Security Issues in Mobile Computing
CS 690B

N. Asokan
Department of Computer Science
University of Waterloo

April 23, 1995

Contents

1 Introduction

Recent advances in wireless networking, portable computing devices and the growth of networking have changed the nature of large distributed systems. Users will demand access to services[1] from a much more diverse set of situations. This phenomenon has brought forth new issues in the security of distributed systems while adding emphasis to some of the other longstanding ones.

The most prevalent computing environments today are *static*: computing devices are fixed, users expect to access resources from a single (or a small set of) points, and interconnection is achieved by using wireline networks that are easy to control and protect. In addition, most of the computing takes place within the administrative boundaries of an organization. In contrast, in emerging *mobile computing* environments, the user population and/or the computing devices can move from place to place. User mobility and computing device mobility are orthogonal factors: for example, even on a fixed network, mobile users may wish to use the network from access points that are far apart. Further, some portions of the environment may consist of wireless networks that facilitate continuous device mobility.

Mobility will be an integral aspect of large distributed systems in the near future. It is important to identify the security issues and engineer appropriate solutions during the design phase of these systems. This report attempts to outline these issues and survey the solutions that have been described in the literature with a view to identifying potential research issues. Section 2 gives a basic introduction to security issues, particularly in distributed systems. Section 3 gives an outline of security techniques used in static computing environments. Section 4 describes the issues raised by introducing mobility. Section 5 is a summary of some proposed solutions to the security issues in emerging mobile computing environments. Section 6 is a discussion of the various solutions described in section 5; it also identifies some potential research issues. The subsequent sections elaborate on individual research issues that have been identified. The report ends with a section containing a summary and conclusions.

2 Security Issues

Several authors [VK85, For94] have presented classifications of the security issues in communication networks. There are five fundamental goals of security in information systems:

- *confidentiality* involves preventing unauthorized or unintended parties from gaining access to private or sensitive information,
- *integrity* ensures that unauthorized modification, destruction or creation of information cannot take place,
- *availability* ensures that authorized entities have access to services when they are needed,
- *legitimate use* guarantees that only authorized entities have access to services, and
- *accountability* ensures that entities are held responsible for their security-related activities by arranging that entities and activities can be linked if and when necessary.

[1] The term *services* includes access to resources since it can be considered to constitute a service.

The exact definition of what constitutes a "security-related" activity, and the details of which entity is authorized to perform which activities depend on the *security policy* adopted by the administrators of services.

One can argue that the value of an asset depends on its actual or potential use. The potential usefulness of an information service is predicated on two factors: the value of the information itself and the correspondence between its current state and a potential user's expectations about it. The former implies confidentiality while the latter implies integrity.

Actual use of an asset has two pre-requisites which correspond to the notions of correctness and completeness: any use should be "correct" (i.e. mechanisms should be in place to guarantee legitimate use) and all attempts at correct use should succeed (i.e. the asset should be available). Actual use generates new information about the process itself. This information must not be lost (accountability) and must be protected from unauthorized disclosure (confidentiality).

Ford [For94, Ch. 2] identifies the first four of the above as fundamental goals of information security. He considers accountability to be "a fundamental principle underlying any security policy." Before elaborating on these goals and how they are to be met, it is necessary to define the context.

2.1 Model

I will assume the standard model of a distributed system (such as the one described in [NS78]) for the purposes of this document. A distributed system consists of a network of several *nodes* that are connected by communication links. Nodes may have computing capabilities. While active, entities in the system are attached to nodes. However, this association is not necessarily permanent. Each entity has a unique identifier. An entity is free to send and receive information over the links. It can also see and modify information that travels along the links. Service providers (servers) and service users (clients) are entities. Servers are processes running on nodes, located in various parts of a network. A client (a user, or a process acting on behalf of a user) needs to authenticate itself by proving its identity to a server (i.e., by presenting its identifier to the server and convincing the server that it is indeed the entity that corresponds to the identifier) before the server can determine whether the particular service can be provided to the client. A group of entities that fall under the same administrative control is called a *domain.*[2] Finally, an active communication channel between two entities is an "association." It may traverse a path consisting of several intermediate network elements and the links that connect them.

2.2 Threats and Safeguards

The goals identified earlier correspond to the fundamental threats of *information leakage*, *integrity violation*, *denial of service*, *illegitimate use*, and *unaccountability.* These fundamental threats are enabled by *underlying threats* such as protocol or system flaws, misconfiguration, trapdoors, breaches of trust, masquerading and eavesdropping. There is no universally applicable classification of underlying threats. Ford [For94, Ch, 2] provides a lucid description of the various threats that are possible in different environments. Protections against the fundamental threats are classified into the various security services.

[2] Also referred to as a *cell* or a *realm.*

- *Authentication:* Authentication is the verification of the claimed identity of an entity. It is a fundamental security service: other services often depend on appropriate authentication having taken place. An entity can authenticate itself by a demonstration of its knowledge (e.g., a password), possessions (e.g., a key to a lock), or characteriscs (e.g., fingerprints). Typically, an entity is expected to be authenticated using more than one of the above methods([DP89, Ch. 7], [For94, Ch. 5]). Authentication can also be transitive: an entity can authenticate another entity by choosing to trust the word of a third.

- *Authorization:* Authorization or access-control is the mechanism used to determine who can legitimately access what services in accordance with the security policy of the administrative domain. It is often based on authentication. Authorization contributes directly to meeting each security goal identified earlier.

- *Confidentiality:* Confidentiality seeks to protect information from being revealed to unauthorized or unintended parties. This includes direct information, such as the contents of messages, or indirect information, such as the information that can be inferred by *traffic analysis.* Data encryption is a common technique used to provide this service.

- *Data Integrity:* Integrity of information is preserved by making sure that information cannot be changed, created or destroyed by unauthorized entities. Techniques such as *message digests* can be used to ensure integrity. A message digest $h()$ operates on a message M of arbitrary length to produce a fixed length hash value $H = h(M)$. If $h()$ is such that it is computationally infeasible to determine M given H, then it is called a *one-way hash function.* One-way hash functions can also be based on a secret key (i.e. $H = h(M, K)$, where K is the secret key). Hash values computed in this manner are called *message authentication codes.*

- *Non-repudiation:* Non-repudiation is a way to meet the goal of accountability. Provision of this service ensures that an entity cannot deny a security-related action it committed. Non-cryptographic techniques to implement non-repudiation include the use of a trusted third party. Cryptographic techniques like *digital signatures* can also be used. Digital signatures are in essence message digests that can also unambiguosly identify the entity that computed the digest. For example, assuming a public key cryptosystem, an entity A can sign a message M by first computing a one-way hash of M and encrypting the result using A's private key.

- *Logging:* Keeping track of the security-related activities of entities is a way to meet the goal of accountability. The nature of logging will depend on the security policies of the domain. Logs can be used in subsequent security audits.

- *Intrusion detection:* While the other services can ensure that the security objectives are met, detection of suspicious activities is useful to alert administrators of attempted attacks. Intrusion detection is usually based on statistical analyses of events.

Most existing static user systems use a password-based scheme by which users authenticate themselves to the systems that provide services. Typically, the operating system and/or the hardware will impose certain system security measures. However, the emphasis from the point of view of security in distributed systems is on *network security.* Physical protection of network elements and their connections can help in security. In most static computing environments, any guarantee of network security is predicated on the physical security of the network. When physical protection is not possible, cryptographic techniques must be used to secure the associations among entities in an open network.

An association manifests itself in the various layers of the protocol stack. One way to secure an association is to secure each link at the lowest layer. Both network elements at the ends of a given link will have to agree on a common secret key to encrypt the traffic that flows through the link. Such *link oriented* measures are entirely transparent to the end users and applications.

At the other extreme, the entities involved in an association can agree on a key and secure the connection at the application level itself. Such *end-to-end* measures do not require any trust in the intermediate elements and links. A discussion of the relative merits of these two types of measures can be found in [VK85] and [For94, Ch. 3].

Regardless of the exact placement of the security services in the protocol stack, mechanisms are necessary to establish the identities of communicating parties to each other and, if necessary, to establish information required to ensure the secrecy of any subsequent traffic between the parties during a given session. Such mechanisms can be based on shared key cryptosystems or public key cryptosystems.

3 Security Techniques in Static Environments

A common approach to providing security services is to build them upon the authentication service. In a distributed system, an association can be secured by first authenticating the entities involved, and then having them agree on one or more session keys to encrypt any subsequent confidential traffic. In this section, I describe this basic approach and implementations of it.

3.1 Standalone systems

The degenerate case of a distributed system consists of a single node. Standalone systems that have no network connections belong to this category. In such systems, a simple password mechanism was often sufficient to meet the security goals. When a user is given access to the system, he/she is assigned (or is allowed to choose) an initial password. Thereafter, presenting that password is sufficient to authenticate him/her. The computing base, consisting of the hardware and the operating system, is tightly controlled and therefore can be trusted. Any server running on the system can trust the system itself to correctly identify a client asking for service.

3.2 Intra-domain authentication

In a non-trivial distributed system consisting of multiple nodes, mechanisms like the above fail to meet the authentication requirements. In order to carry out the verification within its trusted computing base, every server must retain the information necessary to authenticate every potential client. The amount of information can become prohibitive, not to mention the duplication of information and resources that are necessary for such an arrangement. Further, if the server and the client are not co-located, the client must send its credentials over communications links. If the links are physically secure, it is safe to send the credentials without any protection. But, in the most pessimistic interpretation of the model assumed earlier, communication links will not be secure. A malicious eavesdropper can intercept the credentials of a client and present them to the server while claiming to be that client.

In spite of these limitations, the most prevalent authentication mechanisms in large distributed systems are those carried over from the days of standalone systems, sometimes with *ad-hoc* additions (like the Berkeley rcmd authentication) which can provide temporary solutions.

There have been several attempts to design and implement solutions to this network authentication problem using data encryption and one or more *authentication servers*("AS"s). The basic approach is described in [NS78]. It can be implemented using either shared key cryptosystems (SKCS) or public key cryptosystems (PKCS).

3.2.1 Solution Based on SKCS

Notation
X_K - A message X encrypted by a key K.
K_{AB} - A secret key shared between parties A, B.
$Tick_{AB}$ - A *ticket* (explained below) usedby A to authenticate itself to B.
N_A - A *nonce* (explained below) made/used by A.

