Systematic Design of Two-Party Authentication Protocols*

Ray Bird Inder Gopal Amir Herzberg Phil Janson
Shay Kutten Refik Molva Moti Yung

Abstract

We investigate protocols for *authenticated exchange* of messages between two parties in a communication network. Secure authenticated exchange is essential for network security. It is not difficult to design simple and seemingly correct solutions for it, however, many such 'solutions' can be broken. We give some examples of such protocols and we show a useful methodology which can be used to break many protocols. In particular, we break a protocol that is being standardized by the ISO.

We present a new authenticated exchange protocol which is both *provably secure* and *highly efficient and practical*. The security of the protocol is proven, based on an assumption about the the cryptosystem employed (namely, that it is secure when used in CBC mode on a certain message space). We think that this assumption is quite reasonable for many cryptosystems, and furthermore it is often assumed in practical use of the DES cryptosystem. Our protocol cannot be broken using the methodology we present (which was strong enough to catch all protocol flaws we found). The reduction to the security of the encryption mode, indeed captures the non-existence of the exposures that the methodology catches (specialized to the actual use of encryption in our protocol). Furthermore, the protocol prevents chosen plaintext or ciphertext attacks on the cryptosystem.

The proposed protocol is efficient and practical in several aspects. First, it uses only conventional cryptography (like the DES, or any privately-shared one-way function) and no public-key. Second, the protocol does not require synchronized clocks or counter management. Third, only a small number of encryption operations is needed (we use no decryption), all with a single shared key. In addition, only three messages are exchanged during the protocol, and the size of these messages is minimal. These properties are similar to existing and proposed actual protocols. This is *essential* for integration of the proposed protocol into existing systems and embedding it in existing communication protocols.

1 Introduction

The extensive use of open networks and distributed systems poses increasing threats to the security of communications and operations involving end-users and network com-

*R. Bird is with IBM Networking Systems, I. Gopal, A. Herzberg, S. Kutten and M. Yung are with IBM T. J. Watson Research Center, P. Janson and R. Molva are with IBM Zurich Research Laboratory.

ponents [29]. One essential function for achieving security in a network is a mechanism to reliably authenticate the exchange of messages between two communicating parties. Such an authenticated exchange allows the establishment of the fact that the exchange of messages have passed via the other (legal) party, which provides to each party some weak proof of the identity of the other party (in some sense). This operation further enables various applications on top of it, e.g., verifying that a fresh session key agreement is taken place between the legal parties.

The basic idea of cryptographic authentication is to authenticate a message from A to B we use a *challenge* which B previously sent to A. Usually, A cryptographically combines the challenge with the authenticated message, and B verifies this combination. These cryptographic operations are done usually, and in our protocol, using conventional cryptosystem such as DES, with a key known to both parties. (Alternately, the cryptographic operations may use public key cryptosystems [19], Digital signatures [22], or Zero-Knowledge based methods [10, 11], but these alternatives require much more processing.)

Since the key changes quite rarely, the challenge should ideally be different in every authentication instance (and from security standpoint it better be "random"). There are three alternative techniques to guarantee that the challenge is different. First, the challenge may be derived from a real-time clock reading; this is called *time-stamp* challenge. Second, the challenge may be a counter that is incremented after each operation. Third, the challenge may be selected randomly from a huge space; this is called a *nonce* challenge. We concentrate on nonce-based methods which does not require clock synchronization or consistent counter maintenance, both of which are difficult to maintain especially when dealing with parallel sessions. (Note that, if desired, a nonce can be replaced by a counter value or the time-stamp, we assume a good random source is generating the nonces in use).

1.1 Scenario and attacks

We consider two parties A and B which share a key to a 'secure' cryptosystem E. The parties execute possibly many instances of the protocol, where each instance is an authenticated exchange independent of the other instances (exchanges can be executed in parallel and in an interleaving fashion, representing multiple connections between "parties" in an open network environment). Whenever a party completes an instance, it marks the instance as either *accept* (for successful authentication) or otherwise *reject*. The goal is that instances marked as accept were really an exchange of messages with a specific instance of the protocol at the other party. An *error-free history*, is one in which if one takes the history of all instances in both parties except rejected instances, the remaining accepted instances *match exactly*, except, maybe, for some last messages being still in transit. Note that this captures that indeed accepted instances indeed "passed" via the other party.

An attacker on such a protocol can be intuitively described as a third party who has

no access to the key. However, the attacker has access to past legal communication. In addition, the attacker is able to start or interfere in the middle of such protocol instances many times. The attacker tries that a party will mark an instance as "accept" incorrectly. Namely, without the other legal party recording this instance (so that the exchange is not recorded correctly). This captures the fact that the adversary is able to fool one side into accepting an exchange without the other party being actually involved.

Note that the attacker may adaptively send any message to both parties, initiate new instances of the protocol, and intercept messages sent by the parties. We do not impose response-time constraints on these actions. As noted above, in real networks, two parties that share a key may often initiate many instances simultaneously.

Our definition of correctness does not prevent the attacker from acting as a 'relay' between the two parties (by being in the middle). This is equivalent to cutting the communication lines *after* or just before the last message of the instance (and then taking the role of the legal party). However, this does not contradict the authentication of the exchange of the messages. Note that the requirement is that the *exchange* be authenticated, and not the parties themselves. Also, an attacker that removes messages on links between the parties only creates a long delay in the execution of the instance, and the correctness (and authentication) is preserved, thus, we do not consider such attacks.

We investigate both one-way and two-way authenticated exchange protocols. The difference is in the requirements in executions where messages are exchanged correctly between the two parties. In one-way protocols, it is sufficient that one party marks accept; in two-way protocols we require both parties to mark accept. In most one-way protocols, only the initiator of the protocol accepts, and therefore these protocols are not applicable to most tasks. We discuss one-way protocols mainly in order to simplify the discussion of two-way protocols.

