Formal Analysis of IP Layer Security*

Sarah Mocas and Tom Schubert

{sarah,schubert}@cs.pdx.edu

Department of Computer Science

Portland State University

Abstract

The use of formal methods during the development of a security protocol permits the designer to identify subtle assumptions that can lead to security flaws. Automated support for using formal methods can greatly assist this process, however, for many applications, correct formal modeling of a protocol and identification of its underlying assumptions is initially of greater importance than automated proofs. With this in mind, we model several different security mechanisms and protocols that are currently being developed for use at the IP network layer. Specifically we model the proposed IPSEC security extensions, AH and ESP, to determine the soundness of these propositions. In the context of Mobile-IP, we investigate the use of formal logics to examine IP layer security mechanisms but especially mobile node registration.

1 Introduction

An area of current interest is the addition of security to transactions over the network layer of the Internet. Inserting security mechanisms is possible at several different points, all the way from the application level to the hardware level. A natural location is at the IP level of the protocol stack. This particular idea is receiving much attention and several mechanisms for achieving this result have been proposed for possible standardization [2, 5, 6, 7]. There is also a natural combination of IP layer security mechanisms and mobile networking to create secure mobile networks (secure Mobile-IP). In this case, mobility within a network refers to the ability of a network entity, a *mobile node*, to send and receive datagrams using an IP address that is not recognized as local by the subnet in which the node is connected. This requires that there are

*Supported by the U.S. Department of Defense Advanced Research Projects Agency and the U.S. Air Force Rome Laboratory under contract F30602-95-1-0046.

additional entities that act as agents on behalf of a mobile node. In this context, a mobile node's IP address is used to indicate the identity of the node with respect to a home network (where its IP address is local). Datagrams destined for the mobile node continue to be sent to the mobile node's home network and are then forwarded on to the mobile node's actual (registered) location. Both the application of security mechanisms to the Internet and the development of mobile networks are evolving areas.

We have modeled several of the proposed IPSEC security extensions to determine the soundness of these propositions. Specifically, we examine the services that should be provided by the addition to an IP packet of the authentication header, AH, and the privacy header, ESP (encapsulation security payload). Through formal methods, we show where protocols using "secure packets" succeed in providing the service for which they are intended and where they fall short.

In the context of Mobile-IP, we are investigating the use of belief logics to examine IP layer security mechanisms. There are several security mechanisms needed for a reasonable implementation of a secure mobile network. Initially, mobile registration requires the use of an untrusted route to establish a secure tunnel. We model the registration services that are provided via a routing daemon using a UDP port.

Several logics for analyzing cryptographic protocols and authentication schemes have been proposed (see for example [1, 3, 4, 13, 14]). Belief logics are designed to consider what conclusions individual parties (principals) in a communication dialogue can deduce based on messages received and a set of initial assumptions or beliefs. Belief analysis attempts to show that only desired properties are guaranteed by the communication (data confidentiality, message authorship, non-repudiation, no replayed transactions, etc). Note that proofs about idealized protocols are not a guarantee that the concrete protocols are correct. There are many implementation assumptions that if invalid, would cause a secure, idealized protocol to actually be insecure. For example, these logics all assume that the cryptoalgorithm is secure.

Our logic for IP layer security mechanisms closely follows the logic used for analyzing protocols that was developed by Syverson and vanOorschot [13]. Syverson and vanOorschot's logic was chosen as it encompasses many of the desirable features from earlier developed logics in an integrated approach. Using our mechanization of SVO, we formally demonstrate the danger of preserving backward compatibility of the ESP header.

2 Background

The following standard definitions are primarily taken from [7], which outlines a security architecture for the Internet.

Authentication is the property of knowing that the data received is the same as the data that was sent and that the claimed sender is in fact the actual sender. A standard cryptographic hash algorithm that is used as a part of an authentication scheme is (keyed) MD5.

A slightly stronger property is given by a *digital signature* which, in addition to authentication, also guarantees that the message is uniquely associated with the sender and *unforgeable* by the recipient.

Integrity is the property of ensuring that data is transmitted from source to destination without undetected alteration.

Confidentiality is the property of communicating such that the intended recipients know what was sent but unintended parties cannot determine what was sent. *Encryption* is a mechanism used to provide confidentiality. In general, confidentiality is never a certainty[1] but a goal is to guard against known attacks against confidentiality. A standard symmetric encryption/decryption algorithm is DES.

Each of the above properties represent a "service" that may be provided by a set of cryptographic functions used in conjunction with a protocol and keying material. These services will be isolated to the IP layer and not rely on computation performed in higher layers. Strong cryptography is often weakened by subtle assumptions made during implementation. To guard against this type of weakness, security assumptions should not cross layer boundaries.