A solution based on SKCS was first outlined in [NS78]. Several variations on the same basic theme have been described. The scheme presupposes the existence of a trusted third party called the *authentication server*. Every entity in the system shares a secret key with the authentication server. If an entity A wants to perform mutual authentication with another entity B, it engages in the following protocol using a common authentication server S:

Step 1: $A \Rightarrow S : A,\ B,\ N_A$
Step 2: $S \Rightarrow A : Tick_{AB},\ \{K_{AB},\ N_A\}_{K_{AS}}$
where, $Tick_{AB} = \{A, K_{AB}, < validity\ information >\}_{K_{BS}}$
Step 3: $A \Rightarrow B : Tick_{AB},\ N_A^{'}$
Step 4: $B \Rightarrow A : \{N_A^{'}\}_{K_{AB}},\ N_B$
Step 5: $A \Rightarrow B : \{N_B\}_{K_{AB}}$

N_A, $N_A^{'}$ and N_B are *nonces.* Nonces are random numbers that are used just once to guarantee the freshness of messages. They can be used to make sure that an attacker cannot replay captured messages without being detected.

In essence, the authentication server S chooses a random session key for A and B, called K_{AB}. It sends this session key to A and to B separately, encrypted with the secret key which it shares with each party. Apart from S, the only entity that knows the secret K_{AS} is A and the only entity that knows K_{BS} is B. A and B can therefore convince each other of their identities by demonstrating their knowledge of the assigned session key K_{AB}. Furthermore, K_{AB} can be used to encrypt any further private traffic during that particular session.

3.2.2 Solution Based on PKCS

Notation

$K_A,\ K_A^{-1}$ - Public and private key pair of A.

$h(X)$ - The result of applying a one-way hash function $h()$ to a message X.

$< X >_A = X,\ \{h(X)\}_{K_A^{-1}}$ - A message X *signed* by A.

$C_A = < K_A, ... >_{CA}$ - Public key certificate of A, signed by CA.

The same problem can be solved using public key (also known as "asymmetric key") cryptosystems. This scheme presupposes the existence of a *certification authority* (CA). Every entity trusts the CA to vouch for the public keys of other entities. The CA's public key is known to every entity. When an entity (say A) chooses a public key, it registers the binding (between its identity and its public key) with the CA. The CA, having satisfied itself of the *bona fides* of A, generates a *certificate* by signing the binding. Any other entity can verify that the certificate was indeed signed by the CA since it knows the CA's public key.

Given this scenario, two entities A and B can carry out the following protocol:

Step 1: $A \Rightarrow B\ :\ A,\ C_A,\ N_A$
Step 2: $B \Rightarrow A\ :\ C_B,\ < \{N_A\}_{K_A}, N_B >_B$
Step 3: $A \Rightarrow B\ :\ < \{N_B\}_{K_B} >_A$

Again the nonces N_A and N_B are used to deter replay attacks. A variation of this protocol can have a *directory server* which acts as a repository of public key certificates. In this case, A and B need not exchange public key certificates as shown above. Instead, each will request the directory server to supply the public key certificate of the other party.

In *identity-based PKCSs*[vT91] the public key of an entity can be derived from the identity information. This obviates the need for a directory server or extra messages. A and B can unilaterally determine the public keys of each other. While this reduces the number and sizes of messages in the protocol, there is no means of changing the binding between the key and the entity, should the need arise.

Unlike the solution based on an SKCS, the trusted entity (in this case, the CA) need not be online during an execution of the protocol. While this can be an advantage, it also poses the problem of revocation of privileges. Revocation of privileges becomes necessary in several situations. For instance, if an entity discovers that its secret information has been discovered by an intruder, any further damage can be contained by revoking the privileges associated with that secret. In a solution based on an SKCS, this can be achieved by changing the long-term secret key shared between the entity and the authentication server. Several techniques can help solve or mitigate this problem. Internet Privacy Enhanced Mail(PEM) [Ken93] uses a *Validity Period* field in the certificate. A universal means of representing time as well as loosely synchronized clocks are required to enforce such validity restrictions. When the validity period of a certificate is long, as is likely to be the case except for temporary certificates, other techniques are necessary for more timely damage control. PEM also uses the notion of *certificate revocation lists* (CRLs). A CRL of a CA consists of the list of revoked certificates. Each CA advertises the frequency of its CRL updates. Each CRL will contain the time of issue and the expected time for the next CRL to be issued. This CRL issuance convention enables an entity to ensure that a the certificate is currently valid.

3.3 Discussion

Computation in a PKCS is usually several orders of magnitude more complex than in an SKCS of comparable strength. Typically, it is the private key operations in public key cryptosystems that are computationally expensive. Schneier summarizes the current performance figures for DES (Data Encryption Standard, a widely used symmetric key cryptosystem) and RSA (Rivest-Shamir-Adleman, a popular public key cryptosystem named after the researchers who invented it) operations in his book [Sch94, page 285]. The fastest RSA hardware implementations are slower than DES by a factor of about 1000. In software implementations, RSA is slower than DES by a factor of about 100.

Also, note that both systems presume the existence of a common trusted entity. It is easy to mandate such a common trusted entity in a network that is under a single administrative control. Such a network is called a *domain*. In a global distributed system, there will be several thousands of domains. However, the interaction between domains is likely to be less common in the case of static environments.

In summary, security in static computing environments often relies on the strengths of individual system security and the physical security of the network. Where stronger security requirements exist, various authentication schemes based on cryptography have been developed. A common feature of these schemes is the requirement for a single trusted entity. Such schemes work well within one administrative domain.

4 Issues in Mobile Computing

In this section, I outline the characteristics of mobile computing and describe how these impact the security issues discussed in earlier sections.

The three distinguishing features of emerging mobile computing environments are: mobility of users, mobility of network elements (i.e. portable computing devices), and wireless networking.

4.1 Mobility of Users

With mobile users, the likelihood of inter-domain interaction becomes more pronounced: users are much more likely to want access to services and resources from a distance. Increased inter-domain interactions imply that mechanisms used to implement the distributed systems must scale to a global level. As far as security services are concerned, the following ramifications become apparent.

- *Global Authentication:* A mechanism for flexible global authentication is essential to support user mobility. As seen previously, authentication often forms the basis of other security services such as authorization.

- *Privacy:* The need to ensure privacy of users becomes more pronounced. In static environments, the location of a user or network element is unlikely to be secret information. Users and network elements are stationary. However, in a mobile computing environment, it may be necessary to protect information about the locations, itineraries and activities of users.

- *Eavesdropping:* Assumptions about physical security of the network no longer hold true when inter-domain interactions enter the picture. Even if the foreign domain claims its network to be physically secure, a visiting user may not be willing to accept this assurance. Thus, some sort of cryptographic protection becomes unavoidable.

An issue that is common to the mobility of users and network elements is the availability of resources. Mobility necessarily restricts the computing resources available. This is elaborated below.

4.2 Mobility of Network Elements

Mobile users may carry portable computing devices. Portability introduces the following issues.

- *Asymmetry in resources:* Portable devices have comparatively fewer resources available to them. While techological advancements will improve the quantity and quality of the available resources, the *asymmetry* between mobile users/devices and their fixed counterparts is likely to persist.

- *Risks to data:* Due to the higher risks of physical damage, loss, or theft, mobility of devices implies that there is a higher risk of loss for the data stored on them [Geo93].

4.3 Wireless Networking

Wireless networking is necessary to support continuous user and device mobility. This introduces additional issues of concern.

- *Eavesdropping:* The convenience of wireless networks has a cost: it is more convenient for an eavesdropper to listen in on the traffic. It is often asserted that the primary security concern with wireless networks is that communication is susceptible to eavesdropping and tampering. While it is true that in wireless networks, links have no physical security at all, the situation is not much better with wired but open networks. Thus cryptographic protection is necessary in both wireless and fixed networks.

- *Handoff:* Another issue that arises when dealing with multiple domains is that of *handoff.* In a wireless network with mobile users, a user might wander into neighbouring cells while a session is active. Security systems must provide convenient and fast means for the session to be transferred ("handed off") from a cell in one domain to a cell in a different domain.

- *Bandwidth/Error rate:* Wireless networks typically have lower bandwidth and suffer from higher error rates compared to wireline networks. Consequently, traffic over wireless networks is expensive. Security protocols meant to be used over wireless networks should therefore pay special attention to minimizing the number of messages, message sizes and frequencies of exchange.

- *Frequent Disconnections:* In contrast to wireline networks, disconnections will be frequent in wireless networks. Many researchers are working on finding ways to design robust protocols and applications that can withstand such disconnections. There are security implications as well. Bellovin [Bel89] showed how TCP sessions in progress can be "hijacked." Intrusions using an attack based on guessing TCP starting sequence numbers(also described in [Bel89]) have already been reported [CER95]. Both these attacks are easier to carry out if the network layer entity that is being impersonated happens to be off-line. With frequent disconnections, such threats assumes greater importance.

4.4 Other Constraints

In addition to the distinguishing features mentioned above, security solutions for mobile computing environments depend on other factors as well.

- **Compatibility and Simplicity:** Compatibility and simplicity as goals have many aspects in this context. A common concern in developing mobile security solutions is the need for *seamless integration* with existing static environments. In a wireless network for example, the first goal is to achieve "wired-equivalency": using cryptography to secure the wireless links so that their security is comparable to the physical security often found in the wired networks. This can be done with link-oriented measures which can be completely transparent to the end users as well as to the wired portions of a specific connection. Link-oriented measures do not preclude any end-to-end measures being employed at a higher layer in the network stack. But unlike other measures, link-oriented measures will not require changes in the static parts of the network. Hence, the general tendency in the proposed solutions is to use such measures first. Another general concern is to *minimize overhead* in terms of the number and sizes of messages that need to be exchanged during a protocol run.

In summary, assumptions about the physical security of networks, while unreasonable in static computing environments, become quite unacceptable when mobility is introduced as a factor. Solutions to the security issues in large distributed systems should have *scalability* and *simplicity* as design goals.

