Errors in Computational Complexity Proofs for Protocols*

Kim-Kwang Raymond Choo & Colin Boyd & Yvonne Hitchcock

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

Abstract. Proofs are invaluable tools in assuring protocol implementers about the security properties of protocols. However, several instances of undetected flaws in the proofs of protocols (resulting in flawed protocols) undermine the credibility of provably-secure protocols. In this work, we examine several protocols with claimed proofs of security by Boyd & González Nieto (2003), Jakobsson & Pointcheval (2001), and Wong & Chan (2001), and an authenticator by Bellare, Canetti, & Krawczyk (1998). Using these protocols as case studies, we reveal previously unpublished flaws in these protocols and their proofs. We hope our analysis will enable similar mistakes to be avoided in the future.

1 Introduction

Despite cryptographic protocols being fundamental to many diverse secure electronic commerce applications, and the enormous amount of research effort expended in design and analysis of such protocols, the design of secure cryptographic protocols is still notoriously hard. The difficulty of obtaining a high level of assurance in the security of almost any new or even existing protocols is well illustrated with examples of errors found in many such protocols years after they were published. The many flaws discovered in published protocols for key establishment and authentication over many years, have promoted the use of formal models and rigorous security proofs, namely the computational complexity approach and the computer security approach.

Computer Security Approach. Emphasis in the computer security approach is placed on automated machine specification and analysis. The Dolev & Yao (1983) adversarial model is the de-facto model used in formal specifications, where cryptographic operations are often used in a

* The abridged version of this paper is going to appear in the proceedings of Asiacrypt 2005, LNCS 3788/2005 (pp. 624–643).

"black box" fashion ignoring some of the cryptographic properties, resulting in possible loss of partial information. The main obstacles in this automated approach are undecidability and intractability, since the adversary can have a large set of possible actions which results in a state explosion. Protocols proven secure in such a manner could possibly be flawed – giving a false positive result.

Computational Complexity Approach. On the other hand, the computational complexity approach adopts a deductive reasoning process whereby the emphasis is placed on a proven reduction from the problem of breaking the protocol to another problem believed to be hard. The first treatment of computational complexity analysis for cryptography began in the 1980s (Goldwasser & Micali 1984) but it was made popular for key establishment protocols by Bellare & Rogaway. In fact, Bellare & Rogaway (1993) provided the first formal definition for a model of adversary capabilities with an associated definition of security. These human-generated proofs provide a strong assurance that the security properties of the protocols are satisfied. However, it is often difficult to obtain correct proofs of security and the number of protocols that possess a rigorous proof of security remains relatively small. Furthermore, such proofs usually entail lengthy and complicated mathematical proofs, which are daunting to most readers (Koblitz & Menezes 2004). The breaking of provably-secure protocols after they were published is evidence of the difficulty of obtaining correct computational proofs of protocol security. Despite these setbacks, proofs are invaluable for arguing about security and certainly are one very important tool in getting protocols right.

Importance of Specifications and Details. Rogaway (2004) pointed out the importance of robust and detailed definitions in concrete security. In fact, specifications adopted in the computer security approach are expected to be precise (without ambiguity) and detailed, as such specifications are subjected to automated checking using formal tools. Boyd & Mathuria (2003) also pointed out that it is the responsibility of the protocol designers and not the protocol implementers to define the details of protocol specifications. Protocol implementers (usually non-specialists and/or industrial practitioners) will usually plug-and-use existing provably-secure protocols without reading the formal proofs of the protocols (Koblitz & Menezes 2004). Bleichenbacher (2005) also pointed out that important details are often overlooked in implementations of cryptographic protocols until specific attacks have been demonstrated. Flaws in security proofs or specifications themselves certainly will have a damaging effect on the trustworthiness and the credibility of provably-secure protocols in the real world.

In this work, we advocate the importance of proofs of protocol security, and by identifying some situations where errors in proofs arise, we hope that similar structural mistakes can be avoided in future proofs. We use several protocols with claimed proofs in the Bellare–Rogaway model as case studies, namely the conference key agreement protocol due to Boyd & González Nieto (2003), the mutual authentication and key establishment protocols (JP-MAKEP) due to Jakobsson & Pointcheval (2001) and WC-MAKEP due to Wong & Chan (2001). We also examine an encryption-based MT authenticator due to Bellare et al. (1998).

In the setting of the reductionist proof approach for protocols, the security model comprises protocol participants and a powerful probabilistic, polynomial-time (PPT) adversary $\mathcal{A}$, where the latter is in control of all communication between all parties in the model. The original BR93 proof model was defined only for two-party protocols. In subsequent work, the model is extended to analyse three-party server-based protocols (Bellare & Rogaway 1995) and multi-party protocols (Bresson et al. 2001).

Boyd–González Nieto Protocol. An inappropriate proof model environment is one of the likely areas where protocol proofs might go wrong. In the existing proof of the Boyd–González Nieto conference key agreement protocol (Boyd & González Nieto 2003), we observe that the proof model environment has the same number of parties in the model as in the protocol. This effectively rules out a multi-user setting in which to analyse the signature and encryption schemes. This shortcoming fails to include the case where $\mathcal{A}$ is able to corrupt a player who does not participate in the particular key agreement protocol session, and obtains a fresh key of any initiator principal by causing disagreement amongst parties about who is participating in the key exchange.

The attack we reveal on Boyd–González Nieto conference key agreement protocol is also known as an *unknown key share attack*, first described by Diffie et al. (1992). As discussed by (Boyd & Mathuria 2003, Chapter 5.1.2), $\mathcal{A}$ need not obtain the session key to profit from this attack. Consider the scenario whereby A will deliver some information of value (such as e-cash) to B. Since B believes the session key is shared with $\mathcal{A}$, $\mathcal{A}$ can claim this credit deposit as his. Also, a malicious adversary, $\mathcal{A}$, can exploit such an attack in a number of ways if the established session key is subsequently used to provide encryption (e.g., in AES) or integrity (Kaliski 2001).

In the attack on Boyd–González Nieto protocol, $\mathcal{A}$ is able to reveal the key of a non-partner oracle whose key is the same as the initiator principal, thus violating the key establishment goal. The existence of this attack means that the proof of Boyd–González Nieto's protocol is invalid, since the proof model allows Corrupt queries. Protocols proven secure in a proof model that allows the "Corrupt" query (in the proof simulation) ought to be secure against the unknown key share attack, since if a key is to be shared between some parties, U_1, U_2, and U_3, the corruption of some other (non-related) player in the protocol, say U_4, should not expose the session key shared between U_1, U_2, and U_3. In the proof simulations of the protocols on which we perform an unknown key share attack, $\mathcal{A}$ does not corrupt the owner or the perceived partners of the target Test session, but instead corrupts some other (non-related) player in the protocol that is not associated with the target Test session or a member of the "attacked" protocol session.

JP-MAKEP. We also describe an unknown key share attack on the JP-MAKEP which breaks the reduction of the proof from JP-MAKEP to the discrete logarithm problem. Similarly to the Boyd–González Nieto protocol, the proof model allows Corrupt queries for clients, and hence secure protocols ought to be immune to unknown key share attacks.

WC-MAKEP. An attack against WC-MAKEP is described where an adversary $\mathcal{A}$ is able to obtain a fresh key of an initiator oracle by revealing a non-partner server oracle sharing the same session key. The proof was sketchy and failed to provide any simulation.