A *security association*, SA, is a relationship between a given set of network connections that establishes a set of shared security information. Typically, a security association will be specified by a group of secret keys, cryptographic algorithms, initialization vectors and usage modes for the algorithms. In practice, two entities that intend to communicate securely negotiate a SA for a particular connection or time period.

SPI is an acronym for "Security Parameter Index". It is an unstructured opaque index that is used in conjunction with the IP source address to identify a particular security association.

Note that neither a security association nor a SPI specify information about the usage of a protocol, they only give information about the implementation of the cryptographic algorithms. In all examples and arguments, we will assume that the cryptographic algorithms are computationally secure. Further material on cryptography can be found in [11].

[1] A one time pad is provably secure, this is an exception.

2.1 Basic IP Security Headers

The IP level security headers, AH and ESP, were originally proposed by the Network Working Group focused on IP security mechanisms, IPSEC [5, 6, 2, 7]. The term *IPSEC* will be used loosely to refer to packets, keys and routes that are associated with these headers. The IP Authentication Header (AH) is used to provide authentication. The IP Encapsulating Security Header (ESP) is used to provide confidentiality to IP datagrams. This paper will refer to two versions of the ESP header, an early version and a current version.

2.2 Managing Security Associations and Secret Keys

No one method of managing security associations and secret keys has been universally accepted but several ideas are important. A detailed discussion of Internet Security Association and Key Management Protocols, ISAKMP, can be found in [8]. In the simplest case, security association information and secret keys can be transfered in a secure manner to all parties statically. The harder case is the dynamic exchange of this information. ISAKMP assumes that entities that are negotiating a security association exchange digitally signed messages. Ideally, ISAKMP exists in an environment where there is an infrastructure available, such as one with certificates, that supports dynamically verifiable digital signatures. Digital signatures are sufficient to allow an authenticated exchange of messages for setting up an initial security association and the dynamic establishment of secret keys. The negotiated algorithms and keys are then used to encrypt/decrypt and authenticate a series of messages that set up a second security association that will be used for the subsequent set of datagrams. For example, algorithms and keys for use with the AH and ESP headers can be negotiated.

2.3 Mechanized Belief Logics

Belief logics[3] are designed to consider what conclusions individual parties (principals) in a communication dialogue can deduce based on messages received and a set of initial assumptions or beliefs. Logics devised to reason about cryptographic protocols generally consider only idealized protocols; there are no bit streams, but rather typed messages. Thus, all parties are presumed to recognize varying message formats, despite the lack of this information in the bit stream. Belief analysis attempts to show that only desired properties are guaranteed by the communication (data security, non-repudiation, no replayed transactions, etc).

Note that proofs about idealized protocols are not a guarantee that the concrete protocols are correct. There are many implementation assumptions that if invalid, would cause a secure, idealized protocol to actually be insecure. For example, these logics all assume that the cryptoalgorithm is secure.

Using the HOL system's type definition mechanisms, we define a number of application specific data types. HOL's recursive type definition facility [9] automates the process of defining new data types in terms of already existing types. Both new type constants and type constructors (operators) can be defined. Additional (recursive) functions can be defined to operate on the concrete data representation of the type. The properties of new types must be derived by formal proof. This guarantees that the type does not introduce inconsistency into the logic.

The SVO logic[13] defines a number of property predicates to describe the set of beliefs present in a system. This set may also include "facts" that may not be believed by all principals. Informally, a message can be *received*, but it may be encrypted. If a principal *has* the key, then the plaintext may be obtained. However, the principal has no assurance who the plaintext came from unless the message is signed. If a message is signed, then a principal can assume that the signing principal *said* the plaintext. But we don't know when the message was said, unless it is *fresh*, in which case, the principal can assume the signer effectively, *says* the message at the same time it is received (the message is timely, and thus not a replay). Still, the message may not come from an authority (e.g. "Simon says") unless the signer *controls* the message. If so, then whatever the message *asserts* is considered accurate.

Since performing proofs using only the above axioms is exceedingly tedious, we developed a collection of semi-automated tactics to simplify proof manipulations.

3 Formalizing Security Services

In our model, an IP packet is abstractly defined as a *(type, hash-data, encrypted-data, data-list)* tupple, where *type* is either NoProtection, ESP, or AH; *data-list* is a list of all of the fields in the IP packet; *hash-data* is a list of indices into *data-list* that indicate the data in the IP packet that is covered by a hash function[2]; and *encryption-data* indicate the data in the IP packet that is covered by an encryption function. Note that the value of a hash (*hash-value*), which results from hashing the data referenced by hash-data, is usually contained in the data-list. Likewise, for mobile registration, UDP packets are abstracted in the same fashion where the type is Mobile. Whether or not the UDP packet payload is a registration request or reply is determined by information in the payload. The intention is to prove that certain security services are provided based on what data is covered by a hash or encryption.

