INFORMATICA, 2007, Vol. 18, No. 4, 547–568

An Integrative Framework to Protocol Analysis and Repair: Bellare–Rogaway Model + Planning + Model Checker

Kim-Kwang Raymond CHOO *
Australian Institute of Criminology
GPO Box 2944, Canberra ACT 2601 Australia
e-mail: raymond.choo@aic.gov.au

Received: December 2006

Abstract. A modified version of the Bellare and Rogaway (1993) adversarial model is encoded using Asynchronous Product Automata (APA). A model checker tool, Simple Homomorphism Verification Tool (SHVT), is then used to perform state-space analysis on the Automata in the setting of planning problem. The three-party identity-based secret public key protocol (3P-ID-SPK) protocol of Lim and Paterson (2006), which claims to provide explicit key authentication, is used as a case study. We then refute its heuristic security argument by revealing a previously unpublished flaw in the protocol using SHVT. We then show how our approach can automatically repair the protocol. This is, to the best of our knowledge, the first work that integrates an adversarial model from the computational complexity paradigm with an automated tool from the computer security paradigm to analyse protocols in an artificial intelligence problem setting – planning problem – and, more importantly, to repair protocols.

Key words: key establishment protocols, model checker, key authentication, provable security, planning problem.

1. Introduction

Key establishment protocols are used for distributing shared keying material in a secure manner. The first key establishment protocol with public key properties is published by Diffie and Hellman (1976). Although a later protocol of Merkle (1978) – Merkle's puzzles – also achieves the key distribution goal, the Diffie–Hellman protocol enjoys a better ratio between security and efficiency (Wolf, 1999).

Despite an enormous amount of research effort has been expended in the design and analysis of such protocols, there are still worthwhile contributions to be made even in the simple scenario of two users with an on-line server (e.g., the recently proposed three-party identity-based secret public key protocol (3P-ID-SPK) protocol of Lim and Paterson (2006)). The difficulties associated in obtaining a high level of assurance in the security

*The views and opinions expressed in this paper are those of the author and do not reflect those of the Australian Government and Australian Institute of Criminology. Research was performed while the author was with the Intelligent Systems Laboratory / University of Western Sydney.

of almost any new or even existing key establishment protocols are well illustrated with examples of errors found in many such protocols years after they were published (Choo, 2006b; Phan and Goi, 2006). The study of such protocols has led to a dichotomy in cryptographic protocol analysis techniques between the computational complexity approach (Bellare *et al.*, 2000; Bellare and Rogaway, 1993; Canetti and Krawczyk, 2001) and the computer security approach (Lowe, 1996; Meadows, 2003).

Computational Complexity Paradigm. The emphasis in this (provable security) paradigm for protocols 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 and Micali, 1984). It was made popular for key establishment protocols by Bellare and Rogaway (1993). They provided the first formal definition for a model of adversarial capabilities with an associated definition of security (hereafter referred to as the BR93 model).

The difficulty of obtaining correct computational proofs of security is, however, dramatically illustrated by the well-known problem with the OAEP mode for public key encryption (Shoup, 2001). Problems with proofs of protocol security have occurred too, evidenced by the breaking of several provably-secure protocols after they were published (Abdalla *et al.*, 2006; Choo, 2006b). 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) (Boyd and Mathuria, 2003).

Computer Security Paradigm. Since the 1990s, many researchers have shifted their attention to using formal methods (also known as the computer security approach). Emphasis in this approach is placed on automated machine specification and analysis. The main goal of this approach is to relieve humans of the tedious and error prone parts of the mathematical proofs.

