

Security Architectures Using Formal Methods

Colin Boyd

Abstract—A model describing secure communications architectures is developed using the formal language Z. The model is based on fundamental cryptographic properties. Some basic constraints are derived for the design of secure architectures which allow problems to be identified prior to design of security protocols. A simple criterion is derived for ensuring that all pairs of users can set up secure communications channels.

I. INTRODUCTION

IN the area of secure communications, the need for mathematical rigor has long been appreciated. Formality has been used at all levels from analysis of cryptographic algorithms up to definitions of security policy. However, the need to formally define security requirements, and to show that specific security protocols satisfy these requirements, has only recently been recognized.

One important recent piece of work in this direction is the *Logic of Authentication* of Burrows, Abadi, and Needham [2]. The authors define a formal logic specifically for the analysis of authentication protocols. A number of variations and enhancements to this work have been published [6], [8]. An alternative state-based method for protocol analysis has been developed by Meadows [9] which includes a tool for automatic analysis. It is a common feature of these approaches that they make very general assumptions about the cryptographic algorithms underlying the protocols which are analyzed.

A different approach has been taken by Rueppel [12]. He introduces a formal language that allows any security requirement to be decomposed into simple security mechanisms. These mechanisms can be realized as specific cryptographic techniques. The advantage of this approach is that it allows the description of many levels of design to be considered in the same language. However, the formal language does not have any formal semantics—the statements in the language have no logical meaning. As a consequence, it is not possible to formally analyze the descriptions obtained or prove any properties about them. The productions which are possible in the language have to be decided by the user in the light of the security mechanisms available.

In this paper, a formal model is developed for analyzing security designs using the formal language Z. In contrast to the formalisms mentioned above, the model is not intended for analysis of particular security protocols. Instead, its goal is to describe security architectures in terms of the secure channels available. The model may thus be useful at the design stage

before particular protocols are even considered. Some general and intuitively reasonable properties of security designs can be deduced from the model, and it is shown what types of security architectures are possible if secure communications are required. Once the formal model is established, a simpler graphical notation is introduced.

The model is based on the fundamental cryptographic properties of confidentiality and authentication, which also underlie Rueppel's language. In essence, the model describes what secure channels may be formed cryptographically once initial keys are in place. Some of the consequences of the model are intuitively obvious and are widely known as empirical facts. For example, it is shown that no secure channels can be formed between users who possess no secrets. However, it is useful to be able to prove formally that these intuitive properties do indeed always hold.

Apart from cryptography, the other way of providing secure channels is by physical means. All security services can be provided by suitable physical security measures; for example, all messages could be communicated by a trusted courier. Such measures have many drawbacks in a modern communications environment and are therefore normally used only to initialize the system (or periodically reinitialize it). For this reason, physical channels are not treated in this paper as part of the formal model. It is assumed that physical measures must be used to initialize the system in terms of initial distribution of secret keys.

Although the model could certainly have been developed using standard mathematics, there are advantages in using an established formal notation. The language Z has a sound and stable mathematical basis including tools to test for correct syntax and typing. A guide to the language Z is provided by Spivey [13]. The reader who has not seen Z before but is familiar with logic and set theory can follow the main points. A brief summary of the Z features used in the model is contained in the Appendix. An English commentary is provided with all the Z used in the text.

A. Secure Channels and Cryptographic Keys

A security architecture, called the OSI Security Architecture, has been defined as part of open systems communications standards [7]. This document defines a number of security services that might be required in a system, and a number of security mechanisms that can be used to provide these security services. The definitions are all given in English with no formal statements, and so it is not possible to prove whether or not a given system provides a particular security service. The need to make good this deficiency is the main motivation behind this and related work.

Manuscript received May 1992. This work was supported by SERC Research Grant GR/G19787.

The author is with the Communications Research Group, Electrical Engineering Laboratories, University of Manchester, Manchester M13 9PL, U.K. IEEE Log Number 9206681.

From the viewpoint of cryptography, there are only two fundamental services that can be provided, namely, *confidentiality* and *authentication*. All communications security services are concerned with the identity of the senders or recipients of information. Cryptography allows users to be identified by allocating secret keys to them. There are only two ways that cryptography can work, and these define the fundamental services (this characterization is used by Rueppel and also in [1]).

- *Confidentiality*—Only that user (or set of users) in possession of the secret key can read the message.
- *Authentication*—Only that user (or set of users) in possession of the secret key can write the message.

