On Session Identifiers in Provably Secure Protocols

The Bellare-Rogaway Three-Party Key Distribution Protocol Revisited*

Kim-Kwang Raymond Choo, Colin Boyd, Yvonne Hitchcock, and Greg Maitland

Information Security Research Centre
Queensland University of Technology
GPO Box 2434, Brisbane, QLD 4001, Australia
{k.choo,c.boyd,y.hitchcock,g.maitland}@qut.edu.au

Abstract. We examine the role of session identifiers (SIDs) in security proofs for key establishment protocols. After reviewing the practical importance of SIDs we use as a case study the three-party server-based key distribution (3PKD) protocol of Bellare and Rogaway, proven secure in 1995. We show incidentally that the partnership function used in the existing security proof is flawed. There seems to be no way to define a SID for the 3PKD protocol that will preserve the proof of security. A small change to the protocol allows a natural definition for a SID and we prove that the new protocol is secure using this SID to define partnering.

1 Introduction

An important direction in the computational complexity approach for protocol proofs was initiated by Bellare and Rogaway in 1993 with an analysis of a simple two party entity authentication and key exchange protocol [5]. They formally defined a model of adversary capabilities with an associated definition of security, which we refer to as the BR93 model in this paper. Since then, the BR93 model has been further revised several times. In 1995, Bellare and Rogaway analysed a three-party server-based key distribution (3PKD) protocol [6] using an extension to the BR93 model, which we refer to as the BR95 model. The most recent revision to the model was proposed in 2000 by Bellare, Pointcheval and Rogaway [4], hereafter referred to as the BPR2000 model. The proof approach by Bellare *et al.* has been applied to the analysis of public key transport based protocols [9], key agreement protocols [10, 20], password-based protocols [4, 7, 8], conference key protocols [11–14], and smart card protocols [22].

An important difference between the various models is in the way partner oracles are defined (i.e. the definition of partnership). The BR93 model defines

* This work was partially funded by the Australian Research Council Discovery Project Grant DP0345775. This is the full version of the conference proceedings that appears in 4th Conference on Security in Communication Networks - SCN 2004, volume 3352/2005 of Lecture Notes in Computer Science, pages 352–367, Springer-Verlag.

partnership using the notion of matching conversations, where a conversation is a sequence of messages exchanged between some instances of communicating oracles in a protocol run. Partnership in the BR95 model is defined using the notion of a partner function, which uses the transcript (the record of all SendClient and SendServer oracle queries) to determine the partner of an oracle. The BPR2000 model defines partnership using the notion of session identifiers (SIDs) and it is suggested that SIDs be the concatenation of messages exchanged during the protocol run. We examine partnering in the BR95 model and observe that the specific partner function defined in the proof of security for the 3PKD protocol is flawed. Consequently, the BR95 proof is invalidated, although not irreparably so. More interestingly, we also demonstrate that it does not seem possible to introduce a practical definition of partnership based on SIDs in the 3PKD protocol.

In a real world setting, it is normal to assume that a host can establish several concurrent sessions with many different parties. Sessions are specific to both the communicating parties. In the case of key distribution protocols, sessions are specific to both the initiator and the responder principals, where every session is associated with a unique session key. To model the real world implementation, the most recent definition of partnership based on SIDs in the BPR2000 model seems most natural. SIDs enable unique identification of the individual sessions. Without such means, communicating hosts will have difficulty determining the associated session key for a particular session.

We consider the use of SIDs to establish partnership analogous to the use of sockets in establishing connections between an initiating client process and a responding server process in network service protocol architecture [23]. A socket [19, 18] is bound to a port number so that the TCP layer can identify the application to which that data is destined to be sent, analogous to a SID being bound to a particular session enabling communicating principals to determine to which session messages belong. Since the initial development of sockets in the early 1980s, the use of sockets has been prevalent in protocols such as TCP/IP and UDP. In fact, Bellare *et al.* [4] recognised that SIDs are typically found in protocols such as SSL and IPSec.

The inability to define a unique SID in the 3PKD protocol so that the communicating principals can uniquely distinguish messages from different sessions leads one to question the practicality and usefulness of the protocol in a real world setting. In our view, the design of any entity authentication and/or key establishment protocol should incorporate a secure means of uniquely identifying a particular communication session among the many concurrent sessions that a communicating party may have with many different parties. One outcome of this work is such a means of session identification.

We consider the main contributions of this paper to be:

1. the observation that session identifiers are necessary for real world use of provably secure protocols,
2. demonstration of a flaw in the specific partner function used in the BR95 proof of security that invalidates the proof, and

3. proposal of an improved 3PKD protocol with a proof of security using a definition of partnership based on SIDs.

The remainder of this paper is structured as follows: Section 2 briefly explains the Bellare-Rogaway models. Section 3 describes the 3PKD protocol and the specific partner function used in the existing proof of security for the protocol. It also contains a description of a 3PKD protocol run that demonstrates a flaw in the proof due to its use of an inadequate partner function, followed by a description of how to fix it. Section 4 demonstrates that it does not seem possible to successfully introduce a definition of partnership based on SIDs to the 3PKD protocol. We then propose improvements to the 3PKD protocol. Section 5 describes the general notion of the proof of security for the improved protocol. Finally, Section 6 presents the conclusions.

2 Overview of the Bellare-Rogaway Model

Both the BR93 model [5] and the BPR2000 model [4] define provable security for entity authentication and key distribution goals. In the same flavour, the BR95 model [6] specifically defines provable security for the key distribution goal. In this section, we will focus on the BR95 and the BPR2000 definitions of security.

In all three models, the adversary $\mathcal{A}$ is a probabilistic machine that controls all the communications that take place between parties by interacting with a set of $\Pi^i_{U_1,U_2}$ oracles ($\Pi^i_{U_1,U_2}$ is defined to be the i^{th} instantiation of a principal U_1 in a specific protocol run and U_2 is the principal with whom U_1 wishes to establish a secret key). $\mathcal{A}$ also interacts with a set of $\Psi^j_{U_1,U_2}$ oracles, where $\Psi^j_{U_1,U_2}$ is defined to be the j^{th} instantiation of the server in a specific protocol run establishing a shared secret key between U_1 and U_2. The predefined oracle queries are described informally as follows.

- The $\mathsf{SendClient}(U_1, U_2, i, m)$ query allows $\mathcal{A}$ to send some message m of her choice to $\Pi^i_{U_1,U_2}$ at will. $\Pi^i_{U_1,U_2}$, upon receiving the query, will compute what the protocol specification demands and return to $\mathcal{A}$ the response message and/or decision. If $\Pi^i_{U_1,U_2}$ has either accepted with some session key or terminated, this will be made known to $\mathcal{A}$.
- The $\mathsf{SendServer}(U_1, U_2, i, m)$ query allows $\mathcal{A}$ to send some message m of her choice to some server oracle $\Psi^i_{U_1,U_2}$ at will. The server oracle, upon receiving the query, will compute what the protocol specification demands and return the response to $\mathcal{A}$.
- The $\mathsf{Reveal}(U_1, U_2, i)$ query allows $\mathcal{A}$ to expose an old session key that has been previously accepted. $\Pi^i_{U_1,U_2}$, upon receiving this query and if it has accepted and holds some session key, will send this session key back to $\mathcal{A}$.
- The $\mathsf{Corrupt}(U_1, K_E)$ query allows $\mathcal{A}$ to corrupt the principal U_1 at will, and thereby learn the complete internal state of the corrupted principal. The corrupt query also gives $\mathcal{A}$ the ability to overwrite the long-lived key of the corrupted principal with any value of her choice (i.e. K_E). This query can be used to model the real world scenarios of an insider cooperating with

the adversary or an insider who has been completely compromised by the adversary.
- The $\mathsf{Test}(U_1, U_2, i)$ query is the only oracle query that does not correspond to any of $\mathcal{A}$'s abilities. If $\Pi^i_{U_1,U_2}$ has accepted with some session key and is being asked a $\mathsf{Test}(U_1, U_2, i)$ query, then depending on a randomly chosen bit b, $\mathcal{A}$ is given either the actual session key or a session key drawn randomly from the session key distribution. The use of the $\mathsf{Test}(U_1, U_2, i)$ query is explained in Section 2.4. Note that $\Pi^i_{U_1,U_2}$ must be fresh, as defined in Section 2.3.

