

# A Procedure for Verifying Security Against Type Confusion Attacks

Catherine Meadows  
Code 5543  
Naval Research Laboratory  
Washington, DC 20375  
meadows@itd.nrl.navy.mil

## Abstract

A type confusion attack is one in which a principal accepts data of one type as data of another. Although it has been shown by Heather et al. that there are simple formatting conventions that will guarantee that protocols are free from simple type confusions in which fields of one type are substituted for fields of another, it is not clear how well they defend against more complex attacks, or against attacks arising from interaction with protocols that are formatted according to different conventions.

In this paper we show how type confusion attacks can arise in realistic situations even when the types are explicitly defined in at least some of the messages, using examples from our recent analysis of the Group Domain of Interpretation Protocol. We then develop a formal model of types that can capture potential ambiguity of type notation, and outline a procedure for determining whether or not the types of two messages can be confused. This work extends our earlier work on the subject in that it includes an explicit model of attacker and defender and extends the informal model of the type confusion attack in terms of a game between an intruder and a set of honest principals in or earlier work to a more formal model in which actions of intruder and honest principals are described explicitly. This gives us a simpler, more intuitive approach that allows us to calculate probabilities in a more systematic manner, and to compare different intruder strategies and different assumptions about the way in which the protocol is implemented in terms of their effects on type confusion.

## 1 Introduction

Type confusion attacks arise when it is possible to confuse a message, which we will refer to as the *masquerading message*, containing data of one type with a message, which we will refer to as the *spoofed message*, containing data of another. The most simple type confusion attacks are

ones in which fields of one type are confused with fields of another type, such as is described in [9], but it is also possible to imagine attacks in which fields of one type are confused with a concatenation of fields of another type, as is described by Sneekenes in [12], or even attacks in which pieces of fields of one type are confused with pieces of fields of other types.

The technique of tagging data with its type has been shown to provide security against simple type confusion attacks involving the confusion of one field with another in the Dolev-Yao model [5], and we believe that these techniques could easily be extended to more complex type confusion attacks (see [10] for a discussion). But, although a tagging technique may work within a single protocol in which the technique is followed for all authenticated messages, it does not prevent type confusion of a protocol that uses the technique with a protocol that does not use the technique, but that does use the same authentication keys. Since it is not uncommon for master keys (especially public keys) to be used with more than one protocol, it may be necessary to develop other means for determining whether or not type confusion is possible. In this paper we explore these issues further, and describe a procedure for detecting the possibility of the more complex varieties of type confusion.

The remainder of this paper is organized as follows. In Section Two, in order to motivate our work, we give a brief account of a complex type confusion flaw that was recently found during an analysis of the Group Domain of Authentication Protocol, a secure multicast protocol being developed by the Internet Engineering Task Force. In Section Three we give a formal model for the use of types in protocols that takes into account possible type ambiguity. This is similar to an earlier one we developed in [10], except that it takes into account the causal order among message fields as well as the order in which they appear in a message. In Section Four we develop the notion of a type confusion attack as a game between an intruder and a set of honest principals. We use this notion of a game to develop what we call the *gap-toothed zipper*, a generalization of the zipper proce-

dure developed in [10]. We show how the gap-toothed zipper can be used to compare different intruder strategies and help determine whether or not a successful strategy exists. In Section Five we conclude the paper and give suggestions for further research.

## 2 The GDOI Attack

In this section we describe a type flaw attack that was found on an early version of the GDOI protocol [2].

The Group Domain of Interpretation protocol (GDOI), is a group key distribution protocol that is undergoing the IETF standardization process. It is built on top of the ISAKMP [8] and IKE [4] protocols for key management, which imposes some constraints on the way in which it is formatted. GDOI consists of two parts. In the first part, called the Groupkey Pull Protocol, a principal joins the group and gets a group key encryption key from the Group Controller/Key Distributor (GCKS) in a handshake protocol protected by a pairwise key that was originally exchanged using IKE. In the second part, called the Groupkey Push Message, the GCKS sends out new traffic encryption keys protected by the GCKS's digital signature and the key encryption key.

Both pieces of the protocol can make use of digital signatures. The Groupkey Pull Protocol offers the option of including a Proof-of-Possession field, in which either or both parties can prove possession of a public key by signing the concatenation of a nonce NA generated by the group member and a nonce NB generated by the GCKS. This can be used to show linkage with a certificate containing the public key, and hence the possession of any identity or privileges stored in that certificate.

As for the Groupkey Push Message, it is first signed by the GCKS's private key, and then encrypted with the key encryption key. The signed information includes a header HDR, (which is sent in the clear), and contains, besides the header, several different types of message payload, and it ends in a Key Download Payload which will generally end in a random number (the key).

According to the conventions of ISAKMP, HDR must begin with a random or pseudo-random number. In pairwise protocols, this is jointly generated by both parties, but in GDOI, since the message must go from one to many, this is not practical. Thus the number is generated by the GCKS. Similarly, it is likely that the Key Download message will end in a random number: a key. Thus it is reasonable to assume that the signed part of a Groupkey Push Message both begins and ends in a random number.

We found two type confusion attacks. In both, we assume that the same private key is used by the GCKS to sign POPs and Groupkey Push Messages. In the first of these, we assume a dishonest group member who wants to pass

off a signed POP from the GCKS as a Groupkey Push Message. To do this, she creates a fake plaintext Groupkey Push Message GPM, which is missing only the last (random) part of the Key Download Payload. She then initiates an instance of the Groupkey Pull Protocol with the GCKS, but in place of her nonce, she sends GPM. The GCKS responds by appending its nonce NB and signing it, to create a signed (GPM,NB). If NB is of the right size, this will look like a signed Groupkey Push Message. The group member can then encrypt it with the key encryption key (which she will know, being a group member) and send it out to the entire group.

The second attack requires a few more assumptions. We assume that there is a group member A who can also act as a GCKS, and that the pairwise key between A and another GCKS, B, is stolen, but that B's private key is still secure. Suppose that A, acting as a group member, initiates a Groupkey Pull Protocol with B. Since their pairwise key is stolen, it is possible for an intruder to insert a fake nonce for B's nonce NB. The nonce he inserts is a fake Groupkey Push Message GPM' that it is complete except for a prefix of the header consisting of all or part of the random number beginning the header. A then signs (NA,GPM'), which, if NA is of the right length, will look like the signed part of a Groupkey Push Message. The intruder can then find out the key encryption key from the completed Groupkey Pull Protocol and use it to encrypt the resulting (NA,GPM') to create a convincing fake Groupkey Push Message.

A more complete account of both these attacks may be found in [16].

Fortunately the fix was simple. Although GDOI was constrained by the formatting required by ISAKMP, this was not the case for the information that was signed within GDOI. Thusrrr the protocol was modified so that, whenever a message was signed within GDOI, information was prepended saying what the purpose was (e.g. a member's POP, or a Groupkey Push Message). This eliminated the type confusion attacks.

There are several things to note here. The first is that existing protocol analysis tools are not very good at finding these types of attacks. Most assume that some sort of strong typing is already implemented. Even when this is not the case, the ability to handle the various combinations that arise is somewhat limited. For example, we found the second, less feasible, attack automatically with the NRL Protocol Analyzer, but the tool could not have found the first attack, since the ability to model it requires the ability to model the associativity of concatenation, which the NRL Protocol Analyzer lacks. Moreover, type confusion attacks do not require a perfect matching between fields of different types. For example, in order for the second attack to succeed, it is not necessary for NA to be the same size as the random number beginning the header, only that it be

no longer than that number. Again, this is something that is not within the capacity of most crypto protocol analysis tools. Finally, most crypto protocol analysis tools are not equipped for probabilistic analysis, so they would not be able to find cases in which, although type confusion would not be possible every time, it would occur with a high enough probability to be a concern.