5 Proposed Solutions

This section has brief overviews of the various solutions described in the literature. I have tried to use a consistent scheme of notation; consequently the notation for a specific solution may be different from that used in the original source.[3]

Notation
M - Mobile entity.
R - Remote domain M is visiting.
H - M's home domain.
D - M's current domain. (H or some R)
T_A - A time stamp made/used by entity A.
N_A - A nonce made/used by entity A.

[3] The same identifiers are used here for both the name of the domain and the identity of the domain server (i.e., CA or authentication server). For instance, R is also used to denote the domain server of domain R.

5.1 Global System for Mobile Communications

Global System for Mobile Communications(GSM)[ETS93] is the European standardization effort for personal communication networks. GSM attempts to provide user authentication, data confidentiality and key distribution. In the GSM authentication protocol, each mobile entity has a unique identifier called the *International Mobile Station Identity (IMSI)* which binds it to a specific home domain. The authentication server in a home domain shares a secret key with every entity in its domain. When the mobile entity M "pops up" in a remote domain R, it presents its *IMSI*. R infers the home domain H and sends a request to it. H replies with a set of triplets that R can use to authenticate M. Each triplet consists of a random challenge $RAND$, a response to the challenge $SRES$ which is a function of $RAND$ and K_{MH}, and a session key K_{MR} which is a different function of the same arguments. R can now challenge M by presenting a $RAND$. Only an entity that knows K_{MH} can compute the corresponding $SRES$. Hence it can be authenticated as M. Once authenticated, M and R can use K_{MR}, which M can compute similarly, for encrypting private data.

More formally,
Step 1: $M \Rightarrow R : M$
Step 2: $R \Rightarrow H : M,\ R$
Step 3: $H \Rightarrow R :$ A Set of $< RAND,\ SRES,\ K_{MR} >$ triplets
where,

$RAND$ - a random number challenge.
$SRES$ - $\{RAND\}_{K_{MH}}$ (using encryption algorithm A_3) and
K_{MR} - $\{RAND\}_{K_{MH}}$ (using encryption algorithm A_8).

Step 4: $R \Rightarrow M : RAND$
Step 5: $M \Rightarrow R : SRES$

Subsequent traffic can be encrypted with K_{MR} using a third algorithm, A_5. A_3, A_5, and A_8 are all proprietary shared key encryption algorithms. It is assumed that the portion of the network between H and R is secure. While this is a reasonable assumptions in certain contexts, it is not, in an open network. Provision of anonymity is based on the use of temporary aliases called *Temporary Mobile Station Identities (TMSIs)*. However, the $IMSI$ needs to be revealed in order to initialize the $TMSI$.

5.2 Cellular Digital Packet Data

Cellular Digital Packet Data (CDPD) (see [M$^+$94] for references) is an architecture developed by a consortium of companies based in the United States. As with GSM, each mobile has a unique name and belongs to a specific home domain. To authenticate itself to H, M is required to present a triplet $< M,\ ARN,\ ASN >$, where ARN is an 'Authentication Random Number' and ASN is an 'Authentication Sequence Number'. After a successful authentication run, H can choose to change the values of ARN and ASN and communicate them to M. In this case, M is expected to use the new values during the next authentication attempt.

The authentication protocol works as follows: when M pops up in foreign domain, it engages in a run of the Diffie-Hellman (DH) exponential key exchange protocol with R. (See [DP89, pages

219-222] for a detailed description). The DH protocol enables the two parties of a connection over an insecure channel to negotiate a shared key in such a way that an observer cannot determine the key. Once the key is established, M presents its credentials in the form of the $< M,\ ARN,\ ASN >$ triplet. R then forwards these credentials to H. If the credentials are what H expects, it sends a confirmation message back to R. Optionally, H may also send a new triplet with fresh values for ARN and ASN.

As in GSM, the wired network is assumed to be secure. Authentication in CDPD is based on shared secret state between M and H (i.e., the current values of ARN and ASN). This implies that (a) R or an intruder in the wired network between R and H can impersonate M once the shared state has been revealed to it, (b) if such impersonation takes place, it may result in a denial of service when M attempts to authenticate to H next time (since the shared state may have been altered).

5.3 Mobile IP Proposals

The problem of authenticating the mobile node or user to a remote domain is not addressed in the various mobile IP proposals described so far in the literature. The IETF draft proposal [Per94] and the Harvard architecture [B$^+$94] both use a one-way hash function based on a shared secret key (i.e., a message authentication code) to authenticate M to H during the registration process. Both proposals recommend the use of keyed MD5 [Riv92] for this purpose since it has no export restrictions and is quite fast.

The Harvard architecture uses a "magic-cookie" mechanism (similar to the X authority mechanism used by the X11 window system) to provide limited protection to routing re-direction messages. A correspondent host CH that wishes to receive routing re-direct messages concerning a particular mobile M sends a subscription request to H. H responds with a magic-cookie (a random string) to CH. Subsequent routing re-direct messages about M should contain this magic cookie. Otherwise, CH will discard the re-direct message. This prevents intruders who do not have access to the path between H and CH from sending spurious routing re-directs.

5.4 KryptoKnight Mobility Extensions

KryptoKnight is a network authentication/key distribution system developed by IBM. Recently, extensions to KryptoKnight to support mobile entities were described in [M$^+$94]. Each mobile M is assumed to share a secret key K_{MH} with its home domain authentication server. The key to be used between M and a remote domain R is computed as a function of K_{MH} and R. Further, every pair of domains H and R is assumed to share a secret key K_{HR}. With these assumptions, the proposal presents a way to carry out authentication in a mobile environment. The protocols described in [M$^+$94] are built using the *Two Party Authentication Protocols* of KryptoKnight. However, the basic ideas can be used with any standard authentication protocol. The description here attempts to capture the essence of their proposal without making specific references to KryptoKnight protocol building blocks.

The protocol works as follows: when M pops up in R, it computes K_{MR}. It then presents its credentials to R, encrypted by K_{MR}, along with its own identity. At this point, R cannot verify the credentials since it does not know K_{MR}. The structure of M's identity presented allows R to determine the identity of the home domain H of M. R then passes the credentials of M, along with its own credentials, to H. Since H knows K_{MH}, it can compute K_{MR}, and verify that the credentials sent by M via R are genuine and fresh. If the verification succeeds, H can send the key K_{MR} to R, which can then use this key to verify M's authenticity.

More formally,

Step 1: $M \Rightarrow R : M, T_M, N_M, Cred_{MR} = \{T_M, N_M, M\}_{K_{MR}}$

K_{MR} is computed as $K_{MR} = f(R, M, K_{MH})$. Thus M and H can easily compute K_{MR} for a given R. When R receives this message, it will not be able to authenticate M since it does not share any secret information with M as yet. It has to communicate with H, the entity that has jurisdiction over M, in order to do this:

Step 2: $R \Rightarrow H : R, T_M, N_M, N_R, \{M, Cred_{MR}\}_{K_{HR}}$

On receiving this, H will be able to decrypt the last part first to recover N_R and M. It can then compute K_{MR} and therefore decrypt $Cred_{MR}$. If H is satisfied that T_M is satisfactorily recent, it can inform R of the key K_{MR}:

Step 3: $H \Rightarrow R : \{H, N_R, K_{MR}, M, R\}_{K_{HR}}$

Now, R can convince itself of M's identity. It can choose a session key K' to be used for subsequent private traffic with M:

Step 4: $R \Rightarrow M : \{M, N_M, K'\}_{K_{MR}}$

For subsequent authentication while M remains in R, H does not need to be contacted since M and R can both use K_{MR} at this point.

The authors also suggest that if a handoff takes place (say from domain R to domain R') while a session is in progress, R can pass along the session key K' to R' using their shared inter-domain key, $K_{RR'}$. For subsequent authentication to R', M can go through the regular registration process involving H.

Note that the identity of M is not revealed to anyone eavesdropping on the connection between H and R, since it is not sent in the clear. However, for someone on the same local network segment as M and R, M's identity, passed in the clear in the first message, is visible. In an attempt to provide some anonymity for M, the authors also suggest that M can assume a *traveling alias* (say M_t) when it roams out of its home domain. The home domain H is the only other entity that will know the mapping $M_t \rightarrow M$.

5.5 Proposal by Aziz and Diffie

In [AD94], Aziz and Diffie consider privacy and authentication issues in wireless networks. Their major goal is seamless integration of wireless networks with existing, widely-deployed wireline networks. Consequently, they only concern themselves with authenticating mobiles to base stations and arranging to provide a secure channel at the link layer between the two. They use certificates (using a public key cryptosystem) to provide authentication. Subsequent privacy of data communication can be provided by using a shared key cryptosystem (SKCS). The particular SKCS to be used can be negotiated by the two parties involved. Each entity E is assumed to possess a certificate C_E, signed by the certification authority. Each entity is also aware of the certification authority's public key with which they can verify the signatures on the certificates presented by other entities. The protocol works as follows:

M presents its certificate, a challenge nonce N^c and the list of SKCSs supported to the base station B.

Step 1: $M \Rightarrow B$: C_M, N^c, List of SKCSs

B can now verify the certificate and extract M's public key, K_M from it. If the certificate is valid, it chooses an acceptable SKCS from the list, generates a nonce N_B and sends the following reply back:

Step 2: $B \Rightarrow M$: C_B, $\{N_B\}_{K_M}$, Chosen SKCS, $<$ $\{N_B\}_{K_M}$, Chosen SKCS, N^c, List of SKCSs$>_B$

M can now verify B's certificate and extract its public key, K_B. With this, it can (a) decrypt B's nonce N_B, and (b) decrypt the signed component of the message and check for the presence of N^c to ensure that the message is fresh and that the other parts have not been tampered with. If the checks succeed, M can pick a new nonce N_M and return it to B

Step 3: $M \Rightarrow B$: $\{N_B\}_{K_B}$, $< \{N_M\}_{K_B}$, $\{N_B\}_{K_M} >_M$

B can ensure the freshness of the message by checking for the presence of its nonce, and can then extract M's nonce using K_M. Note that the nonces N_B and N_M were never revealed in public. Therefore, each party can form a shared key by XORing the two nonces.

The authors suggest that in the case where there is no common CA between M and B (e.g., when M roams into a remote domain), B can be asked to form a *certificate chain.* Certificate chains are discussed further in Section 8. The chain starts with a certificate signed by a CA known to M. Each certificate in the chain certifies the public key of the next CA. The last certificate in the chain certifies the public key of B. Any applicable certificate revocation lists must accompany each certificate in the chain. The mobile is responsible for checking the validity of each certificate.