The definition of security depends on the notions of partnership of oracles and indistinguishability. In the BR95 model, partnership of oracles is defined using a partner function whose purpose is to enable a mapping between two oracles that should share a secret key on completion of the protocol execution. In the BPR2000 model, partnership of oracles is defined using SIDs. The definition of partnership is used in the definition of security to restrict the adversary's Reveal and Corrupt queries to oracles that are not partners of the oracle whose key the adversary is trying to guess. To avoid confusion, we will explicitly indicate which definition of partnership is used.

2.1 Notion of partnership in the BR95 Model: a Partner Function

No explicit definition of partnership was given in the BR95 model since there is no single partner function fixed for any protocol. Instead security is defined predicated on the existence of a suitable partner function. Before defining the partner function, we need the notion of a transcript. A transcript T is defined to be a sequence of communication records, where a communication record is a combination of SendClient and SendServer queries and responses to these queries. At the end of a protocol run, T will contain the record of the Send queries and the responses.

Definition 1 (BR95 Partner Function) *A partner function f in the BR95 model is syntactically defined to be a polynomial-time mapping between an initiator oracle and a partnering responder oracle (if such a partner exists), which uses the transcript T to determine the partner of an oracle.*

Let A and B be some initiator and responder principals, and also i and j be some instances of A and B respectively. The notation $f^i_{A,B}(T) = j$ denotes that the partner oracle of $\Pi^i_{A,B}$ is $\Pi^j_{B,A}$. The initial values $f^i_{A,B}(T) = *$ and $f^j_{B,A}(T) = *$ mean that neither $\Pi^i_{A,B}$ nor $\Pi^j_{B,A}$ has a BR95 partner. Two oracles are BR95 partners if, and only if, the specific BR95 partner function in use says they are. The specific BR95 partner function used in the proof of security for the 3PKD protocol will be discussed in Section 3.3.

2.2 Notion of partnership in the BPR2000 Model: SIDs

Partnership in the BPR2000 model is given by Definition 2. It is defined using the notion of SIDs, whose construction is by the concatenation of message flows in the protocol. In the BPR2000 model, an oracle who has accepted will hold the associated session key, a SID and a partner identifier (PID). Note that any oracle that has accepted will have at most one BPR2000 partner, if any at all. In Section 4.1, we demonstrate that it does not seem possible to define partnership based on SIDs for the 3PKD protocol.

Definition 2 (BPR2000 Definition of Partnership) *Two oracles, $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$, are BPR2000 partners if, and only if, both oracles have accepted the same session key with the same SID, have agreed on the same set of principals (i.e. the initiator and the responder of the protocol), and no other oracles besides $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ have accepted with the same SID*[1].

2.3 Notion of Freshness

Definitions of security in both BR95 and BPR2000 models depend on the notion of freshness of the oracle to whom the Test query is sent. Freshness is used to identify the session keys about which $\mathcal{A}$ ought not to know anything because $\mathcal{A}$ has not revealed any oracles that have accepted the key and has not corrupted any principals knowing the key. Definition 3 describes freshness in the BR95 model, which depends on the notion of partnership in Definition 1.

Definition 3 (BR95 Definition of Freshness) *Oracle $\Pi^i_{A,B}$ is fresh (or it holds a fresh session key) at the end of execution, if, and only if, oracle $\Pi^i_{A,B}$ has accepted with or without a partner oracle $\Pi^j_{B,A}$, both oracle $\Pi^i_{A,B}$ and its partner oracle $\Pi^j_{B,A}$ (if such a partner oracle exists) have not been sent a* Reveal *query, and the principals A and B of oracles $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ (if such a partner exists) have not been sent a* Corrupt *query.*

The definition of freshness in the BPR2000 model restricts the adversary $\mathcal{A}$ from sending a Corrupt query to any principal in the protocol. We adopt the BR95 version because it offers a tighter definition of freshness since for $\Pi^i_{A,B}$ to be fresh, the adversary is not restricted from sending Corrupt queries to principals apart from the principals of oracle $\Pi^i_{A,B}$ and its partner oracle $\Pi^j_{B,A}$ (if such a partner exists).

2.4 Definition of Security

Security in both the BR95 and BPR2000 models is defined using the game $\mathcal{G}$, played between a malicious adversary $\mathcal{A}$ and a collection of $\Pi^i_{U_x,U_y}$ oracles for players $U_x, U_y \in \{U_1, \ldots, U_{N_p}\}$ and instances $i \in \{1, \ldots, N_s\}$. The adversary $\mathcal{A}$ runs the game simulation $\mathcal{G}$, whose setting is as follows.

[1] Although the original paper required both parties to accept with the same PID, we have corrected this typographical error.

- **Stage 1:** $\mathcal{A}$ is able to send any $\mathsf{SendClient}$, $\mathsf{SendServer}$, Reveal, and $\mathsf{Corrupt}$ oracle queries at will in the game simulation $\mathcal{G}$.
- **Stage 2:** At some point during $\mathcal{G}$, $\mathcal{A}$ will choose a fresh session on which to be tested and send a Test query to the fresh oracle associated with the test session. Note that the test session chosen must be fresh (in the sense of Definition 3). Depending on the randomly chosen bit b, $\mathcal{A}$ is given either the actual session key or a session key drawn randomly from the session key distribution.
- **Stage 3:** $\mathcal{A}$ continues making any $\mathsf{SendClient}$, $\mathsf{SendServer}$, Reveal, and $\mathsf{Corrupt}$ oracle queries of its choice. (In the BR95 model, this stage is omitted and $\mathcal{A}$ was required to output the guess bit b' immediately after making a Test query. However, such a requirement is not strong enough, as discussed by Canetti and Krawczyk [15]. They mentioned including this stage to fix the problem, as proposed by Bellare, Petrank, Rackoff, and Rogaway in an unpublished paper.)
- **Stage 4:** Eventually, $\mathcal{A}$ terminates the game simulation and outputs a bit b', which is its guess of the value of b.

Success of $\mathcal{A}$ in $\mathcal{G}$ is measured in terms of $\mathcal{A}$'s advantage in distinguishing whether $\mathcal{A}$ receives the real key or a random value. $\mathcal{A}$ wins if, after asking a $\mathsf{Test}(U_1, U_2, i)$ query, where $\Pi^i_{U_1,U_2}$ is fresh and has accepted, $\mathcal{A}$'s guess bit b' equals the bit b selected during the $\mathsf{Test}(U_1, U_2, i)$ query. Let the advantage function of $\mathcal{A}$ be denoted by $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k})$, where $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k}) = 2 \times \mathsf{Pr}[\mathsf{b} = \mathsf{b}'] - 1$.

The BPR2000 model defines security for both entity authentication and key establishment goals, whilst the BR95 model defines security only for key establishment. In this paper, we are interested only in the notion of key establishment in the BPR2000 model since the 3PKD protocol does not consider entity authentication as its security goal. We require the definition of a negligible function.

Definition 4 ([1]) *A function $\epsilon(k) : \mathbb{N} \to \mathbb{R}$ in the security parameter k, is called negligible if it approaches zero faster than the reciprocal of any polynomial. That is, for every $c \in \mathbb{N}$ there is an integer k_c such that $\epsilon(k) \leq k^{-c}$ for all $k \geq k_c$.*

The definition of security for the protocol is identical in both the BR95 model and the BPR2000 model, with the exception that different definitions of partnership and freshness are used in the respective models.

Definition 5 (Definition of Security [4, 6]) *A protocol is secure in the BR95 model and secure under the notion of key establishment in the BPR2000 model if both the validity and indistinguishability requirements are satisfied:*

1. *Validity: When the protocol is run between two oracles in the absence of a malicious adversary, the two partner oracles accept the same session key.*
2. *Indistinguishability: For all probabilistic, polynomial-time (PPT) adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k})$ is negligible.*

3 A Flaw in the BR95 Proof of the 3PKD Protocol

In this section, we describe the 3PKD protocol and an execution of the protocol run in the presence of a malicious adversary, followed by an explanation of the specific partner function used in the BR95 proof. Using an execution of the protocol as a case study, we demonstrate that the specific partner function used in the BR95 proof enables a malicious adversary to reveal a session key at one oracle, where the same session key is considered fresh at a different, non BR95 partner oracle.