The other thing to note is that, as we said before, even though it is possible to construct techniques that can be used to guarantee that protocols will not interact insecurely with other protocols that are formatted using the same technique, it does not mean that they will not interact insecurely with protocols that were formatted using different techniques, especially if, in the case of GDOI's use of ISAKMP, the protocol wound up being used differently than it was originally intended (for one-to-many instead of pairwise communication). Indeed, this is the result one would expect given previous results on protocol interaction [7, 1]. Since it is to be expected that different protocols will often use the same keys, it seems prudent to investigate to what extent an authenticated message from one protocol could be confused with an authenticated message from another, and to what extent this could be exploited by a hostile intruder. The rest of this paper will be devoted to the discussion of a procedure for doing so.

### 3 The Model

#### 3.1 Overview

In this section we will describe the model that underlies our procedure. It is motivated by the fact that different principals may have different capacities for checking types of messages and fields in messages. Some information, like the length of the field, may be checkable by anybody. Other information, like whether or not a field is a random number generated by a principal, or a secret key belonging to a principal, will only be checkable by the principal who generated the random number in the first case, and by the possessor(s) of the secret key in the second place. In order to do this, we need to develop a theory of types that take differing capacities for checking types into account. In Section 3.2 we set forth our basic theory of types. In Section 3.3 we show how we construct messages out of types.

#### 3.2 Types

We assume an environment consisting of principals who possess information and can check properties of data based on that information. As in the Dolev-Yao model, we assume principals are either honest, in which case they obey the rules of whatever communication protocols are defined, or

dishonest, in which case they are in league with an intruder who is trying to implement a type confusion attack.

**Definition 3.1** *A field is a sequence of bits. We let  $\iota$  denote the empty field. If  $x$  and  $y$  are two fields, we let  $x||y$  denote the concatenation of  $x$  and  $y$ .*

**Definition 3.2** *A type is a variable whose range is a set of fields, which can include the empty field. A probabilistic type is a random variable whose range is a set of fields. A type member choice is the act of choosing a member of a type (according to its probability distribution, if one exists) by a principal engaging in the protocol. We say that a type is under the control of a principal  $A$  if  $A$  is the principal who performs the type member choice.*

For the purposes of this paper, we will assume that any finite-domain type generated by a pseudo-random number generator (under which we will include cryptographic operations such as encryption, MACs, digital signatures, etc.) that is under the control of an honest principal is given a uniform distribution over its domain. We do not rule out assignments of distributions that correspond more closely to cryptographic assumptions, however, and this is something we intend to investigate more closely in the future.

We assume that each type is under the control of a single principal who may be either an honest principal or the intruder. If a type is under the control of an honest principal, it chooses a member of that type according to the rules of the protocol. On the other hand, there are actually two ways in which a type can be under the control of the intruder. The first way is directly. For example, suppose that the intruder sends a principal a nonce  $N_I$ , and the principal produces the message  $S_A(N_A, N_I)$ . Then the value of  $N_I$  is directly controlled by the intruder. Suppose on the other hand that the principal is expecting to receive a message  $S_B(N_A, N_B)$  where all it knows about  $N_B$  is that it is length  $N$ . If the intruder could trick  $B$  into producing some  $S_B(N_A, X)$ , where  $X$  is some other term of length  $N$ , then the intruder would have tricked  $A$  into accepting  $X$  as of the same type as  $N_B$ . Here, the intruder may not have complete control of the type, since it may not be able to trick  $B$  into accepting all strings of length  $N$ , but it does have some control. We will say that the type is under indirect control of the intruder in this case.

We assign probability distributions to types according to whose control they are under, and how. If a type is under the control of an honest principal, then the probability distribution is defined by the rules of the protocol. If the type is under the direct control of the intruder, then the probability distribution is initially undefined, but will be chosen by the intruder to maximize the likelihood of a type confusion attack. If the type is under the indirect control of the intruder, then no probability distribution is associated with that type.

Rather, the value of the type is determined by the values of the variables it is being matched against in a type confusion attack.

We now consider what an honest principal  $A$  who receives a field  $x$  that is supposed to be in the domain of a type  $T$  is able to tell about it. If  $T$  is under the control of  $A$  itself, then  $A$  will be able to tell, not only whether or  $x$  belongs to  $T$ , but whether or not  $x$  was the value that  $A$  chose. On the other hand, if  $A$  receives a field  $x$  purporting to come from the domain of a  $T$  under the control of an intruder, then all  $A$  can tell is whether or not  $x$  is in the domain of  $T$ .

The domain of a type, from  $A$ 's point of view, will also depend on  $A$ 's own individual knowledge. For example, suppose that  $A$  receives a MAC computed over a message  $M$ . If the MAC  $F$  is computed using a key  $K$  that  $A$  knows, then  $A$  will be able to verify that the MAC was computed over  $M$  using  $K$ . If  $A$  does not know  $K$  then  $A$  will only be able to verify syntactic properties of  $F$  such as the length.

This leads us to the following definition.

**Definition 3.3** *We say that a type  $T$  is local to  $A$  if  $A$  is able to verify membership in the domain of the type.*

Note that, if a type local to  $A$  is also under the control of an honest principal  $B$ , then  $A$  should be able to verify, not only membership in the domain of the type, but whether or not a member of that type was chosen by  $B$ .

We are now ready to consider the roles that types play in a type confusion attack. Let  $M_1$  be a masquerading message constructed by an honest principal  $A$ , and let  $M_2$  be a spoofed message expected by  $B$ . From  $A$ 's point of view,  $M_1$  will be constructed from the following types:

1. types controlled by  $A$ ,  
corresponding to data that it generated itself;
2. types controlled by other honest principals,  
corresponding to data it received from other honest principals, and whose origin and purpose it is able to verify;
3. and types directly controlled by the intruder,  
corresponding to data that it received whose origin and purpose it is unable to verify, either because it came from a dishonest principal, or because it was not authenticated, or because the authentication failed.

On the other hand, from  $B$ 's point of view  $M_2$  will be constructed from

1. types controlled by  $B$ ,  
corresponding to data that it generated itself that it is expecting to see in  $M_2$ ;

2. types controlled by other honest principals,  
corresponding to data it is expecting to see in  $M_2$  that it received from other honest principals previous to receiving  $M_2$ , and whose origin and purpose it is able to verify;
3. types directly controlled by the intruder,  
corresponding to data it is expecting to see in  $M_2$  that it received from elsewhere previous to receiving  $M_2$ , but whose origin and purpose it is unable to verify;
4. and types indirectly controlled by the intruder,  
corresponding to data that it is seeing now for the first time

Let  $A$  be an honest principal. Here are some examples of the types local to  $A$  that we will be interested in.

1. Random number of length  $N$ .

If this is a type under the control of an honest principal, it will be the set of all numbers of length  $N$ , together with the uniform distribution. If it is under the direct control of the intruder, it will be the set of all numbers of length  $N$  with an undefined probability distribution. If it is under the indirect control of the intruder, it will be the set of all numbers of length  $N$ .