The Dolev and Yao (1983) adversarial model is often the de-facto standard used in formal specifications, where cryptographic operations are used in a "black box" fashion. Such an approach, however, ignores some of the cryptographic properties, resulting in possible loss of partial information. Cervesato *et al.* (1999) also pointed out that the main obstacles in this automated approach are undecidability and intractability. Messages from an adversary are unbounded in structural depth and the number of possible values for some data fields is infinite. The adversary can have a large set of possible actions, which results in a state explosion. It is generally acknowledged that protocols proven secure in such a manner could possibly be flawed (Backes and Jacobi, 2003). However, this approach has the benefit of providing unambiguous specification of system requirements and mathematically precise proofs of system properties. This approach should also be credited for finding both known and previously unknown flaws in protocols (Lowe, 1996; Meadows, 1999).

An Integrated Approach. Recognising the disparity in the two different approaches to protocol analysis, Abadi and Rogaway (2002) started the trend of unifying the two

paradigms. In this integrated approach, the aim is to provide abstract models of cryptographic primitives suitable for machine analysis in some well defined sense. More recent comprehensive efforts have been under way in several independent yet related projects by Backes *et al.* (Backes, 2004a; Backes, 2004b; Backes and Jacobi, 2003), Blanchet *et al.* (Blanchet, 2006; Blanchet and Pointcheval, 2006), Canetti *et al.* (Canetti, 2000; Canetti and Fischlin, 2001), and Choo *et al.* (Choo, 2006a; Choo *et al.*, 2004).

Motivations of Paper. One may note that attacks and proofs on protocols are hard to find and check respectively. Moreover, once an attack has been identified (e.g., by using some automated tools from the computer security approach), protocol modifications by hand are also prone to errors and may not necessarily lead to secure protocols. One recent example is the original version of Jakobsson–Pointcheval protocol that appeared in the unpublished pre-proceedings of Financial Crypto 2001 (Jakobsson and Pointcheval, 2001) with a claimed proof of security in the BR93 model. A flaw in the original protocol was discovered by Wong and Chan (2001). Fixes were proposed by Jakobsson and Pointcheval and by Wong and Chan independently; and had claimed security proofs in the BR93 model. However, both fixes were later shown to be flawed (Choo *et al.*, 2005b).

We are motivated by the observation that despite recent advances in the integrative approach, no researchers have propose an integrated framework to analyse and (automatically) repair protocols that were found to be flawed (during the analysis). For example, in the approaches of Backes *et al.*, Blanchet *et al.*, and Canetti *et al.*, the focus is to prove the protocols secure using some automated tools whilst in the approach of Choo *et al.* – which is more closely related to our proposed framework – the focus is to analyse protocols that were proven secure in the Bellare–Rogaway model (Bellare *et al.*, 2000; Bellare and Rogaway, 1993) and the Canetti–Krawczyk model (Canetti and Krawczyk, 2001).

Our Proposed Integrative Framework. In this work, we propose an integrative framework that allows us to analyse protocols using a modified version of the BR93 adversarial model[1] and, more importantly, to repair protocols that are found to be insecure against certain types of attacks[2]. Note that our approach differs from the automatic protocol generation (APG) approach of Song *et al.* (Song, 1999; Song *et al.*, 2001). In the APG approach, the focus is on how to develop a particular protocol in some automatic way in terms of a set of property specifications about the application, and not to analyse and repair protocols.

We now present an overview of our framework.

1: The Adversarial Model. We provide a formal specification and machine analysis of a *modified* version of the widely accepted indistinguishability-based model of Bellare and Rogaway (1993), the BR93 model, from computational proofs in the setting of a planning problem.

[1]We remark that another distinctive feature of our framework as compared to other existing computer security approaches – with the exception of the approach by Choo *et al.* (Choo, 2006a; Choo *et al.*, 2004) – is that we are able to analyse protocols proven secure in the BR93 model (and also the CK2001 model with some minor modifications).

[2]To the best of our knowledge, this is the first work that utilises an automated tool to update “insecure" protocols specifications.