A security association is abstractly modeled as a table where the entries are a *principle*, which is considered to be an address; a *SPI*; a *type*, which is either ESP, AH

[2]The term hash-data is used loosely to refer to the data that is covered by an authentication transform which is not required to be a hash function.

or Mobile; and a *key-list*, which is an ordered list of keys. In each case discussed, a packet will include a source IP address and SPI which can be used to index the security association table so that the correct secret key can be retrieved.

3.1 Integrity

A primary use of a cryptographic hash function is to provide an integrity check over data. Informally, if A and B share a secret key, K, for use with a cryptographic hash function, H, then if A creates a message xyz where $H_K(x) = z$, on receipt of xyz, B can verify the integrity of x and z combined by computing $H_K(x) = z$. If either of these where changed then, assuming a cryptographically strong keyed hash function, B will detect the change. Using the abstract IP packet described above, the integrity of the hash-data, hash-value, IP source address and SPI can be verified as a unit. Since the source IP address and SPI are used to index the security association information and determine what key is used with the hash function, then if any of these fields were modified, it follows that the computation of the hash-value will be incorrect. This notion is mechanized HOL using two auxiliary definitions that validate the datagram AH hash with respect to a security association table and that the relevant field is covered by the hash.

```
 aPrincipal RECEIVED datagram
 datagram HASH_OK security_association_table
 message_field INDEXED_BY (hash-list datagram)
--------------------------------------------
 INTEGRITY message_field security_association_table aPrincipal
```

3.2 Authentication

Authentication of data is shown by proving that there is integrity over both the data and the source. This can be done directly, as in the AH header where the source IP address is included in the hash-data or indirectly as in the ESP and Mobile-IP headers. Again, if A and B share a secret key, K, for use with a cryptographic hash function H, then if A creates a message xyz where $H_K(x) = z$, on receipt of xyz, B can verify the integrity of x and z by computing $H_K(x) = z$. The important issue is the determination of the correct key K for use with a message. In the case of ESP and the Mobile-IP headers, the key is found by using the source IP address and SPI to find the corresponding entry in the list of security associations. Therefore, even though the source IP address and the SPI are not included in the hash-data, modification to these fields will result in, at best, a bad key and an incorrectly computed hash-value. Directly integrity is provided over the hash-data and indirectly integrity is provided over the source IP address and SPI, therefore, the hash-data is authenticated.

```
aPrincipal RECEIVED datagram
datagram HASH_OK security_association_table
message_field INDEXED_BY (hash-list datagram)
sender INDEXED_BY (hash-list datagram)
---------------------------------------------
AUTHEN sender message_field security_association_table aPrincipal

aPrincipal RECEIVED (encrypted mobile_reg_datagram)
(encrypted mobile_reg_datagram) HASH_OK security_association_table
message_field INDEXED_BY (encrypt-list datagram)
---------------------------------------------
AUTHEN sender message_field security_association_table aPrincipal
```

Encryption is sometimes incorrectly assumed to provide authentication. It is convenient to assume that if two parties, A and B, use a secret key (symmetric) cipher system, and secret key, K, then messages sent between A and B are also authenticated. We will assume that only A and B have the ability to encrypt/decrypt messages using K. In this case, A may be convinced that if she receives an encrypted message that decrypts correctly using K, and if she did not author the message, then B must have. Indirectly, the fact that only A and B own K supplies authentication. There are several flaws to this reasoning when this scenario is placed in the environment of the Internet. The most obvious flaw is that even though B may have encrypted the message, he did not necessarily send this copy of the message to A at this particular point in time. This type of replay attack is well studied. A second, subtler problem is that it is assumed in the above case that A can identify a correctly decrypted message. If the entity A is a set of algorithms used to provide a service at the IP layer, then this assumption is incorrect. The originally proposed ESP header did not verify the integrity of the cipher text or that decrypted data was meaningful. The "message" could in fact consist of random bits. Consequently there was no guarantee at the IP level that the decrypted data was the same data encrypted by the sender. One can argue that higher layers in the protocol stack will detect incorrectly decrypted header information and therefore there is a form of authentication that occurs. This is a poor form of authentication at best, since checks performed at higher levels typically have no cryptographic properties. Further, there is no guarantee that any checks are performed at higher layers. For example, if the decrypted data is passed to a UDP layer, a checksum may not even be used (this is OS kernel configuration, not application dependent).