5.6 Proposal by Beller *et al* and Carlsen

In general, public key cryptosystems involve computationally intensive operations. Several proposals for mobile authentication systems decided against using public key cryptosystems since the computation power available to a mobile entity is at a premium (see [B+93] for some quantitative analysis of the computational complexity and for further references). Beller *et al* [B+93] argue that public key systems can be designed so that the computational requirements for the two parties involved are not symmetric. In particular, they show that a protocol based on Rabin's modular square root technique (with speed-up provided by using a fast multiplication technique) can provide acceptable performance. However, the asymmetry is achieved at the cost of relying on the mobile entity to choose a good random number.

The attraction in using public key cryptosystems for authentication is that privacy is then easier to provide (with some caveats). The authors note that while there are proposals to provide privacy and anonymity with protocols based on symmetric key cryptosystems (e.g., by having a "temporary id" for mobile entities that can be changed by mutual agreement with the home domains), they require synchronization, and recovery is cumbersome when synchronization is lost.

Carlsen [Car94] improved these protocols by making them resistant to replay attacks. Carlsen also explains that the issue of anonymity has two parts: privacy of location and privacy of data origin/destination. The former issue arises during the registration of the mobile entity (i.e., the "login" process); the latter issue arises when a particular "connection" is begun. Privacy of location and privacy of data origin cannot be provided by a protocol based purely on shared key cryptosystems. However, Carlsen describes a method by which privacy of data destination can be provided. In essence, the protocol is as follows. Each entity is assumed to share a key with the authentication server, S. When an incoming call for M arrives, R needs to get M's attention without revealing M's identity. R requests a session key from S. S presents the session key $K^{'}$ encrypted with K_{SR} (which R can decrypt) as well as with K_{SM}. R then multicasts the second encrypted message. Only M can decrypt it and proceed to complete the call which will then be protected by the session key $K^{'}$. In other words, all the mobile entities need to continuously decrypt every message that appears on the network. Carlsen asserts that while this can be done with a symmetric key cryptosystem, public key cryptosystems are too expensive for it.

6 Discussion and Research Directions

Privacy and authentication, provision for anonymity, global scalability and simplicity are the goals of a security system in a mobile environment. While designing such a system, several other factors need to be taken into consideration. Vulnerability to eavesdropping in foreign and/or wireless networks and the low computing power available to mobile entities are two of those factors.

All of the proposals described above attempt to provide authentication and privacy, to varying extents. In this section, I attempt to identify issues and open problems based on the descriptions provided in the previous section. The list of items considered is presumably not exhaustive; the intent is to identify the more pressing issues.

6.1 Issues to Consider

The following factors must be borne in mind when security solutions are designed for mobile computing environments.

6.1.1 Asymmetry

Asymmetry in available resources becomes crucial when mobile entities are involved. The proposal by Aziz and Diffie does not take this asymmetry into account. The proposals by Beller *et al* [B+93] and Carlsen [Car94] do. In general, the public key operations (verification and encryption) are less expensive than the private key operations (signing and decryption) [AD94]. This fact can be exploited to design practical solutions.

6.1.2 Scalability

Global scalability is a problem that has not been fully addressed in any of the proposals above. GSM and CDPD simply assume that the wired network is secure, thereby removing the need to consider global scalability as a problem. While this may be true in networks under a single administrative control, secure networks with global scope will become increasingly unlikely. Molva *et al* [M+94] describe a solution to the problem, assuming that a global key distribution mechanism exists. While it is by no means a requirement of security systems for mobile networks alone, the provision of a global key distribution mechanism is crucial for the scalability of such a system. Internet Privacy Enhanced Mail [Ken93] and ISO/ITU X.509 authentication proposals both assume a hierarchical organization of trust in attempting to provide a solution. Shortcomings of such assumptions have been described in [G+92] and [Y+94].

6.2 Problems to Solve

The subsequent sections elaborate on the following security problems while keeping the above issues in mind.

6.2.1 Anonymity

None of the solutions described earlier attempts to seriously address the question of providing anonymity for mobile users. The GSM authentication protocol uses temporary aliases which provide a certain degree of anonymity in the wireless parts of the network. The CDPD authentication protocol uses a negotiated session key to protect the initial exchanges involved in the authentication protocol; but since the session key is negotiated *before* authentication takes place, it is not a very effective way of providing anonymity. In Section 7, the issue is discussed in detail and a solution is presented.

6.2.2 Global Authentication

A flexible, scalable authentication system needs to be developed. A subsequent section describes the issues involved in designing global authentication systems and in the evaluation and distribution of trust in a global scale. In the absence of such a system, applications will resort to relying on weaker authentication methods (or none at all).

7 Anonymity in a Mobile Computing Environment

7.1 Introduction to the Problem

In a distributed system where some entities (i.e., users and computing devices) are mobile, it is often desirable to keep information about their movements, locations and activities confidential. The modalities of providing such *anonymity* depend on the underlying security infrastructure of the system. Typically, this infrastructure is built using a shared key cryptosystem (SKCS), a public key cryptosystem (PKCS), or a hybrid system.

Within the context of this report, anonymity as a goal does not imply unrestrained anonymity; it implies the protection of relevant information from *unintended* parties. For example, I assume that a server may demand that a potential client reveal its true identity and perhaps provide adequate authentication as a pre-requisite to the use of services.

In traditional computing environments, with fixed entities and wireline networks, anonymity of location has not been considered a significant issue. In a mobile computing environment, it assumes greater importance. It is likely to be an important security problem to solve before mobile computing gains wide acceptance. Anonymity is especially desirable when an entity travels outside of its 'home domain.' In spite of that, it has not been considered a basic requirement in designing protocols for mobile computing environments (for example, the IETF mobile-IP effort [Per94] has not yet addressed this issue). Like every other aspect of security, provision of anonymity is tightly coupled with the architecture of the rest of the system. For instance, providing anonymity of network layer identity will be of little use if the routing protocol uses re-direction: anyone who has access to a router elsewhere on the system can determine the whereabouts of a mobile entity by trying to send a message to it. Therefore, the problem of providing anonymity should be considered during the design stage of building mobile computing environments.

There are philosophical motivations for providing anonymity as well. An entity that chooses to protect its location and activities from unintended third parties must have the mechanisms available to do so even if openness and full disclosure is a common policy. Formal proofs of security in complex systems are expensive or infeasible. The principle of least information, disclosing just enough information necessary to perform a particular function and no more, is therefore a good rule to live by in designing security systems.

The rest of this document describes some proposed solutions to the anonymity problem and makes suggestions for improving them. The basic theme behind the suggestions is that effective anonymity can be achieved by making a *limited disclosure* of information.

7.2 Solutions

7.2.1 Shared Key Cryptosystems

In a distributed system with a security infrastructure based on an SKCS, an entity (the client) authenticates itself to another entity (the server) by demonstrating the possession of a secret key shared between them. To do this, the client has to first announce its (claimed) identity in the clear. This makes it hard to provide anonymity of location.

An intuitive solution is to use the notion of *nicknames*. Molva *et al* [M$^+$94] suggest that a mobile entity could use a *traveling alias* when it travels in foreign domains. Only the mobile entity and its home domain will know the mapping between the traveling alias M_t and the real identity M (see Section 5.4 for more details). They go on to suggest that M_t be changed "at regular intervals; perhaps as often as passwords or PINs." However, in the absence of a protocol to change traveling aliases over an insecure channel, entities can change traveling aliases only when they return to their home domain. Such traveling aliases with fairly long lifetimes may be problematic in certain circumstances. For example, as mentioned before, the mobile entity may be called upon to reveal (and even prove) its real identity M before being granted certain services in foreign domains. With time, the set of entities that know the mapping between M_t and M will grow. Further, an onlooker will be able to use a long-lived alias M_t to link the various activities of M for the period during which the alias remains in effect.

An alternative is to use short-lived aliases [ETS93]. Aliases are changed by mutual agreement between the mobile entity and its home domain server. This implies that the mobile and the home domain stay synchronized. If synchronization is lost, a sub-protocol is necessary for re-synchronization. Achieving re-synchronization securely is a challenge. In [ETS93], the mobile entity may be required to send its identity in the clear in order to achieve resynchronization. This, as observed in the same document, constitutes a "breach in the provision of the service."

Loss of synchronization is likely to be rare. In addition, the home domain is likely to have significant computing resources at its disposal. Therefore, in the event of loss of synchronization, I suggest that it may be reasonable to expect the mobile M to simply send a message to H encrypted with K_{HM}. H can then search through its database of shared secret keys until it finds one that can extract the message. If the number of entities in H's jurisdiction renders this an expensive operation, then I suggest that H can divide the principals into groups of manageable sizes, each with a unique group identifier. When synchronization is lost, M can send an encrypted message along with its group identifier. This way, a reduced level of anonymity can be provided by making a *limited disclosure* of information.

Preserving anonymity of data origin is similar to preserving anonymity of location. Anonymity of data destination is somewhat different when the destination is a mobile entity. Carlsen's solution [Car94] suggests essentially the following (see below for a clarification): a domain server D that wants to send a message to a mobile M, currently present in its domain, will multicast a message $\{M, K_s\}_{K_{DM}}$ to all the mobile entities currently present in the domain. Each mobile will attempt to decrypt every such multicast message. Only M will be able to recover the message and the session key K_s which can be used to secure the subsequent session. Carlsen argues that in a portable (voice) communication system, such a computing load on mobile entities is reasonable assuming 3 calls per hour. The arguments may not be applicable in a data communication system where the frequency of messages to a mobile entity can be potentially much higher. Carlsen's model actually consists of various *ports* (or base stations in more common parlance). Each port serves a number of mobile entities that happen to be inside its region. Thus, it is a port which attempts to send a message to a mobile M without revealing M's identity to an onlooker. The description here is conceptually similar to Carlsen's solution.