A *secure channel* can be thought of as a relationship between two system users which provides some security services. In the model of this paper, two basic types of secure channel are considered, namely, confidentiality channels and authentication channels. In addition, we define symmetric channels, each of which may coincide with a confidentiality channel, an authentication channel, or both.

For example, in a conventional symmetric cipher, a pair of users share the same key. This symmetric channel typically provides both a confidentiality and an authentication channel between a specific pair of users. Symmetric ciphers can also be used to provide authentication alone, as when a message authentication code is used [4]. A public key cryptosystem usually has one public key and a corresponding secret key. For certain algorithms, such as RSA [11], the public key may be used for confidentiality and the secret key may be used for authentication. However, for other algorithms, it may be that only one channel is provided. For example, signature schemes associated with zero knowledge protocols [5] provide authentication but not confidentiality, while schemes such as the McEliece algorithm [4] are suitable for authentication but not confidentiality.

There are three components that make up a *security architecture* in the model of this paper:

- users
- trusted users, including information on who trusts whom, and
- secure channels which may provide confidentiality, authentication, or both.

II. THE FORMAL MODEL

The formal model is now presented. It is a relatively simple specification and defines a state-based sequential system as described by Spivey [13]. The first line of the specification defines the abstract types which will be used. These are users and keys and are not defined further as they are fundamental values in the model. Other components, such as secure channels and trusted users, will be defined in terms of these sets.

$$\{User, Key\}$$

Next, some global, or axiomatic, definitions are made. These are things that are not expected to change within the model and so can be excluded from the rest of the system state. Five

sets of keys are defined concerning two separate properties. A key must be in exactly one of the sets *Public*, *Secret*, *Shared*; in other words, these sets partition the keys. In addition, a key must be a confidentiality key (in the set *Conf*) or an authentication key (in the set *Auth*) or both. The dual, or inverse, is defined for each key. Taking the dual of a key is a self-inverse operation. The dual of a confidentiality key is still a confidentiality key and similarly for authentication keys. Secret and public keys are interchanged under the dual map, while the dual of a shared key is still shared. The trusted users are defined by a map which defines those users trusted by each user.

$$\begin{aligned} & \text{Shared, Public, Secret, Auth, Conf} : \mathbb{P}Key \\ & \text{dual} : Key \rightarrow Key \\ & \text{Trusted} : User \rightarrow \mathbb{P} User \end{aligned}$$

$$\begin{aligned} & \langle \text{Shared, Public, Secret} \rangle \text{ partition } Key \\ & \text{Auth} \cup \text{Conf} = Key \\ & \text{dual} \circ \text{dual} = \text{id } Key \\ & \text{dual}(\text{Conf}) = \text{Conf} \\ & \text{dual}(\text{Auth}) = \text{Auth} \\ & \text{dual}(\text{Shared}) = \text{Shared} \\ & \text{dual}(\text{Public}) = \text{Secret} \\ & \text{dual}(\text{Secret}) = \text{Public} \end{aligned}$$

The first ordinary schema defines the variable that records what keys are known by each user, and with whom they are associated.

$$\begin{aligned} & \text{Keys} \\ & \text{keys} : User \rightarrow \mathbb{P}(User \times Key) \end{aligned}$$

Thus, to each user is associated a set of $(user, key)$ pairs, where if x maps to (y, k) , then this is thought of as meaning that x knows key k and uses it in communications with user y . The following three schemas define the state space of the model by giving formal definitions of secure channels in terms of possession of keys.

$$\begin{aligned} & \text{ConfidentialityChannels} \\ & \text{Keys} \\ & \text{ConfChannels} : User \leftrightarrow User \end{aligned}$$

$$\begin{aligned} & \forall x, y : User \bullet (x, y) \in \text{ConfChannels} \Leftrightarrow \\ & (\exists k : \text{Conf} \setminus \text{Secret}; z : User \bullet \\ & (y, k) \in \text{keys}(x) \wedge (z, \text{dual}(k)) \in \text{keys}(y)) \end{aligned}$$

Confidentiality channels define relations between pairs of users. These are ordered pairs as the channel may be in only one direction. It will be noticed that the definition is not symmetrical with respect to x and y . The predicate states that for a confidentiality channel to exist from x to y , there must be a key whose use includes confidentiality and is either shared or public. x must associate this key with y . y must know the dual of the key. In the model, this means that y must associate

it with some user(s), but it is not of any concern to y exactly which users know the key. To see why this is reasonable, consider a public key system providing confidentiality to y . Any user z who knows the public key of y has a confidentiality channel to y , and it is important to z that it is only y who has the secret dual key. y must of course know the secret key, but it does not matter to y who knows the public key. This corresponds to the viewpoint that confidentiality is a service provided to the sender of information.