1.2 Related Works

Many works dealing with authentication in networks combine the issues of key distribution with the issues of authentication. These works avoid our assumption that the two parties share a secret key. They use an entity, trusted by all network processors, called usually a *key distribution center* (KDC). The KDC initially shares a secret key with each of the two parties. These protocols are called *three party protocols*, and have been studied extensively, e.g. in [23, 2, 9, 24, 5, 28, 17, 3, 18]. Also, most of these protocols, e.g. [17, 23, 28], use long messages which makes them unsuitable for low network layers (where the field size devoted to security overhead should be small). Some require synchronized clocks, e.g. [23, 28], or counters, e.g. [17]. While some others require heavy computations such as public key cryptography [18].

Two-party protocols received less attention in the literature, despite their application in many networks. Some works achieve this by using public-key cryptography, e.g. [19, 21]. With a public-key cryptosystem, each party only has to know or verify the public key of

the other party, and there is no need to share secret keys. However, for reasons such as efficiency we want to use only private key cryptosystems (specifically, we believe that the block size of public-key and the computation involved with it are too large an overhead for frequent authentication of entities). The basic published proposal using private key that we have found is the ISO proposed standard [16], which we break in this paper.

Many practical authentication protocols were proposed without a convincing proof of (or argument for) security. We prove the security of our protocol by showing how one can successfully forge CBC-mode encryption using the cryptosystem, given an attacker that breaks the protocol using this cryptosystem. Since cryptosystems, e.g. DES, are usually considered to be provide secure CBC-mode, it is reasonable to consider the protocol quite secure. This is basically an application of the basic method of proof used in many of the recent works in cryptography, originating e.g. in [25, 26].

A different method of analyzing the security of protocols was presented in [5, 1] and used in other works (e.g., [6, 15]). This method applies formal logic to state assumptions and analyze the properties of protocols. This innovative approach enables better comparison of protocols, often revealing critical weaknesses or possible improvements, as was successfully done for several protocols in [5] and in subsequent works. However, the proofs of security obtained using this logic depends on assumptions which concern the protocol itself, not only the specific cryptosystem. Furthermore, the assumptions and the analysis view the cryptographic primitive (cryptosystem) as secure in a very idealized sense which is obviously much stronger than that of any candidate cryptosystem, our approach is to try to quantify the relationship between the cryptosystem and the protocol in a complexity-theoretic sense. Also, the axiomatic system may assume certain mode of use of the system and certain basic believes which may or may not exist in real environments (thus, again, assumptions may be too strong).

For example, one common assumption in logic-of-authentication is that if A and B share a secret key and A receives a message encrypted using that key with source field B, then A believes that B sent that message. This basically means that it is impossible to find an encrypted strings with a specific source field, without knowing the key. However, if the space of messages is small (say we encrypt one bit using half of the ciphertexts as zero and half as one as in "probabilistic encryption" [14]), by guessing enough strings, the attacker can find such an encryption. Also note that the assumption makes use of a property of the protocol, i.e. the use of a source field in the message. Hence, the logic cannot be used for protocols which do not follow several implicit requirements (but it may be extendable?). In particular, the protocols we investigate *cannot be analyzed* using this logic, since they combine the identity of the sender within the message, rather than having a separate field. This was required in order to use short messages, which in turn is essential for integration into existing systems. It seems that the combination of the logic approach with approaches like we take here (computational-complexity and reducing the properties of protocols to basic properties of cryptographic tool) is a useful research direction.

The present work addresses only the exchange of a single message in each instance. Obviously, in many applications we need to exchange reliably many messages, preserving

the order between them. A solution to this problem was presented in [12], however it uses longer messages since it concatenates random fields into them. It seems possible to combine our results and [12] and provide a solution which does not increase the amount of bits communicated.

1.3 Objectives and Results

The goal of this work is to design two-way authentication protocols that are provably secure yet remain realistic, efficient and simple. We require that security be proven (assuming that the cryptosystem used in the protocol is secure in some reasonable sense). Indeed, the protocol we present is secure if the cryptosystem may be used to generate secure CBC-encryption (on some message space). While we define this concept and assumption in this paper, it nevertheless appears to be an assumption made, implicitly, in many systems which employ cryptosystems such as DES in CBC mode to encrypt or to generate hash or 'fingerprint' also called MAC (Message Authentication Code).

Another security requirement is to prevent the attacker from using strong cryptanalysis techniques. In particular, we prevent chosen ciphertext attacks, which enable powerful methods such as the differential cryptanalysis [4].

The non-security requirements are all motivated by the need to present alternative to existing and proposed insecure protocols such as [16]. Such insecure protocols are already designed into systems (using existing flows of messages). It is very difficult to replace them by secure protocols which have significantly higher requirements, are substantially less efficient or behave very differently. This motivated the following requirements, all of which we meet:

- Nonce-based protocol, not requiring either synchronized clocks or stable counters. (The use on nonce implies the possibility of using time or counter value, but vice versa is not true).
- The protocol uses short messages, to be usable in low layers of the network where messages often have fixed sizes.
- Only one key is shared between the two parties, and only this key is used with the cryptosystem. This is both for compatibility with existing systems and to save loading time and secure storage requirements.
- Any secure cryptosystem may be used (in fact, we do not require decryption, so a secure one-way (or random) function suffices).
- Only three flows are used in the protocol.
- A small number of encryption operations (3) is made by each party.
- No context information is used, but the simple fact that "different parties have different names". (No use is made of assumptions like: order of names, number of key owners, temporal constraints, sequential mode of operation, and so on).

Note that by avoiding decryption, it may become easier to obtain governments approval for the (international) use and the export of the protocol. A well-defined interface to an encryption-only device/system suffices.

For simplicity, we only include the cryptographic data fields in the protocols.