- The planning problem, an ongoing research area in artificial intelligence, is about composing a workable plan (a sequence of actions) that allows the agent to achieve its given goal(s) from the initial state (Lifschitz, 1999; Weld, 1999). This is also known as goal-directed reasoning. In our context, the given goal is defined as a state of insecurity and if a workable plan exists, then we have found an attack on the protocol that we are analysing.
- Our choice of formalism for this work is Asynchronous Product Automata (APA), a universal state-based formal method. APA is supported by the Simple Homomorphism Verification Tool (SHVT) (Ochsenschläger *et al.*, 1998) for analysis and verification of cooperating systems and communicating automata. Once the possible state transitions of each automaton have been specified, SHVT (the planner) can be used to automatically search the state space of the model. SHVT provides a reachability graph of the explored states. In our APA specification, the abstract communication model captures the representation of the protocol, the message transmission, and the communication channels.

2: Protocol Specification. As a case study we analyse the three-party identity-based secret public key (3P-ID-SPK) protocol of Lim and Paterson (2006). The protocol claims to provide explicit key authentication – implicit key authentication[3] and key confirmation[4].

3: Protocol Analysis. Our planner, SHVT, reveals a workable plan that allows us to achieve our defined goals (states of insecurity) from the given initial state. In other words, previously unpublished flaw on the Lim–Paterson protocol is revealed by the automated state space analysis performed with SHVT.

4: Protocol Repair. In our framework, we have a set of generic patches to overcome certain attacks, such as the known session key attacks, unknown key share attacks, and reflection attacks, which we term the "repairable" attacks. Once some attack has been revealed using SHVT in the earlier analysis, we check to determine if the revealed attack is in the list of our pre-defined "repairable" attacks. This list can be updated when new patches are discovered.

If revealed attack $\in$ "repairable" attacks,
then modify the existing specification of the protocol accordingly,
otherwise do nothing.
End If.

Contributions of Paper. We regard the main contributions of this paper to be two-fold:

1) confirmation of the feasibility of using formal specifications to identify problems in human-generated computational complexity proofs by revealing previously unpublished flaw in the case study protocol, and

[3]The property whereby one party is assured that no other party aside from a specifically identified second party (and possibly additional identified trusted parties) may gain access to a particular secret key.

[4]The property whereby one party is assured that a second (possibly unidentified) party actually has possession of a particular secret key.

2) an automated approach to repair protocols that were found to be insecure against certain types of attacks.

Foreword. Demonstrating that this integrative framework is useful for analysing and repairing protocols for key establishment is the main conceptual contribution of our work. However, presenting the full APA specifications and a comprehensive description of the planning problem turns out to be rather "challenging" due to space constraints. Moreover, reader not familiar with APA / SHVT might not find the specifications interesting. Therefore, we postpone the presentation of the full APA specifications and instead present only sufficient formalization necessary to understand our approach. Interested reader can refer to the SHVT manual (SHVTManual, 2004). Our framework can also be instantiated with other specification languages and model checker tools.

Outline of Paper. In Section 2, we revisit the three-party identity-based secret public key (3P-ID-SPK) protocol of Lim and Paterson (2006), which we will use as a case study. We then present an overview of our formal specification framework in Section 3. We briefly explain how the protocols are specified in our framework, followed by the results of the protocol analysis using SHVT. Using our approach, we then repair the revealed flaw in the case study protocol. Finally, Section 4 presents the conclusions. Analysis and repair of another case study protocol, the conference key agreement protocol of Boyd and González Nieto (2003), is presented in Appendix C.

2. Case Study Protocol

Protocol 1 involves three parties, a trusted server S and two principals, A and B, who wish to establish communication. The security goal of Protocol 1 is to distribute a session key between two communication principals (i.e., implicit key authentication), which is suitable for establishing a secure session. Key confirmation is also provided by Protocol 1, and hence, achieving explicit key authentication (Menezes *et al.*, 1997; Definition 12.8).

The notation used in Protocol 1 is as follows.

- $\{\cdot\}_K$ denote the encryption of some message under some key K,
- r_U denote some randomly chosen nonce, $||$ denote concatenation of messages,
- $MAC_{K^{MAC}(\cdot)}$ denote the computation of a MAC digest under some MAC key K^{MAC},
- $Sig_{K^{Sign}}(\cdot)$ denote the signature of some message under some signature key K^{Sign},
- $K_{U_1U_2}$ denote the session key shared between users U_1 and U_2,
- pwd_{US} denote some secret password shared between some user U and the server S, and
- $\mathcal{H}$ denote a secure cryptographic hash function.