2. Public key belonging to a designated principal  $B$ .  
This is a type consisting of one member.
3. Digital signature on a message  $M$  using a public key  $P$ .  
This is a type whose domain is the set of all expressions  $E$  satisfying the digital signature relationship with  $M$  and  $P$ . Note that, if the signature scheme is deterministic, this will have only one member. If the type is under the control of an honest principal, the distribution will be uniform over  $E$ .
4. MAC taken over a message  $M$ , using a key  $K$  that  $A$  knows.

This is a type uniformly distributed over the set of all expressions  $E$  satisfying the MAC relation with  $K$  and  $M$ . Again, if the MAC is deterministic, this will have only one member. If the type is under the control of an honest principal, the distribution will be uniform over  $E$ .

### 3.3 Type Function Trees

We are now ready to use types to construct messages. The most obvious way would be to represent messages as

lists of types. However, this is not adequate, because the types that may be used in a message may depend on choices made previously for other fields in that or other messages. Consider the following example:

**Example 3.1** Let  $M$  be the message created by  $A$  of the form  $["nonce", N, NONCE_A]$ , where  $NONCE_A$  is a nonce of length  $N$ . The type of  $NONCE_A$  is the set of numbers of length  $N$ , and so depends upon the second field of the message. On the other hand, suppose that  $A$  computes  $NONCE_A$ , and sends it to  $B$ , who computes the message  $["nonce", N, NONCE_A]$ , where  $N$  is the length of  $NONCE_A$ . In that case the integer  $N$  depends on  $NONCE_A$ .

We formalize the dependence of later choices of types upon previous choices by defining the notion of a type function tree as follows:

**Definition 3.4** A type function tree is a function  $\mathcal{R}$  from lists of fields to types, such that:

1. The empty list  $\langle \rangle$  is in  $\text{Dom}(\mathcal{R})$ ;
2. The list of fields  $\langle x_1, \dots, x_k \rangle$  is in  $\text{Dom}(\mathcal{R})$  if and only if  $\langle x_1, \dots, x_{k-1} \rangle \in \text{Dom}(\mathcal{R})$  and  $x_k \in \mathcal{R}(\langle x_1, \dots, x_{k-1} \rangle)$ ;
3. There exists an integer  $h$ , called the height of  $\mathcal{R}$ , such that for any  $n > h$ ,  $\mathcal{R}(\langle x_1, \dots, x_n \rangle) = \{\iota\}$  where  $\iota$  is the empty string.

We let  $\mathcal{R}^k$  denote the restriction of  $\mathcal{R}$  to  $k$ -tuples.

The order in which types appear in a type function tree should reflect the temporal order in which types are chosen and the causal relationship between types, not necessarily the order in which they appear in a message. We thus need to define the relationship between a type function tree and the message it represents as follows:

**Definition 3.5** Let  $\mathcal{R}$  be a type function tree of height  $h$ . Let  $\rho$  be a map from  $\langle 1, \dots, q \rangle$  onto some  $\langle 1, \dots, h \rangle$ . We say that  $M$  is a message type constructed from  $\mathcal{R}$  via  $\rho$  if  $M$  consists of all fields of the form  $y_1 || \dots || y_q$  such that there exists an  $\langle x_1, \dots, x_h \rangle$  in the domain of  $\mathcal{R}_h$  such that  $y_i = x_j$  whenever  $j = \rho(i)$ . We call  $\rho$  a message surjection.

Thus, in Example 3.1 the first message type is constructed via the identity function, while the second is constructed via a  $\rho$  defined as  $\rho(1) = 1, \rho(2) = 3$ , and  $\rho(3) = 2$ .

We note, in particular, that if  $\mathcal{R}$  is a type function tree corresponding to a spoofed message, then all types under indirect control of the intruder should appear at the end of the tree. This is because the members of these types are not

chosen until the spoofed message is matched with a masquerading message, while the members of the other types will have been chosen prior to a principal's receiving a masquerading message.

Our purpose in constructing type function trees will, of course, be the construction of messages of one type that can be mistaken for messages of another type. Consider, for example, the following protocol:

**Example 3.2** We consider two instances of a simple challenge-response protocol:

1.  $A \rightarrow B : N_A$ ; where  $N_A$  is an arbitrary nonce of length  $N$ ;
2.  $B \rightarrow A : N_B, S_B(N_A, N_B)$ ; where  $N_B$  is an arbitrary nonce of length  $N$ ;
3.  $A \rightarrow B : S_A(N_B, N'_A)$ ; where  $N'_A$  is an arbitrary nonce of length  $N$ ;

and

1.  $B \rightarrow A : N''_B$ ;
2.  $A \rightarrow B : N''_A, S_A(N''_B, N''_A)$ ;
3.  $B \rightarrow A : S_B(N''_A, N'''_B)$

We want to see if it is possible to trick  $A$  into accepting a second message from an honest principal  $B$  in the first instance of the protocol as a third message from  $B$  in the second instance of the protocol. That is, we want to see if it is possible to trick  $A$  into accepting a message  $S_B(X, N_B)$ , as one of the form  $S_B(N''_A, Y)$ , where  $X$  and  $Y$  are supplied by the intruder. At first this seems easy; we let  $X = N''_A$  and then we get  $Y = N_B$ . Suppose that  $N''_A$  is generated, and learned by the intruder, before  $X$  and  $N_B$  is generated before  $Y$ . Since  $Y$  is generated after  $N''_A$  and  $X$  before  $N_B$ , this gives us a possible type function tree as follows:

1.  $\mathcal{R}(\langle \rangle) = N''_A$   
 $N''_A$  is a type under control of  $A$  consisting of all integers of a fixed length  $N$ , uniformly distributed.
2.  $\mathcal{R}(\langle x_1 \rangle) = X$   
 $X$  is a type under direct control of the intruder. It corresponds to the first field in the signed part of the second message of the protocol.
3.  $\mathcal{R}(\langle x_1, x_2 \rangle) = N_B$   
 $N_B$  is a type under control of  $B$  consisting of all integers of length  $N$ , also uniformly distributed.
4.  $\mathcal{R}(\langle x_1, x_2, x_3 \rangle) = Y$   
 $Y$  is a type under indirect control of the intruder. It corresponds to the first field in the signed part of the third message of the protocol.

$$5. \mathcal{R}(\langle x_1, x_2, x_3, x_4 \rangle) = \iota$$

We begin by having  $A$  choose a field  $x_1$  randomly from  $N''_A$ . Clearly, the only strategy available to the intruder is to choose  $x_2 = x_1$ , which, since  $x_1$  has already been revealed, can be done with probability one. We next let  $B$  choose  $x_3$  randomly from  $N_A$ . Once that is done, we can let  $x_4 = x_3$ .

On the other hand, suppose that the intruder generates  $X$  before learning  $N''_A$ . In that case the type function tree could be defined as follows:

1.  $\mathcal{R}(\langle \rangle) = X$
2.  $\mathcal{R}(\langle x_1 \rangle) = N_B$
3.  $\mathcal{R}(\langle x_1, x_2 \rangle) = N''_A$
4.  $\mathcal{R}(\langle x_1, x_2, x_3 \rangle) = Y$
5.  $\mathcal{R}(\langle x_1, x_2, x_3, x_4 \rangle) = \iota$

where the types are defined as above. We now begin by having the intruder choose a field  $x_1$  from  $X$  according to some probability distribution  $\delta$  and  $B$  choose  $x_2$  randomly from  $N_B$ . But now when  $A$  chooses  $x_3$  randomly from  $N''_A$  the probability that  $x_3 = x_1$  is only  $1/2^N$ . Thus, the probability of a successful type confusion attack changes from certain to negligible, no matter what the choice of  $\delta$  is.