Encryption-Based Authenticator. In the Bellare–Canetti–Krawczyk encryption-based authenticator, we demonstrate that an adversary $\mathcal{A}$ is able to use a Session-State Reveal query to find the one-time MAC key and use it to authenticate a fraudulent message. We identify the problem (in its proof) to be due to an incomplete proof specification (Session-State Reveal queries not adequately considered), which results in the failure of the proof simulation where the adversary has a non-negligible advantage, but the MAC forger, $\mathcal{F}$, does not have a non-negligible probability of forging a MAC digest (since it fails). This violates the underlying assumption in the proof. We also demonstrate how the flaw in this MT authenticator invalidates the proof of protocols that use the MT-authenticator using protocol 2DHPE (Hitchcock et al. 2003) as a case study.

Organization of Paper. Section 2 briefly explains the Bellare-Rogaway and the Canetti–Krawczyk models. Section 3 revisits the Boyd–González Nieto conference key agreement protocol, the JP-MAKEP, and the WC-MAKEP. Previously unpublished attacks on these protocols are demonstrated and flaws in the existing proofs are revealed. We conclude this section by proposing fixes to the protocols. Fixed protocols are not proven secure, and are presented mainly to provide a better insight into the proof failures. Section 4 revisits the encryption-based MT-authenticator proposed by Bellare et al. (1998). Finally, Section 5 presents the conclusions.

2 Informal Overview of the Bellare-Rogaway and Canetti–Krawczyk Models

Throughout this paper, the Bellare & Rogaway 1993 model, 1995 model (Bellare & Rogaway 1993, 1995), the Bellare, Pointcheval, & Rogaway 2000 model (Bellare et al. 2000), and the Canetti & Krawczyk 2001 model (Bellare et al. 1998, Canetti & Krawczyk 2001) model will be referred to as the BR93, BR95 BPR2000, and CK2001 models respectively. Collectively, the BR93, BR95, and BPR2000 models are known as the Bellare-Rogaway model.

2.1 Bellare-Rogaway Models

In the Bellare-Rogaway model, the adversary, $\mathcal{A}$, is defined to be a probabilistic machine that is in control of all communications between parties and is allowed to intercept, delete, delay, and/or fabricate any messages at will. $\mathcal{A}$ interacts with a set of $\Pi^i_{U_u,U_v}$ oracles (i.e., $\Pi^i_{U_u,U_v}$ is defined to be the i^{th} instantiation of a principal U_u in a specific protocol run and U_v is the principal with whom U_u wishes to establish a secret key). Let n denote the number of players allowed in the model, where n is polynomial in the security parameter k. The predefined oracle queries are shown in Table 1.
The definition of security depends on the notions of partnership of oracles and indistinguishability. The definition of partnership is used in the definition of security to restrict the adversary's Reveal and $\mathsf{Corrupt}$ queries to oracles that are not partners of the oracle whose key the adversary

A $\mathsf{Send}(U_u, U_v, i, m)$ query to oracle $\Pi^i_{U_u,U_v}$ computes a response according to the protocol specification and decision on whether to accept or reject yet, and returns them to the adversary $\mathcal{A}$. If the client oracle, $\Pi^i_{U_u,U_v}$, has either accepted with some session key or terminated, this will be made known to $\mathcal{A}$.
The $\mathsf{Reveal}(U_u, U_v, i)$ query captures the notion of **known key security**. Any client oracle, $\Pi^i_{U_u,U_v}$, upon receiving such a query and if it has accepted and holds some session key, will send this session key back to $\mathcal{A}$.
The $\mathsf{Corrupt}(U_u, K_E)$ query captures **unknown key share attacks** and **insider attacks**. This query allows $\mathcal{A}$ to corrupt the principal U_u at will, and thereby learn the complete internal state of the corrupted principal. Notice that a $\mathsf{Corrupt}$ query does not result in the release of the session keys since $\mathcal{A}$ already has the ability to obtain session keys through Reveal queries. In the BR95 model, this 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).
The $\mathsf{Test}(U_u, U_v, i)$ query is the only oracle query that does not correspond to any of $\mathcal{A}$'s abilities. If $\Pi^i_{U_u,U_v}$ has accepted with some session key and is being asked a $\mathsf{Test}(U_u, U_v, i)$ 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.

Table 1. Informal description of the oracle queries

is trying to guess. An important difference between the three Bellare–Rogaway models is in the way partner oracles are defined (i.e. the definition of partnership). The BR93 model defines partnership using the notion of matching conversations, where a conversation is defined to be the sequence of messages sent and received by an oracle. The sequence of messages exchanged (i.e., only the Send oracle queries) are recorded in the transcript, T. At the end of a protocol run, T will contain the record of the Send queries and the responses. Definition 1 gives a simplified definition of matching conversations.

Definition 1 (BR93 Matching Conversations) *Let n_S be the maximum number of sessions between any two parties in the protocol run. $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ are said to be partners if they both have matching conversations, where*

$$C_A = (\tau_0, 'start', \alpha_1), (\tau_2, \beta_1, \alpha_2)$$
$$C_B = (\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, *), \text{ for } \tau_0 < \tau_1 < \ldots$$

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. However, no explicit definition of partnership was provided in the original paper since there is no single partner function fixed for any protocol. Instead, security is defined predicated on the existence of a suitable partner function. Two oracles are BR95 partners if, and only if, the specific partner function in use says they are.

BPR2000 partnership is defined based on the notion of session identifiers (SIDs) where SIDs are suggested to be the concatenation of messages exchanged during the protocol run. In this 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 partner, if any at all. Definition 2 describes the definition of partnership in the BPR2000 model.

Definition 2 (BPR2000 Partnership) *Two oracles, $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$, are 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.*

2.2 Canetti-Krawczyk Model

In the CK2001 model, there are two adversarial models, namely the unathenticated-links adversarial model (UM) and the authenticated-links adversarial model (AM). Let $\mathcal{A}_{\mathrm{UM}}$ denote the adversary in the UM, and $\mathcal{A}_{\mathrm{AM}}$ denote the adversary in the AM. The difference between $\mathcal{A}_{\mathrm{AM}}$ and $\mathcal{A}_{\mathrm{UM}}$ lies in their powers. Table 2 provides an informal description of the oracle queries allowed for both $\mathcal{A}_{\mathrm{AM}}$ and $\mathcal{A}_{\mathrm{UM}}$. Let n denote the number of players allowed in the model, where n is polynomial in the security parameter k.

Oracle $\Pi^i_{U_u,U_v}$, upon receiving a Session-State Reveal(U_u, U_v, i) query and if it has neither accepted nor held some session key, will return all its internal state (including any ephemeral parameters but not long-term secret parameters) to the adversary.
Session − Key Reveal, Corrupt, and Test are equivalent to the Reveal, Corrupt, and Test queries in Table 1 respectively.
Send(U_u, U_v, i, m) is equivalent to the Send query in Table 1. However, $\mathcal{A}_{\mathrm{AM}}$ is restricted to only delay, delete, and relay messages but not to fabricate any messages or send a message more than once.

Table 2. Informal description of the oracle queries allowed for $\mathcal{A}_{\mathrm{AM}}$ and $\mathcal{A}_{\mathrm{UM}}$

A protocol that is proven to be secure in the AM can be translated to a provably secure protocol in the UM with the use of an authenticator. Definition 3 provides the definition of an autheticator.

Definition 3 (Definition of an Authenticator) *An authenticator is defined to be a mapping transforming a protocol* π_{AM} *in the* AM *to a protocol* π_{UM} *in the* UM *such that* π_{UM} *emulates* π_{AM}*.*

In other words, the security proof of a UM protocol depends on the security proofs of the MT-authenticators used and that of the associated AM protocol. If any of these proofs break down, then the proof of the UM protocol is invalid. CK2001 partnership can be defined using the notion of matching sessions, as described in Definition 4.