AuthenticationChannels

Keys

$AuthChannels : User \leftrightarrow User$

$\forall x, y : User \bullet (x, y) \in AuthChannels \Leftrightarrow$
 $(\exists k : Auth \setminus Public; z : User \bullet$
 $(z, k) \in keys(x) \wedge (x, dual(k)) \in keys(y))$

Authentication channels also define relations between pairs of users. The definition of authentication channels is dual to the definition of confidentiality channels and corresponds to the viewpoint that authentication is a service provided to the receiver of information.

SymmetricChannels

Keys

$SymmChannels : \mathbb{P}(\mathbb{P}(User))$

$SymmChannels \subseteq \{x, y : User | x \neq y \bullet \{x, y\}\}$
 $\forall x, y : User \bullet \{x, y\} \in SymmChannels \Leftrightarrow$
 $(\exists k : Shared \bullet (y, k) \in keys(x) \wedge (x, dual(k)) \in keys(y))$

Symmetric channels are in both directions and so are defined as sets of two different users. They correspond to the situation where neither key is public, and in practice the key and its dual are usually equal.

State

ConfidentialityChannels

AuthenticationChannels

SymmetricChannels

The system state is defined exactly by what keys are known by each user, thereby defining what secure channels exist.

Transfer

$\Delta State$

$orig?, dest?, recip?, sender? : User$

$k? : Key$

$orig?, k?) \in keys(sender?)$
 $keys' = keys \oplus$
 $\{recip? \mapsto keys(recip?) \cup (orig?, k?)\}$

This schema says that if a key $k?$ is sent from one user to another, then the keys known to the recipient are updated to associate the key sent with the originator.

In this model, the only state changes are those which happen as a result of passing keys from one user to another. Such a key exchange may or may not result in new channels being formed. The key passes from the sender to the recipient. It may be that the users between whom communication is intended (originator and destination) are different from the sender and recipient involved in a particular exchange. This is the situation if the sender is a key server. The recipient will therefore associate the received key with the originator, who may or may not be the sender.

SecureTransfer

Transfer

$k? \in Secret \Rightarrow (sender?, recip?) \in ConfChannels$
 $k? \in Public \Rightarrow (sender?, recip?) \in AuthChannels$
 $k? \in Shared \Rightarrow (sender?, recip?) \in ConfChannels \cap$
 $AuthChannels$
 $orig? \neq sender? \wedge k? \in Public \cup Shared \Rightarrow$
 $sender? \in Trusted(recip?)$
 $recip? \neq dest? \wedge k? \in Secret \cup Shared \Rightarrow$
 $recip? \in Trusted(sender?)$

This schema details the conditions which must exist if a key exchange may be performed securely. The first is that secret keys may only be transferred over a confidentiality channel. The second is that public keys must be transferred over authentication channels. If the key is to be shared, the third condition tells us that the channel should provide both confidentiality and authentication since both users need to associate the key only with each other.

The last two conditions relate to trust. If the key is public or shared then the recipient must trust the sender, unless the sender is the originator. This is because the key must be correctly assigned to the originator. Similarly, if the key is secret or shared, then the sender must trust the recipient not to reveal it, unless the recipient is the destination.

A. User to User Key Exchange

In the case where two users exchange a key for use between themselves, the originator coincides with the sender and the recipient coincides with the destination. We shall call this a simple transfer.

SimpleTransfer

Transfer

$orig? = sender?$

$dest? = recip?$

A simple transfer can result in a new channel, the details of which depend on the kind of key transferred. The following schema illustrates one way this can happen. The code U1 in

the schema name refers to the summary of transition rules given in Section II-C.

SimpleSecretAuthTransfer—U1

Simple Transfer

$k? \in \text{Auth} \cap \text{Secret}$
 $(\text{dest?}, \text{dual}(k?)) \in \text{keys}(\text{orig?})$
 $(\text{sender?}, \text{recip?}) \in \text{ConfChannels}$

In this exchange, the key is secret and intended for authentication. The originator already knows the dual public key. The following theorem says that the previous state change allows a secure transfer which results in a new authentication channel. It can now easily be proved in the Z model.