3.3 Confidentiality

Informally, if A and B share a secret key, K, for use with a cipher system, then if A creates a message $E_K(p) = c$ where E is the encryption function, on receipt of c, B can decrypt c by computing $D_K(c) = p$. The message p is confidential if only A and B know K and $D_K(c) = D_K(E_K(p))$. We will show under one circumstance integrity is needed over the message c, the source IP address and the SPI to guarantee confidentiality but that there is also a method for using encryption such that encryption can be used to imply integrity over c, the source IP address and the SPI.

Consider showing confidentiality over a group of fields, c, in an IP packet. Since the decryption key is retrieved from the security association table using the source IP address and SPI as an index, then integrity over these fields implies that the correct key, K, is gotten. If the ciphertext that is received has not been modified in transit and the *correct* key is used for decryption, then $D_K(c) = D_K(E_K(p))$ will be true. Therefore, under the assumption that the secret key is known only to the sender and receiver, if there is integrity over the message c, the source IP address and the SPI, then this guarantees confidentiality for the plaintext p. Indirectly, the plaintext is also authenticated. This is the method by which confidentiality is supplied using the current version of the ESP header including the optional authentication mechanism provided with that header.

```
  aPrincipal RECEIVED datagram
  INTEGRITY (SRC datagram) security_association_table aPrincipal
  INTEGRITY (SPI datagram) security_association_table aPrincipal
  INTEGRITY message_field security_association_table aPrincipal
  isESPkey datagram k security_association_table
  message_field INDEXED_BY (encrypt-list datagram)
 --------------------------------------------
  CONFID message_field security_association_table aPrincipal
```

Another possibility does not require integrity over the received data in order to prove confidentiality of the message but instead uses encryption and verification of correct decryption to imply integrity. Suppose that A and B share a secret key, K, and that A creates a message $E_K(px) = c$ where x is an additional piece of information that the receiver knows or can calculate to verify that the message was correctly decrypted. On receipt of c, B can decrypt c by computing $D_K(c) = px$ and then check that x is correct. The message px is confidential if only A and B know K, B either knows or can compute x, and $D_K(c) = D_K(E_K(px))$. Two possibilities for the extra information x are: a second secret key K' shared only by A and B or an unkeyed hash of a part of p (a group of fields from an IP packet, before encryption is applied). In the case that x is a hash, the ability to recognize, by verifying the value of x, that the ciphertext was decrypted correctly implies that the correct key was used for decryption, hence for an IP packet the SPI and IP source address were not modified during transmission. Therefore, the plaintext is also authenticated.

4 The AH Header

The AH header consists of the following fields:

$$NextHeader|PayloadLen|Reserved|SPI|SeqNo|AuthenticationData$$

and is preceded by an IP header and followed by the data [5]. The $AuthenticationData$ field is the hash-value. In transport mode an AH packet is:

$$IPHeader|NextHeader|PayloadLen|Reserved|SPI|SeqNo$$
$$|AuthenticationData|Data$$

In our abstract packet format this is:

$$(AH, data-list, (IPHeader, NextHeader, PayloadLen, Reserved, SPI,$$
$$SeqNo, AuthenticationData, Data), ())$$

where data-list is the entire packet.

Since **all** fields in a packet using the AH header are included in the hash-data, it is easy to prove that integrity and authentication are provided over the entire packet. AH may also be used in tunnel mode (see [5]) but again since all fields are covered by an authenticator, such as a hash, it is easy to show that both integrity and authentication are provided for the packet.

5 The ESP Header

The newest version of the ESP header consists of the following fields:

$$SPI|SeqNo|Payload|Pad|PadLen|NextHeader$$
$$|AuthenticationData$$

and is preceded by an IP header [6]. In transport mode an ESP packet is:

$$IPHearder|SPI|SeqNo|Payload|Pad|PadLen|NextHeader$$
$$|AuthenticationData$$

where Payload is a TCP header followed by data. In our abstract packet format this is:

$$(ESP, data-list, (SPI, SeqNo, Payload, Pad, PadLen, NextHeader),\\ (Payload, Pad, PadLen, NextHeader))$$

where data-list is the entire packet. Note that ESP uses both an authentication transform and an encryption/decryption transform, so there will be two keys associated with an ESP security association. When a packet is sent, it is assumed that an encryption function is first used to create ciphertext form the contents of the encrypted-data fields, then an authentication function, such as a keyed hash, is used over the fields in the hash-data. The first version of the ESP header did not include an authentication mechanism [2].

