A Proof of Revised Yahalom Protocol in the Bellare and Rogaway (1993) Model*

Kim-Kwang Raymond Choo**

Canberra, Australia
raymond.choo.au@gmail.com

Abstract. Although the Yahalom protocol, proposed by Burrows, Abadi, and Needham in 1990, is one of the most prominent key establishment protocols analyzed by researchers from the computer security community (using automated proof tools), a simplified version of the protocol is only recently proven secure by Backes and Pfitzmann (2006) in their *cryptographic library* framework. We present a protocol for key establishment that is closely based on the Yahalom protocol. We then present a security proof in the Bellare and Rogaway (1993) model and the random oracle model. An extension to our proposed protocol results in an unusual feature, that is session key can be renewed for subsequent communication without the server's involvement (i.e., re-authentication). We also observe that no partnering mechanism is specified within the Yahalom protocol. We then present a brief discussion on the role and the possible construct of session identifiers as a form of partnering mechanism, which allows the right session key to be identified in concurrent protocol executions. We then recommend that session identifiers should be included within protocol specification rather than consider session identifiers as artefacts in protocol proof.

1 Introduction

The establishment of session keys often involves interactive cryptographic protocols (or also known as authentication and/or key establishment protocols). Such protocols are the cornerstone of any secure communication and increasingly being considered as the *sine qua non* of many diverse secure electronic communications and electronic commerce applications.

It is generally regarded that the design of secure key establishment protocols is notoriously hard. The study of such protocols has resulted in

** The views and opinions expressed in this paper are those of the author and do not reflect those of any other organisation with which the author is currently affiliated. Research was performed while the author was with the Information Security Institute / Queensland University of Technology.

a dichotomy in cryptographic protocol analysis techniques between the computational complexity approach [1,9,16,38] and the computer security approach [34].

The emphasis of this paper is on the current computational complexity (provable security paradigm) approach to proofs for protocols. In this paradigm for protocols, a deductive reasoning process is adopted whereby emphasis is placed on a proven reduction from the problem of breaking the protocol to another problem believed to be hard. A complete mathematical proof with respect to cryptographic definitions provides a strong assurance that a protocol is behaving as desired. The history of mathematics is, however, full of erroneous proofs [14]. One such example is illustrated in the *virtuoso* work of Lakatos [31] where the many proofs and refutations for Euler's characteristic in algebraic topology are presented as a comedy of errors. Many formulations for Euler's characteristic in algebraic topology, a theorem about the properties of polyhedra, have been tried, only to be refuted and replaced by another formulation.

The difficulty of obtaining correct computational proofs of security is also dramatically illustrated by the well-known problem with the OAEP mode for public key encryption [39]. Although OAEP was one of the most widely used and implemented algorithms, it was several years after the publication of the original proof that a problem was found (and subsequently fixed in the case of RSA). Problems with proofs of protocol security have occurred too, evidenced by the breaking of several provably-secure protocols after they were published.

Despite these setbacks, proofs are invaluable for arguing about security and certainly are one very important tool in getting protocols right. Moreover, having security proofs allow protocol designer to formally state the desirable properties / goals that a protocol offers (giving assurance to protocol implementors).

Motivations of Paper.

1. Despite the popularity of the Yahalom protocol [15] – especially with researchers using formal methods for protocol verification [37] – the protocol does not possess a security proof within a computational complexity framework (e.g., within the widely accepted indistinguishability-based model). We note that in a recent work of Backes and Pfitzmann [3], a simplified version of this protocol is proven in the cryptographic library that corresponds to a slightly extended Dolev–Yao model [22]. We hope that by providing such a proof for a slightly modified Ya-

halom protocol, this will be of interest to the researchers, in particular to researchers from the computer security community.
2. We observe that session identifiers (SIDs) do not form part of the protocol specification for the Yahalom protocol (as in the case for many other key establishment protocols). 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. SIDs enables unique identification of the individual sessions.

Contributions of Paper.

1. We work in the widely accepted indistinguishability-based model of Bellare and Rogaway (hereafter referred to as the BR93 model) [9] and the random oracle model (also known as the ideal hash model) [10][1]. In this paper, we present a revised version of the Yahalom protocol and a formal statement of its security in the BR93 model and the random oracle model.
2. We highlight the importance of SIDs for practical key establishment protocols. We briefly discuss possible constructs of SIDS. We then recommend that SIDs should be included in protocol specification rather than be considered as artefacts in the protocol proof noting that not many protocols are proven secure.

Roadmap. Section 2 reviews the BR93 model and the necessary mathematical preliminaries. Section 3 revisits the Yahalom protocol and the simplified version proven secure by Backes and Pfitzmann [3]. In Section 4, a protocol closely based on the Yahalom protocol is then described, followed by a proof of its security. We then describe how our proposed protocol can be extended to allow session keys to be renewed in subsequent sessions without the server's further involvement. Section 5 presents a brief discussion on the role of SIDs in protocols and our recommendations. Section 6 concludes the paper with a comparitive summary of the proven secure protocol.

[1] Some might argue that a proof in the random oracle model is more of a heuristic proof than a real one. However, despite the criticism, this model is still widely accepted by the cryptographic community. We remark that recently, the first practical and provable-secure oblivious transfer password-based protocol whose proof of security relies on the random oracle model was published in ACM CCS 2005 [24]. Moreover, in many applications, a very efficient protocol with a heuristic security proof is preferred over a much less efficient one with a complete security proof [17].

2 Provable Security Paradigm for Protocols

Although the first treatment of computational complexity analysis for cryptography began in the 1980s [25], it was made popular for key establishment protocols by Bellare and Rogaway [9]. They provide the first formal definition for a model of adversary capabilities with an associated definition of security (which we refer to as the BR93 model in this paper) where they provide mathematical proofs for two-party entity authentication protocols. In the model, there exist a powerful adversary who can interact with all the participants, with an aim to learn some information about one session key. Therefore, one tries to prove the indistinguishability of the session key (from a random key) for the adversary.