3.1 3PKD Protocol

The 3PKD protocol in Figure 1 involves three parties, a trusted server S and two principals A and B who wish to establish communication. The security goal of this protocol is to distribute a session key between two communication principals (i.e. the key establishment goal), which is suitable for establishing a secure session. Forward-secrecy and mutual authentication are not considered in the protocol. However, concurrent executions of the protocol are possible.

In the protocol, the notation $\{message\}_{K_{AS}^{enc}}$ denotes the encryption of some message under the encryption key K_{AS}^{enc} and the notation $[message]_{K_{AS}^{MAC}}$ denotes the computation of MAC digest of some message under the MAC key K_{AS}^{MAC}. K_{AS}^{enc} is the encryption key shared between A and S, and K_{AS}^{MAC} is the MAC key shared between A and S. Both keys, K_{AS}^{enc} and K_{AS}^{MAC}, are independent of each other.

1. $A \longrightarrow B : R_A$
2. $B \longrightarrow S : R_A, R_B$
3a. $S \longrightarrow A : \{SK_{AB}\}_{K_{AS}^{enc}}, [A, B, R_A, \{SK_{AB}\}_{K_{AS}^{enc}}]_{K_{AS}^{MAC}}$
3b. $S \longrightarrow B : \{SK_{AB}\}_{K_{BS}^{enc}}, [A, B, R_B, \{SK_{AB}\}_{K_{BS}^{enc}}]_{K_{BS}^{MAC}}$

Fig. 1. 3PKD protocol

The protocol begins by having A randomly select a k-bit challenge R_A and send it to the B with whom she desires to communicate. Upon receiving the message R_A from A, B also randomly selects a k-bit challenge R_B and sends R_B together with R_A as a message (R_A, R_B) to the server S. S, upon receiving the message (R_A, R_B) from B, runs the session key generator to obtain a session key SK_{AB}, which has not been used before. S then encrypts SK_{AB} with K_{AS}^{enc} and K_{BS}^{enc} to obtain ciphertexts α_A and α_B, and computes the MAC digests β_A and β_B of the strings $(A, B, R_A, \{SK_{AB}\}_{K_{AS}^{enc}})$ and $(A, B, R_B, \{SK_{AB}\}_{K_{BS}^{enc}})$ under the keys K_{AS}^{MAC} and K_{BS}^{MAC} respectively. S then sends messages (α_A,β_A) and (α_B,β_B) to A and B respectively in Steps 3a and 3b of the protocol.

3.2 Execution of Protocol Run in the Presence of a Malicious Adversary

Figure 2 depicts an example execution of the 3PKD protocol run in the presence of a malicious adversary, which will be used to demonstrate that the specific partner function used in the BR95 proof enables a malicious adversary to reveal a session key at one oracle, where the same session key is considered fresh at a different, non partner oracle. Consequently, the BR95 proof will be shown to be invalid.

1.	$A \longrightarrow B$ (intercepted by $\mathcal{A}$) :	R_A
1($\mathcal{A}$).	$\mathcal{A}$ (impersonating A) $\longrightarrow B$:	R_E
2.	$B \longrightarrow S$ (intercepted by $\mathcal{A}$) :	R_E, R_B
2($\mathcal{A}$).	$\mathcal{A}$ (impersonating B) $\longrightarrow S$:	R_A, R_B
3a.	$S \longrightarrow A$:	$\{SK_{A,B}\}_{K_{AS}^{enc}}, [A, B, R_A, \{SK_{A,B}\}_{K_{AS}^{enc}}]_{K_{AS}^{MAC}}$
3b.	$S \longrightarrow B$:	$\{SK_{A,B}\}_{K_{BS}^{enc}}, [A, B, R_B, \{SK_{A,B}\}_{K_{BS}^{enc}}]_{K_{BS}^{MAC}}$

Fig. 2. Execution of protocol run in the presence of a malicious adversary

An active adversary $\mathcal{A}$ intercepts and deletes the message R_A sent by A to B. $\mathcal{A}$ then sends a fabricated message R_E to B impersonating A. B, upon receiving the message R_E, and believing that this message originated from A, also randomly selects a k-bit challenge R_B and sends R_B together with R_E as a message (R_E, R_B) to the server S. $\mathcal{A}$ then intercepts and deletes this message (R_E, R_B), and sends the fabricated message (R_A, R_B) to S impersonating B. S, upon receiving the message (R_A, R_B) from $\mathcal{A}$, and believing that this message originated from B, runs the session key generator to obtain a unique session key SK_{AB}, which has not been used before. S encrypts SK_{AB} with the respective principals' encryption keys (i.e., K_{AS}^{enc} and K_{BS}^{enc}) to obtain the ciphertexts α_A and α_B respectively. S also computes the MAC digests (i.e., β_A and β_B) of the strings $(A, B, R_A, \{SK_{AB}\}_{K_{AS}^{enc}})$ and $(A, B, R_B, \{SK_{AB}\}_{K_{BS}^{enc}})$ under the respective keys K_{AS}^{MAC} and K_{BS}^{MAC}. S then sends the messages (α_A, β_A) and (α_B, β_B) to A and B respectively in Steps 3a and 3b of the protocol.

Immediately after both A and B have verified and accepted with the session key SK_{AB}, $\mathcal{A}$ sends a Reveal query to A and obtains the session key SK_{AB} from A. This enables the adversary $\mathcal{A}$ to break the protocol as shown in the following section. Figure 3 shows the oracle queries associated with Figure 2.

3.3 The Partner Function used in the BR95 Proof

The specific partner function used in the BR95 proof is defined in two parts, namely the partner of the responder oracle and the partner of the initiator oracle. Let f be the partner function defined in the BR95 proof, $\Pi_{A,B}^{i}$ be the initiator oracle, and $\Pi_{B,A}^{j}$ be the responder oracle. Both values $f_{A,B}^{i}(T)$ and

On query of q:	Return:	Append to T:
$\mathsf{SendClient}(A, B, i, *)$	R_A	$\langle q, R_A \rangle$
$\mathsf{SendClient}(B, A, j, R_E)$	(R_E, R_B)	$\langle q, (R_E, R_B) \rangle$
$\mathsf{SendServer}(A, B, s, (R_A, R_B))$	$((\alpha_{A,i}, \beta_{A,i}), (\alpha_{B,j}, \beta_{B,j}))$	$\langle q, ((\alpha_{A,i}, \beta_{A,i}), (\alpha_{B,j}, \beta_{B,j})) \rangle$
$\mathsf{SendClient}(A, B, i, (\alpha_{A,i}, \beta_{A,i}))$	$Accept_{A,i}$	$\langle q, Accept_{A,i} \rangle$
$\mathsf{SendClient}(B, A, j, (\alpha_{B,j}, \beta_{B,j}))$	$Accept_{B,j}$	$\langle q, Accept_{B,j} \rangle$
$\mathsf{Reveal}(A, B, i)$	$SK_{A,B,i}$	

Fig. 3. Oracle queries associated with Figure 2

$f^j_{B,A}(T)$ are initially set to $*$, which means that neither $\Pi^i_{A,B}$ nor $\Pi^j_{B,A}$ is BR95 partnered. The description of f is now given, where T is the transcript with which the adversary terminates the execution of the protocol run.

BR95 partner of the initiator oracle: The first two records of T associated with queries of the oracle $\Pi^i_{A,B}$ are examined. If the first record indicates that $\Pi^i_{A,B}$ had the role of an initiator oracle, was sent a $\mathsf{SendClient}(A, B, i, *)$ query and replied with R_A, and the second record indicates that $\Pi^i_{A,B}$'s reply to a $\mathsf{SendClient}(A, B, i, (\alpha_A, \beta_A))$ was the decision *Accept*, then T is examined to determine if some server oracle, $\Psi^k_{A,B}$, sent a message of the form (α_A, β'_A) for some β'_A. If so, determine if this message was in response to a $\mathsf{SendServer}(A, B, k, (R_A, R_B))$ query for some R_B, and if this is also true, determine if there is a unique j such that an oracle $\Pi^j_{B,A}$ generated a message (R_A, R_B). If such an oracle $\Pi^j_{B,A}$ is found, then set $f^i_{A,B}(T) = j$, meaning that the BR95 partner of $\Pi^i_{A,B}$ is $\Pi^j_{B,A}$.