In the earliest version of the ESP header, it was claimed that in addition to confidentiality, the ESP header also supplied authentication even though no authentication transform was used (see the discussion on authentication) [2]. This claim is misleading. The current proposed ESP header does not provide either authentication or confidentiality unless the optional authentication feature of the header is used. If the integrity of the ciphertext and/or the fields used to determine the correct security association and key are not guaranteed, then the decryption of the ciphertext may be incorrect but this may go undetected at the IP layer. In the worst case, only a block of the ciphertext has been modified, so decryption will continue and the resulting bits will be passed up the protocol stack. The use of the self-correcting Cipher Block Chaining Mode increases this possibility. In an attempt to make the latest version of the ESP header backward compatible, the use of a cryptographic hash over parts of an ESP packet is an optional default. As is mentioned in [6], if this option is not used, confidentiality is not guaranteed.

Processing the ESP header by first encrypting and then authenticating allows for fast detection of improperly constructed or replayed packages since only the authentication function needs to be computed. This is done to reduce the success of denial of service attacks and also allows processing of the hash and decryption transformations to be done in parallel [6].

5.1 Hash then Encrypt

The newest version of the ESP header first encrypts the encrypted-data and then applies an authentication transform to the hash-data, next we examine the reverse process. The idea is to encrypt some information, in this case, a hash that when decrypted, can be verified to be correct. The hash-value can be used to verify that the information that was decrypted is in fact the information that was encrypted. Note that in general, the hash-value does not contain any special properties for use in this verification process and that any value that was *expected* to be encrypted and then correctly decrypted would also suffice. In particular, a second shared secret key could be used instead of the hash-value in our proof. In this case the authentication computations are not needed. A shared secret key can also be used instead of a hash-value, while this is true it is not recommended for use with IP headers. Typically, encryption algorithms are used in combination with encryption block modes which allow an association between different encrypted blocks of the same message (see [11] for details). A commonly used mode, Cipher Block Chaining (CBC), has a self correcting property which would allow an attacker to cut and paste early encrypted blocks without effecting the decryption of later encrypted blocks. Hence, the blocks containing a key might decrypt correctly even though part of the ciphertext was modified. This type of attack highlights the importance of integrity checks over data in IP packets for guaranteeing confidentiality.

6 Mobile-IP

When a mobile node enters a non-local subnet, it requests the services of a foreign agent, to obtain a connection to its home subnet. This connection is established through the mobile node's home host or router, called a home agent. An initial protocol is used to register the mobile node's location with its home agent. It is necessary, even if further communications require no security, that control messages between a mobile node and a home agent that are used for mobile registration be authenticated [10]. Therefore, as a minimum requirement, a mobile node and home agent must share the basic information needed for a security association to support authentication during registration.

After registration, the next step in adding security features to a mobile environment consists of grafting security mechanisms that are currently proposed for use on the Internet. Specifically, additional IP layer headers are used to supply confidentiality and authentication over datagrams through binding security to routes. A mobile node that is "abroad" can be viewed as creating an extension of a local secure enclave. The mobile node and home agent, by definition, belong to one administrative/security domain.

6.1 Background

We will consider three types of functionality that can be associated with a network entity in an environment that supports mobility. Functionality is defined in terms of services that can be provided or are used. The following definition are similar to those found in [10].

A *mobile node*, MN, is a host or router that can change its point of attachment from one network or subnetwork to another. A change of location does not necessitate a change of IP address. Every mobile node has a fixed IP address on a fixed home network.

A *home agent*, HA, is a router or host which is capable of establishing a tunnel from a home network, on which it is located, to a mobile node which shares the same home network but is not physically located at the home network. Tunnels are used for the delivery of datagrams which are addressed to the mobile node's home network IP address.

A *foreign agent*, FA, is a router which supports the attachment of mobile nodes. In the context of this paper, a foreign agent supports wireless connections and therefore beacon its availability as an agent. A foreign agent provides routing services to any connected, registered mobile node. These services consist of establishing and maintaining a tunnel from a home agent to a mobile node; and as a default router for datagrams sent by the mobile node or for tunneling from the mobile node back to the home agent. Tunnels are unidirectional.

Note, any node may provide more than one type of functionality given that it contains the appropriate hardware and software. Therefore, it is practical to consider entities which function both as a home agent and a foreign agent or both as a foreign agent and a mobile node, etc. Based on the above definitions, it is not reasonable to consider an entity as both a home agent and a mobile node to the same subnet, since one by definition is located in its home network and the other is not. In this context, a node that is connected via a wireless connection directly to its home network is considered to be local and not mobile. For details on proposed mobile standards see [10].