Theorem 1:

$\text{SimpleSecretAuthTransfer} \vdash \text{SecureTransfer} \wedge$
 $\text{AuthChannels}' = \text{AuthChannels} \cup \{(\text{recip?}, \text{sender?})\}$

Proof: We need to show that all the conditions of *Secure Transfer* are satisfied. Since $k? \in \text{Secret}$, we need $(\text{sender?}, \text{recip?}) \in \text{ConfChannels}$ which is true since it is a predicate in *SimpleSecretAuthTransfer*. Since this is a simple transfer, the last conditions of *SecureTransfer* are always true so the whole schema is true.

Next we show that the new authentication channel exists. From the schema *Transfer*, we have $(\text{orig?}, k?) \in \text{key}'(\text{recip?})$. From the schema *SimpleTransfer*, we have $\text{orig?} = \text{sender?}$. From *SimpleSecretAuthTransfer*, we have $(\text{dest?}, \text{dual}(k?)) \in \text{keys}(\text{orig?})$. Since no keys are lost (from *Transfer*), this implies $(\text{dest?}, \text{dual}(k?)) \in \text{keys}'(\text{orig?})$. Hence, from the definition of *AuthChannels*, there is an authentication channel from *recip?* to *sender?*. \square

The next schema gives a way to form a new confidentiality channel between two users.

SimplePublicConfTransfer—U2

Simple Transfer

$k? \in \text{Conf} \cap \text{Public}$
 $(\text{dest?}, \text{dual}(k?)) \in \text{keys}(\text{orig?})$
 $(\text{sender?}, \text{recip?}) \in \text{AuthChannels}$

The corresponding theorem is as follows, with proof entirely analogous to the previous theorem.

Theorem 2:

$\text{SimplePublicConfTransfer} \vdash \text{SecureTransfer} \wedge$
 $\text{ConfChannels}' = \text{ConfChannels} \cup \{(\text{recip?}, \text{sender?})\}$

Symmetric channels may also be formed as in the following transfer.

SimpleSharedTransfer—U3

SimpleTransfer

$k? \in \text{Shared}$
 $(\text{dest?}, \text{dual}(k?)) \in \text{keys}(\text{orig?})$
 $(\text{sender?}, \text{recip?}) \in \text{AuthChannels} \cup \text{ConfChannels}$

The following is the corresponding theorem for symmetric channels which can also be proved similarly.

Theorem 3:

$\text{SimpleSharedTransfer} \vdash \text{SecureTransfer} \wedge$
 $\text{SharedChannels}' = \text{SharedChannels} \cup$
 $\{\{\text{sender?}, \text{recip?}\}\}$

Other key transfers could be formed in a similar manner; but although they are perfectly valid, they do not result in any new channels. For example, the schema *SimpleSecretConfTransfer* could be defined in the obvious manner and proved secure. However, the precondition would be that a confidentiality channel must already exist from sender to recipient, and so the confidentiality key exchanged does not form a new channel. Such an operation may be useful in practice, for example, when updating keys via a master key.

Consider how the schema *SimplePublicConfTransfer—U2* reflects a real protocol. The sender may have generated a public/secret key pair for confidentiality (say, an RSA key) but before the exchange possesses only an authentication channel to the recipient (possibly via a signature scheme based on zero knowledge). A signed version of the public key can be sent, thus convincing the recipient of the correctness of the public key. The recipient can now use this key when sending information back to the sender, thus forming a confidentiality channel from the recipient to the sender. The schema *SimpleSecretAuthTransfer—U1* represents a dual situation where the sender uses a confidentiality function to transfer a secret key for authentication.

The schema *SimpleSharedTransfer—U3* is illustrated by the (perhaps more common) situation where the sender and recipient know each other's public key. The confidentiality and authentication channels can then be used to exchange a shared secret key for regular communication. The purpose for which the key exchanged will be used is not specified, but it would typically be for both confidentiality and authentication.

It may already be clear that it is not possible to form channels by transfers between two users in conditions other than those defined in the three schemas above. In order to show that this is indeed the case, we prove the following general theorems.