We see from the above examples that we can think of the intruder's attempt to pass off a message of one type as a message of another type as a game between the intruder and the honest principals. The intruder and the honest principals choose various members of types in a type function tree, according to whether the type is under control of the intruder or an honest principal. If the honest principal is doing the choosing, it uses the probability distribution specified in the protocol. If the intruder is doing the choosing directly, it uses a strategy most likely to maximize the probability of one message being accepted as another. If the type is under the indirect control of the intruder we attempt to determine if there is any value satisfying the constraints of the type that will make the two messages equal. In the next section, we will formalize this and make it explicit.

## 4 Type Confusion Games

In this section we show how we can model an attempt by an intruder to convince an honest principal  $A$  to construct a masquerading message that can be accepted as a spoofed message by an honest principal  $B$  in terms of a game between the intruder and the honest principals. We also describe a procedure, similar to the "zipper" described in [10] for verifying that no type confusion attack is possible, and for narrowing down the search for type confusion attacks if one is possible.

We start by building a type function tree that represents the construction of both masquerading and spoofed messages. This is because, as was made clear in our discussion of Example 3.2, we need to keep the relative timing of the creation of the various fields of the two messages straight. However, we also need to describe the two messages as type function trees. We describe how to build a type function tree out of two type function trees as follows:

**Definition 4.1** *Let  $\mathcal{R}_1$  and  $\mathcal{R}_2$  be two type function trees of height  $h_1$  and  $h_2$ , respectively. We define an interleaving  $\mathcal{I}$  of  $\mathcal{R}_1$  and  $\mathcal{R}_2$  inductively as follows. Let  $\theta_1$  and  $\theta_2$  be monotone increasing injections of  $\langle 1, \dots, h_1 \rangle$  and  $\langle 1, \dots, h_2 \rangle$ , respectively into  $\langle 1, \dots, h \rangle$ , such that each member of  $\langle 1, \dots, h \rangle$  is in the image of  $\theta_1$  or  $\theta_2$ .*

1. *If  $1$  is in the image of  $\theta_i$ , we define  $\mathcal{I}(\langle \rangle) = \mathcal{R}_i(\langle \rangle)$ .*
2. *Suppose that  $\mathcal{I}(\langle x_1, \dots, x_{k-1} \rangle) = T$ , and that  $k$  is in the image of  $\theta_i$ . For each  $x_k \in T$  we define  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$  to be  $\mathcal{R}_i(\langle x_{j_1}, \dots, x_{j_t} \rangle)$ , where  $\langle j_1, \dots, j_t \rangle$  is the maximal subsequence of  $\langle 1, \dots, k-1 \rangle$  in the image of  $\theta_i$ .*

We leave it as an exercise to the reader to show that an interleaving of two type function trees is a type function tree if the images of  $\theta_1$  and  $\theta_2$  are disjoint or if  $\mathcal{R}_1^i = \mathcal{R}_2^j$  whenever  $\theta_1(i) = \theta_2(j)$ .

The reason we allow the possibility of  $\theta_1(i) = \theta_2(j)$  is that the two messages might make use of common data. For example, consider a protocol, such as the Internet Key Exchange protocol, which operates in two stages, the first in which principals establish (among other things) data that will appear in the headers of any messages passed in the second stage. If we then want to compare two messages passed in the second stage, we might want to make use of the fact that they contain this common information that was created in the first stage.

The purpose behind the definition of an interleaving of two type function trees is to preserve the causal ordering of data in two messages. If the choice of a member of a type  $X$  influences the choice of a member of a type  $Y$  in another, then  $X$  should precede  $Y$  in the interleaving of the two trees. In particular, types under indirect control in the spoofed message will always come after any type from a masquerading message, since the choice of the members of the types under indirect control of the intruder in the spoofed message will be determined by the choices of the members of the types in the masquerading message. Since moreover types under indirect control of the intruder come last in the spoofed message function tree, we conclude that types under indirect control of the intruder come last in the interleaved type function tree.

We are now finally ready to define a type confusion game between the intruder and the honest principals in a protocol.

**Definition 4.2** Let  $\mathcal{S}_1$  and  $\mathcal{S}_2$  be two type function trees of height  $h_1$  and  $h_2$  respectively, and corresponding to masquerading message and spoofed message respectively. Let  $\rho_1$  and  $\rho_2$  be the message surjections from  $\langle 1, \dots, t_1 \rangle$  to  $\langle 1, \dots, h_1 \rangle$  and from  $\langle 1, \dots, t_2 \rangle$  to  $\langle 1, \dots, h_2 \rangle$ , respectively, belonging to  $\mathcal{S}_1$  and  $\mathcal{S}_2$ . Let  $\mathcal{I}$  be an interleaving of  $\mathcal{S}_1$  and  $\mathcal{S}_2$ . We define a type confusion game between the intruder and the honest principals as follows:

1. If  $\mathcal{I}(\langle \cdot \rangle)$  is a type under control of an honest principal, let  $p_1$  be the probability distribution associated with it. For each member  $x_1$ , let  $q(\langle x_1 \rangle) = p_1(x_1)$ .
2. If  $\mathcal{I}(\langle \cdot \rangle)$  is a type under direct control of the intruder, choose a probability distribution  $\delta_1$  and choose a member  $x_1$  of  $\mathcal{I}(\langle \cdot \rangle)$ . Let  $q(\langle x_1 \rangle) = \delta_1(x_1)$ .
3. Suppose that  $\langle x_1, \dots, x_k \rangle$  have already been chosen, and that  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$  is a type under the control of an honest principal. Let  $p_{k+1}$  be the probability distribution associated with  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$ . Then for each member  $x_{k+1}$  of  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$ , let  $q(\langle x_1, \dots, x_k, x_{k+1} \rangle) = p_{k+1}(x_{k+1})$ .
4. Suppose that  $\langle x_1, \dots, x_k \rangle$  have already been chosen, and that  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$  is under the direct control of the intruder. Choose a probability distribution  $\delta_{k+1}$  on  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$ . For each member  $x_{k+1}$  of  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$ , let  $q(\langle x_1, \dots, x_k, x_{k+1} \rangle) = \delta_{k+1}(x_{k+1})$ .
5. Suppose that  $\langle x_1, \dots, x_k \rangle$  have already been chosen, and that  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$  is under the indirect control of the intruder. Then choose a member  $x_{k+1}$  of  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$ . Let  $q_{k+1}(\langle x_1, \dots, x_k, x_{k+1} \rangle)$  be 1, and  $q_{k+1}(\langle x_1, \dots, x_k, y \rangle)$  be 0 for all other members of  $\mathcal{I}(\langle x_1, \dots, x_k \rangle)$ .

We define a strategy for the intruder to be a choice of probability distributions for the types under the intruder's direct control and members of types under the intruder's indirect control, which may be dependent upon previous choices made by the honest principals.

Given a strategy  $ST$  (that is, a particular choice  $ST$  of probability distributions and type members), we let  $Q_{ST}$  be the probability distribution defined by  $Q_{ST}(\langle x_1, \dots, x_h \rangle) = \prod_{i=1}^h q(\langle x_1, \dots, x_i \rangle)$ , where  $q$  is defined as above.