6.2 Mobile Registration

There are several levels of security needed for a reasonable implementation of a secure mobile network and therefore, several different classes of cryptographic keys. The key material used for mobile registration will be referred to as the *Mobil-IP authentication key*. In addition, *IPSEC* keys, which are the keys associated with a SPI for using either the AH or ESP header need to be negotiated or fixed in advance of mobility. The last might be viewed as "routing" keys.

Mobile-IP registration requires that control messages between the home agent and mobile node are authenticated [10]. A general assumption is that the home agent and mobile node have agreed on the information needed for a *Mobility Security Association* in advance of registration. In this case, a SA, based on the Mobile-IP authentication key, exists between the home agent and mobile node prior to mobile registration. Optionally, messages between the mobile node and the foreign agent or the home agent and the foreign agent may also be authenticated depending in the security policy of these entities.

6.2.1 Mobile Node/Home Agent Authentication

If there is only authentication over control messages between the home agent and mobile node, it is assumed that foreign agents permit any mobile node attempt to register. During mobile registration, the mobile node sends authenticated messages to a home agent via the foreign agent. The foreign agent relies on the home agent's reply message which grants or denies permission for the registration. The security association between the mobile node and the home agent is based on a predetermined shared key or certificate.

The mobile registration request packet consists of the following fields:

$$Type|Flags|Lifetime|HomeAddress|HomeAgent|CareOfAddress|Id|Ext$$

and is preceded by an UDP header [10]. In this case, Type will indicate either a registration request of reply message. Home Address is the mobile node's address in its home subnet. Home Agent is the IP address of the home agent. Care Of Address is either the IP address of the foreign agent or a temporary "co-located care of address" externally assigned to the mobile node within a foreign subnet. Id's are unique to each registration request. The Mobile-Home Authentication Extension is:

$$ExtType|Len|SPI|AuthenticationData$$

For our purposes, the Ext Type indicates that the extension is either for Mobile-Home, Mobile-Foreign or Foreign-Home authentication. The Authentication Data is the result of using an authentication transform over parts of the UDP payload and has been referred to as the hash-value in this paper. A UDP mobile registration packet is:

$$UDPHeader|Type|Flags|Lifetime|HomeAddress|HomeAgent$$
$$|CareOfAddress|Id|EarlierExt|ExtType|Len|SPI|AuthenticationData$$

In our abstract packet format, a registration message is:

$$(Mobile, data-list, (Type, Flags, Lifetime, HomeAddress, HomeAgent,$$
$$CareOfAddress, Id, EarlierExt, ExtType, Len), ())$$

where data-list is the entire packet. The mobile registration reply packet consists of the following fields:

$$Type|Code|Lifetime|HomeAddress|HomeAgent|CareOfAddress|Id|Ext$$

and is preceded by an UDP header [10]. Code indicates the status of the registration request with the same Id. The Mobile-Home Authentication Extension format is the same as that for registration request messages. A UDP mobile registration reply packet is:

$$UDPHeader|Type|Code|Lifetime|HomeAddress|HomeAgent$$
$$|CareOfAddress|Id|EarlierExt|ExtType|Len|SPI|AuthenticationData$$

In our abstract packet format a registration reply message is:

$$(Mobile, data-list, (Type, Code, Lifetime, HomeAddress, HomeAgent,$$
$$CareOfAddress, Id, EarlierExt, ExtType, Len), ()).$$

In either the case of registration or reply there is an authentication transform used over all of the fields in the UDP packet except the UDP Header, SPI and Authentication Data. As with the IP security headers, the SPI and, in this case, Home Address (or Home Agent for reply) are used to index information for the security association and specifically the key. Integrity is supplied directly over the UDP payload (except the SPI and Authentication Data) and indirectly over the SPI when the Authentication Data (hash-value) is computed correctly. Since there is integrity over the source address then the packet fields contained in the hash-data are authenticated.

6.2.2 Optional Authentication

In addition to the Mobile-Home authentication extension, the optional Mobile-Foreign and Foreign-Home authentication extensions are defined. Based on security policy, a FA, MN or HA may require that the identity of other entities be authenticated. For example, the function of the foreign agent is to provide a routing service and based on security policy the foreign agent may wish to refuse service to a home agent or mobile node. In this case registration messages from the home agent to the foreign agent, and/or from the mobile node to the foreign agent should also be authenticated. For example, a foreign agent may charge mobile nodes as customers for a routing service. As a result, a likely policy would require that mobile nodes prove their identity.