Definition 4 (Matching Sessions) *Two sessions are said to be matching if they have the same session identifier (SIDs) and corresponding partner identifier (PIDs).*

2.3 Definition of Freshness

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 5 describes freshness, which depends on the respective notion of partnership. The following definition of freshness does not incorporate the notion of forward secrecy, or the notions of session expiry and exposure in the Canetti–Krawczyk model since these notions are not necessary to explain our attacks.

Definition 5 (Definition of Freshness) *Oracle* $\Pi^i_{A,B}$ *is fresh (or holds a fresh session key) at the end of execution, if, and only if, (1)* $\Pi^i_{A,B}$ *has accepted with or without a partner oracle* $\Pi^j_{B,A}$*, (2) both* $\Pi^i_{A,B}$ *and* $\Pi^j_{B,A}$ *oracles have not been sent a* Reveal *query (or* Session-State Reveal *in the CK2001 model), and (3) A and B have not been sent a* Corrupt *query.*

2.4 Definition of Security

Security in the 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 $\mathcal{G}$, whose setting is explained below:

Stage 1: $\mathcal{A}$ is able to send any oracle queries at will.

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. 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 oracle queries at will but cannot make $\mathsf{Corrupt}$ or $\mathsf{Session\text{-}State/Key\ Reveal}$ queries that trivially expose the test session key.

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 notions of security for entity authentication are client-to-server authentication, server-to-client authentication, and mutual authentication. An adversary is said to violate client-to-server authentication if some fresh server oracle terminates with no partner. Similarly, an adversary is said to violate server-to-client authentication if some fresh client oracle terminates with no partner. An adversary is said to violate mutual authentication if some fresh oracle terminates with no partner.

Definitions 6, 7, and 8 describes the definition of security for the BR95 model, the BPR2000 model, and both the BR93 and CK2001 models respectively.

Definition 6 (BR95 Definition of Security) *A protocol is secure in the BR95 model if both the following requirements are satisfied: (1) When the protocol is run between two oracles $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ in the absence of a malicious adversary, both $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ accept and hold the same session key. (2) For all probabilistic, polynomial-time (PPT) adversaries $\mathcal{A}$, $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k})$ is negligible.*

Definition 7 (BPR2000 Definition of Security) *A protocol is secure in the BPR2000 model if both the following requirements are satisfied: (1)*

When the protocol is run between two oracles $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ in the absence of a malicious adversary, both $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ accept and hold the same session key. (2) For all probabilistic, polynomial-time (PPT) adversaries $\mathcal{A}$, the advantage that $\mathcal{A}$ has in violating entity authentication is negligible, and $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k})$ is negligible.

Definition 8 (BR93 and CK2001 Definitions of Security) *A protocol is secure in the BR93 and CK2001 models if both the following requirements are satisfied: (1) When the protocol is run between two oracles $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ in the absence of a malicious adversary, both $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ accept and hold the same session key, and (2) For all PPT adversaries $\mathcal{A}$, (a) If uncorrupted oracles $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ complete matching sessions, then both $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ must hold the same session key, and (b) $\mathsf{Adv}^{\mathcal{A}}(\mathsf{k})$ is negligible.*

3 Flawed Proofs in the Bellare–Rogaway Model

3.1 Boyd–González Nieto Conference Key Agreement Protocol

The conference key agreement protocol (Boyd & González Nieto 2003) shown in Figure 1 carries a claimed proof of security in the BR93 model, but uses a different definition of partnership than that given in the original model description. Although this protocol was proposed fairly recently, it has been widely cited and used as a benchmark. In the protocol, the notation (e_U, d_U) denotes the encryption and signature keys of principal U respectively, $\{\cdot\}_{e_U}$ denotes the encryption of some message under key e_U, $\sigma_{d_U}(\cdot)$ denotes the signature of some message under the signature key d_U, N_U denotes the random nonce chosen by principal U, $\mathcal{H}$ denotes some secure one-way collision-resistant hash function, and SK_U denotes the session key accepted by U. The protocol involves a set of p users, $\mathcal{U} = \{U_1, U_2, \ldots, U_p\}$.

The initiator, U_1, randomly selects a k-bit challenge N_1, encrypts N_1 under the public keys of the other participants in the protocol, signs the encrypted nonces $\{N_1\}_{e_{U_2}}, \ldots, \{N_1\}_{e_{U_p}}$ and broadcasts these messages in protocol flows 1 and 2 as shown in Figure 1. The other principals, upon receiving the broadcast messages, will respond with their identity and a random nonce. All principals are then able to compute the shared session key $SK_{U_i} = \mathcal{H}(N_1||N_2||\ldots||N_p)$. The session identifier (SID) in the protocol is defined to be the concatenation of messages received and sent. Note that the adversary, $\mathcal{A}$, is allowed to capture and suppress any broadcasted messages in the network.

1. $U_1 \rightarrow * : \mathcal{U} = \{U_1, U_2, \ldots, U_p\}, \sigma_{d_{U_1}}(\mathcal{U}, \{N_1\}_{e_{U_2}}, \ldots, \{N_1\}_{e_{U_p}})$
2. $U_1 \rightarrow * : \{N_1\}_{e_{U_i}}$ for $1 < i \leq p$
3. $U_i \rightarrow * : U_i, N_i$
The session key is $SK_{U_i} = \mathcal{H}(N_1 \| N_2 \| \ldots \| N_p)$.

Fig. 1. Boyd–González Nieto conference key agreement protocol

3.1.1 Unknown Key Share Attack Figure 2 shows the execution of the Boyd–González Nieto conference key agreement protocol in the presence of a malicious adversary, $\mathcal{A}$. For simplicity, let $\mathcal{U} = \{U_1, U_2, U_3\}$ and $\mathcal{U}_\mathcal{A} = \{\mathcal{A}, U_2, U_3\}$, which denote two different sessions.

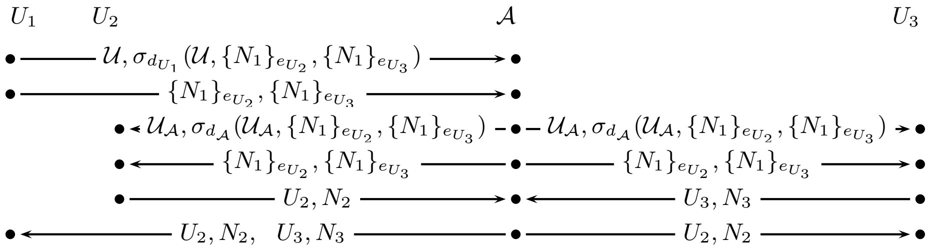

Fig. 2. Unknown key share attack

In Figure 2, the actions of the entities are as follows:

1. The initiator, U_1, encrypts N_1 under the public keys of the other participants in the protocol (i.e., $\mathcal{U} \setminus U_1$), signs the encrypted nonces $\{N_1\}_{e_{U_2}}, \{N_1\}_{e_{U_3}}$ together with $\mathcal{U}$, and broadcasts these messages in protocol flows 1 and 2.
2. A malicious adversary, $\mathcal{A}$, intercepts the broadcasted messages sent by U_1; that is, the broadcast messages sent by U_1 never reach the intended recipients, U_2 and U_3.
   - $\mathcal{A}$ then signs the intercepted encrypted nonces $\{N_1\}_{e_{U_2}}, \{N_1\}_{e_{U_3}}$ together with $\mathcal{U}_\mathcal{A}$ (instead of $\mathcal{U}$) under $\mathcal{A}$'s signing key

– $\mathcal{A}$ now acts as the **initiator** in a **different session** and broadcasts these messages in protocol flows 1 and 2.