1. Protocol 1 begins by having A randomly select a random string, ST_A, and a nonce, r_A. A then computes the encryption key, $PK_{A1} = \mathcal{H}(A||S||pwd_{AS})$, and encrypts (A, ST_A) using PK_{A1}. The message, $A, r_A, \{A, ST_A\}_{PK_{A1}}$, is then sent to B with whom A desires to communicate.

1. $A \longrightarrow B$: $A, r_A, \{A, ST_A\}_{PK_{A1}}$
2. $B \longrightarrow S$: $B, \{B, ST_B\}_{PK_{B1}}, A, r_A, \{A, ST_A\}_{PK_{A1}}$
3. $S \longrightarrow B$: $Sig_{SK_{B2}}(K_{AB}), Sig_{SK_{A2}}(K_{AB})$
4. $B \longrightarrow A$: $Sig_{SK_{A2}}(K_{AB}), MAC_{K_{AB}}(r_A), r_B$
5. $A \longrightarrow B$: $MAC_{K_{AB}}(r_B)$

Protocol 1. Lim–Paterson three-party identity-based secret public key (3P-ID-SPK) protocol.

2. Upon receiving the message from A, B also randomly selects a random string, ST_B, computes the encryption key, $PK_{B1} = \mathcal{H}(B||S||pwd_{BS})$, and encrypts (B, ST_B) using PK_{B1}. The message, $B, \{B, ST_B\}_{PK_{B1}}, A, r_A, \{A, ST_A\}_{PK_{A1}}$, is then sent to the server, S.
3. S computes the corresponding private key pairs for PK_{A1} and PK_{B1}, SK_{A2} and SK_{B2} respectively. Using SK_{A2} and SK_{B2}, the respectively ciphertexts are decrypted and ST_A and ST_B obtained. S then runs the session key generator to obtain a session key K_{AB}, which has not been used before. K_{AB} is then signed under the respectively SK_{A2} and SK_{B2}, and sent to B.
4. B recovers K_{AB} from $Sig_{SK_{B2}}(K_{AB})$ using PK_{B1}, computes the MAC digest $MAC_{K_{AB}}(r_A)$, and randomly chooses a nonce, r_B. Message $Sig_{SK_{A2}}(K_{AB})$, $MAC_{K_{AB}}(r_A)$, r_B is then sent.
5. A recovers K_{AB} from $Sig_{SK_{A2}}(K_{AB})$ using PK_{A1}. If the MAC digest received verifies true, then A will compute MAC digest $MAC_{K_{AB}}(r_B)$ and sends $MAC_{K_{AB}}(r_B)$ to B.
6. At the end of Protocol 1's execution, both A and B have accepted a session key of the same value, K_{AB}.

3. Our Integrative Framework

In this section, we present an overview of our APA formal specification framework. We follow the general adversarial formalism of the BR93 model, except that we now have a definition of insecurity rather than a definition of security. We then specify Protocol 1 using APA and demonstrate that SHVT (or any other model checker tool) can be used to find previously unknown flaws in the protocols. We conclude this section by showing a repaired Lim–Paterson protocol.

3.1. *The Adversarial Model*

In the setting of the reductionist proof approach for protocols, the security model comprises protocol participants and a powerful probabilistic, polynomial-time (PPT) adversary, $\mathcal{A}$. The adversary, $\mathcal{A}$, is in control of all communication between all parties in the model. Each party P_i can run multiple sessions concurrently. The action of U_u running a session, i, is modelled as an oracle. The adversary, $\mathcal{A}$ controls the communications between parties by interacting with a set of oracle, $\Pi^i_{U_u,U_v}$, where $\Pi^i_{U_u,U_v}$ 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.