2.1 The Adversarial Model

Informally the adversary, $\mathcal{A}$, is allowed to fully control the communication network by injecting, modifying, blocking, and deleting any messages at will. $\mathcal{A}$ can also request for any session keys adaptively. The adversary interacts with a set of *oracles*, each of which represents an instance of a principal in a specific protocol run. Each principal has an identifier U and oracle $\Pi^s_{U,\cdot}$ represents the actions of principal U in the protocol run indexed by integer s. Formally, $\mathcal{A}$ can adaptively query the following oracles, as follows:

$\mathsf{Send}(U_1, U_2, s, m)$ This query allows the adversary to make the principal, U_1, run the protocol normally (with some responder). The oracle $\Pi^s_{U_1,U_2}$ will return to the adversary the same next message that an honest principal U_1 would if sent message m according to the conversation so far. This includes the possibility that m not be of the expected format in which case $\Pi^s_{U_1,U_2}$ may simply halt. If $\Pi^s_{U_1,U_2}$ accepts the session key or halts this is included in the response. The adversary can also use this query to initiate a new protocol instance by sending an empty message m. For simplicity in the proof simulation, we separate the simulation of the Send queries into $\mathsf{SendClient}(U_1, U_2, s, M)$ and $\mathsf{SendServer}(U_1, U_2, s, M)$ queries where $\mathsf{SendClient}$ queries are directed at client oracles and $\mathsf{SendServer}$ queries are directed at server oracles.

$\mathsf{Reveal}(U, s)$ This query models the adversary's ability to find session keys. If a session key K_s has previously been accepted by $\Pi^s_{U,\cdot}$ then it is returned to the adversary. An oracle can only accept a key once (of course a principal can accept many keys modelled in different oracles). An oracle is called *opened* if it has been the object of a Reveal query.

Corrupt(U, K) This query models *insider attacks* and *unknown-key share attacks* by the adversary. The query returns the oracle's internal state. $\mathcal{A}$ can choose to replace the long-term secret key of the principal with a key of $\mathcal{A}$'s choice, K. A principal is called *corrupted* if it has been the object of a Corrupt query.

Test(U_1, U_2, s) Once the oracle has accepted a session key K_s the adversary can attempt to distinguish it from a random key as the basis of determining security of the protocol. A random bit b is chosen; if $b = 0$ then K_s is returned while if $b = 1$ a random string is returned from the same distribution as session keys. This query is only asked once by the adversary.

Note that in the original BR93 model, the Corrupt query is not allowed. However, we consider the BR93 model which allows the adversary access to a Corrupt query because later proofs of security in the BR93 model allow the Corrupt query. The omission of such a (Corrupt) query may also allow a protocol vulnerable to insider and unknown key share attacks to be proven secure in the model [18].

2.2 Definition of Security

Definition of security in the BR93 model depends on the notion of the *partner* oracles to any oracle being tested. The way of defining partner oracles has varied in different papers using the model. In more recent proofs (e.g., [30,33,32]), partners have been defined by having the same session identifier (SID) which consists of a concatenation of the messages exchanged between the two. We define $SID(\Pi^s_U)$ as the concatenation of all messages that oracle Π^s_U has sent and received. Let $PID(\Pi^s_U)$ denote the perceived partner of Π^s_U.

Definition 1. *Two oracles, $\Pi^i_{U_1}$ and $\Pi^j_{U_2}$, are* partnered *if:*

- *each believes that the other is its partner (i.e., $PID(\Pi^i_{U_1}) = U_2$ and $PID(\Pi^j_{U_2}) = U_1$),*
- *they agree on the same session identifier (i.e., $SID(\Pi^i_{U_1}) = SID(\Pi^j_{U_2})$).*

Definition 2. *An oracle, $\Pi^i_{U_1}$, is* fresh *at the end of its execution if:*

- *$\Pi^i_{U_1}$ and its partner $\Pi^j_{U_2}$ (if such a partner exists) have not been asked any* Reveal *queries, and*
- *both principals U_1 and U_2 have not been asked any* Corrupt *queries.*

The security of the protocol is defined by the following game played between the adversary and an infinite collection of client and server oracles. Note that a protocol participant is either a client or a server but not both. An overview of the game simulation is as follows:

Stage 0. The long-term secret keys are assigned to each client and server participants in the protocol by running the key distribution algorithm $\mathcal{G}_k$ on input of the security parameter, k.

Stage 1. The challenger now simulates the view of the adversary, $\mathcal{A}$, by answering all Send, Reveal and Corrupt queries of the adversary.

Stage 2. At some stage during the game simulation, a Test query is asked by the adversary to a fresh oracle.

Stage 3. The challenger continues simulating the view of the adversary, $\mathcal{A}$, by answering all Send, Reveal and Corrupt queries of the adversary. However, the adversary is not allowed to ask any Reveal or Corrupt queries that will trivially expose the Test key (i.e., renders the Test key unfresh in the sense of Definition 2).

Stage 4. Eventually the adversary outputs a bit b' and terminates. Success of the adversary $\mathcal{A}$ in this game is measured in terms of its *advantage* in distinguishing the session key of the Test query from a random key, i.e. its advantage in outputting $b' = b$. This advantage must be measured in terms of the security parameter k. If we define success to be the event that $\mathcal{A}$ guesses correctly whether $b = 0$ or $b = 1$ then

$$\mathsf{Adv}^{\mathcal{A}}(k) = |2 \cdot \Pr[\mathsf{success}] - 1|.$$

Definition 3. *A protocol is a secure key establishment protocol if both properties are satisfied:*

1. *If fresh oracles $\Pi^i_{U_1}$ and $\Pi^j_{U_2}$ are partners in the sense of Definition 1, then $\Pi^i_{U_1}$ and $\Pi^j_{U_2}$ conclude with the same session key except for a negligible probability.*
2. *For every probabilistic, polynomial-time adversaries, $\mathcal{A}$, the function $\mathsf{Adv}^{\mathcal{A}}(k)$ is negligible.*

3 The Yahalom Protocol and its Simplified Version

We now revisit the Yahalom protocol [15] described in Protocol 1. At the end of Protocol 1's execution, both users A and B will accept the session key (SK_{AB}) generated by the trusted server, S. Other notation in Protocol 1 is as follows: $\mathcal{E}(m)_K$ denotes an encryption of some message

m under symmetric key K; S denotes a server who shares long-term symmetric keys K_{AS} and K_{BS} with A and B respectively; N_A and N_B denote nonces generated by A and B respectively.