Suppose that the adversary terminates the execution of the protocol run in Figure 3 with some transcript T_1. According to the BR95 partner function f, $\Pi^i_{A,B}$ has no BR95 partner because although there is a $\mathsf{SendServer}(A, B, k, (R_A, R_B))$ query for some R_B, there does not exist a unique j such that an oracle $\Pi^j_{B,A}$ generated a message (R_A, R_B). Hence, $f^i_{A,B}(T_1) = *$.

BR95 partner of the responder oracle: The first two records of T associated with queries of the oracle $\Pi^j_{B,A}$ are examined. If the first record indicates that $\Pi^j_{B,A}$ had the role of a responder oracle, and was sent a $\mathsf{SendClient}(B, A, j, R_A)$ query, and the second record indicates that $\Pi^j_{B,A}$ accepted, then determine if there is a unique i such that an oracle $\Pi^i_{A,B}$ generated a message R_A. If such an oracle $\Pi^i_{A,B}$ is found, then set $f^j_{B,A}(T) = i$, meaning that the BR95 partner of $\Pi^j_{B,A}$ is $\Pi^i_{A,B}$.

For the execution of the protocol run in Figure 3, $\Pi^j_{B,A}$ has no BR95 partner because although $\Pi^j_{B,A}$ accepted, there does not exist a unique oracle $\Pi^i_{A,B}$ that it generated a message R_E (recall R_E is fabricated by $\mathcal{A}$). Hence, $f^j_{B,A}(T_1) = *$. Hence, we have shown that the protocol state is not secure since $\mathcal{A}$ can reveal a fresh non partner oracle, either $\Pi^i_{A,B}$ or $\Pi^j_{B,A}$, and find the session key accepted by $\Pi^j_{B,A}$ or $\Pi^i_{A,B}$ respectively. It is possible to fix the flawed partner function used in the BR95 model, as shown below.

The only differences between the fixed definition of an initiator's partner and the original definition are that the server may think that the initiator and responder roles are swapped, and that the nonce output by B on behalf of A, R'_A, need not be identical to the nonce output by A itself, R_A. The definition of a responder's partner has been made analogous to that of an initiator's partner. Using the fixed partner function in our example execution, $\Pi^i_{A,B}$'s partner is $\Pi^j_{B,A}$ and $\Pi^j_{B,A}$'s partner is $\Pi^i_{A,B}$.

Fixed BR95 partner of the initiator oracle: The first two records of T associated with queries of the oracle $\Pi^i_{A,B}$ are examined. If the first record indicates that $\Pi^i_{A,B}$ had the role of an initiator oracle, was sent a $\mathsf{SendClient}(A, B, i, *)$ query and replied with R_A, and the second record indicates that $\Pi^i_{A,B}$'s reply to a $\mathsf{SendClient}(A, B, i, (\alpha_A, \beta_A))$ was the decision *Accept*, then T is examined to determine if some server oracle, $\Psi^k_{A,B}$ or $\Psi^k_{B,A}$, sent a message of the form (α_A, β'_A) for some β'_A. If so, determine if this message was in response to a $\mathsf{SendServer}(A, B, k, (R_A, R_B))$ or $\mathsf{SendServer}(B, A, k, (R_B, R_A))$ query for some R_B, and if this is also true, determine if there is a unique j such that an oracle $\Pi^j_{B,A}$ generated a message (R'_A, R_B) for any R'_A. If such an oracle $\Pi^j_{B,A}$ is found, then set $f^i_{A,B}(T) = j$, meaning that the BR95 partner of $\Pi^i_{A,B}$ is $\Pi^j_{B,A}$.

Fixed BR95 partner of the responder oracle: The first two records of T associated with queries of the oracle $\Pi^j_{B,A}$ are examined. If the first record indicates that $\Pi^j_{B,A}$ had the role of a responder oracle, was sent a $\mathsf{SendClient}(B, A, j, R'_A)$ query and replied with (R'_A, R_B), and the second record indicates that $\Pi^j_{B,A}$'s reply to a $\mathsf{SendClient}(B, A, j, (\alpha_B, \beta_B))$ was the decision *Accept*, then T is examined to determine if some server oracle, $\Psi^k_{A,B}$ or $\Psi^k_{B,A}$, sent a message of the form (α_B, β'_B) for some β'_B. If so, determine if this message was in response to a $\mathsf{SendServer}(A, B, k, (R_A, R_B))$ or $\mathsf{SendServer}(B, A, k, (R_B, R_A))$ query for some R_A, and if this is also true, determine if there is a unique i such that an oracle $\Pi^i_{A,B}$ generated a message R_A. If such an oracle $\Pi^i_{A,B}$ is found, then set $f^j_{B,A}(T) = i$, meaning that the BR95 partner of $\Pi^j_{B,A}$ is $\Pi^i_{A,B}$.

4 A Revised Protocol

We now revisit the construction of SIDs in the BPR2000 model and demonstrate that it does not seem possible to define partnership based on SIDs in the 3PKD protocol. We then propose an improvement to the 3PKD protocol with a natural candidate for the SID. Consequently, the protocol is practical in a real world setting.

4.1 Defining SIDs in the 3PKD Protocol

Bellare, Pointcheval, and Rogaway [4] suggested that SIDs can be constructed on-the-fly using fresh unique contributions from the communicating participants. Uniqueness of SIDs is necessary since otherwise two parties may share a key but not be BPR2000 partners, and hence the protocol would not be considered secure. Within the 3PKD protocol, the only values that A and B can be sure are unique are R_A and R_B. However, the integrity of only one of R_A and R_B is preserved cryptographically for each party in the protocol. Since the integrity of a SID consisting of R_A and R_B is not preserved cryptographically, attacks such as the one proposed in Section 3 are possible. An alternative would be to use an externally generated SID, such as a counter, but the use of such a SID would be inconvenient. Hence, it does not seem possible to use SIDs to successfully define partnership in the 3PKD protocol.

4.2 An Improved Provably Secure 3PKD Protocol

In order for partnership to be defined using the notion of SIDs in the 3PKD protocol, we propose an improvement to the protocol as shown in Figure 4. In the improved 3PKD protocol, S binds both values composing the SID, R_A and R_B, to the session key for each party, using the MAC digests in message flows 3a and 3b.

$$\begin{array}{|l|}\hline 1. \;\; A \longrightarrow B : R_A \\ 2. \;\; B \longrightarrow S : R_A, R_B \\ 3a. \; S \longrightarrow A : \{SK_{AB}\}_{K_{AS}^{enc}}, [A, B, R_A, R_B, \{SK_{AB}\}_{K_{AS}^{enc}}]_{K_{AS}^{MAC}}, R_B \\ 3b. \; S \longrightarrow B : \{SK_{AB}\}_{K_{BS}^{enc}}, [A, B, R_A, R_B, \{SK_{AB}\}_{K_{BS}^{enc}}]_{K_{BS}^{MAC}} \\ \hline \end{array}$$

Fig. 4. An Improved Provably Secure 3PKD Protocol

The primitives used in the protocol are the notions of a secure encryption scheme [16] and a secure message authentication scheme [17]. Both notions are now relatively standard. For the security of the underlying encryption scheme, we consider the standard definitions of *indistinguishability of encryptions* (IND) due to Goldwasser and Micali [16] and *chosen-plaintext attack* (CPA). For the security of the underlying message authentication scheme, we consider the standard definition of existential unforgeability under *adaptive chosen-message attack* (ACMA) due to Goldwasser, Micali, and Rivest [17].

Theorem 1 *The improved 3PKD protocol is a secure key establishment protocol in the sense of Definition 5 if the underlying message authentication scheme is secure in the sense of existential unforgeability under ACMA and the underlying encryption scheme is indistinguishable under CPA.*

5 Security Proof

The proof of Theorem 1 generally follows that of Bellare and Rogaway [6], but is adjusted to the different partnering function used. The validity of the protocol is straightforward to verify and we concentrate on the indistinguishability requirement. The security is proved by finding a reduction to the security of the underlying message authentication scheme and the underlying encryption scheme.

The general notion of the proof is to assume that there exists an adversary $\mathcal{A}$ who can gain a non-negligible advantage in distinguishing the test key in game $\mathcal{G}$ (i.e. $\mathsf{Adv}^{\mathcal{A}}(k)$ is non-negligible), and use $\mathcal{A}$ to break the underlying encryption scheme or the message authentication scheme. In other words, we consider an adversary $\mathcal{A}$ that breaks the security of the protocol.