Theorem 4:

$\text{SecureTransfer} \vdash$
 $\text{ran}(\text{ConfChannels}' \setminus \text{ConfChannels}) \subseteq$
 $\text{ranConfChannels} \cup \text{dom AuthChannels}$

Proof: Suppose y is a member of

$\text{ran}(\text{ConfChannels}' \setminus \text{ConfChannels})$.

Then (x, y) is a new confidentiality channel for some user x . The key transfer that defines the state change must involve at least one of x and y . Furthermore one of these must be the recipient of the key since a new channel can only be formed by one of the users receiving a key.

Suppose first that x is the recipient of $k?$. Then y is the originator and, furthermore, we must already have $(w, \text{dual}(k?)) \in \text{keys}(y)$ for some user w , since this must hold

if $(x, y) \in \text{ConfChannels}'$. Also $k? \in \text{Shared} \cup \text{Public}$ by definition of ConfChannels . Suppose z is the sender (where $z \neq y$), then by *Transfer*, $(y, k?) \in \text{keys}(z)$. But this shows that $(z, y) \in \text{ConfChannels}$ so $y \in \text{ran ConfChannels}$ as required. On the other hand, if the sender is y , then by *SecureTransfer*, $(y, x) \in \text{AuthChannels}$ so $y \in \text{dom AuthChannels}$ as required.

Finally, suppose that y is the recipient. Then $\text{dual}(k?) \in \text{Shared} \cup \text{Public}$ and so $k? \in \text{Shared} \cup \text{Secret}$. If the sender is any user z , then by *SecureTransfer* we have $(z, y) \in \text{ConfChannels}$ as required. \square

The following dual theorem also holds by a dual proof.

Theorem 5:

$$\begin{aligned} \text{SecureTransfer} \vdash \\ \text{dom}(\text{AuthChannels}' \setminus \text{Authchannels}) \subseteq \\ \text{ranConfChannels} \cup \text{domAuthChannels} \end{aligned}$$

The theorems may be restated using English as the following.

Theorem 6: Suppose that a user has either a confidentiality channel to her, or an authentication channel from her, at some state of the system. Then in the previous state of the system such a channel must also exist. By an inductive argument, such a channel exists at all previous states.

Another way to interpret the theorem is that no secure channels may be formed between any users who do not already possess secret or shared keys. The result seems quite natural—it is not expected to get something from nothing.

B. A Simple Example

A simple example shows that Theorem 6 may be useful in designing a security architecture. This example was introduced by Rueppel [12] to illustrate his own formal approach. The system consists of a set of users and a Key Management Center (KMC). The KMC uses a public key cryptosystem to distribute session keys to the users. The public key of the KMC is assumed to be known by all users. Session keys are used with a symmetric cryptosystem to provide secure communications. A protocol is proposed to allow any users A and B to establish secure communications as follows.

- 1) A and B choose random keys k_a and k_b which they send to the KMC encrypted with the public key of the KMC.
- 2) The KMC chooses a random session key k_s for use by A and B .
- 3) The KMC sends k_s to A and B encrypted with k_a and k_b , respectively.

This protocol is insecure because the messages sent in step 1 have no authentication, and so the KMC has no idea of their source. Thus, although A and B can correctly deduce that messages in step 3 come from the KMC, they have no assurance about who else shares the session key. Rueppel's language reveals this because it is necessary to use an illegal production (one not in the database) in order to arrive at the conclusion that secure communication is achieved.

The KMC has a secret key whose dual public key is known to all other users. This establishes a confidentiality channel from each user to the KMC. But apart from the KMC, all

TABLE I
TRANSITIONS FOR SIMPLE TRANSFERS

Existing Channels	New Channel	Rule Name
$A \xrightarrow{c} B$	$B \xrightarrow{a} A$	U1
$A \xrightarrow{a} B$	$B \xrightarrow{c} A$	U2
$A \xrightarrow{c} B$ and $A \xrightarrow{a} B$	$A \longleftrightarrow B$	U3

other users initially have no secret or shared keys at all. In other words, these users have no confidentiality channel to them, or authentication channel from them. Thus, by Theorem 6, it may be concluded, *without even considering the protocol that is suggested*, that it will not be possible to establish a confidentiality channel to, or an authentication channel from, any user apart from the centre. There is an architectural flaw in the system which needs to be addressed before considering what protocols may be used.