7.2.2 Public Key Cryptosystems

Public key cryptosystems can provide location and data origin anonymity in a natural way. The mobile M can use the domain server D's public key K_D to encrypt messages to it. Within an administrative domain, each M can be required to know the public key K_H of its home domain server. However, when M wanders into a different domain R, it cannot be expected to know the public key K_R. It has to identify itself first so that R can determine how to *mutually* authenticate M. The decrypt-on-the-fly solution by Carlsen [Car94] to provide anonymity of data destination (as described in the previous subsection) will be prohibitively expensive in the case of a PKCS.

Some researchers [ETS92][F$^+$94] have discouraged the use of a PKCS to build authentication protocols for environments with mobile entities. The rationale for this recommendation is that the computational complexity of known PKCSs are beyond the limited resources available to mobile entities. Even if this argument is valid for a specific environment, it is likely to be a temporary limitation. What is more relevant is the *asymmetry* in available resources between mobile entities and their stationary counterparts. Beller *et al* observe [B$^+$93] that it is feasible to design practical authentication mechanisms using PKCSs keeping this asymmetry in mind.

7.2.3 A Hybrid Solution

Complete anonymity may be hard to achieve. But it might also not be an absolute requirement. The earlier notion of *limited disclosure* can again be used to achieve practical anonymity. For instance, it might be acceptable to reveal that the visiting mobile entity is from domain H (or a suitably large super-domain of H), without revealing its exact identity. A hybrid protocol can provide such limited anonymity without being too complicated. In the rest of this subsection I describe an example protocol for authenticating a mobile entity using the above notion of limited disclosure.

The following assumptions describe the model on which the subsequent discussion is based. The system is partitioned into domains. The authentication server in domain H shares a secret key with every entity in the domain. H's public key is known to every entity in its domain. Each pair of domains H and R has (or using an inter-domain key distribution protocol, can dynamically arrange to have) a shared key K_{HR}.[4] The remote server R will provide services to the visiting mobile M only if *(a)* M's true identity is revealed to it and *(b)* the authenticity of the claimed identity is adequately demonstrated. Finally, all network links are considered to be insecure channels in which an intruder can observe, modify or destroy data. The goal is to achieve satisfactory mutual authentication between M and R while minimizing the set of entities that can infer the presence of M in R by watching the traffic.

The authentication protocol works as follows.

Protocol P1:

When a mobile M shows up in a remote domain R, it begins the registration process by identifying its home domain H and presenting its credentials encrypted with the public key of H. I assume that M can compute the key K_{MR} which is to be used for the mutual authentication. I elaborate on the computation of K_{MR} below. The first message is sent from the M to R as follows:

M1: $M \Rightarrow R : Cred_{MR},\ H,\ \{M, R, N_M\}_{K_{MR}}$

Where,

$$Cred_{MR} \;=\; \{M, R, H, \{T_M\}_{K_{MH}}\}_{K_H}$$

[4] Designing a scalable key distribution mechanism is another security problem that has been made more urgent by the advent of mobile computing. While there have been proposals to solve this problem by imposing a hierarchy of trust, such a hierarchy is not acceptable in all situations. This open issue is discussed in Section 8.

T_M is either a time stamp (which will imply the need for loosely synchronized clocks) or some indicator of freshness (e.g. a nonce that was handed to M the last time it authenticated successfully to H). In this description, I will assume that T_M is the current timestamp. R cannot verify the credentials. It can however find and mutually authenticate H and pass the credentials to it. The second message is from R to H as follows:

M2: $R \Rightarrow H : \{R, Cred_{MR}, N_R\}_{K_{HR}}$, R

H can verify the veracity and timeliness of the message from M. If the verification succeeds, H has to send the necessary information back to complete mutual authentication between M and R. The exact nature of this information depends on the assumptions made. For example, if T_M is a nonce and not a timestamp, H has to generate a new nonce to be sent back to M so that it can be used in the next protocol run. H also needs to inform R (and perhaps M) of the key K_{MR} that is to be used to secure the channel between M and R. H can pick a random key K_{MR} to be used by M and R. Or K_{MR} can be derived from other information that is unique to a given protocol run. For example, K_{MR} can be a one-way hash function $h(K_{MH}, R, T_M)$. In this case, M already possesses enough information to compute K_{MR} when it initiates the protocol. I use the latter. This is why I stated that M can compute K_{MR} and use it to encrypt the second part of the message $M1$. The third message is from H to R as follows:

M3: $H \Rightarrow R : Tick_{HR}$

Where,

$Tick_{HR} = \{R, H, M, K_{MR}, N_R\}_{K_{HR}}$

R can now know the true identity of M. It can also extract the key K_{MR} using which it can extract the second part of message $M1$. It can use these to complete the mutual authentication with M as follows:

M4: $R \Rightarrow M : \{N_R^{'}, N_M, M, R\}_{K_{MR}}$

M can use the already computed K_{MR} to extract this message. It can verify that this response is timely by checking for the presence of its nonce N_M inside. It can then respond to the challenge by demonstrating to R that it was able to extract the nonce $N_R^{'}$.

M5: $M \Rightarrow R : \{M, R, N_R^{'}\}_{K_{MR}}$

Several variations to this example protocol are possible. For example, in message **M5**, R can allocate a temporary identifier to M as in [M^{+}94] to be used in subsequent messages from and to M while it is in R. The same K_{MR} is expected to be used for mutual authentication between M and R for subsequent connection attempts for the entire duration of M's stay in R. But R or M can force a re-authentication involving H at any time, should the need arise.

This protocol provides limited anonymity since an onlooker can only know that an entity from H

visited R but cannot determine the exact identity of the entity. It is scalable in that it can provide anonymity to an arbitrary level of granularity: instead of specifying H, M could have identified some parent domain H' of H and used $K_{H'}$ to encode its identity relative to H'. While H' shares no secret with M and therefore cannot verify the authenticity directly, it can form a trusted path to H and pass along the credentials. This protocol also takes the asymmetry of computing power into account: the mobile entities transfer the burden of verifying authenticity to their home domains. Even if the inter-domain key distribution protocol was based on a PKCS, the expensive task of verifying a chain of certificates provided by R will fall on H and not on M. There is only one public key operation (encryption using H's public key) that M has to perform. M has to store a small, finite number of keys in its permanent storage(e.g. K_{MH} and K_H). It also needs to store K_{MR} (or enough information to re-compute K_{MR}) for the duration of its stay in R.

7.3 Related Work

David Chaum and others have done extensive work on unconditionally anonymous protocols to implement, among other things, the digital equivalent of cash. The basic theme of security (for providing and/or using services and products) without identification is expounded by Chaum in [Cha85]. By requiring that the security of a transaction not be dependent on establishing the identities of participants to one another, it is possible to design protocols that achieve complete unconditional anonymity. Unconditional anonymity also opens the possibility of "perfect crime" (see [Sch94, page 123] for a reference). Chaum uses the phrase "limited anonymity" in [Cha81] to denote the anonymity provided by a special set of protocols. These protocols, like the unconditional anonymity protocols, preserve the anonymity of participants from onlookers and from each other. However, if necessary (for example, by a court warrant), the anonymity of a transaction can be undone at a later time.

In this report, I assume that the legitimate parties involved in a transaction can demand to know the real identities of the other parties. The concern here is to protect the identities of certain parties involved in the transaction from other onlookers who are not legitimate parties to the transaction. By "limited anonymity" I mean that some information about the identity of a participant is leaked to an eavesdropping third party.

The ideas presented here are intuitive. Similar ideas have been used in different contexts. In [ML94], Maxemchuk *et al* describe various protocols for providing anonymity using a building block called "Double Locked Box Protocol." Using this protocol, an entity A can send a message to another entity B via an intermediary I. A starts by sealing the identity of B in a box that only B's computer can open. This box is placed inside another box, along with the identity of B's computer and sealed so that only the intermediary I can open the outer box. The box is then handed to A's computer. A's computer cannot see what is inside but knows that it needs to be passed to I. I can get no information about the sender or recipient except the identities of their computers. B's computer, which receives the inner box from I knows nothing about the sender. As is evident, the basic idea is that by disclosing a limited amount of information in the clear, significant anonymity can be obtained.

7.4 Summary

Anonymity has not been a major issue in traditional distributed systems like the Internet. But the advent of mobile computing has provided strong motivations to address this problem. Providing complete anonymity is often hard, infeasible or undesirable. This document has made suggestions that can help provide limited but practical anonymity by using *limited disclosure* of information. Provision of anonymity, and security issues in general, must be taken into account while mobile computing environments are being designed.

A correctness proof of protocol **P1** using the BAN logic [B+89] appears in Appendix A. However, the proof only establishes that the authentication protocol is correct.

This work needs extensions in several directions. Existing techniques to analyze the correctness of security protocols (such as BAN logic) are not powerful enough to draw formal conclusions about the preservation of anonymity. Further investigation is necessary to determine how this can be achieved.

Another extension is to develop a classification scheme for various kinds and levels of anonymity. Such a classification will be useful in determining what levels of anonymity service can be justifiable in a given context.

8 Global Authentication

8.1 Motivation

Access to computers and communication networks has risen dramatically in recent years. A wide variety of services is now available from servers located at different parts of a large network like the Internet. It is customary for groups of people in different locations to work together on short- or long-term projects, requiring them to access services at remote locations. Absence of a global authentication mechanism (i.e. a mechanism to authenticate across domains) results in either inadequate authentication (making both the servers and clients vulnerable to attacks from intruders) or inconvenience to the users.

Another notable development is the interest in mobile computing. It is envisaged that a user will be able to access services regardless of their location. When a user pops up in a remote domain and demands services, the server will have to verify the user's authenticity as a first step. The ability to perform global authentication is indispensable to support mobile clients.

8.2 Survey of Solutions

The basic idea behind global authentication is the generalization of the intra-domain authentication mechanism described above. To prove its identity to a server S, a potential client C has to provide an *authentication chain*:

$$\{E_i, i = 0 \ldots n+1\}$$

Each E_i is an entity in the system. E_{i+1} should be able to trust the capacity of E_i to assure the authenticity of E_{i-1} (for values of i in $1 \ldots n$). C is entity E_0 and S is entity E_{n+1}. The process of authentication with three participants is identical to the one described in the intra-domain authentication case. This implies that two adjacent entities in an authentication chain need to have prior knowledge of each other. Typically, E_i is an AS for domain i; prior knowledge of AS E_j in domain j means that domains i and j had arranged for prior cross-registration of ASs.