3. U_2 and U_3 upon receiving the broadcasted messages, will reply to $\mathcal{A}$ with their identity and a random nonce.
4. $\mathcal{A}$ impersonates U_2 and U_3 and forwards the messages from U_2 and U_3 to U_1.
5. U_1, U_2, and U_3 are then able to compute the shared session key $SK_{U_i} = \mathcal{H}(N_1||N_2|, | \ldots ||N_n)$.

Table 3 describes the internal states of players U_1, U_2, and U_3 at the end of the protocol execution shown in Figure 2. We observe that U_1 is not partnered with either U_2 or U_3 according to Definition 2, since U_1 does not have matching SIDs or agreeing PIDs (Krawczyk termed such an attack a *key-replication attack* (Krawczyk 2005) whereby $\mathcal{A}$ succeeds in forcing the establishment of a session, S_1, other than the Test session or its matching session that has the same key as the Test session. In this case, $\mathcal{A}$ can distinguish whether the Test-session key is real or a random value by asking a Reveal query to the oracle associated with S_1).

U	sid_U	pid_U
U_1	$\mathcal{U}, \sigma_{d_{U_1}}(\mathcal{U}, \{N_1\}_{e_{U_2}}, \{N_1\}_{K_{U_3}}), \{N_1\}_{e_{U_2}}, \{N_1\}_{e_{U_3}}, U_2, N_2, U_3, N_3$	$\{U_2, U_3\}$
U_2	$\mathcal{U}_\mathcal{A}, \sigma_{d_\mathcal{A}}(\mathcal{U}_\mathcal{A}, \{N_1\}_{e_{U_2}}, \{N_1\}_{e_{U_3}}), \{N_1\}_{e_{U_2}}, \{N_1\}_{e_{U_3}}, U_2, N_2, U_3, N_3$	$\{\mathcal{A}, U_3\}$
U_3	$\mathcal{U}_\mathcal{A}, \sigma_{d_\mathcal{A}}(\mathcal{U}_\mathcal{A}, \{N_1\}_{e_{U_2}}, \{N_1\}_{e_{U_3}}), \{N_1\}_{e_{U_2}}, \{N_1\}_{e_{U_3}}, U_2, N_2, U_3, N_3$	$\{\mathcal{A}, U_2\}$

Table 3. Internal states of players U_1, U_2, and U_3

U_1 believes that the session key SK_{U_1} is being shared with U_2 and U_3, but U_2 (and U_3 respectively) believes the key $SK_{U_2} = \mathcal{H}(N_1||N_2||N_3) = SK_{U_3} = SK_{U_1}$ is being shared with $\mathcal{A}$ and U_3 (and U_2 respectively), when in fact, the key is being shared among U_1, U_2, and U_3. However, $SK_{U_1} = SK_{U_2} = SK_{U_3} = \mathcal{H}(N_1||N_2||N_3)$. Although the adversary $\mathcal{A}$ does not know the value of the session key (since $\mathcal{A}$ does not know the value of N_1), $\mathcal{A}$ is able to send a Reveal query to the session associated with either U_2 or U_3 and obtain $SK_{U_2} = \mathcal{H}(N_1||N_2||N_3) = SK_{U_3}$, which has the same value as SK_{U_1}. Hence, the Boyd–González Nieto conference key agreement protocol shown in Figure 1 is not secure in the BR93 model since the adversary $\mathcal{A}$ is able to obtain the fresh session key of the initiator U_1 by revealing non-partner oracles of U_1 (i.e., U_2 or U_3), in violation of the security definition given in Definition 8.

3.1.2 An Improved Conference Key Agreement Protocol It would appear that by changing the order of the application of the signature and encryption schemes, the attack shown in Figure 2 can be avoided. At a first glance, however, this may appear to contradict the result of An et al. (2002) that no matter what order signature and encryption schemes are applied, the result can still be secure. A closer inspection reveals that our observation actually supports the findings of An *et al.*, since the protocol operates in a multi-user setting. Although An *et al.* found that signature and encryption schemes can be applied in either order in the two user setting, they found some further restrictions in the multi-user setting. These restrictions are that the sender's identity must be included in every encryption and the recipient's identity must be included in every signature. In this case, swapping the order of the encryption and signature schemes happens to cause the protocol to fulfil these requirements.

An alternative way to prevent the attack is to include the sender's identity in each encryption and also the session identifier, *sid*, in the key derivation function. We use the same construct for *sid* (i.e., the concatentation of all messages received) as used by Boyd & González Nieto. In the improved protocol, the adversary $\mathcal{A}$ will not be able to "claim" ownership of the encrypted message $\{N_1, U_1\}_{e_{U_i}}$ since the identity of the initiator is included in the encryption. Since the construct of the session key in the improved protocol comprises the associated *sid*, a different *sid* will imply a different session key. Hence, the attack shown in Figure 2 will no longer be valid against this improved protocol. Figure 3 describes the improved protocol.

$$\begin{array}{l}
1.\ U_1 \rightarrow * : \mathcal{U} = \{U_1, U_2, \ldots, n\}, \sigma_{d_{U1}}(\mathcal{U}, \{N_1, U_1\}_{K_{U_2}}, \ldots, \{N_1, U_1\}_{K_{U_n}}) \\
2.\ U_1 \rightarrow * : \{N_1, U_1\}_{e_{U_i}} \text{ for } 1 < i \leq n \\
3.\ U_i \rightarrow * : U_i, N_i \\
\hline
sid = \mathcal{U} || \sigma_{d_{U1}}(\mathcal{U}, \{N_1, U_1\}_{K_{U_2}}, \ldots, \{N_1, U_1\}_{K_{U_n}}) || \{N_1, U_1\}_{e_{U_i}} || U_i || N_i \\
\text{The session key is } SK_{U_i} = \mathcal{H}(N_1 || sid).
\end{array}$$

Fig. 3. Improved Boyd–González Nieto conference key agreement protocol

3.1.3 Limitations of Existing Proof In the existing proof, the security of the protocol is proved by finding a reduction to the security of the encryption and signature schemes used. The number of protocol participants in the proof simulation, p, is assumed to be equal to the number of players allowed in the model, n, where n is polynomial in the security parameter k. In its reductionist approach, the proof assumes that there exists an adversary $\mathcal{A}$ who can gain a non-negligible advantage, $\mathsf{Adv}^{\mathcal{A}}(k)$, in distinguishing the test key from a random one. An attacker is then constructed that uses $\mathcal{A}$ to break either the underlying encryption scheme or the signature scheme.

In the context of the attack shown in Figure 2, assume that the number of protocol participants in the proof simulation is three. The proof then assumes that the number of parties in the model is also three. However, in order to carry out the attack, we have to corrupt a 4$^{\text{th}}$ player (i.e., U_4, an outsider as shown in Figure 4) to obtain the signature key of U_4.