Let  $p$  be a number between 0 and 1. We say that the intruder has a winning strategy with respect to  $p$  if there is some strategy  $ST$  such that

$$Q_{ST}(\bar{x} \text{ s.t. } x_{\rho_1(1)} \parallel \dots \parallel x_{\rho_1(t_1)}) = x_{\rho_2(1)} \parallel \dots \parallel x_{\rho_2(t_2)} \geq p.$$

We now construct a procedure, similar to the ‘‘zipper’’ defined in [10], for helping to determine if the intruder has

a winning strategy. It is based on the fact that generally, the intruder's success in inducing type confusion will depend on which types he tries to match with each other. The probability of success will thus depend on which types in the masquerading message overlap with which types in the spoofed message. This will induce constraints on lengths of fields in the respective messages. Thus it will be important to have a complete list of the possible constraints. We do this by computing all possible length constraints on the two sequence of message fields being matched, as follows.

**Definition 4.3** If  $x$  is a bitstring, we let  $l(x)$  denote the length of  $x$ . Let  $\langle i_1, \dots, i_m \rangle$  and  $\langle j_1, \dots, j_n \rangle$  be two sets of indices. We construct a constraint tree as follows:

1. The root of the constraint tree is the empty set. We call this the 0'th level of the tree.
2. The children of the root, referred to as the first level of the tree, are the nodes

- $\mathbf{C}_1 = \{l(x_{i_1}) \leq l(x_{j_1})\}$
- $\mathbf{C}_2 = \{l(x_{i_1}) > l(x_{j_1}), l(x_{i_1}) \leq l(x_{j_1}) + l(x_{j_2})\}$
- $\vdots$
- $\mathbf{C}_n = \{l(x_{i_1}) > l(x_{j_1}) + \dots + l(x_{j_{n-1}}), l(x_{i_1}) \leq l(x_{j_1}) + \dots + l(x_{j_n})\}$

3. We construct the  $s + 1$ 'th level of the tree, where  $s < n - 1$ , as follows. If  $\mathbf{D}$  is a node such that the largest  $v$  such that  $l(x_{i_1}) + \dots + l(x_{i_v}) \leq l(x_{j_1}) + \dots + l(x_{j_t})$  appears in  $\mathbf{D}$  for some  $t$  is  $s$ , construct the child nodes of  $\mathbf{D}$  as follows:

- $\mathbf{D}_1 = \mathbf{D} \cup \{l(x_{i_1}) + \dots + l(x_{i_{s+1}}) \leq l(x_{j_1}) + \dots + l(x_{j_t})\} - \{l(x_{i_1}) + \dots + l(x_{i_s}) \leq l(x_{j_1}) + \dots + l(x_{j_t})\}$
- $\mathbf{D}_2 = \mathbf{D} \cup \{l(x_{i_1}) + \dots + l(x_{i_{s+1}}) > l(x_{j_1}) + \dots + l(x_{j_t}), \{l(x_{i_1}) + \dots + l(x_{i_{s+1}}) \leq l(x_{j_1}) + \dots + l(x_{j_{t+1}})\}\}$
- $\vdots$
- $\mathbf{D}_{n-t} = \mathbf{D} \cup \{l(x_{i_1}) + \dots + l(x_{i_{s+1}}) > l(x_{j_1}) + \dots + l(x_{j_{n-1}}), \{l(x_{i_1}) + \dots + l(x_{i_{s+1}}) \leq l(x_{j_1}) + \dots + l(x_{j_n})\}\}$

4. We construct the  $n$ 'th level of the tree as follows. Suppose that  $\mathbf{D}$  is a node in the  $n - 1$ 'st level such that the constraint  $l(x_{i_1}) + \dots + l(x_{i_{n-1}}) \leq l(x_{j_1}) + \dots + l(x_{j_t})$  appears in  $\mathbf{D}$ . Then

- $\mathbf{D}_1 = \mathbf{D} \cup \{l(x_{i_1}) + \dots + l(x_{i_n}) = l(x_{j_1}) + \dots + l(x_{j_m})\} - \{l(x_{i_1}) + \dots + l(x_{i_{n-1}}) \leq l(x_{j_1}) + \dots + l(x_{j_t})\}$ .

**Example 4.1** To see how this works, consider two sequences  $\langle x_1, x_2, x_3 \rangle$  and  $\langle x_4, x_5 \rangle$ . The nodes at level one are:

- $D_1 = \{l(x_1 \leq l(x_4))\};$
- $D_2 = \{l(x_1) > l(x_4), l(x_1) \leq l(x_4) + l(x_5)\}.$

The nodes at level two are:

- $D_{(1,1)} = \{l(x_1) + l(x_2) \leq l(x_4)\};$
- $D_{(1,2)} = \{l(x_1) \leq l(x_4), l(x_1) + l(x_2) > l(x_4), l(x_1) + l(x_2) \leq l(x_4) + l(x_5)\};$
- $D_{(2,2)} = \{l(x_1) > l(x_4), l(x_1) + l(x_2) \leq l(x_4) + l(x_5)\}.$

The nodes at level three are:

- $D_{(1,1,1)} = \{l(x_1) + l(x_2) \leq l(x_4), l(x_1) + l(x_2) + l(x_3) = l(x_4) + l(x_5)\};$
- $D_{(1,2,1)} = \{l(x_1) \leq l(x_4), l(x_1) + l(x_2) > l(x_4), l(x_1) + l(x_2) + l(x_3) = l(x_4) + l(x_5)\}$
- $D_{(2,2,2)} = \{l(x_1) > l(x_4), l(x_1) + l(x_2) + l(x_3) = l(x_4) + l(x_5)\}.$

We now need to define what it means for a sequence of fields to be consistent with a set of inequalities.

**Definition 4.4** Let  $\mathbf{Q}$  be a set of inequalities and equalities defined in terms of variables  $\langle X_1, \dots, X_M \rangle$ . We will say that a sequence of fields  $\langle x_1, \dots, x_r \rangle$  where  $r \leq M$  is consistent with (or  $\triangleleft$ )  $\mathbf{Q}$  if the result of substituting  $x_1$  for  $X_1$  through  $x_r$  for  $X_r$  does not imply any contradictions.

We are now ready to define a procedure for verifying security against type confusion attacks. As we said before, it is similar to the “zipper” of [10]. The main difference is that instead of matching up fields according to the order in which they appear in the message, we match them in a way consistent with the causal order in which they are computed. This allows us to compute the probability of a successful type confusion using the probabilities taken from a type function tree instead of computing probabilities in an ad hoc fashion. We refer to this new version of the zipper as a “gap-toothed zipper”.

We proceed as follows.

**Definition 4.5** Let  $\mathcal{R}$  and  $\mathcal{S}$  be two type function trees of height  $h_1$  and  $h_2$ , respectively, where a masquerading message is constructed from  $\mathcal{R}$  using a function  $\rho_1$  from  $\langle 1, \dots, t_1 \rangle$  onto  $\langle 1, \dots, h_1 \rangle$  and a spoofed message is constructed from  $\mathcal{S}$  using a function  $\rho_2$  from  $\langle 1, \dots, t_2 \rangle$  onto  $\langle 1, \dots, h_2 \rangle$ . Let  $\mathcal{I}$  be an interleaving of  $\mathcal{R}$  and  $\mathcal{S}$ , constructed using injections  $\theta_1$  and  $\theta_2$ . Let  $p$  be a number

between zero and one. We define  $\mathcal{Z}(\mathcal{I}, p)$ , the gap-toothed zipper over  $\mathcal{I}$  and  $p$  as follows.