Gligor *et al* [G$^+$92] identify the following desirable properties that a solution to this problem should possess. It should be *simple* (e.g: requiring few new types of interactions between clients and servers beyond those required by authentication within a domain). It should be *general* enough to not require the existence of strict trust hierarchies. It should be *transparent* to the end users, hiding the myriad details of the authentication protocol from them. Some of these goals are contradictory, thereby making appropriate trade-offs necessary.

At one extreme is a scheme like *Pretty Good Privacy(PGP)*, which is a public domain software package based on asymmetric key cryptography [Zim94]. PGP provides no transparent key distribution mechanism. It uses the "web of trust" model. Each user keeps a personal *public key ring.* Some of these public keys may belong to entities that are trusted to vouch for the authenticity of other public keys. The degree of such trust can be specified on a per-key basis. One can add new keys provided they are signed by these trusted entities. This model is completely general in that it imposes no particular trust hierarchy whatsoever. Each user is free to choose his/her own. It does not support the notion of inter-domain authentication: in effect, each user is in a private domain and there is no means to communicate with someone whose public key is not available in the personal public key ring. Not surprisingly, such a system does not scale well. As PGP becomes more prevalent, the need for a convenient key distribution mechanism has become increasingly apparent. There have been attempts to build "key servers" using electronic mail and other means of network access.

Most of the other solutions have a clearly defined notion of a domain. In the trivial case, if complete cross-registration of domains were possible, the authentication chains would contain at most two domains. But cross-registration of a large number of domains is neither feasible nor efficient. The questions then become, "Given the name of a server, how can a client construct an authentication chain that will be acceptable to that server?" and "Presented with an authentication chain from a client, how can a server verify that the chain is acceptable?" All the current solutions mentioned in the literature attempt to address these questions by assuming the presence of trust hierarchies.

Entities in a distributed system need to be named. The most prevalent, scalable, system of nomenclature corresponds to a tree. The Internet uses such a hierarchical naming scheme to identify Internet hosts and other network layer entities. Entities are grouped into domains, such as the EDU domains which consists of US educational institutions or the CA domain which consists of Canadian organizations. The top-level domains are further subdivided in a hierarchical fashion, leading to domains like UWaterloo.CA and MFCF.UWaterloo.CA. The domains, particularly near the leaves of the tree, often correspond to administrative groupings. This last observation has been exploited to develop a mechanism to derive and verify authentication chains, as described below.

SPX [TA91] imposes a strict policy for inter-domain authentication. SPX supposes the presence of a forest of sub-trees of the naming tree, forming trust hierarchies. A trust descendant will trust its ancestor to certify other domains whose names are subordinate to that of the parent. Special "cross-certifying CAs" are trusted to certify any CA to any other. Only one cross-certifying domain will be allowed in an authentication chain. Despite its obvious limitations, this model of inter-domain authentication allows for the generation and verification of authentication chains in a convenient

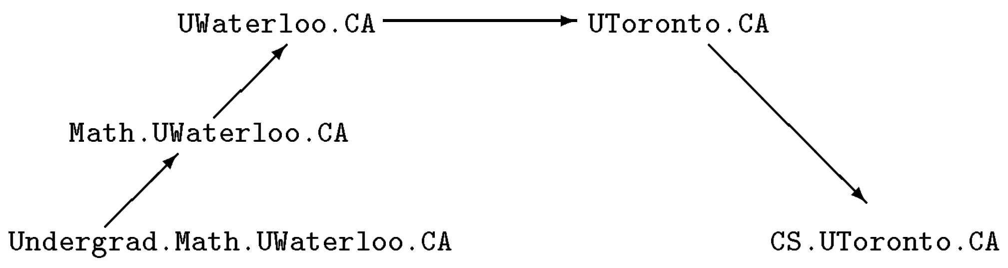

Figure 1: Chain derivation assuming an up-across-down policy

manner: given the name of a remote domain, it will be possible to automatically construct an authentication chain if one exists. This is accomplished by walking up the naming tree until an ancestor of the target domain is reached (either directly or using a cross-certifying CA) and then walking down the tree until the target domain is reached.

The SPX infrastructure is similar to the one recommended by ISO/ITU in the X.509 recommendation [OSI89]. Another implementation of a similar architecture is the one proposed for Internet Privacy Enhanced Mail [Ken93]. Current and future Internet proposals for various distributed system services are likely to built assuming the presence of the Internet PEM key-distribution architecture. However, PEM implementations are still at the experimental stage.

Kerberos [KN93] also allows for the possibility of authentication via an authentication chain (consisting of a sequence of domains) as described above. A client can pass the name of the target domain to a Kerberos library routine that can construct an authentication chain and obtain the necessary authentication certificates, *assuming* that the trust hierarchy coincides with the naming hierarchy. Such a chain may or may not be acceptable to the end server. The client has no way of predicting whether the server will accept the authentication chain. It is the server's responsibility to determine if the chain is acceptable. In other words, an entity may need to have a complete understanding of the global trust relationships in order to perform inter-domain authentication.

As an example using Internet style naming, suppose a client on domain

Undergrad.Math.UWaterloo.CA

wants to use a service provided by a server located on

CS.UToronto.CA.

The client will construct the authentication chain based on the names as in figure 1.

Gligor *et al* [G$^+$92] presented a formal characterization of the "up-across-down" policy of inter-domain authentication used in SPX and Kerberos. They define a trust hierarchy based on the *has-jurisdiction-over* relationship. If a domain A has jurisdiction over a domain B (denoted $A > B$), then any domain X will accept authentication from A about B (note that X will have to first ascertain the identity of A). Further, B will accept from A authentication about *any* domain X. The has-jurisdiction-over relation therefore defines a trust hierarchy. Separate trust hierarchies can be linked by the *peer-trust* relationship: if a domain A trusts a peer-domain X, A has a means of ascertaining X's identity. Therefore A can verify the authenticity of any trust descendant of X since an authentication chain can be constructuted betwen it and A via X.

They go on to argue that the trust hierarchy should have a constrained mapping to the naming

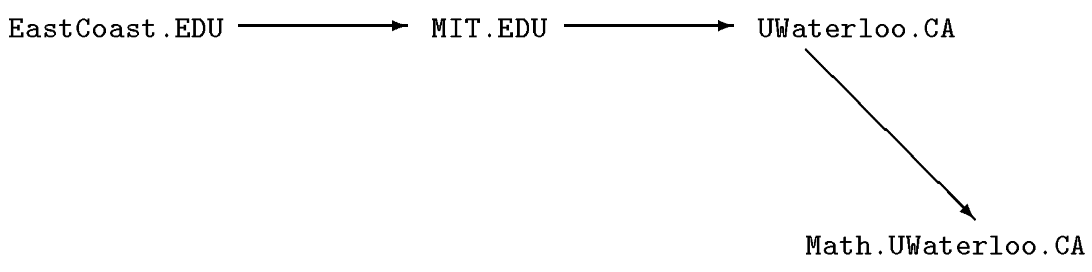

Figure 2: Chain derivation assuming flexible trust relationships

hierarchy (i.e. if $A > B$, then A should be the naming parent of B, but not vice-versa).

It follows that this policy, like that used in SPX or Kerberos, allows for authentication chains that traverse at most one peer-trust link. The policy can be implemented in a manner that is transparent to the user. It is possible to use chains that do not conform to this policy, at the expense of transparency: it is the responsibility of the client to generate and use an authentication chain that might be acceptable to the server. An even harder issue is for a server to decide whether every link in a certain authentication chain does indeed correspond to a trust parent-child or peer-peer relationship.

8.3 Problems with Existing Solutions

Assuming that the trust relationship in a distributed system has a strong correlation with the naming scheme does make it easy to implement a global authentication mechanism. While this assumption is reasonable in some environments, it is likely to be unacceptable in certain situations. Some of these are outlined below:

- **Bottlenecks in the naming tree:** In some places, the naming tree might have a very high fan-out. In the Internet, the EDU domain is an example of such a case. It is impractical to expect the EDU domain to have trust-parent relationships with each of its children. Consider the case of a well-connected (i.e. possessing peer-trust relationships with a large number of domains) child of the EDU domain, say MIT.EDU. Suppose further that EastCoast.EDU is a small institution which is cross-registered with only one outside domain, MIT.EDU. It is reasonable for EastCoast.EDU to choose to treat the well-connected MIT.EDU as a trust parent since it will enable an entity in EastCoast.EDU to authenticate any other entity on the Internet. Imposing a strict trust hierarchy based on the naming hierarchy will preclude trust relationships such as these. This sort of an arrangement is common in international diplomacy: when a small country cannot afford to maintain a direct diplomatic presence in another country, it may choose to operate through the embassy of a third country. Figure 2 illustrates the construction of an authentication chain like this.

- **Partial trust:** Recently, many commercial Internet providers have started selling Internet connectivity to individual users as well as organizations. An organization O that buys its Internet connectivity from such a provider P will be a child of P in the naming tree. However, it is reasonable for O to decide not to trust P's recommendations about the identity of a large majority of other domains. Further, O may prefer some or all other entities not to accept P's recommendation about the authenticity of O.

- **Commercial notarization:** Again, with increasing commercialization of widely accessible networks, it is possible that vouching for or verifying authenticity could be offered as commercial services. In this case, a flexible global authentication mechanism will be necessary so that an organization can specify varying trust relationships and change them when it wants to.

- **Infrastructure:** Above all, requiring the trust relationships to conform to a rigid structure such as a tree necessitates the existence of adequate infrastructure before the scheme can be implemented. This is the most important reason that has kept from PEM from being widely available. Curiously enough, a restrictive global authentication mechanism needs greater infrastructural support than a more flexible one.

8.4 Features of a Flexible Global Authentication System

To address these issues, a flexible global authentication system is required. This suggests the following issues.

- **Expression of trust relationships:** The first step in designing a flexible global authentication mechanism is to investigate a scheme for specifying trust relationships. The solutions described earlier often made implicit assumptions about trust relationships; these specifications were often too strong (such as $A > $ B) or inadequate. The scheme should be flexible enough to express various degrees of trust.

