, RFC 1321, 1992.
15. R.L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
16. J. G. Steiner, C. Neuman, and J. I. Schiller. Kerberos: an authentication service for open network systems. In *Proceedings of the 1988 USENIX Winter Conference Dallas, TX*, pages 191–202, 1988.
17. J. J. Tardo and K. Alagappan. SPX: Global authentication using public key certificates. In *IEEE Symposium on Research on Security and Privacy*, pages 232–244. IEEE, 1991.
18. Y. Zheng, J. Pieprzyk, and J. Seberry. HAVAL - A one-way hashing algorithm with variable length of output. Abstracts of AUSCRYPT'92, Gold Coast, Australia, December 1992.
19. Y. Zheng and J. Seberry. Immunizing public key cryptosystems against chosen ciphertext attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):715–724, 1993.