Using results of Bellare, Boldyreva and Micali [2], we may allow an adversary against an encryption scheme to obtain encryptions of the same plaintext under different independent encryption keys. Such an adversary is termed a multiple eavesdropper, $\mathcal{ME}$. In the 3PKD protocol, the server, upon receiving a message from the responder principal, sends out two ciphertexts derived from the encryption of the same plaintext under two independent encryption keys. Hence, we consider a multiple eavesdropper $\mathcal{ME}$ who is allowed to obtain encryptions of the same plaintext under two different independent encryption keys. The formal definition of $\mathcal{ME}$ is given by Definition 6.

Definition 6 ([2, 6]) *Let $\Omega = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme with security parameter k, $\mathcal{SE}$ be the single eavesdropper and $\mathcal{ME}$ be the multiple eavesdropper, and $\mathcal{O}_{k_A}$ and $\mathcal{O}_{k_B}$ be two different independent encryption oracles associated with encryption keys k_A and k_B. We define the advantage functions of $\mathcal{SE}$ and $\mathcal{ME}$ to be:*

$$\begin{aligned}
Adv^{\mathcal{SE}}(k) &= 2 \times Pr[\mathcal{SE} \leftarrow \mathcal{O}_{k_A}; (m_0, m_1 \overset{R}{\leftarrow} \mathcal{SE}); \theta \overset{R}{\leftarrow} \{0,1\}; \gamma_A \overset{R}{\leftarrow} \mathcal{O}_{k_A}(m_\theta) \\
&\quad : \mathcal{SE}(\gamma_A) = \theta] - 1 \\
Adv^{\mathcal{ME}}(k) &= 2 \times Pr[\mathcal{ME} \leftarrow \mathcal{O}_{k_A}, \mathcal{O}_{k_B}; (m_0, m_1 \overset{R}{\leftarrow} \mathcal{ME}); \theta \overset{R}{\leftarrow} \{0,1\}; \\
&\quad \gamma_A \overset{R}{\leftarrow} \mathcal{O}_{k_A}(m_\theta), \gamma_B \overset{R}{\leftarrow} \mathcal{O}_{k_B}(m_\theta) : \mathcal{ME}(\gamma_A, \gamma_B) = \theta] - 1
\end{aligned}$$

Lemma 1 ([2]) *Suppose the advantage function of $\mathcal{SE}$ against the encryption scheme is ϵ_k. Then the advantage function of $\mathcal{ME}$ is at most $2 \times \epsilon_k$.*

As a consequence of Lemma 1, an encryption scheme secure against IND-CPA in the single eavesdropper setting will also be secure against IND-CPA in the multiple eavesdropper setting [2].

An overview of the proof of Theorem 1 is now provided (a complete proof appears in Appendix A). The proof is divided into two cases since the adversary $\mathcal{A}$ can either gain her advantage against the protocol by forging a MAC digest with respect to some user's MAC key or gain her advantage against the protocol without forging a MAC digest.

5.1 Adaptive MAC Forger $\mathcal{F}$

Following the approach of Bellare, Kilian and Rogaway [3], we quantify security of the MAC scheme in terms of the probability of a successful MAC forgery under adaptive chosen-message attack, which we denote by $\mathsf{Pr}[\mathsf{Succ}^{\mathcal{F}}(k)]$. For the MAC scheme to be secure under chosen-message attack, $\mathsf{Pr}[\mathsf{Succ}^{\mathcal{F}}(k)]$ must be negligible. In other words, the MAC scheme is considered broken if a forger $\mathcal{F}$ is able to produce a valid MAC forgery for a MAC key unknown to it.

The first part of the proof of security for the improved 3PKD protocol assumes that the adversary $\mathcal{A}$ gains her advantage by forging a valid MAC digest for a MAC key that $\mathcal{A}$ does not know. More precisely, we define $\mathsf{MACforgery}$ to be the event that at some point in the game $\mathcal{A}$ asks a $\mathsf{SendClient}(B, A, j, (\alpha_{B,j}, \beta_{B,j}))$ query to some fresh oracle $\Pi^j_{B,A}$, such that the oracle accepts, but the MAC value $\beta_{B,j}$ used in the query was not previously output by a fresh oracle. We then construct an adaptive MAC forger $\mathcal{F}$ against the security of the message authentication scheme using $\mathcal{A}$, as shown in the following attack game, $\mathcal{G}_{\mathcal{F}}$.

- **Stage 1:** $\mathcal{F}$ is provided permanent access to the MAC oracle $\mathcal{O}_{x'}$ associated with the MAC key x' throughout $\mathcal{G}_{\mathcal{F}}$.
- **Stage 2:** $\mathcal{F}$ runs $\mathcal{A}$ to produce a valid MAC forgery for the MAC key x' that is known to neither $\mathcal{F}$ nor $\mathcal{A}$. By examining all oracle queries made by $\mathcal{A}$, $\mathcal{F}$ outputs the MAC forgery.

The objective of $\mathcal{F}$ is to output a valid MAC forgery for a MAC message which was not previously asked of $\mathcal{O}_{x'}$. It is shown in the proof that $\mathsf{Pr}[\mathsf{MACforgery}] \le N_p \cdot \mathsf{Pr}[\mathsf{Succ}^{\mathcal{F}}(k)]$, where N_p is polynomial in the security parameter, k. Hence, $\mathsf{Pr}[\mathsf{MACforgery}]$ is negligible if the message authentication scheme in use is secure.

5.2 Multiple Eavesdropper Attacker $\mathcal{ME}$

The second part of the proof assumes that the adversary $\mathcal{A}$ gains her advantage without forging a MAC digest. We construct another algorithm $\mathcal{ME}$ that uses $\mathcal{A}$ against the security of the encryption scheme, whose behaviour is described by the attack game $\mathcal{G}_{\mathcal{ME}}$ shown below and in Figure 5. The objective of $\mathcal{ME}$ is to correctly predict the challenge bit θ in the game simulation $\mathcal{G}_{\mathcal{ME}}$ (i.e. have $\theta' = \theta$).

- **Stage 1:** $\mathcal{ME}$ is provided permanent access to two different encryption oracles $\mathcal{O}_{k_A}$ and $\mathcal{O}_{k_B}$ associated with encryption keys k_A and k_B respectively throughout the game $\mathcal{G}_{\mathcal{ME}}$.
- **Stage 2:** $\mathcal{ME}$ chooses a pair of messages (m_0, m_1) of equal length and hands them to the challenger. The challenger then chooses a random challenge bit, θ (i.e., $\theta \xleftarrow{R} \{0,1\}$), and returns the ciphertexts γ_A and γ_B to $\mathcal{ME}$, where $\gamma_A = \mathcal{E}_{k_A}(m_\theta)$ and $\gamma_B = \mathcal{E}_{k_B}(m_\theta)$.
- **Stage 3:** $\mathcal{ME}$ runs $\mathcal{A}$ to determine whether m_0 or m_1 was encrypted as γ_A and γ_B. By examining all oracle queries made by $\mathcal{A}$, $\mathcal{ME}$ outputs her prediction, θ'.

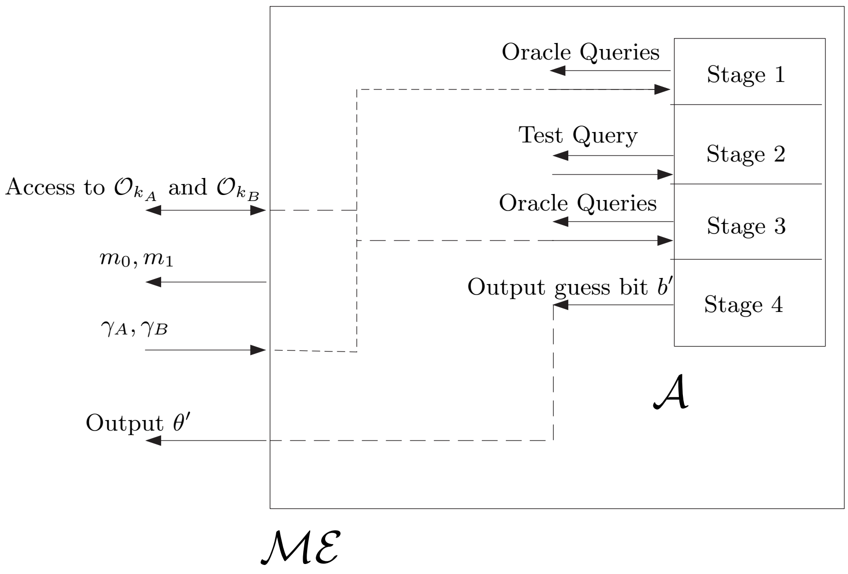

Fig. 5. Game $\mathcal{G}_{\mathcal{ME}}$

We denote the probability that $\mathcal{ME}$ correctly guesses the challenge bit θ by $\mathsf{Pr}[\mathsf{Succ}^{\mathcal{ME}}(k)]$, and observe that for the encryption scheme to be IND-CPA, $\mathsf{Adv}^{\mathcal{ME}}(k) = 2 \times \mathsf{Pr}[\mathsf{Succ}^{\mathcal{ME}}(k)] - 1$ must be negligible. It is shown in the proof that $(\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}}) = N_p^2 N_s \cdot \mathsf{Adv}^{\mathcal{ME}}(k)$, where N_p and N_s are polynomial in the security parameter. Hence, $(\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}})$ is negligible if the encryption scheme in use is secure.