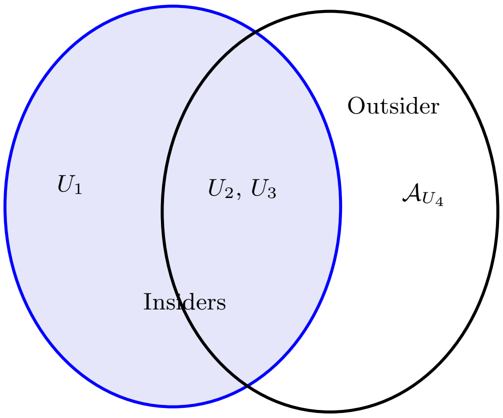

Fig. 4. Insiders vs outsider

In the proof simulation of the protocol execution shown in Figure 2, $\mathcal{A}$ corrupts U_4, an outsider in the target session, and assumes U_4's identity. Since U_4 does not exist in the model assumed by the proof, the attacker against the encryption and signature schemes cannot simulate the $\mathsf{Corrupt}(U_4)$ query for $\mathcal{A}$ and the proof fails (since although $\mathcal{A}$ succeeds, it cannot be used to break either the encryption or signature schemes). Our observation is consistent with the above results of An *et al.*, which highlight the underlying cause of the proof breakdown – the proof environment effectively did not allow a multi-user setting in which to analyse the signature and encryption schemes.

3.2 Jakobsson–Pointcheval MAKEP

Figure 5 describes the published version of JP-MAKEP (Jakobsson & Pointcheval 2001), which was designed for low power computing devices[1]. JP-MAKEP carries a claimed proof of security in the BR93 model but uses the notion of SIDs in the definition of partnership. There are two communicating principals in MAKEP, namely the server B and the client of limited computing resources, A. The security goals of the protocol are mutual authentication and key establishment between the two communicating principals. A and B are each assumed to know the public key of the other party (i.e., g^{x_B} and g^{x_A} respectively).

Client A (x_A, g^{x_A})		Server B (x_B, g^{x_B})
$a, t \in_R \mathbb{Z}_q, c = g^a, T = g^t, K = (g^{x_B})^a$		
$r = \mathcal{H}_1(T, g^{x_B}, c, K), A' = \mathcal{H}_2(g^{x_B}, c, K)$	$\xrightarrow{ID_B, c, r}$	$K = c^{x_B}, A = \mathcal{H}_2(g^{x_B}, c, K)$
$A' \stackrel{?}{=} A$	$\xleftarrow{A, e}$	$0 \leq e < 2^k$
$d = t - ex_A \ mod\ q$	$\xrightarrow{ID_A, d}$	$r \stackrel{?}{=} \mathcal{H}_1(g^d (g^{x_A})^e, g^{x_B}, c, K)$
$sid = (ID_B, c, r, A, e, ID_A, d)$		$sid = (ID_B, c, r, A, e, ID_A, d)$
$sk = \mathcal{H}_0(g^{x_B}, c, K)$		$sk = \mathcal{H}_0(g^{x_B}, c, K)$

Fig. 5. Jakobsson–Pointcheval MAKEP

3.2.1 Unknown Key Share Attack Figure 6 depicts an example execution of JP-MAKEP in the presence of a malicious adversary $\mathcal{A}$. At the end of the attack, B believes he shares a session key, $sk_{B\mathcal{A}} = \mathcal{H}_0(g^{x_B}, c, K)$, with the adversary $\mathcal{A}$, when in fact the key is being shared with A (i.e., unknown key share attack). A and B are not partners since they have different SIDs, $sid_{B\mathcal{A}} = (ID_B, c, r, A, e, ID_{\mathcal{A}}, d - ex_{\mathcal{A}} \ mod\ q) \neq sid_{AB}$, and different perceived partners (i.e., $PID_A = A$ and $PID_B = \mathcal{A}$).

[1] The original version appeared in the unpublished pre-proceedings of Financial Crypto 2001 with a claimed proof of security in the BR93 model. Nevertheless, a flaw in the protocol was discovered by Wong & Chan (2001). In this published version, the flaw found by Wong & Chan in the original version has been fixed.

$$
\begin{array}{ccc}
A\ (x_A, g^{x_A}) & \mathcal{A}\ (x_{\mathcal{A}}, g^{x_{\mathcal{A}}}) & B\ (x_B, g^{x_B}) \\
\xrightarrow{ID_B, c, r} & \xrightarrow{ID_B, c, r} & \\
\xleftarrow{A, e' = 0} & \text{Fabricate} & \xleftarrow{A, e} \\
\xrightarrow{ID_A, d = t} & \text{Fabricate} & \xrightarrow{ID_{\mathcal{A}}, d - ex_{\mathcal{A}}\ mod\ q} \\
 & & r \stackrel{?}{=} \mathcal{H}_1(g^{d - ex_{\mathcal{A}}}(g^{x_{\mathcal{A}}})^e, g^{x_B}, c, K) \\
\multicolumn{3}{c}{sid_{B\mathcal{A}} = (ID_B, c, r, A, e, ID_{\mathcal{A}}, d - ex_{\mathcal{A}}\ mod\ q)} \\
\multicolumn{3}{c}{sid_{AB} = (ID_B, c, r, A, e', ID_A, d) \neq sid_{B\mathcal{A}}} \\
sk_{AB} = \mathcal{H}_0(g^{x_B}, c, K) & & sk_{B\mathcal{A}} = \mathcal{H}_0(g^{x_B}, c, K)
\end{array}
$$

Fig. 6. Unknown key attack on Jakobsson–Pointcheval MAKEP

From Figure 6, we observe that A has terminated the protocol without any partners, in violation of the server-to-client authentication goal. On the other hand, the server, B, has terminated the protocol with the adversary, $\mathcal{A}$, as its partner. Hence, the client-to-server authentication is not violated. Consequently, JP-MAKEP is not secure since the adversary is able to obtain a fresh session key of A by revealing a non-partner oracle of A (i.e., an oracle of B), in violation of the security definition given in Definition 8. A fix for JP-MAKEP is to change $0 \leq e < 2^k$ in the protocol specification to $0 < e < 2^k$.

3.2.2 Flaws in Existing Proof

In the proof simulation of the protocol, let P be another client where $P \neq A, B$. P is clearly the "outsider" in the target session of Figure 6 that $\mathcal{A}$ is attacking. $\mathcal{A}$ then corrupts P, the outsider, and assumes P's identity. This is allowed in the existing proof (Jakobsson & Pointcheval 2001, Lemma 3) for the server-to-client authentication, since it is claimed that the JP-MAKEP provides partial forward-secrecy whereby corruption of the client may not help to recover the session keys.

The proof assumes that the probability of $\mathcal{A}$ violating the server-to-client authentication is negligible. In the context of the attack shown in Figure 6, $\mathcal{A}$ managed to violate the server-to-client authentication by corrupting a non-partner player, P. By violating the server-to-client authentication, $\mathcal{A}$ is then able to distinguish a real key or a random key by asking a Reveal query to a non-partner server oracle of A. This violates the server-to-client authentication with non-negligible probability. The discrete logarithm breaker $\mathcal{A}_{\mathcal{DL}}$ (which is constructed using $\mathcal{A}$) is unable to obtain a non-negligible probability of breaking the discrete logarithm problem, contradicting the underlying assumption in the proof. Consequently, the proof simulation fails (the result of Reveal and Corrupt queries were not adequately considered in the simulation).

3.3 Wong–Chan MAKEP

Figure 7 describes WC-MAKEP (Wong & Chan 2001), which was proposed as an improvement to the original unpublished version of JP-MAKEP. Note that Figure 7 describes the corrected version of WC-MAKEP, where the computation of $\sigma = (r_A \oplus r_B)$ by A is replaced by $\sigma = (r_A \oplus r_B)||ID_B$.