Let  $\mathbf{E}$  be the equation  $x_{\theta_1 \circ \rho_1(1)} \parallel \dots \parallel x_{\theta_1 \circ \rho_1(t_1)} = x_{\theta_2 \circ \rho_2(1)} \parallel \dots \parallel x_{\theta_2 \circ \rho_2(t_2)}$ . Let  $\Gamma$  be the constraint tree constructed from the two sequences of indices  $\langle \theta_1 \circ \rho_1(1), \dots, \theta_1 \circ \rho_1(t_1) \rangle$  and  $\langle \theta_2 \circ \rho_2(1), \dots, \theta_2 \circ \rho_2(t_2) \rangle$ . For each leaf  $\mathbf{C}$  of the constraint tree, construct a sequence of sets of pairs  $G(r, \mathbf{C}) = (\langle x_1, \dots, x_r \rangle, q(\langle x_1, \dots, x_r \rangle))$  as follows:

1. We construct  $G(1, \mathbf{C} \cup \{\mathbf{E}\})$  as follows.

- If  $\mathcal{I}(\langle \rangle)$  is a type under control of an honest principal Let  $G(1, \mathbf{C} \cup \{\mathbf{E}\}) = \{(\langle x_1 \rangle, p_{\langle \rangle}(\langle x_1 \rangle)) \mid x_1 \in \mathcal{I}(\langle \rangle) \wedge x_1 \triangleleft \mathbf{C} \cup \{\mathbf{E}\}\}$ , where  $p_{\langle \rangle}$  is the probability distribution associated with  $\mathcal{I}(\langle \rangle)$ .
- If  $\mathcal{I}(\langle \rangle)$  is a type under direct control of a dishonest principal, let  $G(1, \mathbf{C} \cup \{\mathbf{E}\}) = \{(\langle x_1 \rangle, \delta_{\langle \rangle}(x_1)) \mid x_1 \in \mathcal{I}(\langle \rangle) \wedge x_1 \triangleleft \mathbf{C} \cup \{\mathbf{E}\}\}$ , where  $\delta_{\langle \rangle}$  is the (as yet undefined) probability distribution associated with  $\mathcal{I}(\langle \rangle)$ .

Note that by construction  $\mathcal{I}(\langle \rangle)$  cannot be under the indirect control of the intruder.

2. Suppose that  $G(r-1, \mathbf{C} \cup \{\mathbf{E}\})$  is known. We let  $G(r, \mathbf{C} \cup \{\mathbf{E}\})$  be the union of all  $H(\langle x_1, \dots, x_{r-1} \rangle)$  such that  $(\langle x_1, \dots, x_{r-1} \rangle, g(\langle x_1, \dots, x_{r-1} \rangle)) \in G(r-1, \mathbf{C} \cup \{\mathbf{E}\})$ , where  $H(\langle x_1, \dots, x_{r-1} \rangle)$  is defined as below.

- For each  $(\langle x_1, \dots, x_{r-1} \rangle, g(\langle x_1, \dots, x_{r-1} \rangle))$  in  $G(r-1, \mathbf{C} \cup \{\mathbf{E}\})$  such that  $\mathcal{R}(\langle x_1, \dots, x_{r-1} \rangle)$  is under the control of an honest principal,  $H(\langle x_1, \dots, x_{r-1} \rangle) = \{(\langle x_1, \dots, x_r \rangle, g(\langle x_1, \dots, x_r \rangle)) \mid \langle x_1, \dots, x_r \rangle \triangleleft \mathbf{C} \cup \{\mathbf{E}\} \wedge g(\langle x_1, \dots, x_r \rangle) > 0\}$  where  $g(\langle x_1, \dots, x_r \rangle) = p_{\langle x_1, \dots, x_{r-1} \rangle}(\langle x_1, \dots, x_r \rangle) \cdot g(\langle x_1, \dots, x_{r-1} \rangle)$  where  $p_{\langle x_1, \dots, x_{r-1} \rangle}$  is the probability distribution associated with  $\mathcal{I}(\langle x_1, \dots, x_{r-1} \rangle)$ .
- For each  $(\langle x_1, \dots, x_{r-1} \rangle, g(\langle x_1, \dots, x_{r-1} \rangle))$  in  $G(r-1, \mathbf{C} \cup \{\mathbf{E}\})$  such that  $\mathcal{I}(\langle x_1, \dots, x_{r-1} \rangle)$  is under the direct control of a dishonest principal, let  $g(\langle x_1, \dots, x_r \rangle) = \delta_{\langle x_1, \dots, x_{r-1} \rangle}(\langle x_1, \dots, x_r \rangle) \cdot g(\langle x_1, \dots, x_{r-1} \rangle)$  where  $\delta_{\langle x_1, \dots, x_{r-1} \rangle}$  is the (as yet unknown) probability distribution associated with  $\mathcal{I}(\langle x_1, \dots, x_{r-1} \rangle)$ , and let  $H(\langle x_1, \dots, x_{r-1} \rangle) = \{(\langle x_1, \dots, x_r \rangle, g(\langle x_1, \dots, x_r \rangle)) \mid \langle x_1, \dots, x_r \rangle \triangleleft \mathbf{C} \cup \{\mathbf{E}\} \wedge g(\langle x_1, \dots, x_r \rangle)\}$
- For each  $(\langle x_1, \dots, x_{r-1} \rangle, g(\langle x_1, \dots, x_{r-1} \rangle)) \in G(r-1, \mathbf{C} \cup \{\mathbf{E}\})$  such that  $\mathcal{I}(\langle x_1, \dots, x_{r-1} \rangle)$  is under the indirect control of a dishonest principal, then, if there is a field  $x_r \in$

$\mathcal{I}(\langle x_1, \dots, x_{r-1} \rangle)$  such that  $\langle x_1, \dots, x_{r-1} \rangle \in \mathbf{C} \cup \{\mathbf{E}\}$  (by construction there is at most one such  $x_r$  for each  $\langle x_1, \dots, x_{r-1} \rangle$ ), let  $g(x_1, \dots, x_r) = g(x_1, \dots, x_{r-1})$ , and let  $H(\langle x_1, \dots, x_{r-1} \rangle) = \{(\langle x_1, \dots, x_r \rangle, g(x_1, \dots, x_r))\}$ .

We let  $\Sigma(h, \mathbf{C} \cup \{\mathbf{E}\})$  be the sum of all  $g(\bar{x})$  such that  $(\bar{x}, g) \in G(h, \mathbf{C} \cup \{\mathbf{E}\})$

By construction, for each leaf node  $\mathbf{C}$  of the length constraint tree, we conclude that  $\Sigma G(h, \mathbf{C} \cup \{\mathbf{E}\})$  is the probability that there exists a sequence  $\langle x_1, \dots, x_h \rangle$  satisfying  $\mathbf{E}$  and  $\mathbf{C}$ . Let  $p$  be a number between 0 and 1. Clearly, if  $\Sigma G(h, \mathbf{C} \cup \{\mathbf{E}\}) < p$  for all  $\mathbf{C}$  and all choices for the probability distributions  $\delta$  under the direct control of the intruder, then the intruder has no winning strategy. On the other hand, if there is a  $\mathbf{C}$  and some choices of  $\delta$  that makes  $\Sigma G(h, \mathbf{C} \cup \{\mathbf{E}\}) \geq p$ , then it may be possible, given certain assumptions about the length choices of the honest principals, to find length choices for the intruder that will guarantee consistency with  $\mathbf{C}$ , and thus produce a winning strategy.