5.3 Conclusion of Proof

The proof concludes by observing that:

$$\begin{aligned}\mathsf{Adv}^{\mathcal{A}}(k) &= (\mathsf{Adv}^{\mathcal{A}}(k)|\mathsf{MACforgery}) \times \mathsf{Pr}[\mathsf{MACforgery}] \\ &\quad + (\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}}) \times \mathsf{Pr}[\overline{\mathsf{MACforgery}}] \\ &\leq \mathsf{Pr}[\mathsf{MACforgery}] + (\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}})\end{aligned}$$

Hence, $\mathsf{Adv}^{\mathcal{A}}(k)$ is negligible when the encryption scheme and message authentication scheme in use are secure against IND-CPA and secure against existential forgery under ACMA respectively, and therefore the improved 3PKD protocol is also secure.

6 Conclusion and Future Work

By making a small change to the 3PKD protocol we have allowed SIDs to be defined in a natural way. This makes the improved protocol a more useful tool for practical applications since we have provided a simple way to identify which secure session key should be used on which communication channel. At the same time we would argue that the resulting definition of partnering is more intuitive, and consequently we believe that our proof of security is more straightforward than the one presented by Bellare and Rogaway in their original paper.

As a result of our findings we would recommend that all provably secure protocols should use partnering definitions based on SIDs. This situation is common for two-party protocols [4, 10, 15]; even if a SID is not explicitly used in the security definition, one can easily be defined from the fresh inputs of each principal. When it comes to multi-party protocols the situation is not so clear. While protocols which use only broadcast messages [21] have a natural SID, protocols which utilise point-to-point messages do not have this property [12, 13]. It would be interesting to know whether the protocols without broadcast messages can be provided with a secure means to obtain a shared SID.

References

1. Mihir Bellare. A Note on Negligible Functions. *Journal of Cryptology*, 15(4):271–284, 2002.
2. Mihir Bellare, A. Boldyreva, and Silvio Micali. Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. In Bart Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, pages 259 – 274. Springer-Verlag, 2000. Volume 1807 of Lecture Notes in Computer Science.
3. Mihir Bellare, Joe Kilian, and Phillip Rogaway. The Security of the Cipher Block Chaining Message Authentication Code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000.
4. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In Bart Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, pages 139 – 155. Springer-Verlag, 2000. Volume 1807 of Lecture Notes in Computer Science.
5. Mihir Bellare and Phillip Rogaway. Entity Authentication and Key Distribution. In Douglas R. Stinson, editor, *Advances in Cryptology - Crypto 1993*, pages 110–125. Springer-Verlag, 1993. Volume 773 of Lecture Notes in Computer Science.
6. Mihir Bellare and Phillip Rogaway. Provably Secure Session Key Distribution: The Three Party Case. In *27th ACM Symposium on the Theory of Computing*, pages 57–66. ACM Press, 1995.
7. Steven M. Bellovin and Michael Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *Symposium on Security and Privacy*, pages 72–84. IEEE, 1992.
8. Steven M. Bellovin and Michael Merritt. Augmented Encrypted Key Exchange: A Password-Based Protocol Secure Against Dictionary Attacks and Password File Compromise. In *1st ACM Conference on Computer and Communications Security*, pages 244–250. ACM Press, 1993.

9. Simon Blake-Wilson and Alfred Menezes. Security Proofs for Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques. In Bruce Christianson, Bruno Crispo, T. Mark A. Lomas, and Michael Roe, editors, *Security Protocols Workshop*, pages 137–158. Springer-Verlag, 1997. Volume 1361 of Lecture Notes in Computer Science.
10. Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman Key Agreement Protocols. In Stafford E. Tavares and Henk Meijer, editors, *Selected Areas in Cryptography - SAC 1998*, pages 339–361. Springer-Verlag, 1998. Volume 1556 of Lecture Notes in Computer Science.
11. Colin Boyd and Juan Manuel Gonzalez-Nieto. Round-optimal Contributory Conference Key Agreement. In Yvo Desmedt, editor, *Public Key Cryptography - PKC 2003*, pages 161–174. Springer-Verlag, 2003. Volume 2567 of Lecture Notes in Computer Science.
12. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Provably Authenticated Group Diffie–Hellman Key Exchange — The Dynamic Case. In Colin Boyd, editor, *Advances in Cryptology - Asiacrypt 2001*, pages 209–223. Springer-Verlag, 2001. Volume 2248 of Lecture Notes in Computer Science.
13. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic Group Diffie–Hellman Key Exchange under Standard Assumptions. In Lars R. Knudsen, editor, *Advances in Cryptology - Eurocrypt 2002*, pages 321–336. Springer-Verlag, 2002. Volume 2332 of Lecture Notes in Computer Science.
14. Emmanuel Bresson, Olivier Chevassut, David Pointcheval, and Jean-Jacques Quisquater. Provably Authenticated Group Diffie–Hellman Key Exchange. In *8th ACM Conference on Computer and Communications Security*, pages 209–223. ACM Press, 2001.
15. Ran Canetti and Hugo Krawczyk. Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. In Birgit Pfitzmann, editor, *Advances in Cryptology - Eurocrypt 2001. Available from http://eprint.iacr.org/2001/040/*, pages 453–474. Springer-Verlag, 2001. Volume 2045 of Lecture Notes in Computer Science.
16. S. Goldwasser and S. Micali. Probabilisitic Encryption. *Journal of Computer and System Sciences*, 28:270–299, 1984.
17. S. Goldwasser, S. Micali, and R.L. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM Journal on Computing*, 17(2):281 – 308, 1988.
18. The Internet Engineering Task Force. *RFC 0147 Definition of a Socket*, 1971. http://www.ietf.org/rfc/rfc0147.txt?number=0147.
19. The Internet Engineering Task Force. *RFC 0204 Sockets in Use*, 1971. http://www.ietf.org/rfc.html/.
20. Markus Jakobsson and David Pointcheval. Mutual Authentication and Key Exchange Protocol for Low Power Devices. In Paul F. Syverson, editor, *5th International Conference on Financial Cryptography - FC 2001*, pages 178–195. Springer-Verlag, 2001. Volume 2339 of Lecture Notes in Computer Science.
21. Jonathon Katz and Moti Yung. Scalable Protocols for Authenticated Group Key Exchange. In Dan Boneh, editor, *Advances in Cryptology - Crypto 2003*, pages 110–125. Springer-Verlag, 2003. Volume 2729 of Lecture Notes in Computer Science.
22. Victor Shoup and Aviel D. Rubin. Session Key Distribution Using Smart Cards. In Ueli M. Maurer, editor, *Advances in Cryptology - Eurocrypt 1996*, pages 321–331. Springer-Verlag, 1996. Volume 1070 of Lecture Notes in Computer Science.
23. William Stallings. *Data and Computer Communications – 7th Edition*. Prentice Hall, 2004.

A Proof of Theorem 1

The security of the improved protocol is proved by finding a reduction to the security of the encryption scheme and the message authentication scheme. Let N_s be the maximum number of sessions between any two parties in the protocol run and N_p be the maximum number of players in the protocol run, where both N_s and N_p are polynomial in the security parameter k.

The proof is divided into two cases since the adversary $\mathcal{A}$ can either gain her advantage against the protocol while forging a MAC digest with respect to some user's MAC key or gain her advantage against the protocol without forging a MAC digest. The proof assumes that there exists an adversary $\mathcal{A}$ that has a non-negligible advantage against the protocol, and shows that this implies that either the encryption scheme or the message authentication scheme is insecure.