2 One Way Authentication

2.1 A trivial, but insecure, one way protocol

One-way authentication is simply "authenticated acknowledgment", namely a protocol to let the sender of a message know that it was received. Figure 1 shows a trivial one way authentication protocol. This protocol may be used to authenticate B to A (right hand side) or to authenticate A to B (left hand side). Here, N_1 is a nonce generated by B (or A) and $E(N_1)$ is the value of N_1 enciphered by E. The idea is that since only A and B can decipher, then when B sends $E(N_1)$ and gets it deciphered, it believes that A was the one who deciphered.

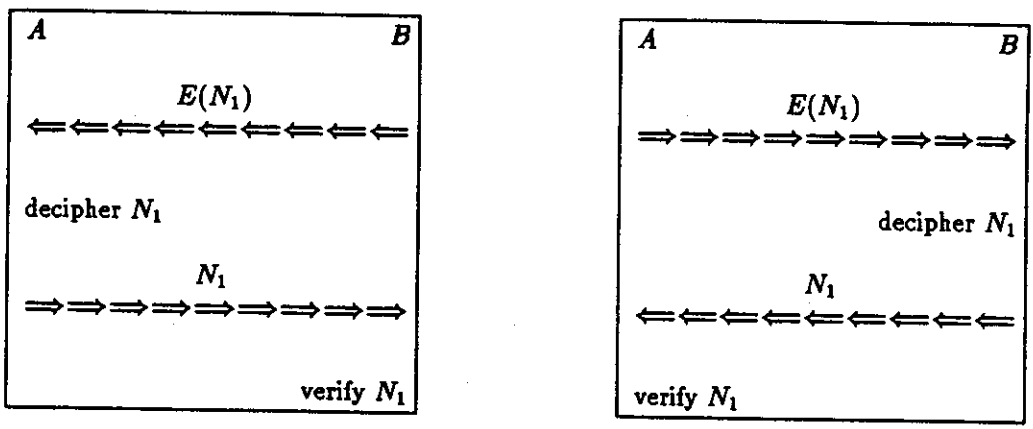

Figure 1: Trivial and insecure one way protocol

We now demonstrate an attack on this protocol. Party A tries to authenticate a message to B. An attacker X intercepts the first flow (i.e. $E(N_1)$) sent by A to B. The attacker wishes to pretend to be B, although it cannot decipher $E(N_1)$ directly. However, the attacker uses A itself to perform this decryption. For this purpose, the attacker starts a second instance with A, pretending to be B starting the protocol. (In figure 2 we use a different kind of arrows to distinguish between instances.) We named this kind of attack a *parallel session attack*, and we say that A served as an *oracle* to the attacker. In addition, notice that the protocol exposes the key to chosen-ciphertext attacks. Using it with e.g. Rabin cipher system [25] is insecure.

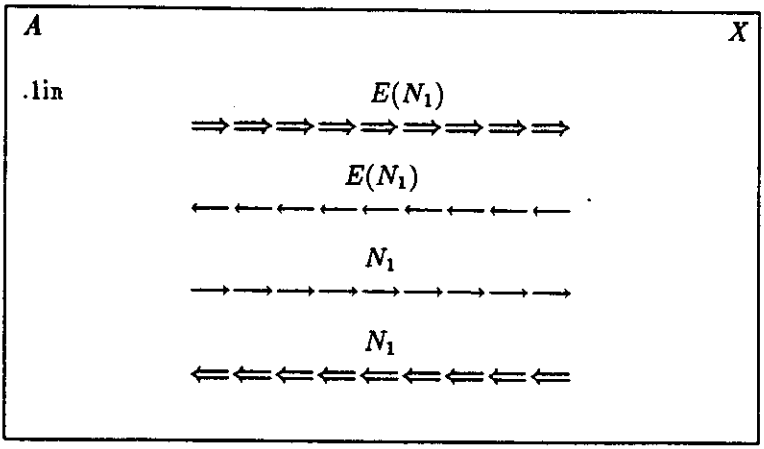

Figure 2: Parallel session attack on the simple one-way protocol

2.2 A Secure One-Way Authentication Protocol

We now modify the protocol of figure 1 slightly, to make it a secure one-way protocol. The weakness of the protocol of figure 1 is that the same key is used for two different purposes: verification of A and verification of B. One trivial solution is to use two different keys, one for each purpose. However, recall that we allow just one key. In addition, the challenging party controlled the entire ciphertext to be decrypted in the protocol, which made chosen-ciphertext attacks possible.

The solution to the first problem is to use the same key, but in two different ways (for A and B). Basically, we implement two different encryption functions E_A and E_B using just one function E. We do this by combining the encryption $E(m)$ of plaintext message m, with the identity of the processor (A or B). In addition, we make the response more complex than merely opening a ciphertext.

The security of this scheme relies on what we may call the 'pseudo-independence' of $E_A(m)$ $E_B(m)$, and $E(m)$ (to be explained in the definition below). As an example for operations that are *not* 'pseudo-independent' the reader is encouraged to break the protocol with $E_A(m) = E(A + m)$ and $E_B(m) = E(B + m)$, where + stands for the bit-wise exclusive or operation over the strings. Intuitively these operations are 'dependent' since given m, it is easy to find m' so that $A + m$ is identical to $B + m'$. We need a more secure way of combining the identity and and m.

Such 'secure combination' of values is assumed to hold for the fingerprint produced by the CBC mode of DES or any good block cipher (also when produces only last block called MAC, given $S = a, b, c$ then $MAC(S) = E(c+(b+E(a)))$, in the context of a fixed length messages and when the first field is chosen at random and the rest is fixed or a fixed function of dependent on this initial choice. The exact context is important for our

assumption to be reasonable. MAC is considered secure in many cases, and furthermore, in our context, the two common attacks on MAC calculations do not apply. The first attack is the *birthday attack* [7] when the message space is cut into two parts and using the known birthday paradox, a new message is found with some much larger probability (rather than being order of $1/n$ for $n = 2^l$, it becomes $1/\sqrt{n}$). The second attack is a *splay attack*, where given two strings $S_1 = b_1, b_2, ..., b_n$ and $S_2 = c_1, c_2, ..., c_m$, and their MAC's: $MAC(S_1) = E(b_n + (E(b_{n-1} + ...(E(b_1))..)))$, and similarly $MAC(S_2)$, one can manipulate and get the following string

$$S' = b_1, b_2, ..., b_n, \{MAC(S_1) + c_1\}, c_2, ..., c_m$$

and notice that $MAC(S_2) = MAC(S')$ so we have a new string and its MAC.