$A\ (a, g^a)$		$B\ (SK_B, PK_B)$
$r_A \in_R \{0,1\}^k, x = \{r_A\}_{PK_B}$		
$b \in_R \mathbb{Z}_q \setminus \{0\}, \beta = g^b$	$\xrightarrow{Cert_A, \beta, x}$	Decrypt x
$\sigma = (r_A \oplus r_B)\Vert ID_B$		
$y = a\mathcal{H}(\sigma) + b\ mod\ q$	$\xleftarrow{\{r_B, ID_B\}_{r_A}}$	$r_B \in \{0,1\}^k$
$SK_{AB} = \mathcal{H}(\sigma)$	$\xrightarrow{y}$	$g^y \stackrel{?}{=} (g^a)^{\mathcal{H}(\sigma)}\beta, SK_{BA} = \mathcal{H}(\sigma)$

Fig. 7. Wong–Chan MAKEP

3.3.1 A New Attack Figure 8 depicts an example execution of WC-MAKEP, where at the end of the protocol execution, A and B accept with the same session key, $SK_{AB} = \mathcal{H}(\sigma) = SK_{BA}$.

A	$\mathcal{A}$	B
$\xrightarrow{Cert_A, \beta, x}$	Fabricate message	$\xrightarrow{Cert_A, \beta \cdot g^e, x}$
$\xleftarrow{\{r_B, ID_B\}_{r_A}}$		$\xleftarrow{\{r_B, ID_B\}_{r_A}}$
$\xrightarrow{y}$	$y' = y + e\ mod\ q$	$g^{y'} \stackrel{?}{=} (g^a)^{\mathcal{H}(\sigma)}(\beta \cdot g^e)$
$SK_{AB} = \mathcal{H}(\sigma)$	$\xrightarrow{y'}$	$SK_{BA} = \mathcal{H}(\sigma)$

Fig. 8. Attack on Wong–Chan MAKEP

However, according to Definition 1, both A and B are not partners as B's replies are not in response to genuine messages sent by A (i.e., both A and B will not have matching conversations given in Definition 1). Since two non-partner oracles, $\Pi_{A,B}$ and $\Pi_{B,A}$, accept session keys with the same value, the adversary $\mathcal{A}$ can reveal a fresh non-partner oracle, $\Pi_{B,A}$, and find the session key accepted by $\Pi_{A,B}$. This violates Definition 8. Both oracles of A and B have terminated the protocol without any partners, in violation of the mutual authentication goal. WC-MAKEP, therefore, is insecure in the BR93 model since the attack outlined in Figure 8 shows that both the key establishment and mutual authentication goals are violated.

3.3.2 Preventing the Attack A possible fix to WC-MAKEP is to change the construction of the session key to $SK = \mathcal{H}(A, B, \beta, x, y, \sigma)$. The inclusion of the sender's and responder's identities and messages (β, x, y) in the key derivation function effectively binds the session key to all messages sent and received by both A and B (Choo et al. 2005). If the adversary changes any of the messages in the transmission, the session key will also be different. Intuitively, the attack shown in Figure 8 will no longer be valid against WC-MAKEP.

3.3.3 Flaws in Existing Proof The existing (sketchy) proof fails to provide a proof simulation. In the absence of a game simulation in the existing proof, we may only speculate that the proof fails to adequately consider the simulation of Send and Reveal queries (in the same sense as outlined in Section 3.2.2).

In the flaws in the AMP protocol (Kwon 2001) and EPA protocol (Hwang et al. 2003) revealed by Wan & Wang (2004), both proofs fail to provide any proof simulations. These examples highlight the importance of detailed proof simulations, as the omission of such simulations could potentially result in protocols claimed to be secure being, in fact, insecure.

4 Flaw in the Proof of an Encryption-Based MT-Authenticator

In this section, we reveal an inadequacy in the specification of the encryption based MT-authenticator proposed by Bellare et al. (1998) and identify a flaw in its proof simulation. We then demonstrate with an example protocol (the protocol 2DHPE (Hitchcock et al. 2003)) how the flaw in the proof of the encryption-based MT-authenticator results in the violation of the key establishment goal in the protocol 2DHPE where a malicious adversary is able to learn a fresh session key. The attack we reveal on the protocol 2DHPE also applies to protocol 14 that appears in the full version of (Hitchcock et al. 2004). Surprisingly, the inadequacy in the specification was not spotted in the proof simulation of the MT-authenticator, and has not previously been spotted in other protocols (Hitchcock et al. 2004, 2003) using this MT-authenticator.

We may speculate that if protocol designers fail to spot this inadequacy in the specification of their protocols, the protocol implementers are also highly unlikely to spot this inadequacy until specific attacks have been demonstrated, as suggested by Bleichenbacher (2005).

Having identified the flaw in the proof of the MT-authenticator, we provide a fix to the MT-authenticator specification. As a result of this fix, protocols using the revised encryption based MT-authenticator will no longer be flawed due to their use of this MT-authenticator. The notation used throughout this section is as follows: the notation $\{\cdot\}_{K_U}$ denotes an encryption of some message m under U's public key, K_U, and $MAC_K(m)$ denotes the computation of MAC digest of some message m under key K.

4.1 Bellare–Canetti–Krawczyk Encryption-Based MT-Authenticator

Figure 9 describes the encryption based MT-authenticator, which is based on a public-key encryption scheme indistinguishable under chosen-ciphertext attack and the authentication technique used by Krawczyk (1996). Note that the specification of the encryption-based MT-authenticator does not specify the deletion of the received nonce v_A (incidentally, v_A is also the one-time MAC key) from B's internal state before sending out the last message.

4.2 Flaw in Existing Proof of MT-Authenticator

In the usual tradition of reductionist proofs, the existing MT-authenticator proof (Bellare et al. 1998) assumes that there exists an adversary $\mathcal{A}$ who can break the MT-authenticator, and an encryption-aided MAC forger, $\mathcal{F}$ is constructed using such an adversary $\mathcal{A}$ against the unforgability of the underlying MAC scheme. Subsequently, the encryption-aided MAC

A		B
Choose nonce v_A	$\xleftarrow{sid, m}$	Choose message m
	$\xrightarrow{sid, m, \{v_A\}_{K_B}}$	Decrypt $\{v_A\}_{K_B}$
Verify $MAC_{v_A}(m, A)$	$\xleftarrow{sid, m, MAC_{v_A}(m, A)}$	Compute $MAC_{v_A}(m, A)$

Fig. 9. Bellare–Canetti–Krawczyk encryption-based MT-authenticator

forger, $\mathcal{F}$, can be used to break the encryption scheme. $\mathcal{F}$ who has access to a MAC oracle, is easily constructed as follows:

- guess at random an index i,
- for all but the i-th session, generate a key v_k and answer queries as expected,
- if $\mathcal{A}$ calls a Session-State Reveal[2] on any session other than the i-th session, the response can easily be simulated,
- if $\mathcal{A}$ calls a Session-State Reveal on the i-th session, $\mathcal{F}$ aborts.

The assumption is that if $\mathcal{A}$ has a non-negligible advantage against the underlying protocol, then $\mathcal{F}$ has a non-negligible probability of forging a MAC digest.

Consider the scenario shown in Figure 10. When $\mathcal{A}$ asks for the one-time MAC key (i.e., v_k) with a Session-State Reveal query, it is perfectly legitimate since this session with SID of sid_j is not the i-th session with SID of sid_i. Recall that sessions with non-matching SIDs (i.e., $sid_i \neq sid_j$) are non-partners.
Clearly, $\mathcal{F}$ is unable to answer such a query since v_A is a secret key (note that the MAC oracle to which $\mathcal{F}$ has access is associated with v_A, but $\mathcal{F}$ does not know v_A). Hence, the proof simulation is aborted and $\mathcal{F}$ fails. Consequently, $\mathcal{F}$ does not have a non-negligible probability of forging a MAC digest (since it fails) although $\mathcal{A}$ has a non-negligible advantage against the security of the underlying protocol, in violation of the underlying assumption in the proof.