$\mathcal{A}$ controls the communication channels via the queries to the targeted oracles, as shown below.

$\mathsf{Send}(U_u, U_v, i, m)$ **query.** This query to an 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 oracle has either accepted with some session key or terminated, this will be made known to $\mathcal{A}$.

$\mathsf{Reveal}(U_u, U_v, i)$ **query.** Any oracle upon receiving such a query and if it has accepted and holds some session key, will send this session key back to $\mathcal{A}$.

$\mathsf{Corrupt}(U_u)$ **query.** This query allows $\mathcal{A}$ to corrupt the principal U_u at will, and thereby learn the complete internal state (e.g., password) of the corrupted principal.

Definition of insecurity for the protocol is dependent on the notions of partnership of oracles and indistinguishability of the session key. The definition of partnership is used in the definition of security to restrict the adversary's Reveal and Corrupt queries to oracles that are not partners of the oracle whose key the adversary is trying to guess.

3.1.1. *Definition of Partnership*

Partnership in the BR93 model is defined using the notions of matching conversations. However, we adopt the recent trend, also adopted by several other researchers (Jeong *et al.*, 2004; Krawczyk, 2005; Kudla and Paterson, 2005), and define partnerships based on the notion of session identifiers (SIDs).

DEFINITION 1 (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.

3.1.2. *Definition of Freshness*

Definition 2 describes the notion of freshness, which depends on the notion of partnership in Definition 1.

DEFINITION 2. Oracle Π^i_A is fresh (or it holds a fresh session key) at the end of execution, if, and only if, (1) oracle Π^i_A has accepted with or without some partner oracle(s), (2) both oracle Π^i_A and its partner oracle(s) (if such a partner oracle exists) are unopened (i.e., have not been sent a Reveal query), and (3) neither A nor any of the partnering players (if such a partner oracle exists) are corrupted (i.e., none of them have been sent any Corrupt query).

3.2. *Protocols Specification*

As mentioned earlier, the setting of our approach is based on the planning problem. Therefore, similar to the formulation of the planning problem (Weld, 1999), we have three inputs in the formulation of our framework, as follows.

Description of Initial State. We describe the internal states and knowledge of (e.g., keys that are known to) the agents. Agents in our formal framework comprise the protocol participants and a malicious adversary (adopted from the BR93 model described in Section 3.1).

Description of Goal State. Before an agent can achieve its goal(s), we need to provide a formal description of the goal state that the agent has to achieve. In our framework, we define a desired goal as a state of insecurity. Hence, if an agent manages to achieve the defined goal, then the protocol is insecure.

Description of Possible Actions. This input provides a formal description for the set of possible actions that can be performed by the agents.

For the rest of the paper, we use the terms agent and automata interchangeably. In our formal framework using APA specification, protocol principals are modelled as a family of elementary automata. The various state spaces of the principals are modelled as a family of state sets. The channel through which the elementary automaton communicates is modelled by the addition and removal of messages from the shared state component Network, which is initially empty. Each of the elementary automata only has access to the particular state components to which it is connected. In addition to the regular protocol principals, we specify an adversary, $\mathcal{A}$, which has access to the shared state component Network, but no access to the internal states of the principals.

This adversary, $\mathcal{A}$, is adopted from the BP93 model described in Section 3.1 whereby $\mathcal{A}$ is able to intercept messages in the Network, swap data components in the intercepted messages to form new messages, remove messages from the Network, or fabricate new messages. $\mathcal{A}$ is then able to send these messages to the oracles via the Network (corresponding to Send queries in the BR93 model). Also, once an oracle, Π_U^i, has accepted and holds a session key, the session identifier associated with that oracle becomes visible to the adversary, $\mathcal{A}$, via the shared state component Transcript. If $\mathcal{A}$ so chooses, $\mathcal{A}$ is able to obtain the session key of Π_U^i (stored in the state component Keys) via a Reveal query. The shared state component Transcript also contains a log of all sent messages. For simplicity, the graphical illustration of a two-party (instead of three-party) protocol in APA specification is presented in Fig. 1.