We now present a formal notion of the security usually attributed to the CBC mode fingerprint [13]. This notion is sufficient to show the security of the selection of E_A, E_B below. We present it in the context of a general size block-cipher (size l) as is customary in complexity theory. It is assumed (for practice) that length of 64 bits DES (or 128) already provides strong enough security.

Definition 2.1 *A cryptosystem $E : \{0,1\}^l \to \{0,1\}^l$ is* **depth-2 CBC secure** *if the following holds. Select arbitrary:*

- *two different identities $A, B \in \{0,1\}^l$.*
- *a small set of chosen messages $CHOSEN \subset \{0,1\}^l$ (size poly(l)).*
- *An efficient algorithm ATTACKER.*

Randomly select a key for E for blocks of size l, and two strings $w, x \in \{0,1\}^l$.

Run ATTACKER on inputs x, $E(B + E(x))$, $E(A + E(w))$, $E(B + E(w))$ and let attacker choose CHOSEN, as long as $x \notin CHOSEN$ (note that $E(B+E(x))$ was given to the attacker). Get $E(A + E(z))$, $E(B + E(z))$ for every $z \in CHOSEN$. The probability that ATTACKER produces either w or $E(A + E(x))$ is negligible (smaller than inverse polynomial in l).

Lemma 1 *If E is a depth-2 CBC secure cryptosystem then the selection $E_A(m) = E(A + E(m))$ and $E_B(m) = E(B + E(m))$ gives a secure one-way authentication protocol.*

The proof reduces all on-line (and replay and off-line) attacks to the adversary attacking the CBC mode as above. It captures any cryptographic attempt to produce the answer to the challenge in some feasible way with some non-negligible probability (as long as the assumption is correct and the query power of the attacker in its definition does not enable the attack). Note that the definition above captures both, the security of the exact challenge in a protocol where the challenge is decryption as in the protocol

above, as well as an attack on a dual protocol where the challenge involves encryption (in the CBC mode) of a challenge (which is just a plaintext w).

If the assumption would have been that the message space encrypted under CBC is not $(m, A)(m, B)$ given all random m's, but actually $(m, A+m)(m, B+m)$, then a similar assumption about CBC on this message space could be postulated. This message space increases the variability of the second block of the strings (and we call it the *variability heuristics*, which combines ciphertext chaining with plaintext chaining mode).

2.3 Two-way authentication is not simply twice one-way!

A natural proposition for a two-way authentication protocol is to combine two applications of a secure one-way authentication protocol, i.e. to use the one-way protocol to authenticate each party to the other. We show this simple two-way protocol in figure 3. The second flow in figure 3 simply sends in parallel both the second flow of the first application and the first flow of the second application.

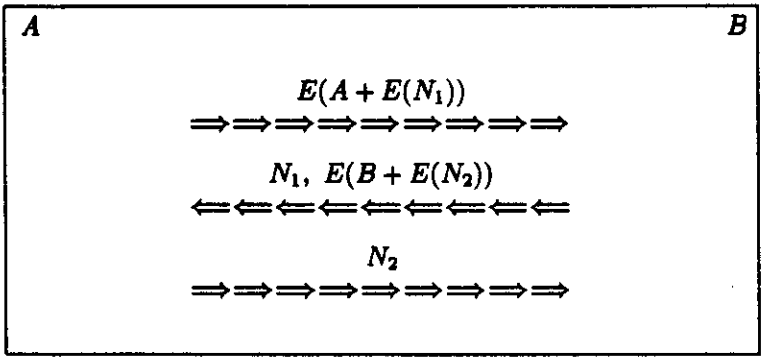

Figure 3: Combination of two runs of one-way authentication

We now show why this protocol is not a secure two way authentication protocol. Basically the same kind of message is used in the first flow and in the second flow. Therefore, an attacker can initiate a few sessions and as a result can successfully impersonate one of the parties by supplying the third flow: It gets the value of the third flow from a legal party in the second flow. This attack is demonstrated in figure 4.

3 A Technique for Breaking Protocols

We have broken many protocols suggested during this investigation. In most cases it was done as follows: we searched for the values the attacker had to send, then we search for

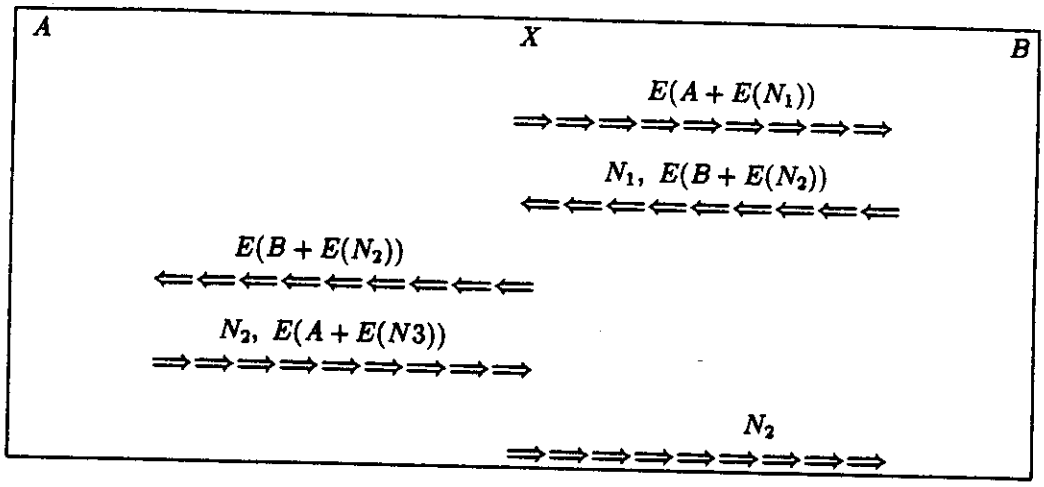

Figure 4: An attack on two-way authentication by twice one-way

a session in which a honest party could send this value (or a response from which the value can be derived by an easy computation).