C. An Alternative Notation

The Z notation allows the model to be defined within the formality of set theory and logic. The model is explicit about the relationship between ownership of keys and existence of confidentiality and authentication channels. However, the previous theorem indicates that the preconditions for forming new channels may be defined in terms of existing channels. Before proceeding to the situation of key transfer through trusted users, a more convenient notation is now introduced which allows the transition rules from existing channels to new channels to be easily summarized.

$A \xrightarrow{a} B$ will mean that there exists an authentication channel from A to B (and hence that A has a secret key which is either shared with B , or for which B has the dual public key). Similarly, $A \xrightarrow{c} B$ will mean that there is a confidentiality channel from A to B , and $A \longleftrightarrow B$ will mean that there is a symmetric channel between A and B .

Using this notation, the transitions defined for user-to-user key exchange are easily stated. The existing channels from each schema can be stated together with the new channel resulting from the key transfer. Thus, the schema *SimplePublicConfTransfer* shows that the precondition $A \xrightarrow{a} B$ allows transition to a state with $B \xrightarrow{c} A$. Table I lists the possible transition rules for creating new channels from user to user key exchange. Each rule can be compared to the associated schema in the Z model via the rule name which is appended to the relevant schema name. Thus, if the channel(s) in the left-hand column exist, then the new channel may be formed. Note, however, that it is assumed that the sender of the key is able to generate it—not always a reasonable assumption. Theorem 6 shows that these are the only possible transitions to new channels that may occur through user to user key exchange.

D. Trusted Users

A useful architecture requires trusted users, via whom keys may be passed, and hence many new secure channels formed. This means that during the transfer of keys, the originator and destination will not always correspond to the sender and

recipient. Although keys may certainly pass from the originator or destination, a new channel will only be formed when one of these is the recipient. Because there is no formal way of differentiating between originator and destination, we may as well assume that the recipient is the destination. In order that the transfer is secure, the recipient must trust the sender (at least in the case that $k?$ is public or shared). These conditions are defined in the next schema.

<i>TrustedTransfer</i> <i>Transfer</i>
$dest? = recip?$ $sender? \in Trusted(recip?)$

We next define some fundamental state transitions which form new channels in this manner.

<i>TrustedPublicAuthTransfer—T1</i> <i>TrustedTransfer</i>
$k? \in Auth \cup Public$ $(dest?, dual(k?)) \in keys(orig?)$ $(sender?, recip?) \in AuthChannels$

<i>TrustedPublicConfTransfer—T2</i> <i>TrustedTransfer</i>
$k? \in Conf \cup Public$ $(dest?, dual(k?)) \in keys(orig?)$ $(sender?, recip?) \in AuthChannels$

As in the simple case, the following theorems are easily proven.

Theorem 7:

$$TrustedPublicAuthTransfer \vdash SecureTransfer \wedge AuthChannels' = AuthChannels \cup \{(orig?, recip?)\}$$

Theorem 8:

$$TrustedPublicConfTransfer \vdash SecureTransfer \wedge ConfChannels' = ConfChannels \cup \{(recip?, orig?)\}$$

Consider, for example, the schema *TrustedPublicConfTransfer—T2*. This specifies how a public confidentiality key may be transferred via a trusted user. The outcome is the same as for the schema *SimplePublicConfTransfer—U2*, but the preconditions are different. Here the recipient needs to trust the sender, and because the sender must know the public key in order to send it, there already exists a confidentiality channel from the sender to the originator. This schema can be interpreted as an abstract description of a familiar situation where a trusted center certifies the public key of the user.

TABLE II
TRANSITIONS FOR TRUSTED CHANNELS

Existing Channels	New Channel	Rule Name
$A \xrightarrow{a} T \xrightarrow{a} B$	$A \xrightarrow{a} B$	T1
$A \xrightarrow{c} T \xrightarrow{a} B$	$B \xrightarrow{c} A$	T2
$A \xrightarrow{c} T \xrightarrow{c} B$ and $A \xrightarrow{a} T \xrightarrow{a} B$	$A \longleftrightarrow B$	T3

The recipient (of the certificate) trusts the center (sender) to provide a correct key and knows the public authenticating key of the center, so that an authentication channel exists from the sender to recipient.

As with simple transfers, other schemas defining transfer of secret keys could be defined in the obvious way. They represent less common situations, but a possible scenario is where the center is trusted by the recipient to generate and distribute public/secret key pairs. The center must distribute the secret key to the recipient via a confidentiality channel, and other users may receive the public key on an authentication channel. Since, in this case, the recipients of the public key must trust the center, the transition of channels is the same as in rule T2.