- **Propagation of trust information:** Next, a suitable protocol must be developed to propagate the information about trust relationships among the entities in the distributed system. Techniques used for routing in networks may be suitably modified for this purpose. Recently, protocols have been developed to facilitate routing, subject to administrative policy restrictions([Est89], [Ste93]). A crucial aspect of the protocol is its own security. Existing routing protocols have minimal or no security associated with them [KC93].

- **Evaluation of trust and mistrust:** Trust-related information (public keys, certificates, recommendations about trustworthiness) about an entity will reach a server in various ways. The trust relationships mentioned earlier will constitute the policy to be used in deciding which bits of this information are to be believed and used. It is necessary to devise a way to evaluate the trustworthiness of information. It will also be necessary to develop heuristics that can be used to evaluate the trustworthiness of information that is obtained from a hitherto unknown entity (i.e. when no *a priori* trust relationship exists).

- **Enforcement of trust relationships:** Finally, a suitable mechanism for the enforcement of trust relationships is needed. Convenient interfaces are needed for clients to extract trust information and construct authentication chains, and for servers to test the validity of trust relationships claimed in an authentication chain submitted by a client.

It will be worthwhile to design and build a mechanism to facilitate the expression and enforcement of trust relationships in distributed systems. Finally, note the following additional observations regarding the design of of such a system to facilitate flexible global authentication.

- Such a system can and should be sufficiently independent of specific authentication protocols. This will in fact enable inter-operability with multiple authentication protocols. It is possible that, at least in the short term, different domains will choose to adopt different authentication systems. Some may be capable of supporting multiple protocols. The only requirement of the protocols used by entities in a valid authentication chain is that any three consecutive entities in the chain should have at least one common protocol among them.

- The nature of trust in human relationships is a complex, and perhaps a not very well understood subject. It is necessary to suitably constrain the subject so that the issue of trust in distributed systems becomes a tractable problem. The issue of assigning quantitative measures to primary (i.e. not derived) trust relationships is beyond the scope of an automated system. It will be the responsibility of human administrators to initialize and maintain these measures. Derivations of trust relationships based on these primary relationships can be automated to some extent. Further investigation is necessary on this subject.

9 Conclusions

The advent of mobile computing has underscored many of these issues that are yet to be solved satisfactorily. It has also introduced some new problems. In preparation for more extensive research, the following steps have been completed as described in this document.

- A study of the security-related issues in large distributed systems that support mobile computing and wireless networks (Sections 1, 2, 3, 4),
- A study of the existing/proposed solutions and their strengths/shortcomings (Sections 5, 6),
- Identification of potential research issues (Section 6), and
- Elaboration of some of these issues and proposed new solutions (Sections 7, 8).

In particular, the issue of anonymity has been explored further and some solutions have been proposed based on the notion of *limited disclosure* of information. An example authentication protocol for mobile computing environments based on the proposed solutions has also been developed. This work was presented at the First IEEE Symposium on Mobile Computing Systems and Applications [Aso94].

The goal of the proposed research is to:

- Identify issues and problems that arise in designing a comprehensive security infrastructure to support mobile computing,
- Explore each issue, in particular the trade-offs involved,
- Propose a design for a scalable security infrastructure, and
- Prototype (and if possible, fully implement) the proposed design.

Currently, two major issues have been identified. Immediate avenues of investigation include the following.

- **Classification of anonymity levels:** The level of anonymity required or provided is likely to span a wide range. A comprehensive scheme for classifying the various levels of anonymity can help to measure, compare, and reason about the anonymity service. The earlier notion of limited disclosure suggests that an intuitive approach is to base the classification scheme on two independent parameters: classes of entities and the amount of identity-related information they know. A particular level of anonymity can then be described as a mapping from the first dimension (classes of entities) to the second.
- **Authentication protocols providing various levels of anonymity:** The basic authentication protocol described in Section 7 can be extended to provide varying levels of anonymity.
- **Techniques for reasoning about anonymity:** A way to classify and measure anonymity can be the first step in reasoning about anonymity preservation in security protocols. Intuitively, preservation of anonymity can be argued as follows. First, describe the desired level

of anonymity using a suitable notation (e.g., the one developed for classifying the levels of anonymity). Then, encode the initial assumptions and the protocol steps as statements in a suitable logic. Then, demonstrate that after each step in the protocol, the level of anonymity does not fall below the specification. That is, each class of entities should not acquire more information than intended.

- **Techniques for trust derivation:** In a system that supports flexible trust relationships, it will be necessary to have techniques by which an entity can estimate the trustworthiness of another entity. In the context of mobile computing, a remote domain server and a mobile user will need to make estimates about the trustworthiness of each other before the mobile user can make use of available services. This evaluation of trust can be implemenented based on a chain of *recommendations* [Y^+93, Y^+94]. The first issue in using such a chain is to determine how the trust values associated with each link in the chain can be combined.

  A second avenue of investigation is to carry out a comparative analysis (e.g., using simulations) to study the effects of a trust hierarchy on the performance of inter-realm authentication systems. The primary reason cited for preferring a trust hierarchy (as opposed to more flexible arrangements) is the resulting simplicity and potential increase in efficiency of the authentication protocols [G^+92].

It is possible that some of the issues may have to be investigated in a more general context. For example, investigation into anonymity issues may evolve to include more than protocols in mobile computing.

If the research is successful, I hope to have

- proposed a new method for classifying, comparing, and reasoning about anonymity in mobile computing environments, and possibly in a more general sense as well,
- proposed a general framework for specifying trust relationships between entities in a distributed system and deriving new trust relationships using these specifications during protocol execution, and
- used the techniques developed to propose a more flexible and scalable security infrastructure that supports mobile computing.

References

[AD94] Ashar Aziz and Whitfield Diffie. Privacy and Authentication for Wireless Local Area Networks. *IEEE Personal Communications*, 1(1):25–31, First Quarter 1994.

[Aso94] N. Asokan. Anonymity in a Mobile Computing Environment. In *Proceedings of Workshop in Mobile Computing Systems and Applications*, pages 200–204, Santa Cruz, California, December 1994.

[B+89] Michael Burrows et al. A Logic of Authentication. Technical Report 39, Digital Systems Research Center, February 1989. Revised February, 1990; Appendix (The Scope of a Logic of Authentication) May, 1994.

[B+93] Michael Beller et al. Privacy and Authentication on a Portable Communications System. *IEEE Journal of Selected Areas in Communications*, 11(6):821–829, August 1993.

[B+94] Trevor Blackwell et al. Secure Short-Cut Routing for Mobile IP. In *Proceedings of the Summer 1994 USENIX Conference*, pages 305–316, Boston, Massachusetts, June 1994.

[Bel89] S. M. Bellovin. Security Problems in the TCP/IP Protocol Suite. *Computer Communication Review*, 19:32–48, April 1989.

[Car94] Ulf Carlsen. Optimal Privacy and Authentication on a Portable Communication System. *Operating Systems Review*, 28(3):16–23, July 1994.

[CER95] CERT (Computer Emergency Response Team Coordination Center). IP Spoofing Attacks and Hijacked Terminal Connections. Computer Emergency Response Team, CA-95:01, January 1995.

[Cha81] David Chaum. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[Cha85] David Chaum. Security Without Identification: Transaction Systems to Make Big Brother Obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[DP89] D. W. Davies and W. L. Price. *Security for Computer Networks.* John Wiley & Sons, Chichester, England, second edition, 1989.

[Est89] D. Estrin. Policy Requirements for Inter Administrative Routing. IETF Network Working Group, RFC 1125, November 1989.

[ETS92] ETSI (European Telecommunications Standards Institute). DECT Common Interface: Part 7 - Security Issues. European Telecommunications Standards Institute, ETSI Standard ETS 300 175.7, October 1992.

[ETS93] ETSI (European Telecommunications Standards Institute). GSM - Security Related Network Functions. European Telecommunications Standards Institute, ETSI Standard GSM 03.20, October 1993.

[F+94] Y. Frankel et al. Fraud Prevention and Availability. Manuscript available in ftp://software.watson.ibm.com/pub/security/cdpd, October 1994.

[For94] Warwick Ford. *Computer Communications Security - Principles, Standard Protocols and Techniques*. PTR Prentice Hall, Englewood Cliffs, New Jersey, first edition, 1994.

[G+92] Virgil D. Gligor et al. Inter-realm Authentication in Large Distributed Systems. In *IEEE Symposium on Security and Privacy*, pages 2–17, Oakland, California, May 1992.

[Geo93] George H. Forman and John Zahorjan. The Challenges of Mobile Computing. Technical Report 93-11-03 (ftp://ftp.cs.washington.edu), Department of Computer Science and Engineering, University of Washington, December 1993.

[KC93] Brijesh Kumar and Jon Crowcroft. Integrating Security in Inter-Domain Routing Protocols. *Computer Communication Review*, 23(5), October 1993.

[Ken93] Stephen Kent. Privacy Enhancement for Internet Electronic Mail: Part II. IETF Network Working Group, RFC 1422, February 1993.

[KN93] John Kohl and B. Clifford Neumann. Kerberos Version 5 RFC. IETF Network Working Group, RFC 1510, September 1993.

[M+94] Refik Molva et al. Authentication of Mobile Users. *IEEE Network*, 8(2):26–34, March/April 1994.

[ML94] N. F. Maxemchuck and S. Low. The Use of Communications Networks to Increase Personal Privacy. Available from ftp://ftp.research.att.com/dist/anoncc, August 1994.

[NS78] Roger Needham and Michael Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[OSI89] OSI (Open Systems Interconnection). The Directory Authentication Framework. ISO/ITU, X.509 9594-8, 1989.

[Per94] C. Perkins. IP Mobility Support. IETF Network Working Group, Internet Draft, version 07, November 1994.

[Riv92] Ronald L. Rivest. The MD5 Message-Digest Algorithm. IETF Network Working Group, RFC 1321, April 1992.

[Sch94] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, New York, first edition, 1994.

[Ste93] M. Steenstrup. An Architecture for Inter-Domain Policy Routing. IETF Network Working Group, FC 1478, June 1993.

[TA91] Joseph J. Tardo and Kannan Alagappan. SPX: Global Authentication Using Public Key Certificates. In *IEEE Symposium on Security and Privacy*, pages 232–244, Oakland, California, May 1991.

[VK85] Victor Voydock and Stephen Kent. Security in High-Level Network Protocols. *IEEE Communications Magazine*, 23(7):12–24, July 1985.