The list can, of course, be used in order to check a candidate protocol. Of course, a protocol not broken by this method may still be breakable by some other method. However, *all the attacks we found* on protocols can be found using this list. Section 4 presents such an attack. Postulating an exact assumption about the underlying cryptosystem which captures the fact that using various activations of the protocol in any context and attacking the protocol directly using cryptanalysis, is the core of our security proof.

Example 3.1 *Consider the attack of figure 4. In order to attack the session with B, called the* attack session, *the attacker X started another session with A, called a* reference session. *In the reference session X sent to A exactly the same challenge, $E(B+E(N_2))$, as X got in the attack session.*

In general, the attacker can send something that depends on the attack session in a more subtle way. For example, change the protocol in such a way that the challenge in the first flow is XORed with the name of the sending party. That is: $A + E(A + E(N_1))$, while the rest of the protocol remains without change. To break this protocol X would need to send in the reference session the message $A + E(B + E(N_2))$, although in the attack session it receives $E(B + E(N_2))$.

The method is, therefore, to try to compute any of the response to challenge flows required to in an attack session, using the flows available in reference sessions (computing in any feasible way in polynomial time is allowed). Since we are going to assume that the responses to challenges include an encryption of a truly random field in each and

every instance in the context of our cryptosystem, combining useful (for attack) information from various reference session is impossible (since these pieces of information are 'independent'. Thus, reference sessions are not combined, but rather used individually. The method, therefore, uses one reference session and one attack session. Notice that there are only three types of reference sessions to consider (R_1: A starts with B; R_2: X starts with A; R_3: X starts with B). There are only two types of attack sessions (A_1: X intercepts a call; A_2: X starts a call). We illustrate R_3 and A_1 in figure 5 where given C_1 message, attacker produces a reference message C_1' to get back a response C_2' from which it attempts to deduce C_2 required as a response in the attack session.

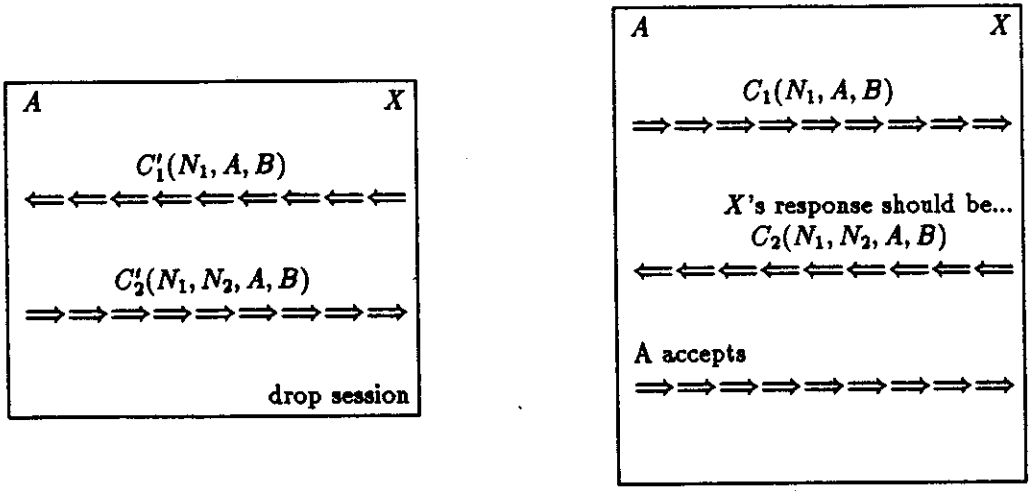

Figure 5: Reference session R_3 (left) and attack session A_1 (right).

4 Attack on ISO Protocol

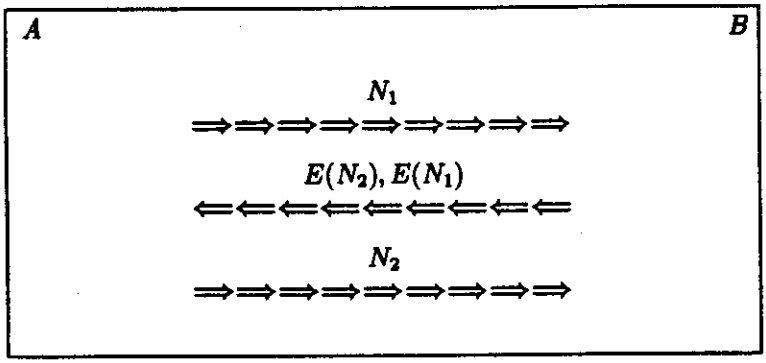

Figure 6: ISO standard proposed protocol

Due to lack of space, this version contains only one example of an attack. In Figure 6

we show a protocol proposed as a standard for "Entity Authentication Using Symmetric Techniques" by the ISO [16]. It seems that the designer of this protocol have realized the problem of using the same flow in both directions, and thus the challenges in the different directions are different.

The reader may exercise now by breaking this protocol, directly by experimentation or by applying the approach presented in the previous section. The attack we found is shown in figure 7. Notice that B is impersonated while not even being present in the communication.