Table II lists transition rules for trusted users written in the graphical notation. In all cases, user B trusts user T . Note that in the schemas corresponding to rules T1 and T2, it is assumed that the sender already knows the public key of the originator. However, even if this is not the case initially, the channels which are assumed to exist may be used for a key transfer to make this condition hold. For example, the authentication channel from A (the originator) to T (the sender) in rule T1 may be used for A to transfer her public key to T . Thus, given the existing channels in the table, the new channel may always be achieved by a sequence of secure key transfers.

The rule T3 allows creation of symmetric channels. A schema for this transition has not been given but can be derived from the previous rules as follows. $A \xrightarrow{c} T \xrightarrow{c} B$ can be converted to $A \xleftarrow{a} T \xrightarrow{c} B$ by U1 and hence to $B \xrightarrow{c} A$ by T2. Similarly, $A \xrightarrow{a} T \xrightarrow{a} B$ converts to $A \xrightarrow{a} B$ by T1. Putting these together gives $A \longleftrightarrow B$ by U3. Thus, we have shown that the new channel in rule T3 can be formed from the existing channels by a series of secure key transfers.

III. DISCUSSION

Only transitions which add new secure channels to the model have been defined. In a real system, secure channels are continually being destroyed as well as built. This is done by destroying key information which is a common occurrence at the end of a communications session. The model could, of course, be extended to allow the destruction of channels—indeed, this would probably be desirable in modeling a specific system. In this paper, however, we are only interested in what secure architectures are possible, defined by what secure channels may be built from existing ones.

In Section II-A, a necessary condition for a user to set up secure channels was given, namely, that he should possess an authentication channel to, or a confidentiality channel from,

at least one other user. It is a natural question to consider what is a *sufficient* condition in a system to allow users to set up secure channels. We will say that *secure communications* exist between two users if there are confidentiality and authentication channels in both directions.

Definition: There is a *trusted link* from A to B if B trusts A and either $A \xrightarrow{a} B$ or $B \xrightarrow{c} A$. There is a *trusted chain* from X to Y if there is a user T with $X \xrightarrow{a} T$ or $T \xrightarrow{c} X$ and a sequence of trusted links joining T to Y .

We may think of the system as a directed graph with the users as nodes and the edges as the trusted links. Then a trusted chain is a path in the graph theoretic sense. The following result gives a sufficient condition for establishing secure communications between any two users in a distributed system.

Theorem 9: Secure communication between any two users may be established by a sequence of secure key transfers if there is a trusted chain from each one to the other.

Proof: Assume that such a path does exist in both directions. By transition rule $U1$, we may assume that all links in both chains are authentication channels in the same direction. But then, using the rule $T1$, each intermediate node may be recursively removed from the chain. Thus, we may arrange the situation $A \xrightarrow{a} B$ and $B \xrightarrow{a} A$. By rule $U2$, secure communications may be established. \square

Using this result, all sorts of security architectures are possible which allow secure communications between various users. If only certain users require (or are allowed by the security policy to establish) secure communications, then unusual architectures may be useful. In general, it will be the case that all users will want the ability to create secure communications with any other users, and so it is to be expected that security architectures should be homogeneous from the viewpoint of ordinary users.

Two examples of familiar architectures which satisfy Theorem 9 are star architectures and tree architectures. In star architectures, all users can be viewed as arranged in a star around the trusted center, with each user having a secure channel with the center. Various protocols exist to allow secure communications to be established from such an architecture. The secure channels must form a trusted chain from all users to all other users. An example is the protocol given in Needham and Schroeder's classic paper [10] in which each user initially shares a key with the trusted center which provides a confidentiality and authentication channel in both directions.