We note that in a later independent yet related work by Tian & Wong (2006),the same flaw in the proof of the encryption-based MT-authenticator described in Figures 9 and 10 is discovered.

[2] Note that in the original paper of Bellare et al. (1998), a Session-State Reveal is known as a Session-Corruption query.

A	$\mathcal{A}$	B
	Intercept	$\xleftarrow{sid_j, m}$
$\xleftarrow{sid_i, m'}$	Fabricate	
$\xrightarrow{sid_i, m', \{v_A\}_{K_B}}$	Intercept	
	Fabricate	$\xrightarrow{sid_j, m, \{v_A\}_{K_B}}$
		$\xleftarrow{sid_j, m, MAC_{v_A}(m, A)}$
		$\xrightarrow{\mathsf{Session{-}State\ Reveal}(sid_j)}$
$\xleftarrow{sid_i, m', MAC_{v_A}(m', A)}$	Fabricate	$\xleftarrow{v_A}$

Fig. 10. Execution of encryption-based MT-authenticator in the presence of a malicious adversary, $\mathcal{A}$

4.3 Proposed Fix to the Encryption-Based MT-Authenticator

In this section, we provide a fix to the encryption-based MT-authenticator by requiring that the party concerned delete the received nonce from its internal state before sending out the MAC digest computed using the received nonce. With the fix, the adversary will not be able to obtain the value of v_A using a Session-State Reveal query. Hence, in the proof of the security of the MT-authenticator, $\mathcal{F}$ will be able to answer such a query because $\mathcal{F}$ is no longer required to return the value of v_A. Therefore, the attack shown in Figure 10 will no longer be valid, since $\mathcal{A}$ will no longer be able to obtain the value of v_A and fabricate a MAC digest.

4.4 An Example Protocol as A Case Study

Figure 11 describes a password-based protocol 2DHPE due to Hitchcock et al. (2003). Using the protocol 2DHPE as an example, we demonstrate that as a result of the flaw in the proof of the encryption-based MT-authenticator, the proof of protocol 2DHPE is also invalid. In the example protocol, both A and B are assumed to share a secret password, $\pi_{A,B}$, and the public keys of both A and B (i.e., K_A and K_B respectively) are known to all participants in the protocol. The protocol uses the encryption-based MT-authenticator to authenticate the message B, sid, g^y from B.
Figure 12 describes an example execution of protocol 2DHPE in the presence of a malicious adversary $\mathcal{A}$ (in the UM). We assume that $\mathcal{A}$ has a shared password with B, $\pi_{\mathcal{A},B}$. At the end of the protocol execution shown

$A\ (\pi_{A,B})$		$B\ (\pi_{A,B})$
$x \in_R \mathbb{Z}_q, v_A \in_R \{0,1\}^k$	$\xrightarrow{A, sid, g^x, \{v_A\}_{K_B}}$	$y \in_R \mathbb{Z}_q$
		$v'_A = \mathcal{D}_{d_B}(\{v_A\}_{K_B}), N_B \in_R \{0,1\}^k$
$SK_{A,B} = (g^y)^x$	$\xleftarrow{B, sid, g^y, N_B, MAC_{v'_A}(B, sid, g^y, A)}$	
	$\xrightarrow{sid, \{A, sid, g^x, N_B, \pi_{A,B}\}_{K_B}}$	$SK_{B,A} = (g^x)^y$

Fig. 11. Hitchcock, Tin, Boyd, González Nieto, & Montague (2003) protocol 2DHPE

in Figure 12, oracle $\Pi^{sid}_{A,B}$ has accepted a shared session key $SK_{A,B} = g^{xz}$ with $\Pi^{sid}_{B,A}$. However, such an oracle (i.e., $\Pi^{sid}_{B,A}$) does not exist. By sending a Session-State Reveal query to oracle $\Pi^{sid_{\mathcal{A}}}_{B,\mathcal{A}}$, $\mathcal{A}$ learns the internal state of $\Pi^{sid_{\mathcal{A}}}_{B,\mathcal{A}}$, which includes v'_A. With v'_A, $\mathcal{A}$ can fabricate and send a MAC digest to A. Hence, the adversary is able to obtain a fresh session key of $\Pi^{sid}_{A,B}$ (i.e., $SK_{A,B} = g^{xz}$) since $\mathcal{A}$ knows z (in fact, z is chosen by $\mathcal{A}$).

A	$\mathcal{A}$	B
$\xrightarrow{A, sid, g^x, \{v_A\}_{K_B}}$	$\xrightarrow{\mathcal{A}, sid_{\mathcal{A}}, g^x, \{v_A\}_{K_B}}$	$v'_A = \mathcal{D}_{d_B}(\{v_A\}_{K_B})$
		$\xleftarrow{B, sid_{\mathcal{A}}, g^y, N_B, MAC_{v'_A}(B, sid, g^y, A)}$
	$\xrightarrow{\mathsf{Session-State\ Reveal}(B, sid_{\mathcal{A}})}$	
	$\xleftarrow{B, sid, g^z, N_B, MAC_{v'_A}(B, sid, g^z, A)}$	$\xleftarrow{v'_A}$
$\xrightarrow{sid, \{A, sid, g^x, N_B, \pi_{A,B}\}_{K_B}}$		
$SK_{A,B} = g^{xz}$		

Fig. 12. Execution of protocol 2DHPE in the presence of a malicious adversary

If the encryption-based MT-authenticator requires B to delete the received nonce v'_A from B's internal state before sending out message 3, then $\mathcal{A}$ will not be able to obtain the value of v'_A with a Session-State Reveal query and fabricate $MAC_{v'_A}(B, sid, g^y, A)$. Protocol 2DHPE will, therefore, be secure.

5 Conclusion

Through a detailed study of several protocols and an authenticator with claimed proofs of security, we have concluded that specifying correct computational complexity proofs for protocols remains a hard problem. We have identified three areas where protocol proofs are likely to fail; namely

- an inappropriate proof model environment,
- Send, Reveal and Corrupt queries not adequately considered in the proof simulations, and
- omission of proof simulations.

We also observe that certain constructions of session keys may contribute to the security of the key establishment protocol. This observation supports the findings of recent work of Choo et al. (2005), who describe a way of constructing session keys, as described below:

- The identities and roles of the participants to provide resilience against unknown key share attacks and reflection attacks since the inclusion of the identities of both the participants and role asymmetry effectively ensures some sense of direction. If the role of the participants or the identities of the (perceived) partner participants change, the session keys will also be different.
- The unique SIDs ensure that session keys will be fresh. If SIDs are defined as the concatenation of messages exchanged during the protocol execution, messages altered during the transmission will result in different session keys. This prevents the key replicating attack (Krawczyk 2005) in the Bellare–Rogaway and Canetti–Krawczyk models.

6 Acknowledgements

This work was partially funded by the Australian Research Council Discovery Project Grant DP0345775. The first author would like to thank Dr. Juan Manuel González Nieto of Information Security Institute for his insightful discussions on the conference key agreement protocol.

Bibliography

An, J. H., Dodis, Y. & Rabin, T. (2002), On the Security of Joint Signature and Encryption, *in* L. R. Knudsen, ed., 'Advances in Cryptology - Eurocrypt 2002', Springer-Verlag, pp. 83–107. Volume 2332/2002 of Lecture Notes in Computer Science.