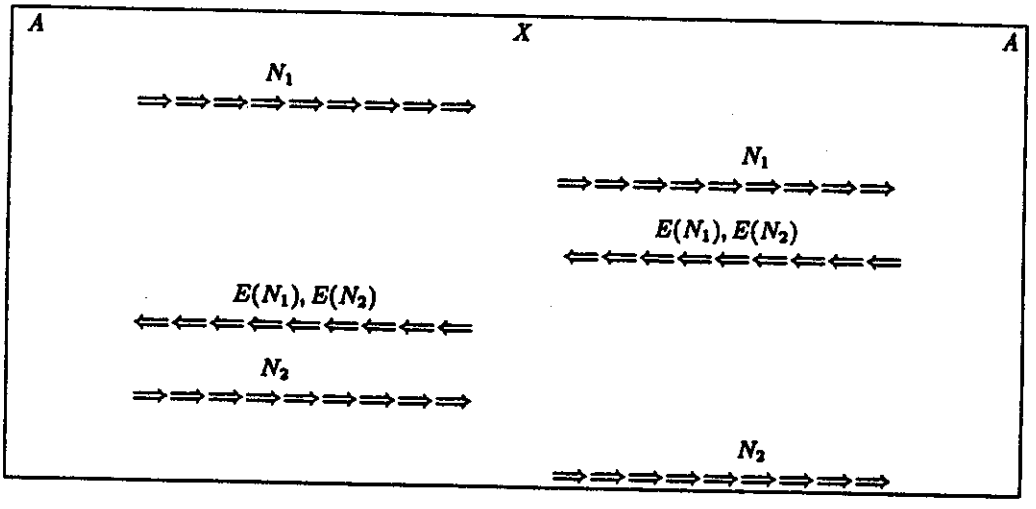

Figure 7: An attack on the ISO protocol

This attack is generated by the technique presented in section 3, as follows. We use attack session A_1 and reference session R_3. The attacker uses the second flow of R_3 to be the second flow of A_1.

5 A Secure Two-Way Authentication Protocol

5.1 Avoiding Security Weaknesses

We have tried to identify and avoid the security weaknesses of other protocols. The attacker cannot directly perform the cryptographic operations required to generate the response to a given challenge, since the key is secret and we made the protocol strong in the sense that it is defined over random instances and does not expose data for attacks (such as chosen message, say). However, the attacker may attempt to generate the re-

sponse by using the the services of a legal party by applying an instance of the protocol itself. This implies several requirements from the protocol:

- The attacker should not be able to use either party as an 'oracle'. In particular, the protocol should never perform a cryptographic operation on inputs which may be completely selected by the attacker. Instead, in every cryptographic operation, there should be at least one field which is selected randomly by a honest party. We achieve this by having a random field selected by the party performing the cryptographic operation. This ensures "pseudo-independence" of responses in various sessions (and combination of many reference sessions). This also prevents the protocol from being a source for chosen plaintext/ciphertext attacks.
- We have to prevent parallel session attacks like the one shown in figure 2. Hence, it should be infeasible to compute the responses of one party from possible responses of the other party without knowing the key. In particular, the responses of A should differ from the responses of B. To achieve this, the identity of the party is a part of the response.
- We have to prevent interleaving attacks, like the one shown in figure 4. Hence, it should be infeasible to compute the responses of one flow of the protocol from available other flows (possibly of other instances of the protocol). To achieve this, each of the flows is different (independent, in some sense).
- The different fields in the message should be cryptographically separated, i.e. the attacker cannot control one field through another field. This is necessary to support the security function of each field, e.g. to prevent attacks as described in example 3.1. We achieve this, while keeping the message short, by sending a MAC (or CBC encryption) of the concatenation of all the fields.

5.2 A secure protocol

In figure 8 we present a secure two-way authentication protocol, following the design considerations presented above. In order to find a secure protocol, we follow the security considerations above. Then, we tried to break the protocol using the methodology presented in section 3.

During this evolutionary development, we realized that it is very helpful for the protocol to use the well established CBC mode of the DES [13], Namely, it seems good to use expressions of the form $E(a + E(b + E(c)))$, where some of c, b and a are random values and others are constants or constants exclusive-ored with random values. The protocol in figure 8 follows from this form for the expressions combined with the security considerations presented above.

The proof of the security of this protocol relies heavily on an assumption about the security of the CBC mode. This assumption is an extension of the depth-2 CBC assumption presented in section 2.2. The assumption first asserts that producing something like

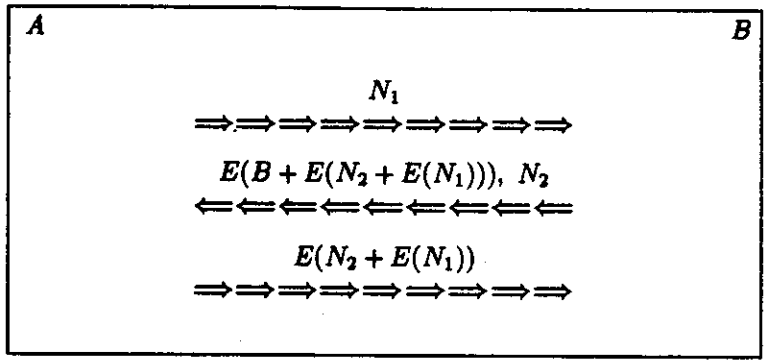

Figure 8: A secure two party authentication protocol

a response on the second message given N_1 is impossible given many choices of other challenges. Then, the assumption also takes care of the independence of the two encryptions used in the second and third flow of the protocol (as explained in the next paragraph). The proof actually reduces the general attack on a session to a direct violating of the CBC-encryption mode on the message space, thus capturing directly the hardness of attacks along the lines of the search methodology above on the protocol above. Both the proof and the exact assumption are omitted from this abstract. We just outline the properties achieved.