In order to show how such a procedure could work, we apply it to Example 3.2. This time, however, we relax the condition that all nonces be the same length, to allow nonces of any length. Thus, the masquerading message, made out of types local to  $B$ , is of the form  $(X, N_B)$  where  $N_B$  is a nonce under the control of  $A$ , and  $X$  is under the direct control of the intruder, and the spoofed message, made out of types local to  $A$ , is  $(N'_A, Y)$ , where  $N'_A$  is under the control of  $A$  and  $Y$  is under the indirect control of the intruder. We assume that the honest principals choose the length of the nonces first, and then choose random nonces of that length. Since  $X$  is chosen before  $N_B$  we then let the function tree  $\mathcal{R}$  for  $(X, N_B)$  be defined as

1.  $\mathcal{R}(\langle \rangle) = X$
2.  $\mathcal{R}(\langle z_1 \rangle) = N_B$
3.  $\mathcal{R}(\langle z_1, z_2 \rangle) = \iota$

Since  $Y$  is under the indirect control of the intruder, it is not generated until the spoofed message is received, which is after  $N'_A$ . Thus the function tree  $\mathcal{S}$  is defined as

1.  $\mathcal{S}(\langle \rangle) = N'_A$
2.  $\mathcal{S}(\langle y_1 \rangle) = Y$
3.  $\mathcal{S}(\langle y_1, y_2 \rangle) = \iota$

In this case,  $\rho_1$  and  $\rho_2$  are both the identity function.

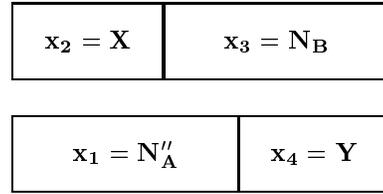
Suppose that we assume that the member of  $X$  is chosen after the member of  $N'_A$ . Then, in our construction of the interleaving  $\mathcal{I}$ , we have  $\theta_1 \circ \rho_1(1) = 2$ ,  $\theta_1 \circ \rho_1(2) = 3$ ,  $\theta_2 \circ \rho_2(1) = 1$ , and  $\theta_2 \circ \rho_2(2) = 4$ .

In this case,  $\mathbf{E}$  is  $x_2 || x_3 = x_1 || x_4$ . The length constraint tree corresponding to this game has only two leaves:  $\mathbf{C}_1 = \{l(x_2) \leq l(x_1), l(x_2) + l(x_3) = l(x_1) + l(x_4)\}$ , and

$\mathbf{C}_2 = \{l(x_2) > l(x_1), l(x_2) + l(x_3) = l(x_1) + l(x_4)\}$ .

Given a number  $p$  between 0 and 1, we wish to determine if there is a winning strategy with respect to  $p$  in the resulting type confusion game. We will restrict ourselves to the case in which the intruder and honest principals choose a length for the values under their direct control (or have it chosen for them) prior to engaging in the type confusion game.

We start with  $\mathbf{C}_1$ . This set of length constraints is illustrated by the figure below:



**Figure 1. Messages obeying  $\mathbf{C}_1$  constraints**

1. We choose  $x_1$  first, which belongs to a type under control of  $A$ . Any choice of  $x_1$  is consistent with  $\mathbf{C}_1 \cup \{\mathbf{E}\}$ , so  $G(1, \mathbf{C}_1 \cup \{\mathbf{E}\})$  is the set of all  $(x_1, 1/2^{l(x_1)})$ . Its cardinality is  $2^{l(x_1)}$ .
2. We then choose  $x_2$ , which belongs to a type under direct control of the intruder. If  $x_1$  and  $x_2$  obey the length constraints in  $\mathbf{C}_1$ , then they are consistent with  $\mathbf{E}$  if and only if  $x_2$  is equal to the first  $l(x_2)$  bits of  $x_1$ . Thus, the only strategy available to the intruder, given a particular value of  $x_1$ , is to choose  $x_2$  equal to the first  $l(x_2)$  bits of  $x_1$ . Thus,  $G(2, \mathbf{C}_1 \cup \{\mathbf{E}\})$  is the set of all such  $(\langle x_1, x_2 \rangle, 1/2^{l(x_1)})$ , and its cardinality is  $2^{l(x_1)}$ .
3. We now choose  $x_3$ , which is under control of  $B$ . The values  $x_3$  and  $x_1$  overlap on the last  $l(x_1) - l(x_2)$  bits of  $x_1$ . Since both values are chosen independently with uniform distribution, the probability of  $E$  being satisfied, that is, that the values agree on these  $l(x_1) - l(x_2)$  bits, is  $1/2^{l(x_1) - l(x_2)}$ . For any  $\langle x_1, x_2 \rangle$  consistent with  $\mathbf{C}_1 \cup \{\mathbf{E}\}$ , the cardinality of the set of  $x_3$  consistent with these constraints is  $2^{l(x_3) - l(x_1) + l(x_2)}$ . Since  $x_1$  and  $x_3$  are chosen with uniform distribution, we have  $G(3, \mathbf{C}_1 \cup \{\mathbf{E}\})$  is the set of all such  $(\langle x_1, x_2, x_3 \rangle, 1/2^{l(x_1) + l(x_3)})$  where

$\langle x_1, x_2, x_3 \rangle$  is consistent with these constraints, and its cardinality is  $2^{l(x_2)+l(x_3)}$ .

4. Finally, we have  $x_4$ , under the indirect control of the intruder. This is set equal to the last  $l(x_4)$  bits of  $x_1$ .

Thus  $G(4, \mathbf{C}_1 \cup \{\mathbf{E}\})$  is the set of all  $(\langle x_1, x_2, x_3, x_4 \rangle, 1/2^{l(x_1)+l(x_3)})$  such that  $x_1, x_2, x_3$ , and  $x_4$  satisfy the constraints of  $\mathbf{C}_1 \cup \{\mathbf{E}\}$ . Its cardinality is  $2^{l(x_2)+l(x_3)}$ . Thus, given any fixed choice for the lengths of  $x_1, x_2, x_3$ , and  $x_4$  satisfying  $\mathbf{C}_1$ , the probability of a successful type confusion attack is  $1/2^{l(x_2)-l(x_1)}$ .

We now look at  $\mathbf{C}_2$ . We assume that  $l(x_1), l(x_2), l(x_3)$ , and  $l(x_4)$  have been chosen to be consistent with  $\mathbf{C}_2$ . This set of constraints is given by the figure below:

$x_2 = \mathbf{X}$	$x_3 = \mathbf{N}_B$
$x_1 = \mathbf{N}''_A$	$x_4 = \mathbf{Y}$

**Figure 2. Messages obeying  $\mathbf{C}_2$  constraints**

1. As in the case of  $\mathbf{C}_1$ , all choices of  $x_1$  are consistent with  $\mathbf{C}_1 \cup \{\mathbf{E}\}$ . Thus  $G(1, \mathbf{C}_1 \cup \{\mathbf{E}\})$  is the set of all  $(x_1, 1/2^{l(x_1)})$ , and its cardinality is  $2^{l(x_1)}$ .
2. For each choice of  $x_1$ , choose  $x_2$ , which belongs to a type under direct control of the intruder. We need to choose the first  $l(x_1)$  bits of  $x_2$  equal to  $x_1$ ; the rest are free. Thus  $G(2, \mathbf{C}_1 \cup \{\mathbf{E}\})$  is the set of all such  $(\langle x_1, x_2 \rangle, 1/2^{l(x_1)} \cdot \delta_{\langle x_1 \rangle}(x_2))$ . Its cardinality is  $2^{l(x_1)+l(x_2)-l(x_1)} = 2^{l(x_2)}$ . The only restriction on  $\delta_{\langle x_1 \rangle}$  so far is that it be nonzero only when the first  $l(x_1)$  bits of  $x_2$  are equal to  $x_1$ .
3. We now choose  $x_3$ , which is under the direct control of  $B$ . According to the length constraints in  $\mathbf{C}_2$ , the value  $x_3$  does not overlap with any of the values previously chosen, so any choice of  $x_3$  is consistent with  $\mathbf{E}$ . The distribution of  $x_3$  is uniform, so we have  $G(3, \mathbf{C}_2 \cup \{\mathbf{E}\}) = (\langle x_1, x_2, x_3 \rangle, 1/2^{l(x_1)+l(x_3)} \cdot \delta_{\langle x_1 \rangle}(x_2))$ , and its cardinality is equal to  $2^{l(x_2)+l(x_3)}$ .
4. Finally, we set  $x_4$ , which is under the indirect control of the intruder, to be equal to the last  $l(x_2) - l(x_1)$  bits of  $x_2$  concatenated with  $x_3$ . We thus have  $G(4, \mathbf{C}_2 \cup \{\mathbf{E}\}) = \{(\langle x_1, x_2, x_3, x_4 \rangle, 1/2^{l(x_1)+l(x_3)} \cdot \delta_{\langle x_1 \rangle}(x_2))\}$