A.1 Adaptive MAC Forger $\mathcal{F}$

For the first part of the proof, assume that at some stage $\mathcal{A}$ asks a $\mathsf{SendClient}(B, A, j, (\alpha_{B,j}, \beta_{B,j}))$ query to some fresh oracle, $\Pi^j_{B,A}$, such that $\Pi^j_{B,A}$ accepts, but the MAC digest value $\beta_{B,j}$ used in the $\mathsf{SendClient}(B, A, j, (\alpha_{B,j}, \beta_{B,j}))$ query was not previously output by a fresh oracle. Hence, $\mathsf{Pr}[\mathsf{MACforgery}]$ is non-negligible. We construct an adaptive MAC forger $\mathcal{F}$ against the security of the message authentication scheme using $\mathcal{A}$. We define an attack game $\mathcal{G}_{\mathcal{F}}$ as follows.

- **Stage 1:** $\mathcal{F}$ is provided permanent access to the MAC oracle $\mathcal{O}_{x'}$ associated with the MAC key x' throughout the game $\mathcal{G}_{\mathcal{F}}$.
  - $\mathcal{F}$ randomly chooses a principal $\overline{U}$, where $\overline{U} \in \{U_1, \dots, U_{N_p}\}$. $\overline{U}$ is $\mathcal{F}$'s guess at which principal $\mathcal{A}$ will choose for the MAC forgery.
  - $\mathcal{F}$ randomly generates the list of MAC keys for the $\{U_1, \dots, U_{N_p}\} \setminus \{\overline{U}\}$ principals.
  - $\mathcal{F}$ randomly generates the list of encryption keys of the $\{U_1, \dots, U_{N_p}\}$ principals.
- **Stage 2:** $\mathcal{F}$ runs $\mathcal{A}$ and answers all oracle queries from $\mathcal{A}$. This can be done in a straightforward manner since $\mathcal{F}$ can respond to all oracle queries from $\mathcal{A}$ as required using the keys chosen in Stage 1 and $\mathcal{O}_{x'}$. In addition, $\mathcal{F}$ records all the MAC digests it receives from $\mathcal{O}_{x'}$. If, during its execution, $\mathcal{A}$ makes an oracle query that includes a forged MAC digest for $\overline{U}$, then $\mathcal{F}$ outputs the MAC forgery as its own, and halts. Otherwise, $\mathcal{F}$ halts when $\mathcal{A}$ halts.

The random choice of $\overline{U}$ by $\mathcal{F}$ means that the probability that $\overline{U}$ is the party for whom $\mathcal{A}$ generates a forgery (if $\mathcal{A}$ generates any forgery at all) is at least $1/N_p$. Hence, the success probability of $\mathcal{F}$ is

$$\mathsf{Pr}[\mathsf{Succ}^{\mathcal{F}}(k)] \geq \frac{\mathsf{Pr}[\mathsf{MACforgery}(k)]}{N_p}.$$

Hence

$$\mathsf{Pr}[\mathsf{MACforgery}] \leq N_p \cdot \mathsf{Pr}[\mathsf{Succ}^{\mathcal{F}}(k)].$$

Since we know that N_p is polynomial in the security parameter k and $\mathsf{Adv}^{\mathcal{F}}(k)$ is negligible by definition of the security of the message authentication scheme, $\Pr[\mathsf{MACforgery}]$ is also negligible.

A.2 Multiple Eavesdropper Attacker $\mathcal{ME}$

For the second part of the proof, assume that $\mathsf{Adv}^{\mathcal{A}}(k)$ is non-negligible, but that $\mathcal{A}$ achieves this advantage without forging a MAC digest. Recall that $\mathcal{ME}$ hands a pair of messages (m_0, m_1) to the challenger and receives $\gamma_A = \mathcal{E}_{k_A}(m_\theta)$ and $\gamma_B = \mathcal{E}_{k_B}(m_\theta)$, which are encryptions under two different keys of one of the messages.

$\mathcal{ME}$ runs $\mathcal{A}$ to determine whether m_0 or m_1 was encrypted as γ_A and γ_B. By examining all oracle queries made by $\mathcal{A}$, $\mathcal{ME}$ outputs her prediction, θ'. The details of the game $\mathcal{G}_{\mathcal{ME}}$ in Figure 5 are explained as follows.

Stage 1: $\mathcal{ME}$ is provided permanent access to two different encryption oracles $\mathcal{O}_{k_A}$ and $\mathcal{O}_{k_B}$ associated with encryption keys k_A and k_B respectively.

Stage 2: $\mathcal{ME}$ chooses a pair of messages (m_0, m_1) of equal length and hands them to the challenger. The challenger then chooses a random challenge bit, θ (i.e., $\theta \stackrel{R}{\leftarrow} \{0, 1\}$), and returns the ciphertexts γ_A and γ_B to $\mathcal{ME}$, where $\gamma_A = \mathcal{E}_{k_A}(m_\theta)$ and $\gamma_B = \mathcal{E}_{k_B}(m_\theta)$. $\mathcal{ME}$ then randomly chooses the following:

- target initiator and responder principals, A and B, where $A, B \stackrel{R}{\leftarrow} \{U_1, \ldots, U_{N_p}\}$,
- target session between A and B whose instance at the server is u (i.e. the session of $\Psi^u_{A,B}$), where $u \stackrel{R}{\leftarrow} \{1, \ldots, N_s\}$,
- list of encryption keys (i.e. $K^{enc}_{U_i}$) for all participants except A and B, and list of MAC keys (i.e. $K^{MAC}_{U_i}$) for all participants.

Stage 3: $\mathcal{ME}$ runs $\mathcal{A}$ to determine whether m_0 or m_1 was encrypted as γ_A and γ_B. By examining all oracle queries made by $\mathcal{A}$, $\mathcal{ME}$ outputs her prediction, θ'. $\mathcal{A}$ makes a series of $\mathsf{SendClient}(U_1, U_2, \iota, m)$, $\mathsf{SendServer}(U_1, U_2, \iota, m)$, $\mathsf{Reveal}(U, \iota)$, and $\mathsf{Corrupt}(U, K)$ oracle queries to $\mathcal{ME}$, which can be answered by $\mathcal{ME}$ as explained below. At some point during $\mathcal{G}_{\mathcal{ME}}$, $\mathcal{A}$ makes a $\mathsf{Test}(U_1, U_2, \iota)$ query on some fresh oracle (in the sense of Definition 3), which $\mathcal{ME}$ must also answer.

On receipt of $\mathsf{SendClient}(U_1, U_2, \iota, m)$ queries:

- If U_1 = initiator, U_2 = responder, and $m = *$, then this will start a protocol run. This query can be successfully answered by $\mathcal{ME}$ and the outgoing message is some randomly chosen k-bit challenge R_{U_1}.
- If U_1 = responder, U_2 = initiator, and m is some k-bit challenge R_{U_1}, then $\mathcal{ME}$ will successfully answer with R_{U_1}, R_{U_2}, where R_{U_2} is some randomly chosen k-bit challenge.

– If U_I = initiator, U_R = responder (where $\{U_1, U_2\} = \{U_I, U_R\}$), and the message $m = (\{SK_{U_1,U_2,\iota}\}_{K^{enc}_{U_1}}, [U_I, U_R, R_{U_I}, R_{U_R}, \{SK_{U_1,U_2,\iota}\}_{K^{enc}_{U_1}}]_{K^{MAC}_{U_1}})$. All session keys (if accepted) are known from the $\mathsf{SendServer}(U_1, U_2, \iota, m)$ queries, since we assume that $\mathcal{A}$ is not able to produce any MAC forgeries. Hence, if the MAC digest verifies correctly, the MAC digest must be authentic (i.e. must have been generated by $\mathcal{ME}$ during a SendServer query) and in this case, $\mathcal{ME}$ will output the decision $\delta = accept$. Otherwise, $\mathcal{ME}$ will output the decision $\delta = reject$, as the protocol specification demands.
– If U_I = initiator, U_R = responder (where $\{U_1, U_2\} = \{U_I, U_R\}$), and $m \in \{(\gamma_A, [U_I, U_R, R_{U_I}, R_{U_R}, \gamma_A]_{K^{MAC}_{U_1}}), (\gamma_B, [U_I, U_R, R_{U_I}, R_{U_R}, \gamma_B]_{K^{MAC}_{U_1}})\}$. Both γ_A and γ_B are given as input to $\mathcal{ME}$ and hence known to $\mathcal{ME}$. With the assumption that $\mathcal{A}$ is not able to produce any MAC forgeries, if the MAC digest verifies correctly, then the MAC digest must have been generated by $\mathcal{ME}$ during a SendServer query. In this case, $\mathcal{ME}$ will output the decision $\delta = accept$, otherwise $\mathcal{ME}$ will output the decision $\delta = reject$, as the protocol specification demands.
– In all other cases the input to the SendClient query is invalid, so $\mathcal{ME}$ will randomly choose a bit θ' as its response and hand it to the challenger. Hence, SendClient queries can be correctly answered by $\mathcal{ME}$.