1. $A \rightarrow B : N_A$
2. $B \rightarrow S : B, \mathcal{E}(A, N_A, N_B)_{K_{BS}^{enc}}$
3. $S \rightarrow A : \mathcal{E}(B, SK_{AB}, N_A, N_B)_{K_{AS}^{enc}}, \mathcal{E}(A, SK_{AB})_{K_{BS}^{enc}}$
4. $A \rightarrow B : \mathcal{E}(A, SK_{AB})_{K_{BS}^{enc}}, \mathcal{E}(N_B)_{SK_{AB}}$

Protocol 1: The Yahalom protocol

Protocol 1 provides key confirmation – B is assured that A actually has possession of the same secret session key, SK_{AB}, since A sends to B the encryption of the nonce chosen by B, N_B, using SK_{AB}.

Choo and Hitchcock [20] pointed out informally that it does not appear possible to prove Protocol 1 secure in the BR93 model due to the encryption of the nonce using the established session key (i.e., $\mathcal{E}(N_B)_{SK_{AB}}$) in the last message (from A to B). In an independent yet related work, Backes and Pfitzmann [3] raise similar observation. In the simplified version proposed by Backes and Pfitzmann [3], the encryption of the nonce using the established session key (i.e., $\mathcal{E}(N_B)_{SK_{AB}}$) in message 4 is removed from the protocol.

4 A New Provably-Secure Protocol

Following the approach of Boyd, Choo, and Mathuria [13], we will define the authenticated encryption scheme in the security proof for our proposed protocol prior to defining our proposed protocol.

4.1 Secure Authenticated Encryption Schemes

Let k denote the security parameter. A *symmetric encryption scheme* $\mathcal{SE} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ consists of three algorithms, namely: the *key generation* algorithm $\mathcal{K}$, the *encryption* algorithm $\mathcal{E}$, and the *decryption* algorithm $\mathcal{D}$ as described below.

- $\mathcal{K}$ is a probabilistic algorithm which, on input 1^k, outputs a key K.
- $\mathcal{E}$ is a probabilistic algorithm which takes a key K and a message M drawn from a message space $\mathcal{M}$ associated to K and returns a ciphertext C. This is denoted by $C \xleftarrow{R} \mathcal{E}_K(M)$.

– $\mathcal{D}$ is a deterministic algorithm which takes a key K and a ciphertext C and returns the corresponding plaintext M or the symbol $\perp$ which indicates an illegal ciphertext. This is denoted as $x \leftarrow \mathcal{D}_K(C)$. We require that $\mathcal{D}_K(\mathcal{E}_K(M)) = M$ for every $K \leftarrow \mathcal{K}(1^k)$.

For security we use the definitions of Bellare and Namprempre [7]. We require that the symmetric encryption scheme provides confidentiality in the sense of indistinguishability under chosen plaintext attacks (*IND-CPA security*) and provides integrity in the sense of preserving integrity of plaintexts (*INT-PTXT security*). We note that each of these is the weakest of the properties defined by Bellare and Namprempre and are provided by either encrypt-then-MAC or by MAC-then-encrypt constructions. Therefore there are many practical ways of implementing our protocol which can reasonably be expected to satisfy these assumptions. We now define these concepts more precisely.

For any efficient (probabilistic polynomial time) adversary $\mathcal{X}$, the confidentiality security is defined in terms of the following game, which we call $\mathcal{G}_1$.

1. The challenger chooses a key $K \leftarrow \mathcal{K}(1^k)$.
2. Given access to the encryption oracle, the adversary outputs two messages of equal length $M_0, M_1 \in \mathcal{M}$ of her choice.
3. The challenger computes $C_b \stackrel{R}{\leftarrow} \mathcal{E}_K(M_b)$ where $b \stackrel{R}{\leftarrow} \{0,1\}$. The bit b is kept secret from the adversary.
4. The adversary is then given C_b and has to output a guess b' for b.

We define the advantage of the adversary $\mathcal{X}$ playing the above game as

$$\mathsf{Adv}_{\mathcal{X}}^{\mathrm{ind-cpa}}(k) = |2 \cdot \Pr[b' = b] - 1|.$$

Definition 4. *The encryption scheme $\mathcal{SE}$ is IND-CPA secure if the advantage of all efficient adversaries playing game $\mathcal{G}_1$ is negligible.*

For any efficient adversary $\mathcal{F}$, the integrity security is defined in terms of the following game, which we call $\mathcal{G}_2$.

1. Choose a key $K \leftarrow \mathcal{K}(1^k)$.
2. The adversary $\mathcal{F}$ is given access to the encryption oracle and also a *verification oracle* which on input a ciphertext C outputs 0 if $\mathcal{D}_K(C) = \perp$ and outputs 1 if C is a legitimate ciphertext.
3. The adversary wins if it can find a legitimate ciphertext C^* such that the plaintext $M = \mathcal{D}_K(C^*)$ was never used as a query to the encryption oracle. In this case we say the event forgery has occurred.

We define the advantage of the adversary playing the above game as $\mathsf{Adv}_{\mathcal{F}}^{\text{int}-\text{ptxt}}(k) = 2 \cdot \Pr[\mathsf{forgery}]$.

Definition 5. *The encryption scheme $\mathcal{SE}$ is INT-PTXT secure if the advantage of all efficient adversaries playing game $\mathcal{G}_2$ is negligible.*

4.2 Our Proposed Protocol

Now that the authenticated encryption scheme to be employed in the protocol has been defined, we can define the protocol that we shall prove secure. New notation introduced here in Protocol 2 are:

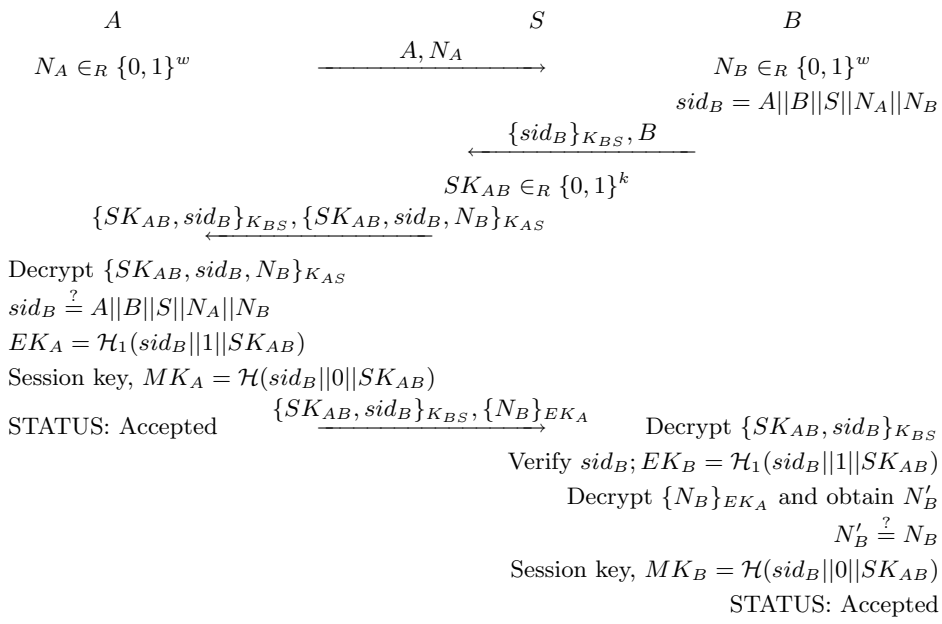
- $\mathcal{H}$ and $\mathcal{H}_1$ denote two secure and independent cryptographic hash functions;
- $\{m\}_K$ denotes an authenticated encryption of some message m under symmetric key K;
- $||$ denotes concatenation of messages;
- sid denotes the session identifier[2];
- $N_U \in_R \{0,1\}^w$ denotes a random w-bit nonce; and
- $SK_{AB} \in_R \{0,1\}^k$ denotes the random k-bit key generated by the server, S, for some session.

Protocol 2 is very similar to Protocol 1 and differences include (but not limited to) the following:

1. In Protocol 1, the session key (SK_{AB}) is contributed by the server, S, whilst for Protocol 2, users A and B as well as the server S contribute to the key value ($MK_{AB} = \mathcal{H}(sid||0||SK_{AB})$).
2. In Protocol 1's specification, there is no partnering mechanism (e.g., sid) specified. Without such partnering mechanism, communicating parties will be unable to uniquely distinguish messages from different sessions. This is further discussed in Section 5.
3. Due to the use of an authenticated encryption scheme in Protocol 2, the computational overhead is slightly more expensive than that of Protocol 1.

Informally, the inclusion of the

[2] Note that sid is made public upon protocol completion, and the security of the protocol does not hinge on the difficulty of predicting a valid sid. In other words, anyone (including the adversary, $\mathcal{A}$) knows what a particular sid is.



Protocol 2: A revised Yahalom protocol

- Identities of the participants[3] and role asymmetry within the session key construction effectively ensures some sense of direction. If the role of the participants or the identities of the (perceived) partner change, the session keys will also be different. Hence, this provides resilience against unknown key share and reflection attacks.
- Unique session identifier (*sid*) within the session key construction ensures that session keys will be fresh. Moreover, it appears that the publication of *sid* upon protocol completion results in $\mathcal{A}$ being unable to get B to accept nonce pair (which is part of the published *sid*) as the session key. Recall a different *sid* also mean a different session key. Hence, it appears that the type flaw attack revealed on Protocol 1 by Basin, Mödersheim, and Viganò [4] is thwarted.

4.3 Proof for Protocol 2

Theorem 1 *Protocol 2 is a secure key establishment protocol in the sense of Definition 3 if the underlying authenticated encryption scheme is INT-*

[3] Such an approach is also recommended by National Institute of Standards and Technology (NIST) [35]

PTXT secure as described in Definition 5 and both $\mathcal{H}$ and $\mathcal{H}_1$ are modelled as independent random oracles.

The proof follows that of Bellare and Rogaway [11] and that of Boyd, Choo, and Mathuria [13] quite closely; differences include the use of a combined authenticated encryption scheme (as opposed to separate encryption and MAC functions), the different partnering function used, and the deployment of the random oracle (note that we model $\mathcal{H}$ and $\mathcal{H}_1$ as random oracles).

The general idea of the security proof is to assume that the protocol adversary can gain an advantage and use this to break the assumptions about the security of the encryption algorithm. Since the adversary relies on its oracles to run we simulate the oracles so that we can supply the answers to all the queries the adversary might ask. We cannot supply answers which rely on knowledge of the encryption keys that we are trying to break, so we use the integrity of plaintexts to show that these queries would, almost certainly, not be answered by any oracle running the protocol. As long as the simulation works with non-negligible probability the assumption about the encryption scheme fails.

Following Bellare and Rogaway [11] we need to extend the definition of a secure encryption scheme to allow the adversary to obtain multiple encryptions of the same plaintext under many different independent encryption keys. Such an adversary is termed a *multiple eavesdropper*. A multiple eavesdropper, $\mathcal{ME}$, is allowed to obtain encryptions of the same plaintext under two different independent encryption keys. We can bound the advantage of a multiple eavesdropper by considering it as a special case of the multi-user setting analysed by Bellare, Boldyreva and Micali [6]. In their notation we have the case of $q_e = 1$, meaning that the adversary can only obtain one encryption for each public key.

Lemma 1 *Suppose that an adversary has advantage at most $\epsilon(k)$ for encryption scheme $(\mathcal{E}, \mathcal{D})$. Then a multiple eavesdropper has advantage not more than $n \cdot \epsilon(k)$.*

Notice that since an authenticated encryption scheme is also a secure encryption scheme in the sense defined by this result, it also holds for an authenticated encryption scheme. This allows us to define a variant of game $\mathcal{G}_1$ which we call $\mathcal{G}_1'$. The only difference between these is that in $\mathcal{G}_1'$ the adversary is given access to two encryption oracles for two independently generated keys, and its challenge consists of two encryptions of either m_0 or m_1 under the two keys.

4.3.1 Integrity attacker We now construct a forger $\mathcal{F}$ against the security of the authenticated encryption scheme, $\mathcal{SE}$, described in Definition 4, using an adversary against Protocol 2, $\mathcal{A}$. We will say that the event $\mathsf{success}_{\mathcal{F}}$ occurs if $\mathcal{F}$ wins game $\mathcal{G}_2$ against $\mathcal{SE}$.

Lemma 2 *There is an efficient algorithm $\mathcal{F}$ defined using $\mathcal{A}$ such that if* forge *occurs with non-negligible probability then* $\mathsf{success}_{\mathcal{F}}$ *occurs with non-negligible probability .*

In order to prove Lemma 2 we describe how $\mathcal{F}$ is constructed. When $\mathcal{F}$ runs it receives access to the encryption and verification oracles of the authenticated encryption scheme $\mathcal{SE}$. Its output must be a forged ciphertext for a message m which was not previously input to the encryption oracle.