Notice that the response on the second flow includes a random data field N_2, so these responses in different instances are 'pseudo-independent', and there is no exposure to chosen plaintext attack. The response in the second flow is independent from the response on the third flow. We, indeed, assume that given $E(B + E(w))$, for a constant B, it is hard to produce $E(w)$ (which is an assumption equivalent to assuming one can get m from $E(m)$. If this is possible with non-negligible probability, then the depth-2 CBC assumption does not hold. Note further that our m in turn is of the form $E(N_2 + E(N_1))$ for known N_1, N_2 and getting it is, in turn, also equivalent to a violation of the depth-2 CBC. Thus, we actually build on a previous assumption in this extension. Therefore, we can justifyingly assume that encryptions done by A at the third-flow response are independent from those done by B on the second response (and vice versa).

Note that this protocol is optimal in all of the efficiency criteria, except for requiring three encryptions. The third encryption assures further strength of the method against possible statistical attacks (like birthday attacks— it isolates every block of plaintext of a given fixed format).

Also note that the use of CBC encryption is extendible in case more fields are needed to be hashed in the messages, e.g. information to be exchanged and authenticated or longer names (as long as the format is determined once and for all.

As a further improvement we suggest the protocol of figure 9. This is the previously

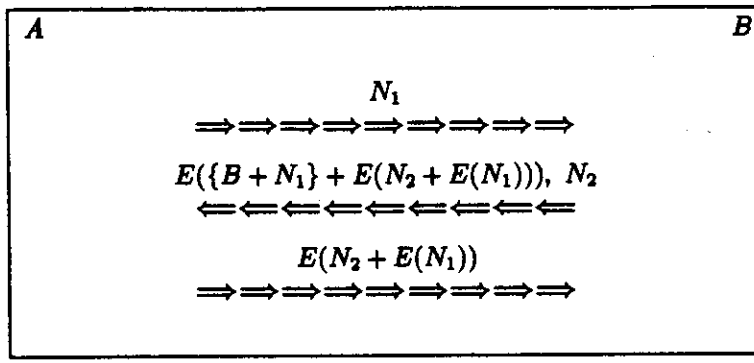

Figure 9: Heuristically improved protocol

mentioned variability heuristics. Heuristically, we change the third block in the plaintext of the response of the second flow from B to $B + N_1$ to increase the variability of this block. The security of this protocol seems to be stronger since the attacker cannot collect many pairs of the form $E(B + m)$, m for a fixed B.

Similar improvement is achieved if the name of B is long (say, mandatorily contains two blocks B_1 and B_2 and the second flow computes $E(B_1 + E(B_2 + E(N_2 + E(N_1))))$; however this introduces an additional encryption. This does not give the attacker any direct pair $m, E(m)$. But this alternative still gives the attacker a collection of pairs forming a deterministic relation $E(B_1 + E(B_2 + m))$ and m, for fixed B_1, B_2. We can combine the last two suggestions and add the variability heuristics here and get

$$E(\{B_1 + N_1\} + E(\{B_2 + N_1\} + E(N_2 + E(N_1))))$$

which seems to be the strongest combination.

6 Open Questions

The work presented a protocol which is shown to be as secure as its underlying block-cipher system in a CBC-mode. That is, this protocol encrypts only in the CBC mode over a plaintext space which includes a random block of a fixed size and fixed format. The CBC-encryptions and CBC-fingerprints, also called Message Authentication Code (MAC) (which was shown strong in our message space), are widely used in practice. Hence, this security assumption is accepted in practice, although we are not aware of it being formally defined so far. The obvious question is whether this assumption can be reduced to the security of a cryptosystem itself (or some weaker assumption). Say, can the security of the protocol be proven based solely on the assumption that we use a block-cipher which is (ideally) a random permutation.

Another question concerns the methodology which we used while developing this

protocol. We used this methodology to check the protocol against a family of 'interleaving' attacks. An obvious challenge is to identify exactly the full generality of this family of attacks, and to prove that the methodology, or some modification of it, ensures security against the general set of attacks, based on any reasonable underlying encryption method (not necessarily our encryptions).

Acknowledgments

The authors are grateful to Reid Sayre for suggesting the problem and useful discussions. It is also a pleasure to thank Martin Abadi, Sekar Chandersekaran, Don Coppersmith, Mark Davis, Bob Elander, Virgil Gligor, Don Johnson, Mike Matyas, and Jim Randall for useful discussions and comments.

References

[1] M. Abadi and M. Tuttle, *A semantics for a logic of authentication*, PODC 1991, pp. 201–216

[2] R. K. Bauer, T. A. Berson and R. J. Freiertag, *A key distribution protocol using event markers*. ACM TOCS 1 3 (1983) pp. 249-255.

[3] S. M. Bellovin and M. Merritt, Limitations of the Kerberos Authentication System. ACM Computer Communication Review 30,5 (1990) 119-132.

[4] E. Biham and A. Shamir, *Differential cryptanalysis of des-like cryptosystems*. Crypto-90.

[5] M. Burrows, M. Abadi and R. M. Needham, *A logic of authentication*. Proc. 12th ACM SOSP, ACM OSR 23 5 (Dec.89) 1 13. (Also in ACM TOCS).

[6] P-C Cheng and V. Gligor, *On the formal specification and verification of a multiparty session protocol*. IEEE Sym. on Research in Security and Privacy (1990), pp. 216–233.

[7] D. Coppersmith, *Another birthday attack*. Crypto-85, pp. 14-17.

[8] D. E. R. Denning, *Cryptography and Data Security*. Addison-Wesley, Reading, MA, 1982.

[9] D. E. Denning and G. M. Sacco, *Timestamps in key distribution systems*. CACM 24 8 (Aug.81) 533-536.

[10] A. Fiat and A. Shamir, *How to Prove Yourself: practical solutions to identification and signature problems*. Proc. of Crypto-86, Springer-Verlag LNCS 263, (1987) pp. 186-194.