Bellare, M., Canetti, R. & Krawczyk, H. (1998), A Modular Approach to The Design and Analysis of Authentication and Key Exchange Protocols, *in* J. Vitter, ed., '30th ACM Symposium on the Theory of Computing - STOC 1998', ACM Press, pp. 419–428.

Bellare, M., Pointcheval, D. & Rogaway, P. (2000), Authenticated Key Exchange Secure Against Dictionary Attacks, *in* B. Preneel, ed., 'Advances in Cryptology – Eurocrypt 2000', Springer-Verlag, pp. 139 – 155. Volume 1807/2000 of Lecture Notes in Computer Science.

Bellare, M. & Rogaway, P. (1993), Entity Authentication and Key Distribution, *in* D. R. Stinson, ed., 'Advances in Cryptology - Crypto 1993', Springer-Verlag, pp. 110–125. Volume 773/1993 of Lecture Notes in Computer Science.

Bellare, M. & Rogaway, P. (1995), Provably Secure Session Key Distribution: The Three Party Case, *in* F. T. Leighton & A. Borodin, eds, '27th ACM Symposium on the Theory of Computing - STOC 1995', ACM Press, pp. 57–66.

Bleichenbacher, D. (2005), Breaking a Cryptographic Protocol with Pseudoprimes, *in* S. Vaudenay, ed., 'Public Key Cryptography - PKC 2005', Springer-Verlag, pp. 9–15. Volume 3386/2005 of Lecture Notes in Computer Science.

Boyd, C. & González Nieto, J. M. (2003), Round-optimal Contributory Conference Key Agreement, *in* Y. Desmedt, ed., 'Public Key Cryptography - PKC 2003', Springer-Verlag, pp. 161–174. Volume 2567/2003 of Lecture Notes in Computer Science.

Boyd, C. & Mathuria, A. (2003), *Protocols for Authentication and Key Establishment*, Springer-Verlag.

Bresson, E., Chevassut, O. & Pointcheval, D. (2001), Provably Authenticated Group Diffie–Hellman Key Exchange — The Dynamic Case, *in* C. Boyd, ed., 'Advances in Cryptology - Asiacrypt 2001', Springer-Verlag, pp. 209–223. Volume 2248/2001 of Lecture Notes in Computer Science.

Canetti, R. & Krawczyk, H. (2001), Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels (Extended version available from http://eprint.iacr.org/2001/040/), *in* B. Pfitzmann, ed., 'Advances in Cryptology - Eurocrypt 2001', Springer-Verlag, pp. 453–474. Volume 2045/2001 of Lecture Notes in Computer Science.

Choo, K.-K. R., Boyd, C. & Hitchcock, Y. (2005), On Session Key Construction in Provably Secure Protocols (Extended version available from http://eprint.iacr.org/2005/206), *in* E. Dawson & S. Vaudenay, eds, '1st International Conference on Cryptology in Malaysia - Mycrypt 2005', Springer-Verlag, pp. 116–131. Volume 3715/2005 of Lecture Notes in Computer Science.

Diffie, W., van Oorschot, P. C. & Wiener, M. J. (1992), 'Authentication and Authenticated Key Exchange', *Journal of Designs, Codes and Cryptography* **2**, 107–125.

Dolev, D. & Yao, A. C. (1983), 'On the Security of Public Key Protocols', *IEEE Transaction of Information Technology* **29(2)**, 198–208.

Goldwasser, S. & Micali, S. (1984), 'Probabilisitic Encryption (Available from http://theory.lcs.mit.edu/~joanne/shafi-pubs.html)', *Journal of Computer and System Sciences* **28**, 270–299.

Hitchcock, Y., Boyd, C. & González Nieto, J. M. (2004), Tripartite Key Exchange in the Canetti-Krawczyk Proof Model (Extended version available from http://sky.fit.qut.edu.au/~boydc/papers/), *in* A. Canteaut & K. Viswanathan, eds, '5th International Conference on Cryptology in India - Indocrypt 2004', Springer-Verlag, pp. 17–32. Volume 3348/2004 of Lecture Notes in Computer Science.

Hitchcock, Y., Tin, Y.-S. T., Boyd, C., González Nieto, J. M. & Montague, P. (2003), A Password-Based Authenticator: Security Proof and Applications, *in* T. Johansson & S. Maitra, eds, '4th International Conference on Cryptology in India - Indocrypt 2003', Springer-Verlag, pp. 388–401. Volume 2904/2003 of Lecture Notes in Computer Science.

Hwang, Y. H., Yum, D. H. & Lee, P. J. (2003), EPA: An Efficient Password-Based Protocal for Authenticated Key Exchange, *in* R. Safavi-Naini & J. Seberry, eds, '8th Australasian Conference on Information Security and Privacy - ACISP 2003', Springer-Verlag. Volume 2727/2003 of Lecture Notes in Computer Science.

Jakobsson, M. & Pointcheval, D. (2001), Mutual Authentication and Key Exchange Protocol for Low Power Devices, *in* P. F. Syverson, ed., '5th International Conference on Financial Cryptography - FC 2001', Springer-Verlag, pp. 169–186. Volume 2339/2002 of Lecture Notes in Computer Science.

Kaliski, B. S. (2001), 'An Unknown Key-Share Attack on the MQV Key Agreement Protocol', *ACM Transactions on Information and System Security (TISSEC)* **4(3)**, 275–288.

Koblitz, N. & Menezes, A. (2004), Another Look at "Provable Security", Technical report CORR 2004-20, Centre for Applied Cryptographic Research, University of Waterloo, Canada.

Krawczyk, H. (1996), SKEME: A Versatile Secure Key Exchange Mechanism for Internet, *in* 'ISOC Network and Distributed System Security - NDSS 1996', IEEE Internet Society Press, pp. 114–127.

Krawczyk, H. (2005), HMQV: A High-Performance Secure Diffie-Hellman Protocol (Extended version available from `http://eprint.iacr.org/2005/176/`), *in* V. Shoup, ed., 'Advances in Cryptology - Crypto 2005', Springer-Verlag, pp. 546–566. Volume 3621/2005 of Lecture Notes in Computer Science.

Kwon, T. (2001), Authentication and Key Agreement via Memorable Passwords, *in* A. Juels & J. Brainard, eds, 'ISOC Networks and Distributed Security Systems - NDSS 2001', Internet Society Press.

Rogaway, P. (2004), On the Role Definitions in and Beyond Cryptography, *in* M. J. Maher, ed., '9th Asian Computing Science Conference - Asian 2004', Springer-Verlag. Volume 3321/2004 of Lecture Notes in Computer Science.

Tian, X. & Wong, D. S. (2006), Session Corruption Attack and Improvements on Encryption Based MT-Authenticators, *in* D. Pointcheval, ed., 'Cryptographers' Track at RSA Conference - CT-RSA 2006', Springer-Verlag. Lecture Notes in Computer Science.

Wan, Z. & Wang, S. (2004), Cryptanalysis of Two Password-Authenticated Key Exchange Protocols, *in* H. Wang, J. Pieprzyk & V. Varadharajan, eds, '9th Australasian Conference on Information Security and Privacy - ACISP 2004', Springer-Verlag. Volume 3108/2004 of Lecture Notes in Computer Science.

Wong, D. S. & Chan, A. H. (2001), Efficient and Mutually Authenticated Key Exchange for Low Power Computing Devices, *in* C. Boyd, ed., 'Advances in Cryptology - Asiacrypt 2001', Springer-Verlag, pp. 172–289. Volume 2248/2001 of Lecture Notes in Computer Science.