[vT91] Johan van Tilburg. Secret-Key Exchange with Authentication. In *Computer Security and Industrial Cryptography, ESAT Course*, pages 71–86, Leuven, Belgium, May 1991.

[Y+93] R. Yahalom et al. Trust Relationships in Secure Systems - A Distributed Authentication Perspective. In *IEEE Symposium on Security and Privacy*, pages 150–164, Oakland, California, May 1993.

[Y+94] R. Yahalom et al. Trust-based Navigation in Distributed Systems. *Computing Systems*, 7(1):45–73, Winter 1994.

[Zim94] Philip Zimmerman. PGP User's Guide Volumes I and II. Available with the PGP distribution from various sources on the Internet, 1994.

A Proof of Protocol P1 using BAN Logic

A.1 The BAN Logic

Authentication is a process by which entities in a distributed system can prove their claimed identities to other entities. The goal of an authentication protocol is to guarantee that each verifier is convinced of the corresponding identities of the provers. Authentication protocols are hard to design. There have been several instances of "holes" being discovered in published protocols. A logic of authentication is used to reason about and prove the correctness of authentication protocols in a formal manner.

I use the Burrows-Abadi-Needham (BAN) logic [B+89] to prove the correctness of protocol **P1**. A short description of the relevant parts of the logic is given below. For a full description, see the original source [B+89].

The following additional notation is used in this section.

Notation
$\stackrel{K}{\longmapsto} A$ - K is a public key of A
$A \stackrel{K}{\longleftrightarrow} B$ - K is a secret key shared between A and B.

The following constructs are also used in the logic.

- A **believes** X - This is the central construct in the logic. This statement is true when A behaves as though X were true.

- A **sees** X - A sees X in a message sent by someone. A can read and repeat X but not necessarily believe it.

- A **once said** X - There is reason to believe A had, at some point in the past, generated the message X. However, A may not necessarily **believe** X at the current time.

- X **is fresh** - There is reason to believe that X was generated "recently", (e.g., during the current run of the protocol).

- A **controls** X - The principal A is considered to be the authority on X.

The following rules of the logic are relevant to this proof:

- **Message Meaning Rule**: If A gets a message $\{X\}_K$ from B and A believes that it shares the secret key K with B, then A can conclude that B once generated the message X. In other words,

  $$\frac{A \textbf{ believes } A \stackrel{K}{\longleftrightarrow} B,\ A \textbf{ sees } \{X\}_K}{A \textbf{ believes } B \textbf{ once said } X}$$

  An analogous version applies in the case of public keys.

- **Component Visibility Rule**: If an entity sees a message, it also sees the components of the message provided it knows the necessary keys. Example rules in this class are:

$$\frac{A \textbf{ sees } (X,Y)}{A \textbf{ sees } X} \quad \frac{A \textbf{ believes } A \overset{K}{\longleftrightarrow} B,\ A \textbf{ sees } \{X\}_K}{A \textbf{ sees } X}$$

  Analogous versions apply in the case of public keys.

- **Nonce Verification Rule:** If a *recent* message is known to have been uttered by a certain entity, then it is reasonable to conclude that the said entity believes the message.

$$\frac{A \textbf{ believes } X \textbf{ is fresh},\ A \textbf{ believes } B \textbf{ once said } X}{A \textbf{ believes } B \textbf{ believes } X}$$

- **Jurisdiction Rule:** If A believes that B has jurisdiction over X then A trusts B on the truth of X. In other words,

$$\frac{A \textbf{ believes } B \textbf{ controls } X,\ A \textbf{ believes } B \textbf{ believes } X}{A \textbf{ believes } X}$$

The first step in the proof is to derive an *idealized* version of the protocol from the description of the real protocol. The idealized protocol is intended to be a "clearer and more complete" specification of the real protocol. There are guidelines that help in this derviation process. For instance, any cleartext communication can be dropped from the idealized form since it does not contribute to the security of the protocol but merely serves as a hint. Steps in the idealized protocol need not have a one-to-one correspondence with the real protocol.

The second step is to identify the assumptions about the initial state. The proof process begins with this set of initial logic statements. After each step in the idealized protocol, applicable rules of the logic are used to derive new statements. A proof is successful if the statements representing the goals of the authentication protocol can be derived. In the case of two entities A and B wishing to authenticate each other and acquire a session key K to protect any subsequent traffic in the session, the goal is for each entity to believe that it shares a secret key with the other. That is, the goals of the authentication process are:

$$A \textbf{ believes } A \overset{K}{\longleftrightarrow} B,\ B \textbf{ believes } A \overset{K}{\longleftrightarrow} B.$$

A.2 The Real Protocol

The real protocol is as follows:

M1: $M \Rightarrow R : Cred_{MR},\ H,\ \{M, R, N_M\}_{K_{MR}}$
Where, $Cred_{MR} \ = \ \{M, R, H, \{T_M\}_{K_{MH}}\}_{K_H}$
M2: $R \Rightarrow H : \{R, Cred_{MR}, N_R\}_{K_{HR}}$, R
M3: $H \Rightarrow R : Tick_{HR}$
Where, $Tick_{HR} = \{R, H, M, K_{MR}, N_R\}_{K_{HR}}$
M4: $R \Rightarrow M : \{N_R^{'}, N_M, M, R\}_{K_{MR}}$
M5: $M \Rightarrow R : \{M, R, N_R^{'}\}_{K_{MR}}$

A.3 The Idealized Protocol and Assumptions

The idealized protocol is as follows:

M1: $M \Rightarrow R : \{N_M\}_{K_{MR}}$

M2: $R \Rightarrow H : \{\{\{T_M\}_{K_{MH}}\}_{K_H}, N_R\}_{K_{HR}}$

M3: $H \Rightarrow R : \{N_R, M \stackrel{K_{MR}}{\longleftrightarrow} R\}_{K_{HR}}$

M4: $R \Rightarrow M : \{N_M, N_R^{'}, M \stackrel{K_{MR}}{\longleftrightarrow} R\}_{K_{HR}}$

M5: $M \Rightarrow R : \{N_R^{'}, M \stackrel{K_{MR}}{\longleftrightarrow} R\}_{K_{HR}}$

The assumptions are as follows:

<u>**Keys**</u>

A1: M **believes** $\stackrel{K_H}{\longmapsto} H$
A2: H **believes** $\stackrel{K_H}{\longmapsto} H$
A3: M **believes** $M \stackrel{K_{MH}}{\longleftrightarrow} H$
A4: H **believes** $M \stackrel{K_{MH}}{\longleftrightarrow} H$
A5: R **believes** $R \stackrel{K_{HR}}{\longleftrightarrow} H$
A6: H **believes** $R \stackrel{K_{HR}}{\longleftrightarrow} H$
A7: R **believes** H **controls** $R \stackrel{K}{\longleftrightarrow} M$

<u>**Nonces**</u>

A8: M **believes** T_M **is fresh**
A9: H **believes** T_M **is fresh**
A10: M **believes** N_M **is fresh**
A11: R **believes** N_R **is fresh**
A12: R **believes** $N_R^{'}$ **is fresh**

<u>**Definition of K_{MR}**</u>

A13: $K_{MR} = h(K_{MH}, R, T_M)$, where $h()$ is a one-way hash function, K_{MH} is the shared secret key between M and H and T_M is a freshness indicator.

A.4 Proof

The definition of K_{MR} ($A13$), $A8$ and $A3$
$\Rightarrow$ **R1:** M **believes** $M \stackrel{K_{MR}}{\longleftrightarrow} R$
$M2$, $A6$ and the *Component Visibility Rule*
$\Rightarrow$ **R2:** H **sees** $\{\{T_M\}_{K_{MH}}\}_{K_H}, N_R$
$A2$ and the *Component Visibility Rule*
$\Rightarrow$ **R4:** H **sees** $\{T_M\}_{K_{MH}}$
$A4$ and the *Message Meaning Rule*
$\Rightarrow$ **R5:** H **believes** M **once said** T_M

$A8$ and the *Nonce Verification Rule*

$\Rightarrow$ **R6:** H **believes** M **believes** T_M

hence, using the definition of $K_{MR}(A13)$

$\Rightarrow$ **R7:** H **believes** $M \overset{K_{MR}}{\longleftrightarrow} R$

hence the inclusion of K_{MR} in $M3$ is valid.

$M3$ and $A5$ and the *Message Meaning Rule*

$\Rightarrow$ **R8:** R **believes** H **once said** N_R, $M \overset{K_{MR}}{\longleftrightarrow} R$

$A11$ and the *Nonce Verification Rule*

$\Rightarrow$ **R9:** R **believes** H **believes** $M \overset{K_{MR}}{\longleftrightarrow} R$

$A7$ and the *Jurisdiction Rule*

$\Rightarrow$ **R10:** R **believes** $M \overset{K_{MR}}{\longleftrightarrow} R$

Results $R1$ and $R10$ are sufficient. But stronger guarantees can be obtained due to messages $M4$ and $M5$:

$M1$, $R9$ and the *Message Meaning Rule*

$\Rightarrow$ **R11:** R **sees** N_M

hence the inclusion of N_M in $M4$ is valid.

$M4$, $R1$ and the *Message Meaning Rule*

$\Rightarrow$ **R12:** M **believes** R **once said** N_M, $N_R^{'}$, $M \overset{K_{MR}}{\longleftrightarrow} R$

$A10$ and the *Nonce Verification Rule*

$\Rightarrow$ **R13:** M **believes** R **believes** $M \overset{K_{MR}}{\longleftrightarrow} R$

$M5$, $R10$, $A12$, the *Message Meaning Rule*, and the *Nonce Verification Rule*

$\Rightarrow$ **R14:** R **believes** M **believes** $M \overset{K_{MR}}{\longleftrightarrow} R$

The derivation of result $R7$ needs a comment. It may require additional assumptions, such as "H **believes** M **controls** T_M," to go from $R6$ to $R7$. However I have not been able to identify a satisfactory assumption to capture the intent in this case. M does not really **control** T_M; nor does H believe it does. If T_M is a time stamp, then M simply assigns the current value of its clock to be T_M. H agrees to use T_M in constructing K_{MR} only if T_M is satisfactorily recent.