In order to obtain the forgery $\mathcal{F}$ runs $\mathcal{A}$ by first choosing a user U_i for $i \in_R [1, Q]$. This user will be simulated as though its long-term key is the one used in $\mathcal{SE}$. For all other $j \in [1, Q]$ with $j \neq i$, $\mathcal{F}$ generates the long-term shared key using the key generation algorithm $\mathcal{K}_k$. This allows $\mathcal{F}$ to answer all the oracle queries from $\mathcal{A}$ as follows.

$\mathsf{Send}(U_1, s, M)$ For any well-formed queries to S, $\mathcal{F}$ can reply with valid ciphertexts, by choosing the session key and forming the ciphertexts, either directly using the known key or using the encryption oracle in the case of U_i. For queries to initiate a protocol run, $\mathcal{F}$ can generate a random nonce and answer appropriately. Finally, consider a query to either an initiator or responder oracle including a claimed server message (corresponding to protocol messages 3 or 4). The relevant ciphertext can be verified either directly using the known key or using the verification oracle. If the ciphertext is verified correctly then the oracle accepts and this information is returned to $\mathcal{A}$.

$\mathsf{Reveal}(U, s)$ Since all session keys are known from running the $\mathsf{Send}(U, s, M)$ queries the query can be trivially answered with the correct session key (if accepted).

$\mathsf{Corrupt}(U)$ As long as $U \neq U_i$ all the private information is available and the query can be answered. In the case $U = U_i$ then the query cannot be answered and $\mathcal{F}$ will abort and fail.

$\mathsf{Test}(U, s)$ Since all the accepted session keys are known from running the Send queries the query can be trivially answered by identifying the correct session key.

$\mathcal{F}$ continues the simulation until a forgery event against $\mathcal{SE}$ occurs, or until $\mathcal{A}$ halts. Note that as long as $\mathcal{F}$ does not abort then the simulation

is perfect. If forge occurs then the probability that the user involved is U_i equals $1/Q$. In this case the event $\mathsf{success}_{\mathcal{F}}$ occurs. Futhermore, in this case $\mathcal{F}$ does not abort since U_i cannot be corrupted before the forge event. Therefore we arrive at the following upper bound.

$$\Pr(\mathsf{forge}) \leq Q \cdot \Pr(\mathsf{success}_{\mathcal{F}}) \tag{1}$$

4.3.2 Confidentiality attacker For the second part of the proof, we assume that $\mathcal{A}$ gains an advantage without producing a forgery. We construct an attacker with a non-negligible advantage against the encryption scheme, $\mathcal{X}$, using the adversary, $\mathcal{A}$.

Lemma 3 *There is an efficient algorithm $\mathcal{X}$ defined using $\mathcal{A}$ such that if* success *occurs but* forge *does not occur, then $\mathcal{X}$ wins game $\mathcal{G}_1'$.*

Two random keys K and K' are chosen by the challenger for $\mathcal{SE}$ and $\mathcal{X}$ is given access to the encryption oracles for these keys. First $\mathcal{X}$ chooses two users U_i and U_j for $i, j \in_R [1, Q]$. For all other $k \in [1, Q]$, $\mathcal{X}$ generates the long-term key using the key generation algorithm $\mathcal{K}_k$. Next $\mathcal{A}$ chooses two random session keys K_0 and K_1. Suppose that Q_S is the maximum number of Send queries that $\mathcal{A}$ will ask of the server and Q_H is the maximum number of hash queries that $\mathcal{A}$ will ask of the server. $\mathcal{X}$ chooses a value s_0 randomly in $[1, Q_S]$. The idea is that $\mathcal{X}$ will inject the ciphertexts C_b, C_b' into a random SendServer query. $\mathcal{X}$ proceeds to simulate responses for $\mathcal{A}$ as follows. Let U_I and U_R denote the initiator and the responder respectively.

Note that we also require two separate lists of tuples, $L_{\mathcal{H}}$ and $L_{\mathcal{H}_1}$ to be maintained. If we are asked queries of the form $\mathcal{H}(SID_i^k||0||SK)$ and $\mathcal{H}_1(SID_i^k||1||SK)$, we check to see if the queries have been previously asked. If so, then the previous answer stored in the respective list will be returned (to maintain consistency). Otherwise, return a random value, $v \in_R \{0,1\}^k$. In addition, store this answer together with the query in the respective list.

SendClient**:** In the case of $U_1 = U_I$, $U_2 = U_R$, and $m = *$, then this will start a protocol run. This query can be successfully answered by $\mathcal{X}$ and the outgoing message is some randomly chosen k-bit challenge N_{U_1}.

SendClient**:** In the case of $U_1 = U_R$, $U_2 = U_I$, and m is some k-bit challenge, then $\mathcal{X}$ will choose a unique k-bit challenge, N_{U_2}; computes the session identitifer, $sid = U_1||U_2||S||m||N_{U_2}$ and the respective ciphertext; and successfully answer this query.

SendServer: In the case of $U_1 = \{U_I, U_R\}$, $U_2 = S$, and m is of the right format (as per message 2 in protocol specification), then S will run the session key generator and output a session key not previously output and generates the respective ciphertexts as the protocol specification demands.

SendClient: In the case of $U_1 = U_I$, $U_2 = U_R$, and m is of the right format (as per message 3 in protocol specification). Since we assume that $\mathcal{A}$ is not able to produce any MAC forgeries, all session keys (if accepted) are known from the $\mathsf{SendServer}(U_1, U_2, \iota, m)$ queries. Hence, if the received ciphertext (MAC digest) verifies correctly, the message must have been generated by $\mathcal{X}$ during a SendServer query and in this case, $\mathcal{X}$ will output the decision $\delta = \mathit{accept}$. Otherwise, $\mathcal{X}$ will output the decision $\delta = \mathit{reject}$, as the protocol specification demands.

SendClient: If $U_1 = U_I$, $U_2 = U_R$, and m is of the right format (as per message 4 in protocol specification). Again under the assumption that $\mathcal{A}$ is not able to produce any MAC forgeries, all session keys (if accepted) are known from the $\mathsf{SendServer}(U_1, U_2, \iota, m)$ queries. Since we also know the keying materials for both the session key and the one-time encryption/MAC key EK (used to encrypt the nonce of U_R) are the same and if received ciphertext (MAC digest) verifies correctly, the message must have been generated by $\mathcal{X}$ during a SendServer query. Therefore, $\mathcal{X}$ will output the decision $\delta = \mathit{accept}$. Otherwise, $\mathcal{X}$ will output the decision $\delta = \mathit{reject}$, as the protocol specification demands.