A tree architecture is a generalization of the star architecture, suitable for large networks. Each user is attached to a trusted center, but there may be many of these. Each trusted center itself is attached to a trusted metacenter. There may be several layers leading up to the root of the tree. Such an architecture is suggested in the CCITT X500 Directory System standard [3]. This may be based on public key cryptography, and each trusted center acts as a certification authority (CA) for the public keys of those nodes below it in the tree. There is thus an authentication channel from the CA to each user below it. In addition, the CA knows the public key of each user below it and so, if this key provides authentication, there is an

TABLE III
SPECIAL Z SYMBOLS

Symbol	Meaning
$f : X \rightarrow Y$	Function between X and Y
$f : X \leftrightarrow Y$	Relation between X and Y
id_X	The identity function on X
$\text{dom } f$	The domain of f
$\text{ran } f$	The range of f
$f(X)$	Image of the set X under the function f
$f \oplus g$	Function which takes values of the function f except on the domain of g , where it takes the values of g
$f \circ g$	Functional composition where the domain of g must equal the range of f
$X \setminus Y$	Set difference of X and Y

authentication channel upwards to each CA. This continues for every CA up the tree. Thus, it can be seen that by traversing the correct nodes in the tree, there is a trusted chain between any two users as required by Theorem 9.

APPENDIX

SUMMARY OF Z NOTATION

A Z specification consists mainly of a collection of units called *schemas* which take the form of a box divided in two by a horizontal bar. Above the bar are a number of declarations (the *signature* of the schema), while below the bar are a number (possibly zero) of predicates which constrain the elements in the signature.

<i>SchemaName</i>
<i>Declarations</i>
<i>Predicates</i>

Declarations define the names of all variables and must include a type. The notation $\mathbb{P}X$ denotes the set of sets with elements in X (the *power set* of X). Thus, variables of type $\mathbb{P}X$ take sets of elements of X as values. Predicates are defined as in standard predicate logic and may include existential and universal quantifiers.

Schemas can be included in other schemas, thus providing easy modularization of the specification. If a schema name is included in the predicate part, this is equivalent to including all the declarations and predicates of the included schema. The Z convention for representing state changes is that the value of each variable in the new state is shown by the dashed variable of the same name. The notation $\Delta State$ indicates that all variables from the schema $State$ are included together with their dashed states. Variables followed by a question mark are conventionally inputs to an operation. Theorems are stated in the form: *premise* \vdash *conclusion*.


Table III covers some of the special symbols used in the text.

ACKNOWLEDGMENT

The author is grateful to R. Lampard of the National Physical Laboratory for useful comments, and to J. Jacob of Oxford University Computing Laboratory for advice on the use of Z.

REFERENCES

- [1] C. A. Boyd, "Hidden assumptions in cryptographic protocols," *Proc. IEE*, Part E, vol. 137, pp. 433-436, 1990.
- [2] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Trans. Comput. Syst.*, vol. 8, no. 1, pp. 18-36, Feb. 1990.
- [3] CCITT, The Directory, Part 8: Authentication Framework, Recommendation X.509, Geneva, 1989.
- [4] D. W. Davies and W. L. Price, *Security in Computer Networks*. New York: Wiley, 1989.
- [5] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Crypto 86*. New York: Springer-Verlag, 1987.
- [6] L. Gong, R. Needham, and R. Yahalom, "Reasoning about belief in cryptographic protocols," in *Proc. 1990 IEEE Computer Soc. Symp. Security Privacy*, IEEE Computer Society Press, 1990, pp. 234-248.
- [7] ISO 7498-2, Information Processing Systems—Open Systems Interconnection Reference Model, Security Architecture, 1988.
- [8] R. Kailar and V. D. Gligor, "On belief evolution in authentication protocols," in *Proc. Computer Security Foundations Workshop IV*. New York: IEEE Press, 1991, pp. 103-116.
- [9] C. Meadows, "A system for the specification and analysis of key management protocols," in *Proc. 1991 IEEE Computer Soc. Symp. Security Privacy*. New York: IEEE Computer Society Press, 1991, pp. 182-195.
- [10] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. ACM*, vol. 21, no. 12, pp. 993-999, Dec. 1978.
- [11] R. Rivest, A. Shamir, and L. Adelman, "A method for obtaining digital signatures and public key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120-126, Feb. 1978.
- [12] R. A. Rueppel, "A formal approach to security architectures," in *Proc. Eurocrypt 91*. New York: Springer Verlag, 1991.
- [13] J. M. Spivey, *The Z Notation*. Englewood Cliffs, NJ: Prentice-Hall, 1989.



Colin Boyd received the B.Sc. and Ph.D. degrees in mathematics from the University of Warwick in 1981 and 1985, respectively.

He subsequently worked for five years at the British Telecom Research Laboratories, specializing in data security. He is currently Lecturer in Communications at the University of Manchester. His research interests include the theory and applications of cryptography and the use of formal methods in computer and communications security.