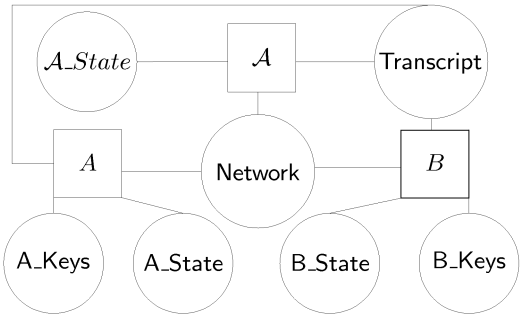

Fig. 1. Graphical illustration of a two-party protocol in APA specification.

Description of Initial State. The first phase of our formal specification is to specify the basic data types necessary to specify our case study protocols. Examples of the data types and the functions we defined are shown in Fig. 4 of Appendix A.

Defining SIDs in Protocol 1. Bellare, Pointcheval, and Rogaway (Bellare *et al.*, 2000) suggested that SIDs can be constructed on-the-fly using fresh unique contributions from the communicating participants. Uniqueness of SIDs is necessary since otherwise two parties may share a key but not be partners (in the sense of Definition 1), and hence the protocol would not be considered secure.

Within Protocol 1, the only values that A and B can be sure are unique are r_A, r_B, ST_A, and ST_B. However, only r_A and r_B are both known to A and B. Hence, a naive construct for SIDs in Protocol 1 is the concatenation of r_A and r_B where r_A and r_B are the unique nonces contributed by individual protocol participant.

Another possible SID construct is the concatenation of the protocol participants' identities and the unique nonces contributed by individual protocol participant. Including the identities of the participating parties is to provide a binding to the session identifier. This avoids scenarios where two or more sessions have identical keys but different session identities as protocol participants have different perceived partners. Such an approach is adopted by several other researchers (Jeong *et al.*, 2004; Krawczyk, 2005; Kudla and Paterson, 2005) and is also recommended by NIST (NIST, 2006). Therefore for Protocol 1, we define SIDs as the concatenation of the protocol participants' identities and the unique nonces contributed by individual protocol participant (Choo, 2007).

Description of Goal State. In using automated tools, we find it more natural to define the goal state to be the **state of insecurity** – reachability of this state implies that the protocol being analysed is insecure – rather than the state of security due to the limitations of model checking tool (i.e., state explosion problem). Definition 3 depends on the notions of partnership in Definition 1 and freshness in Definition 2.

DEFINITION 3. A protocol is insecure in our formal framework if a workable plan is composed by our planner (an agent achieves any of the following goal state(s)).

Goal State 1. Two fresh non-partner oracles accept the same key – in violation of the key establishment goal.

Goal State 2. Some fresh oracle accepts some key, which has been exposed (i.e., is known to $\mathcal{A}$) – in violation of the key establishment goal.

Goal State 3. Some fresh oracle accepts and terminates with no partner – in violation of the key confirmation goal.

Violation of goal state 1 also implies that the protocol is vulnerable to a *key replicating attack* first described by Krawczyk (2005). Definition 4 presents a description of the key replicating attack.

DEFINITION 4 (Key Replicating Attack (Krawczyk, 2005)). A key replicating attack is defined to be an attack whereby the adversary, $\mathcal{A}$, 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 random by asking a Session-Key Reveal query to the oracle associated with S_1.

Since our formal framework is closely based on the BR93 model, protocols that were proven secure in the BR93 model but found to be violating any of the first two goal states (or any of the three goal states if the protocol provides key confirmation) described in Definition 3 will be insecure in the BR93 model. Consequently, the existing proof of the (insecure) protocol will also be invalid.

Description of Possible Actions. Possible actions refer to the message exchanges among the agents (i.e., protocol participants and the adversary). As Russell and Norvig (1995) suggested, actions are represented by logical descriptions of pre-conditions and effects (which we term post-conditions). To model actions, we would now need to specify the properties of all states of components associated with the particular elementary automaton, and the changes of the states caused by the state transition.