[11] Z. Galil, S. Haber, and M. Yung, *Symmetric Public-Key Cryptography*. Crypto 85, pp. 128–137.

[12] O. Goldreich, A. Herzberg and Y. Mansour, *Source to Destination Communication in the Presence of Faults*. PODC 1989, pp. 85–102.

[13] *Data Encryption Standard, FIPS 46, NBS (Jan.77)*..

[14] S. Goldwasser and S. Micali, *Probabilistic encryption*. J. Comp. Systems Sci. 28 (1984), pp. 270–299.

[15] L. Gong, R. Needham and R. Yahalom, *Reasoning about belief in cryptographic protocols*. IEEE Sym. on Research in Security and Privacy (1990), pp. 234–248.

[16] *Working Draft: Entity Authentication Using Symmetric Techniques.* ISO Project JTC1.27.02.2(20.03.1.2) 06/21/1990.

[17] *Banking - Key management (wholesale). ISO 8732, Geneva (1988).*

[18] *OSI Directory - Part 8: Authentication Framework.* ISO 9594-8, Geneva (1988).

[19] J. J. Jueneman, S. M. Matyas, and C. H. Meyer, *Message Authentication. IEEE Communication Magazine*, pp. 29–40, 1985.

[20] M. J. Merritt, *Cryptographic protocols. Ph.D. dissertation GIT-ICS-83/06, The Georgia Institute of Technology, Atlanta, Ga., 1983*

[21] C. H. Meyer and S. M. Matyas, *Cryptography: a new dimension in computer data security.* Willey, New York, 1982

[22] M. Naor and M. Yung, *Universal one-way hash functions and their cryptographic applications.* ACM annual Symp. on Theory of Computing, 1989.

[23] R. M. Needham, M. D. Schroeder, *Using encryption for authentication in large networks of computers.* CACM 21 12 (1978) 993-998.

[24] D. Otway, O. Rees, *Efficient and timely mutual authentication.* ACM OSR 21 1 (Jan.87) 8-10.

[25] M. O. Rabin, *Digital signature and public-key functions as intractable as factoring.* MIT Tech. reprt TM-212 Lab. for Comp. Sci. (1979).

[26] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key crypto-systems.* CACM 21 2 (1978) 120-126 (and CACM 26 1 (1983) 96-99).

[27] D.D. Sidhu, Authentication Protocols for Computer Networks:I. *Computer Networks and ISDN Systems*, 11, pages 297–310, 1986.

[28] J. G. Steiner, et al., *Kerberos: an authentication server for open network systems.* Proc. Usenix Conf. (Winter 88).

[29] V.L. Voydoc and S.T. Kent, *Security Mechanisms in High Level Network Protocols.* Computing Surveys. 15 (1983).

Appendix: an Exhaustive Search of Sessions' Interleaving

Let A$i.j$ {R$i.j$} be the ith flow of attack session Aj {Reference session Rj} in Figure 5. Assume further that the message in the first flow contains the challenge in the first message C_1 which is a function of N_1, and possibly A and B. The message in the second flow containing the challenge C_2: a function of N_2 and A and B and C_1 (and N_1), etc. We deal with the attack session and at least one other session. Thus let C_1', C_2' N_1' and N_2' be the values used in the other session, to distinguish between them and the value used in the attack session in question.

The attacker can break the protocol if it can solve one of the equations below. (We have written just N_1, N_2, but in fact the attacker may control only C_1 and C_2, since it may not know N_1 and N_2.)

Example 6.1 *Example- in Figure 4 Equation 7 is solved trivially by the attacker. In Example 3.1 the attackers does a minor computation on C_2 to obtain C_1.*

The Equations

- (1) $A_1.2 = R_1.3$, where N_2 is free and $N_1 \neq N_1'$
- (2) $A_1.2 = R_1.2$, where N_2 is free and $N_1 \neq N_1'$
- (3) $A_1.2 = R_2.2$, where N_1' and N_2 are free but N_1' cannot depend on N_2' and $N_1' \neq N_2'$ as this would be a trivial relay (observe and cut) attack where X is just a passive observer between A and B, who have actually authenticated one another in real-time
- (4) $A_1.2 = R_3.2$, where N_1' and N_2' are free but N_1' cannot depend on N_2'
- (5) $A_2.3 = R_1.3$, where N_1 is free but cannot depend on N_2
- (6) $A_2.3 = R_1.2$, where N_1 is free but cannot depend on N_2
- (7) $A_2.3 = R2.2$, where N_1 is free but cannot depend on N_2 and N_1' is free but cannot depend on N_2'
- (8) $A_2.3 = R_3.2$, where N_1 is free but cannot depend on N_2 and N_1' is free but cannot depend on N_2'
- (9) $A_2.3 = A2.2$, where N_1 is free but cannot depend on N_2

The following example shows a case of a breakable protocol that cannot be broken by the method as described so far. This is since the attacker does not send in the attack session the *very same* value it received in the reference session. Instead it receives a value, does some trivial operation and sends the result. The method can be generalized further to cope with this additional operation. However, in order to be sure that the protocol is indeed secure one should actually have a formal proof of security stating exactly the attacker's poser and reducing it to some statement about the underlying cryptosystem, (also presented in this paper) rather than to be sure that one has coped with every possible attack. Still, we present the method as a very useful checking tool. An extension of the method should take into account derivation of responses based on easy calculations from reference sessions. Here is the example:

Example 6.2 *Let us change the protocol of Figure 4 slightly. Instead of sending N_1 in the second flow, the party (say B) sends N_1 XORed with the party name ($B + N_1$) in the example of Figure 4). The rest of the algorithm remains the same. In particular, in the third flow N_2 is sent, rather than ($A + N_2$).*

In order to break the protocol we need now to change the attack in Figure 4 slightly. The attacker who receives $A + N_2$ from A in the second flow of the reference session needs to perform an easy computation of XORing this with A to obtain the value N_2 it needs to send B.