In all other cases the input to the SendClient or SendServer is invalid, $\mathcal{X}$ will terminate and halt the simulation. Hence, SendClient and SendServer queries can be correctly answered by $\mathcal{X}$.

This completes the description of $\mathcal{X}$. Since all the accepted session keys are known from running the SendClient and SendServer queries, the Test query can be trivially answered by identifying the correct session key.

Let lucky be the event that $\mathcal{X}$ does not fail during the Test query. When lucky occurs, $\mathcal{X}$ wins game $\mathcal{G}_1'$ whenever $\mathcal{A}$ is successful. This means that $\Pr(\mathsf{success}_{\mathcal{X}}|\mathsf{lucky}) \geq \Pr(\mathsf{success}_{\mathcal{A}}|\overline{\mathsf{forge}})$. We also have $\Pr(\mathsf{lucky}) \geq 1/(Q^2 \cdot Q_S)$. Putting these together we obtain:

$$\Pr(\mathsf{success}_{\mathcal{A}}|\overline{\mathsf{forge}}) \leq Q^2 \cdot Q_S \cdot \Pr(\mathsf{success}_{\mathcal{X}}). \tag{2}$$

4.3.3 Conclusion of Proof for Theorem 1 Since N, Q_S, and Q_H are polynomial in the security parameter k and ϵ is negligible by definition. Therefore, by combining equations 1 and 2 completes the proof for Theorem 1.

4.4 An Extension to Protocol 2

In addition to the basic Protocol 2, there is an extension which allows the session key to be renewed in subsequent sessions without the server's further involvement (i.e., re-authentication). This entails A and B exchanging new nonces N'_A and N'_B and computing the new session key as $MK'_A = \mathcal{H}(sid_{A'}||SK_{AB}) = \mathcal{H}(sid_{B'}||SK_{AB}) = MK'_B$ where $sid_{A'} = sid_{B'} = (A||B||S||N'_A||N'_B)$ as described by Protocol 3.

$$
\begin{array}{ccc}
A & & B \\
N'_A \in_R \{0,1\}^w & \xrightarrow{A, N'_A} \quad \xleftarrow{B, N'_B} & N'_B \in_R \{0,1\}^w \\
\multicolumn{3}{c}{sid_{A'} = (A||B||S||N'_A||N'_B) = sid_{B'}} \\
\multicolumn{3}{c}{MK'_A = \mathcal{H}(sid_{A'}||SK_{AB}) = \mathcal{H}(sid_{B'}||SK_{AB}) = MK'_B}
\end{array}
$$

Protocol 3: An extension to Protocol 2 (i.e., re-authentication)

Protocol 3 can also be enhanced with key confirmation, which consists of a handshake using the shared secret.

Remark. We are unable to prove Protocol 3 secure in the current model we are using. To prove Protocol 3 secure, we would have to modify the definitions of freshness (described in Definition 2) and partnership (described in Definition 1). This is to restrict the adversary from exposing session key agreed by both A and B in their previous session (i.e., SK_{AB}) without rendering the session key unfresh.

5 Partnering Mechanism: A Brief Discussion

In Protocol 1, partnering mechanism does not form part of its specification. Message exchanges in the real world are seldom conducted over secure channels. Therefore, it is realistic to assume that any adversary is able to modify messages at will, which is the case in the Bellare–Rogaway style models. As Goldreich and Lindell [26, Section 1.3] have pointed out, such an adversary capability means that the adversary is able to conduct concurrent executions of the protocol (one with each party).

For protocols proven secure in the Bellare–Rogaway style models or the Canetti–Krawczyk model [16], session identifiers as partnering mechanism are not explicitly part of the protocol specification but rather embedded within the partnership definition (e.g., it is stated that the correctness of session identifiers can be omitted from the formal protocol

specification [28]). We also observe that in the Canetti–Krawczyk model, the values of the session identifiers are not specified. Instead, it is assumed that session identifiers are known by protocol participants before the protocol begins. Such an assumption might not be practical as it requires some forms of communication between the protocol participation prior to the start of the protocol. Furthermore, by assuming that session identifiers are known by protocol participants before the protocol begins indicates that session identifiers do not form part of the protocol specification.

We advocate that session identifiers play a significant role in protocol security as they bind together incoming and outgoing messages, and uniquely identify a particular session. In other words, attacks against protocol is also predicated on the constructions of SIDs chosen as shown by Bohli, González Vasco, and Steinwandt [12] and Choo and Hitchcock [20].

In practice, it seems more intuitive to include session identifiers within the protocol specification since implementation of such protocols (e.g., SSL and IPSec) should allow applications to distinguish between the various concurrent sessions between one or many other applications. In other words, protocol on its own (without the session identifiers component) does not allow concurrent executions since oracles have no means of uniquely identifying one session from another. Moreover, not all protocols are proven secure in the Bellare–Rogaway style models and the Canetti–Krawczyk model or carry any security proofs.

How to Construct SIDs? In practice, session identifiers may be determined during protocol execution [16,21,29], as in the case of the Bellare, Pointcheval, and Rogaway model [8] and recent work of Krawczyk [33] whereby session identifiers are defined to be the concatentation of all incoming and outgoing messages. However, this might not be achievable in some protocols where the protocol participants do not have full view of the messages exchanged (e.g., the inability to define session identifiers in the Bellare–Rogaway 3PKD protocol [11] pointed out by Choo *et al.* [19]). As a bare minimum, session identifiers constructed in this context, should contain some unique contributions from each participant (e.g., random nonces, timestamps) and the identities of the peers (which is the case for Protocol 2).

Recommendations. Therefore, we suggest consider the construction of session identifiers or some forms of partnering mechanism within the protocol specification. Otherwise, this will result in the inability of communicating principals to uniquely distinguish messages from different sessions. Consequently, this leads one to question the practicality and usefulness of

the protocol in a real world setting. Moreover, including session identifiers in the key derivation function ensures that entities who have completed matching sessions, partners, will accept the same session key.

Word of Caution. We do not claim that including session identifiers or some forms of partnering mechanism within protocol specifications is the panacea to the design of secure protocols. The security of the protocol is based on many other factors, such as the underlying cryptographic primitives used. However, 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.