Step 1. The name of the transition pattern has to be first defined, e.g., Protocol Step 1.
Step 2. The variables to be used in this transition pattern, $(x_1, \ldots, x_n)$, is then defined. Note that the variables defined here are local to this transition pattern.
Step 3. The required (pre-)conditions prior to performing a state transition is now defined. If any of the defined conditions is not satisfied, this transition pattern will proceed no further.
Step 4. The changes of the states caused by this state transition is now specified, i.e., the post-conditions / effects.
State Transition. A state transition can only occur when all the above steps are executed.

3.3. *Protocols Analysis*

Fig. 2 presents an example reachability graph describing how searches are performed in the SHVT analysis, namely: forward from the initial state to the finish state when an agent has achieved its goal (i.e., achieving any of the goal states described in Definition 3).

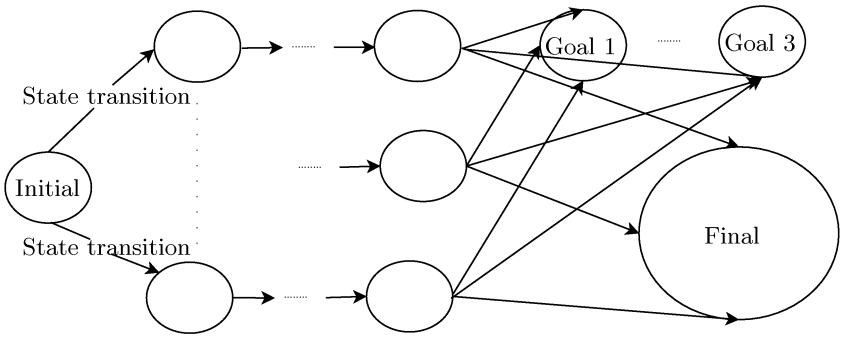

Fig. 2. A reachability graph: protocol analysis in SHVT.

Protocol Analysis	# Players	# Runs	Run-Time	Goal state(s) achieved	New Flaw(s)?
Protocol 1	4	2	Approximately 2 mins	Yes (Goal states 2 and 3 of Definition 3)	Yes

Fig. 3. Analysis statistics.

If a workable plan exists (i.e., the agent manages to reach any of the finish states, achieving goal state 1, goal state 2, and/or goal state 3 of Definition 3), we can then trace the path that it takes and hence, find the attack sequence. This is also known as a solution – a plan that an agent can execute, which guarantees the achievement of the goal (Russell and Norvig, 1995) – in the planning problem. Note that the nodes in the reachability graph described in Fig. 2 represent the various states of the protocol execution and the arcs (arrows) represent the state transitions between two states.

The search approach in our SHVT analysis is also known as a progression planner in the planning problem. Consequently, the inherent limitations associated with a progression planner, i.e., the high branching factor and thus the huge size of the search space (Russell and Norvig, 1995), are also present in our approach. For example, in several of our earlier experiments, we observe that certain interleavings of the protocol message exchanges and actions of the adversary that are "obviously" immaterial in achieving the goal, are explored by SHVT. In other words, what is obvious to us, human, are not necessarily obvious to the progression planner or SHVT.

Hence, for run-time efficiency, and to avoid enormous branching factors in the search space, we restrict the actions of the adversary so that certain actions are possible for only some message types. The attack sequence and the internal states can be examined by viewing the reachability graphs produced by SHVT. The analyses were run on a Pentium IV 2.16 GHz computer with 1024 Mb of RAM and the analysis statistics are shown in Fig. 3.

Analysis of Protocol 1. State space analysis in SHVT reveals that both goal states 2 and 3 of Definition 3 are violated. The attack sequence is described in Attack 1. Let C_B denotes C impersonating B.