In the case of registration, to add authentication between the mobile node and foreign agent a Mobile-Foreign Authentication extension is added by the mobile node to the registration packet after the Mobile-Home Authentication extension. A UDP mobile registration packet is:

$$UDPHeader|Type|Flags|Lifetime|HomeAddress|HomeAgent$$
$$|CareOfAddress|Id|EarlierExt|ExtType|Len|SPI|AuthenticationData$$
$$|EarlierExt|ExtType|Len|SPI|AuthenticationData$$

In our abstract packet format a registration message form the mobile node to the foreign agent is:

$$(Mobile, data-list, (Type, Flags, Lifetime, HomeAddress, HomeAgent, CareOfAddress, Id, EarlierExt, ExtType, Len, SPI, AuthenticationDataEarlierExt, ExtType, Len), ())$$

where data-list is the entire packet. The foreign agent will remove and process the last extension. Note that if there is a pre-existing security association between the mobile node and the foreign agent then showing authentication here is no different then showing that it holds in the case of the mobile node and home agent.

It seems unreasonable to establish shared symmetric keys between the mobile node and foreign agent or even the foreign agent and home agent prior to registration. In the broadest terms, mobility assumes that future locations are not determined in advance. Assuming a certificate infrastructure a foreign agent is able to authenticate either the home agent or mobile node or both in a scalable fashion. If the foreign agent gets a signed registration request and possibly also a certificate from the mobile node then the foreign agent can checks its own CA (certificate authority) to see if the request is signed correctly and if so then decide whether or not to permit registration. Similarly, the foreign agent can request that control messages from the home agent are signed and that a certificate is included. This can be used to verify the identity of the home agent. Using a certificate infrastructure allows the foreign agent to verify signatures when services are requested, so the foreign agent need not have shared secret keys for all potential users in advance.

In particular, using digital signatures would change the above registration packets in that the Authentication Data might now be an un-keyed hash (of the same fields) that was digitally signed. Optionally, a public key might be added in an additional extension. This is an interesting area for future examination.

6.2.3 Tunnel Security

At the end of registration, message confidentiality over datagrams sent from the home agent to the mobile node is implemented through the use of the ESP header and associated encryption/decryption transforms. IPSEC keys may be exchanged between a home agent and a mobile node in advance of mobility or may be negotiated after registration. Such negotiations are outlined in [8]. Typically, an initially agreed on key should be used only to encrypt additional keys that are used with the IPSEC headers. It is standard practice to change data encryption keys regularly.

The addition of ESP to Mobile-IP tunnels is similar to adding ESP to IP packets except ESP in this case is used in tunnel mode. In tunnel mode an ESP packet is:

$$NewIPHeader|SPI|SeqNo|OldIPHeader|Payload|Pad|PadLen$$
$$|NextHeader|AuthenticationData$$

where Payload is a TCP header followed by data. In our abstract packet format this is:

$$(ESP, data-list, (SPI, SeqNo, OldIPHeader, Payload, Pad, PadLen, NextHeader),$$
$$(OldIPHeader, Payload, Pad, PadLen, NextHeader))$$

where data-list is the entire packet. Of course the source IP address for use with the SPI is gotten from the New IP Header. Again, if the optional authentication is used, then this header and security mechanism can be shown to provide authentication and confidentiality over the encryption-data.

This alone only shows that from within a secure enclave, messages can be sent securely to a mobile node. There are interesting problems with allowing the home agent to forward messages to the mobile mode from a source outside of the secure enclave. This allows the possibility of a chosen plaintext attack since an attacker could observe the encrypted messages traveling from the home agent to the mobile node. This is another area for future interesting work.

7 Support for Mechanized Theorem Proving

Using HOL90, we have defined a number of new data types to describe IP datagrams, IPSEC security mechanisms, and mobile registration messages. Performing proofs however, remains a human-intensive process. To better support the use of our mechanized logic, we have developed a general infrastructure base, based on earlier work[12], for creating automated and interactive procedures to prove goals about inductively defined relations.

The infrastructure provides tools to build interactive functions and goal-directed support functions, procedures to eliminate existentially quantified variables from terms, and tactics to generalize existentially quantified rules. Many of the functions are generic, expecting as arguments the list of inference rules returned by `new_inductive_definition` and a list of the relevant combinators. In our case, the list of relevant combinators includes all of the type constructors from the newly defined types (`key`, `keyuse`, `keyfact`, `item`, and `proposition`).

For many applications, validation of the inductively defined rules is, at least initially, of greater importance than automated proofs. For the automated proofs to be useful, their construction must be understood. Thus, it is valuable for the automated procedures to inform the user what rules are relevant and when rules are applied to a goal.

Interactive support is provided by a collection of parameterized functions that suggest an appropriate rule to apply based on the current goal or a term. (`suggested_rules()` and `rules_for_term()`, respectively). Using these mechanisms, users don't need to remember all the rules that may apply, but can request a list of potentially useful rules. Many protocol logics have a significant number of inference rules ([4] lists 44 rules). Even SVO logic, which was designed to significantly reduce the number of rules compared with other protocol logics, requires 23 rules.