6 Summary

Table 1 presents a comparative summary of our proven secure protocol, Protocol 2, with two other similar server-based three-party key establishment protocols, namely the Bauer–Berson–Feiertag protocol [5] and the Otway-Rees protocol [36].
In conclusion, we proved the security of another protocol example (revised Yahalom protocol [15]) in the BR93 model. In terms of both messages and rounds, we observe from Table 1 that all three protocols satisfy the lower bound of four messages obtained by Gong [27] for server-based protocols with similar goals using timestamps. However, an extension to Protocol 2 allows session key to be renewed in subsequent sessions without invoking the server (as described in Protocol 3), which makes it more attractive than the other two protocols (in a realistic setting). As noted,we are unable to prove Protocol 3 secure in the current model we are using. To prove Protocol 3 secure, we would have to modify the definitions of freshness (described in Definition 2) and partnership (described in Definition 1). This is to restrict the adversary from exposing session key agreed by both A and B in their previous session (i.e., SK_{AB}) without rendering the session key unfresh.

We then briefly discussed the role of session identifiers as a form of partnering mechanism and concluded with the recommendation that session identifiers should be included within protocol specification. This will allow concurrent executions and a mean of uniquely identifying one session from another. Furthermore, by including session identifiers in the key derivation function, ensures that entities who have completed matching sessions, partners, will accept the same session key.

Protocols	Computational	Security proof?
Protocol 2	Slightly more expensive due to use of authenticated encryption scheme	BR93 model
Extensions to Protocol 2 allows session key to be renewed in subsequent sessions without invoking the server (see Protocol 3). Moreover, Protocol 2 ensures that entities who have completed matching sessions, partners, will accept the same session key (recall that *sid* is included in the key derivation function) without requiring the server to store every message processed and not issue different session keys for the same input message received.		
Otway-Rees [36]	cheap	Dolev–Yao style model [2]
In the approach taken by Backes [2], the server is required to store every message processed and not issue different session keys for the same input message received. Without this assumption, a malicious adversary is able to make the initiator and the responder agree on a different session key by asking a trusted third party (i.e., server) to create multiple session keys in response to the same message, as revealed by Fabrega, Herzog, and Guttman [23]. However, it has been pointed out that this assumption only works well within a confined implementation and will not scale well to a more realistic environment with a large number of participating parties and a substantial level of traffic to any one server [20].		
Bauer–Berson–Feiertag [5]	cheap	No

Table 1. A comparative summary

As a result of this work, we recommended that session identifiers should be included within protocol specification rather than considering session identifiers as artefacts in protocol proof, even for protocols proven secure in the computational complexity framework.

Acknowledgments

The author would like to thank the anonymous reviewers for their constructive feedback, particularly for pointing out that Protocol 3 cannot be proven secure in the existing (BR93) model. The author would also like to thank Professor Colin Boyd and Dr. Yvonne Hitchcock for their invaluable discussions on related topics when the author was with the Information Security Institute at Queensland University of Technology. Despite their invaluable assistance, any errors remaining in this paper are solely attributed to the author.

References

1. Abdalla, M. and Fouque, P.-A. and Pointcheval, D. (2005). Password-Based Authenticated Key Exchange in the Three-Party Setting. Proceedings of PKC 2005,

Les Diablerets, Switzerland, 23-26 January, vol 3386 of LNCS, pp. 65–84. Springer-Verlag, Berlin.

2. Backes, M. (2004). A Cryptographically Sound Dolev-Yao Style Security Proof of the Otway-Rees Protocol. Proceedings of ESORICS 2004, Sophia Antipolis, France, 13-15 September, vol 3193 of LNCS, pp. 89–108. Springer-Verlag, Berlin.
3. Backes, M. and Pfitzmann, B. (2006).On the Cryptographic Key Secrecy of the Strengthened Yahalom Protocol. Proceedings of IFIP SEC 2006, Karlstad, Sweden, 22-24 May, vol 2808 of IFIP Series, pp. 233–245. Springer-Verlag, Berlin.
4. Basin, D. and Mödersheim, S. and Viganó, L. (2003). An On-the-Fly Model-Checker for Security Protocol Analysis. Proceedings of ESORICS 2003, Gjovik, Norway, 13-15 October, vol 2808 of LNCS, pp. 253–270. Springer-Verlag, Berlin.
5. Bauer, R. and Berson, T. and Feiertag, R. (1983) A Key Distribution Protocol Using Event Markers. ACM Transactions on Computer Systems, 1(3), 249–255.
6. Bellare, M. and Boldyreva, A. and Micali, S. (2000). Public-key Encryption in a Multi-User Setting: Security Proofs and Improvements. Proceedings of EUROCRYPT 2000, Bruges, Belgium, 14-18 May, vol 1807 of LNCS, pp. 259–274. Springer-Verlag, Berlin.
7. Bellare, M. and Namprempre, C. (2000). Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm. Proceedings of ASIACRYPT 2000, Kyoto, Japan, 3-7 December, vol 1976 of LNCS, pp. 531–545. Springer-Verlag, Berlin.
8. Bellare, M. and Pointcheval, D. and Rogaway, P. (2000). Authenticated Key Exchange Secure Against Dictionary Attacks. Proceedings of EUROCRYPT 2000, Bruges, Belgium, 14-18 May, vol 1807 of LNCS, pp. 139–155. Springer-Verlag, Berlin.
9. Bellare, M. and Rogaway, P. (1993a). Entity Authentication and Key Distribution. Proceedings of CRYPTO 1993, Santa Barbara, California, 22-26 August, vol 773 of LNCS, pp. 110–125. Springer-Verlag, Berlin.
10. Bellare, M. and Rogaway, P. (1993b). Random Oracles Are Practical: A Paradigm For Designing Efficient Protocols. Proceedings of ACM CCS 1993, Fairfax, Virginia, 3-5 November, pp. 62–73. ACM, New York.
11. Bellare, M. and Rogaway, P. (1995). Provably Secure Session Key Distribution: The Three Party Case. Proceedings of ACM STOC 1995, Las Vegas, Nevada, 29 May-1 June , pp. 57–66. ACM, New York.
12. Bohli, J.-M. and González Vasco, M. I. and Steinwandt, R. (2005). Secure Group Key Establishment Revisited. Cryptology ePrint Archive, Report 2005/395. `http://eprint.iacr.org/2005/395`.
13. Boyd, C. and Choo, K.-K. R. and Mathuria, A. (2006). An Extension to Bellare and Rogaway (1993) Model: Resetting Compromised Long-Term Keys. Proceedings of ACISP 2006, Melbourne, Australia, 3-5 July, vol 4058 of LNCS, pp. 371–382. Springer-Verlag, Berlin.
14. Bundy, A., Jamnik, M., and Fugard, A. (2005). What is a Proof?. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 363(1835), 2377–2391.
15. Burrows, M. and Abadi, M. and Needham, R. (1989). A Logic of Authentication. ACM Transactions on Computer Systems, 8(1), 18–36.
16. Canetti, R. and Krawczyk, H. (2001). Analysis of Key-Exchange Protocols and Their Use for Building Secure Channel. Proceedings of EUROCRYPT 2001, Innsbruck, Austria, 6-10 May, vol 2045 of LNCS, pp. 453–474. Springer-Verlag, Berlin.