Summing up all the probabilities from  $G(4, \mathbf{C}_2 \cup \{\mathbf{E}\})$  gives a total of 1 no matter what choice of  $\delta$ , as long as it is nonzero only when the first  $l(x_1)$  bits of  $x_2$  equal  $x_1$ .

## 5 Conclusion and Discussion

We have presented a formal model and procedure for determining whether or not type confusions are possible in signed messages in a cryptographic protocol. Our approach has certain advantages over previous applications of formal methods to type confusion; we can take into account the possibility that an attacker could cause pieces of message fields to be confused with each other, as well as entire fields. This allows one to determine whether or not there is any strategy available to the attacker that will raise the probability of a successful attack above some predetermined threshold. The approach is an improvement over our previous work in [10] in that it offers an explicit model of the behavior of attacker and honest principals in terms of a type confusion tree, allowing one to use the probabilities specified in the tree to compute directly the probability a successful attack. Moreover, our model, by separating the causal relationships among types from the order in which they appear in the messages, allows the user to experiment with different assumptions about the causal ordering of message fields, or about which message fields come from trusted and which come from untrusted principals.

There are several ways in which this work could be extended. One would be to extend the method to type function trees of unbounded height. For arbitrary trees, this will probably be impossible, but most messages containing an unbounded number of terms only contain an unbounded list of fields of the same type, e.g. a message used to deliver an unbounded number of keys. Thus it may be possible to develop inductive techniques to deal with this problem.

Another, more longterm goal, is to extend this work to deal with confusion, not only about the content of messages, but the way in which they are encrypted or authenticated. As we see from the work of Bellovin [3] and Stubblebine and Gligor [13] such type confusion, in particular involving modes of encryption, can have serious effects on the security of a system. In an analogy to our experience with type confusion of GDOI, we were able to use the NRL Protocol Analyzer to reproduce some of Bellovin's attacks on the Encapsulating Security Protocol in [15], but we were not able to use the tool to perform a complete analysis of the problem. Moreover, the problem becomes somewhat more complicated than type confusion of message content in that we may need to consider the interaction between two type systems, that of the plaintext and that of the ciphertext. It will be interesting to see if our approach can be extended to this problem, which has seen relatively little exploration in the formal methods community. One exception is the work

of Stubblebine, Gligor, and Kailar on the guarantee of message integrity protection in protocols [6, 14]. The problem that they study, the ability of an intruder to create a recognizable message (instead of spoofing a particular one) is slightly different than ours, but there is enough in common so that many of their techniques may be applicable.

Finally, it might be useful to investigate the integration of this model with more mathematically rigorous models of cryptography. At present we have populated our type function trees with relatively simplistic assumptions about probability distributions related to random number generation, encryption, and so forth. This may be all that we need, but it might be useful to see if applying any of the currently available mathematical models of cryptography, or any of the emerging techniques for wedding these with formal logical models as in [11], would be of help here.

## 6 Acknowledgements

We are grateful to MSec and SMuG Working Groups, and in particular to the authors for the GDOI protocol, for many helpful discussions on this topic. We are also grateful to the participants in the 2002 FCS Workshop for helpful discussions on [10]. This work was supported by ONR.

## References

- [1] J. Alves-Foss. Provably insecure mutual authentication protocols: The two party symmetric encryption case. In *Proc. 22nd National Information Systems Security Conference.*, Arlington, VA, 1999.
- [2] M. Baugher, T. Hardjono, H. Harney, and B. Weis. The group domain of interpretation. available at <http://www.watersprings.org/pub/id/draft-irtf-smug-gdoi-01.txt>, July 2001.
- [3] S. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix UNIX Security Symposium*, San Jose, CA, 1996.
- [4] D. Harkins and D. Carrel. The Internet Key Exchange (IKE). RFC 2409, Internet Engineering Task Force, November 1998. available at <http://ietf.org/rfc/rfc2409.txt>.
- [5] J. Heather, G. Lowe, and S. Schneider. How to prevent type flaw attacks on security protocols. In *Proc. of 13th IEEE Computer Security Foundations Workshop*, pages 255–268. IEEE Computer Society Press, 2000.
- [6] R. Kailar, S. Stubblebine, and V. Gligor. Reasoning about message integrity. In F. Cristian, G. Le Lann, and T. Lunt, editors, *Dependable Computing for Critical Applications 4*, pages 41–53. Springer-Verlag, 1995.
- [7] J. Kelsey and B. Schneier. Chosen interactions and the chosen protocol attack. In *Security Protocols, 5th International Workshop April 1997 Proceedings*, pages 91–104. Springer-Verlag, 1998.
- [8] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP). Request for Comments 2408, Network Working Group, November 1998. Available at <http://ietf.org/rfc/rfc2408.txt>.
- [9] C. Meadows. Analyzing the Needham-Schroeder public key protocol: A comparison of two approaches. In *Proceedings of ESORICS '96*. Springer-Verlag, 1996.
- [10] C. Meadows. Identifying potential type confusion in authenticated messages. In *Workshop on Foundations of Computer Security*, Copenhagen, Denmark, 2002. DIKU Technical Report 02/12.
- [11] J.C. Mitchell, A. Ramanathan, A. Scedrov, and V. Teague. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols. submitted for publication, available online at <http://theory.stanford.edu/people/jcm/publications.htm>, 2002.
- [12] E. Sneekenes. Roles in cryptographic protocols. In *Proceedings of the 1992 IEEE Computer Security Symposium on Research in Security and Privacy*, pages 105–119. IEEE Computer Society Press, 1992.
- [13] S. Stubblebine and V. Gligor. On message integrity in cryptographic protocols. In *IEEE Computer Society Symposium on Research in Security and Privacy*, pages 85–104. IEEE Computer Society Press, 1992.
- [14] S. Stubblebine and V. Gligor. Protocol design and integrity protection. In *Proceedings of the 1993 Symposium on Security and Privacy*, pages 41–53. IEEE Computer Society Press, 1993.
- [15] S. Stubblebine and C. Meadows. Formal characterization and automated analysis of known-pair and chosen-text attacks. *IEEE Journal on Selected Areas on Communication*, 18(4):571–581, April 2000.
- [16] C. Meadows P. Syverson and I. Cervesato. Specification and analysis of the Group Domain of Interpretation Protocol using NPATRL and the NRL Protocol Analyzer. *Journal of Computer Security*, to appear, 2003.