On receipt of $\mathsf{SendServer}(U_1, U_2, \iota, m)$ **queries:**

Message m must be of the form (R_{U_1}, R_{U_2}), where R_{U_1} and R_{U_2} are some k-bit challenges, otherwise $\mathcal{ME}$ will randomly choose a bit θ' as its response and hand it to the challenger.

– If this is the target session (i.e. $U_1 = A$, $U_2 = B$, and $\iota = u$), then $\mathcal{ME}$ will compute the MAC digest using the respective MAC keys and output $(\gamma_A, [U_1, U_2, R_{U_1}, R_{U_2}, \gamma_A]_{K^{MAC}_A})$ and $(\gamma_B, [U_1, U_2, R_{U_1}, R_{U_2}, \gamma_B]_{K^{MAC}_B})$ to U_1 and U_2 respectively.
– If this is not the target session, $\mathcal{ME}$ will compute the MAC digest using the respective MAC keys and output $(\alpha_{U_1}, [U_1, U_2, R_{U_1}, R_{U_2}, \alpha_{U_1})]_{K^{MAC}_{U_1}})$ and $(\alpha_{U_2}, [U_1, U_2, R_{U_1}, R_{U_2}, \alpha_{U_2})]_{K^{MAC}_{U_2}})$ to U_1 and U_2 respectively, where $\alpha_{U_1} = \mathcal{O}_{k_A}(SK_{U_1,U_2,\iota})$ if $U_1 = A$, $\alpha_{U_1} = \{SK_{U_1,U_2,\iota}\}_{K^{enc}_{U_1}}$ if $U_1 \neq A$, $\alpha_{U_2} = \mathcal{O}_{k_B}(SK_{U_2,U_1,\iota})$ if $U_2 = B$, and $\alpha_{U_2} = \{SK_{U_2,U_1,\iota}\}_{K^{enc}_{U_2}}$ if $U_2 \neq B$.
– In all other cases the input to the SendServer query is invalid, so $\mathcal{ME}$ will randomly choose a bit θ' as its response and hand it to the challenger. Hence, SendServer queries can be correctly answered by $\mathcal{ME}$.

On receipt of $\mathsf{Reveal}(U_1, U_2, \iota))$ **queries:**

– If $\Pi^{\iota}_{U_1,U_2}$ has accepted and it forms the target session (i.e. $\{U_1, U_2\} = \{A, B\}$) and the last flow that $\Pi^{\iota}_{U_1,U_2}$ received had the form (α, β), where $\alpha \in \{\gamma_A, \gamma_B\}$), then $\mathcal{ME}$ will randomly choose a bit θ' as its response and hand it to the challenger.
– If this session has accepted but it is not the target session, then $\mathcal{ME}$ will output the session key $SK_{U_1,U_2,\iota}$, since all session keys are known from the $\mathsf{SendServer}(U_1, U_2, \iota, m)$ queries.

– In all other cases the input to the Reveal query is invalid, so $\mathcal{ME}$ will randomly choose a bit θ' as its response and hand it to the challenger. Hence, Reveal queries can be correctly answered by $\mathcal{ME}$.

On receipt of $\mathsf{Corrupt}(U, K)$ queries:

– If $U \in \{A, B\}$ (i.e. $\mathcal{A}$ is trying to corrupt the initiator or responder principal of the target session), then $\mathcal{ME}$ will randomly choose a bit θ' as its response and hand it to the challenger.

– If $U \in \{U_1, \ldots, U_{N_p}\} \setminus \{A, B\}$, then $\mathcal{ME}$ will hand all internal information for U to $\mathcal{A}$, update U as corrupted and also update K_U^{enc} to be K.

On receipt of $\mathsf{Test}(U_1, U_2, \iota)$ query:

– If this is the target session (i.e. $\{U_1, U_2\} = \{A, B\}$ and the last flow that $\Pi^{\iota}_{U_1,U_2}$ received had the form (α, β), where $\alpha \in \{\gamma_A, \gamma_B\}$), then $\mathcal{ME}$ will answer the query with m_0, else $\mathcal{ME}$ will randomly choose a bit θ' as its response and hand it to the challenger. After making a Test query and getting an answer of the form $SK_{U_1,U_2,\iota}$ from $\mathcal{ME}$, $\mathcal{A}$ continues interacting with the protocol and eventually outputs a guess bit b', where $b' \xleftarrow{R} \{0, 1\}$. $\mathcal{ME}$ then outputs $\theta' = b'$ as its response and hands it to the challenger.

The random choices of A, B, and u by $\mathcal{ME}$ mean that the probability that the target session is the same as the session that $\mathcal{A}$ chooses as the Test session is $\frac{1}{N_p^2 N_s}$. We claim that if the target session and Test session are the same, and the Test session is fresh, then $\mathcal{ME}$ will succeed whenever $\mathcal{A}$ succeeds and fail whenever $\mathcal{A}$ fails. To see that this is true, suppose $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ receive γ_A and γ_B and both accept. If $\mathcal{A}$ attempts to reveal either of these oracles, then neither will be considered fresh. This can be seen since the valid MAC digests mean that both oracles have the same SID, and the random generation of both components of the SID implies that no other (honest) party will have accepted with the same SID, and thus $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ are partners (since it is easy to verify that the other partnership requirements are also met). We also observe that no other oracle apart from $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ accepts after receiving γ_A and γ_B since only one server instance outputs a valid MAC digest for this pair of ciphertexts, and this server instance only outputs a valid MAC digest for one pair of nonces.

It is easy to verify that when the target session and Test session are different, or when the Test session is not fresh, $\mathcal{ME}$ outputs a random bit, so has probability of $\frac{1}{2}$ of succeeding. Hence, $\mathcal{ME}$'s success probability is given by

$$\begin{aligned}
\mathsf{Pr}[Succ^{\mathcal{ME}}(k)] &= \frac{N_p^2 N_s - 1}{2 N_p^2 N_s} + \frac{\mathsf{Pr}[b' = b | \overline{\mathsf{MACforgery}}]}{N_p^2 N_s} \\
&= \frac{N_p^2 N_s - 1}{2 N_p^2 N_s} + \frac{(\mathsf{Adv}^{\mathcal{A}}(k) | \overline{\mathsf{MACforgery}}) + 1}{2 N_p^2 N_s} \\
2 \cdot \mathsf{Pr}[Succ^{\mathcal{ME}}(k)] - 1 &= \frac{N_p^2 N_s - 1}{N_p^2 N_s} + \frac{(\mathsf{Adv}^{\mathcal{A}}(k) | \overline{\mathsf{MACforgery}}) + 1}{N_p^2 N_s} - 1
\end{aligned}$$

$$\begin{aligned}
\mathsf{Adv}^{\mathcal{ME}}(k) &= \frac{(\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}})}{N_p^2 N_s} \\
(\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}}) &= N_p^2 N_s \cdot \mathsf{Adv}^{\mathcal{ME}}(k) \quad ,
\end{aligned}$$

where $\mathsf{Pr}[b' = b|\overline{\mathsf{MACforgery}}]$ is the probability that $\mathcal{A}$ succeeds given that $\mathcal{A}$ does not make any MAC forgeries and $(\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}})$ is the advantage of $\mathcal{A}$ given that $\mathcal{A}$ does not make any MAC forgeries. Since we know that both N_p and N_s are polynomial in the security parameter k and $\mathsf{Adv}^{\mathcal{ME}}(k)$ is negligible by definition of the security of the encryption scheme, $(\mathsf{Adv}^{\mathcal{A}}(k)|\overline{\mathsf{MACforgery}})$ is also negligible.