17. Catalano, D. and Pointcheval, D. and Pornin, T. (2006). Trapdoor Hard-to-Invert Group Isomorphisms and Their Application to Password-based Authentication. Journal of Cryptology, To Appear.
18. Choo, K.-K. R. and Boyd, C. and Hitchcock, Y. (2005). Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. Proceedings of ASIACRYPT 2005, Chennai, India, 4-8 December, vol 3788 of LNCS, pp. 585–604. Springer-Verlag, Berlin.
19. Choo, K.-K. R. and Boyd, C. and Hitchcock, Y. and Maitland, G. (2004). On Session Identifiers in Provably Secure Protocols: The Bellare-Rogaway Three-Party Key Distribution Protocol Revisited. Proceedings of SCN 2004, Amalfi, Italy, 8-10 September, vol 3352 of LNCS, pp. 352–367. Springer-Verlag, Berlin.
20. Choo, K.-K. R. and Hitchcock, Y. (2005). Security Requirements for Key Establishment Proof Models: Revisiting Bellare–Rogaway and Jeong–Katz–Lee Protocols. Proceedings of ACISP 2005, Brisbane, Australia, 4-6 July, vol 3574 of LNCS, pp. 429–442. Springer-Verlag, Berlin.
21. Cliff, Y. and Tin, Y.-S. T. and Boyd, C. (2006). Password Based Server Aided Key Exchange. Proceedings of ACNS 2006, Singapore, 6-9 June, vol 3989 of LNCS, pp. 146–161. Springer-Verlag, Berlin.
22. Dolev, D. and Yao, A. (1983). On the Security of Public Key Protocols. IEEE Transaction of Information Technology, 29(2), 198–208.
23. Fabrega, J. T. and Herzog, J. C. and Guttman, J. D. (1999). Strand Spaces: Proving Security Protocols Correct. Journal of Computer Security, 7(2/3), 191–230.
24. Gentry, C. and MacKenzie, P. and Ramzan, Z. (2005). Password Authenticated Key Exchange Using Hidden Smooth Subgroups. Proceedings of ACM CCS 2005, Alexandria, VA, 7-11 November , pp. 299–309. ACM, New York.
25. Goldwasser, S. and Micali, S. (1984). Probabilisitic Encryption. Journal of Computer and System Sciences, 28(3), 270–299.
26. Goldreich, O. and Lindell, Y. (2001). OSession-Key Generation using Human Passwords Only. Proceedings of CRYPTO 2001, Santa Barbara, California, 19-23 August, vol 2139 of LNCS, pp. 408–432. Springer-Verlag, Berlin.
27. Gong, L. (1993). Lower Bounds on Messages and Rounds for Network Authentication Protocols. Proceedings of ACM CCS 1993, Fairfax, Virginia, 3-5 November , pp. 26–37. ACM, New York.
28. Hitchcock, Y. and Boyd, C. and González Nieto, J. M. (2004). Tripartite Key Exchange in the Canetti-Krawczyk Proof Model. Proceedings of INDOCRYPT 2004, Chennai, India, 20-22 December, vol 3348 of LNCS, pp. 17–32. Springer-Verlag, Berlin.
29. Hitchcock, Y. and Boyd, C. and González Nieto, J. M. (2006). Modular Proofs for Key Exchange: Rigorous Optimizations in the Canetti-Krawczyk Model. Applicable Algebra in Engineering, Communication and Computing Journal, 16(6), 405–438.
30. Jeong, I. R. and Katz, J. and Lee, D. H. (2004). One-Round Protocols for Two-Party Authenticated Key Exchange. Proceedings of ACNS 2004, Yellow Mountain, China, 8-11 June, vol 3089 of LNCS, pp. 220–232. Springer-Verlag, Berlin.
31. Lakatos, I. (1976) Proofs and Refutations : The Logic of Mathematical Discover. Cambridge University Press, Cambridge.
32. Kudla, C. and Paterson, K. G. (2005). Modular Security Proofs for Key Agreement Protocols. Proceedings of ASIACRYPT 2005, Chennai, India, 4-8 December, vol 3788 of LNCS, pp. 549–569. Springer-Verlag, Berlin.
33. Krawczyk, H (2005). HMQV: A High-Performance Secure Diffie-Hellman Protocol. Proceedings of CRYPTO 2005, Santa Barbara, California, 14-18 August, vol 3621 of LNCS, pp. 546–566. Springer-Verlag, Berlin.

34. Meadows, C. (2001). Open Issues in Formal Methods for Cryptographic Protocol Analysis. Proceedings of MMM-ACNS 2001, St. Petersburg, Russia, 21-23 May, vol 2052 of LNCS, pp. 237–250. Springer-Verlag, Berlin.
35. NIST. SP 800-56A (2006). Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography. National Institute of Standards and Technology, Computer Security Division, Gaithersburg, MD.
36. Otway, D. and Rees, O. (1987). Efficient and Timely Mutual Authentication. ACM Operating Systems Review, 21(1), 8–10.
37. Paulson, L. C. (2001). Relations between Secrets: Two Formal Analyses of the Yahalom Protocol. Journal of Computer Security, 9(3), 197–216.
38. Shoup, V. (1999). On Formal Models for Secure Key Exchange (Version 4). RZ 3120 (#93166). IBM Research, Zurich.
39. Shoup, V (2001). OAEP Reconsidered. Proceedings of CRYPTO 2001, Santa Barbara, California, 19-23 August, vol 2139 of LNCS, pp. 239–259. Springer-Verlag, Berlin.