We have also developed a tactic generating function, `make_suggest_tac` that returns a tactic specialized for a list of inference rules and a list of combinators. The tactic searches for and then applies, an appropriate rule to apply based on the current goal. In practice, this tactic appears to provide a significant improvement over tactics that try every rule. The tactic also outputs what rules where chosen to apply in a given situation. The displayed list of decisions can assist the user understand why a proof is correct and often, why it fails.

A second general tactic EXISTS_ELIM_TAC has also been developed to search for witnesses to replace existentially quantified variables. The tactic searches the assumption list for candidates that match the variable's usage in the goal. After replacement, goals can often be reduced by rewriting with the assumption list. For performance considerations, the tactic defers rewriting until all existentially quantified variables have been replaced.

Using the above two tactics, a general strategy was developed and programmed into a tactic (IND_RULES_TAC). The initial proofs of the lemmas required a fair number of steps, but with the use of the IND_RULES_TAC, the proofs are performed with only a few tactics. The resulting proofs also require less time with fewer intermediate theorems generated. At this point in development, the tactic does not backtrack and so occasionally, the user must select the correct rule.

The final general support mechanism developed supports generalizing some of the rules returned by `new_inductive_definition`. When creating a new inductive definition, there are often rules with existentially quantified variables in the antecedent that could instead be universally quantified. While this is not always beneficial, in many cases, proofs are simplified.

8 Conclusion

In this paper, we have described proposed IPSEC security mechanisms (AH/ESP) and their use to provide integrity, authentication, and confidentiality use in securing mobile computing. By formalizing definitions of integrity, authentication, and confidentiality, we have shown under what contexts these properties are and are not preserved based on what data is covered by a hash or encryption. In the context of Mobile-IP, we have developed a mechanization of the mobile registration protocol that uses an untrusted route to establish a secure tunnel.

Using the HOL90 system, these definitions have been mechanized and correctness proofs constructed using a belief logic. Future work will focus on automating the decision procedures and mechanise our paper-and-pencil analysis of secure tunnels for use with Mobile IP.

9 Acknowledgments

The authors would like to express there appreciation to Jim Binkley and John McHugh for many helpful conversations on Mobile-IP, especially the details of mobile registration and secure tunnels, and in general to the members of the Secure Mobile Networking Project at Portland State University.

References

[1] M. Abadi and M. Tuttle. A semantics for a logic of authentication. In *Proceedings of the Tenth ACM Symposium on Principles of Distributed Computing*, pages 201–216. ACM Press, 1991.

[2] R. Atkinson. *IP Encapsulating Security Payload (ESP)*. Network Working Group Request for Comments: 1827, August 1996.

[3] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1), Feb. 1990.

[4] L. Gong and R. Needhamand R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Computer Society Press, 1990.

[5] S. Kent and R. Atkinson. *IP Authentication Header*. Network Working Group, Internet Draft: draft-ietf-ipsec-auth-05.txt, July 1997.

[6] S. Kent and R. Atkinson. *IP Encapsulating Security Payload (ESP)*. Network Working Group, Internet Draft: draft-ietf-ipsec-esp-04.txt, July 1997.

[7] S. Kent and R. Atkinson. *Security Architecture for the Internet Protocol*. Network Working Group, Internet Draft: draft-ietf-ipsec–.txt, 1997.

[8] D. Maughan, M. Schertler, M. Schneider, and J. Turner. *Internet Security Associations and Key Management Protocol (ISAKMP)*. IPSEC Working Group, Internet Draft, February 1997.

[9] Tom Melham. Automating recursive type definitions in higher order logic. In G. Birtwhistle and P.A Subrahmanyam, editors, *Current Trends in Hardware Verification and Automated Theorem Proving*, pages 341–386. Springer-Verlag, 1989.

[10] C. Perkins. *IP Mobility Support*. Network Working Group Request for Comments: 2002, October 1996.

[11] Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, Inc, 1994.

[12] Tom Schubert and Sarah Mocas. A mechanized logic for secure key escrow protocol verification. In *International Workshop on the HOL Theorem Proving System and its Applications*. Springer-Verlag, August 1995.

[13] Paul F. Syverson and Paul C. van Oorschot. On unifying some cryptographic protocol logics. In *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy*, May 1994.

[14] P. vanOorschot. Extending cryptographic logics of cryptographic logics of belief to key agreement protocols. In *Proceedings of the First ACM Conference on Computers and Communications Security*, pages 232–243, 1993.