The adversary, $\mathcal{A}$, asks a Corrupt query to C, a client participant – *static corruption*. $\mathcal{A}$ now runs as C.
1. $A \longrightarrow B$: $A, r_A, \{A, ST_A\}_{PK_{A1}}$
The adversary intercepts and deletes this message from the Network.
$2(C)$. $C \longrightarrow S$: $C, \{C, ST_C\}_{PK_{C1}}, A, r_A, \{A, ST_A\}_{PK_{A1}}$
3. $S \longrightarrow C$: $Sig_{SK_{C2}}(K_{AC}), Sig_{SK_{A2}}(K_{AC})$
$4(C)$. $C_B \longrightarrow A$: $Sig_{SK_{A2}}(K_{AC}), MAC_{K_{AC}}(r_A), r_C$
5. $A \longrightarrow B$: $MAC_{K_{AB}}(r_B)$
The adversary intercepts and deletes this message from the Network.

Attack 1. Attack sequence on Protocol 1.

In Attack 1,

Static Corruption. The adversary, $\mathcal{A}$, asks a Corrupt query to C, a client participant, prior to the execution of Protocol 1 between A and B – *static corruption*. $\mathcal{A}$ now

runs as C since the Corrupt query enables the adversary to learn the entire internal state of C including the password shared between C and S, pwd_{CS}.

Step 1 of Protocol 1. A randomly selects a random string, ST_A, and a nonce, r_A. A then computes the encryption key, $PK_{A1} = \mathcal{H}(A||S||pwd_{AS})$, and encrypts (A, ST_A) using PK_{A1}. The message is then sent to B as per protocol specification.

Message Interception. The adversary intercepts and deletes this message sent by A from the Network.

Step 2 of Protocol 1 by C. C also randomly selects a random string, ST_C, computes the encryption key, $PK_{C1} = \mathcal{H}(C||S||pwd_{CS})$, and encrypts (C, ST_C) using PK_{C1}. The message, $C, \{C, ST_C\}_{PK_{C1}}, A, r_A, \{A, ST_A\}_{PK_{A1}}$, is then sent to the server, S.

Step 3 of Protocol 1. S computes the corresponding private key pairs for PK_{A1} and PK_{C1}, SK_{A2} and SK_{C2} respectively. Using SK_{A2} and SK_{C2}, the respectively ciphertexts are decrypted and ST_A and ST_C obtained. S then runs the session key generator to obtain a session key K_{AC}, which has not been used before. K_{AC} is then signed under the respectively SK_{A2} and SK_{C2}, and sent to C.

Step 4 of Protocol 1 by C. C recovers K_{AC} from $Sig_{SK_{C2}}(K_{AB})$ using PK_{C1}, computes the MAC digest $MAC_{K_{AC}}(r_A)$, and randomly chooses a nonce, r_C. The message, $Sig_{SK_{A2}}(K_{AC}), MAC_{K_{AC}}(r_A), r_C$, is then sent to A by C **impersonating** B.

Step 5 of Protocol 1. A recovers K_{AC} from $Sig_{SK_{A2}}(K_{AC})$ using PK_{A1}. The MAC digest received will certainly verified true since C already recovers K_{AC} from $Sig_{SK_{C2}}(K_{AB})$ using PK_{C1} in the earlier step. Hence, A will compute MAC digest $MAC_{K_{AC}}(r_C)$ and sends $MAC_{K_{AC}}(r_C)$ to B.

Message Interception. The adversary intercepts and deletes this message sent by A from the Network.

Session Key. If the MAC digest received verifies true, then C (the adversary) knows that A has accepted session key, K_{AC} and A believes is being shared with B. However, B is unaware of such a session with A since all messages sent by A to B have been intercepted and deleted from the Network by the adversary. Hence, A has accepted and terminated without a partner – goal state 3 of Definition 3. Moreover, the adversary knows the session key accepted by A after receiving the message from S in Step 3 of the protocol execution – goal state 2 of Definition 3.

3.4. *Protocol Repair*

As mentioned earlier, we have a set of generic patches to overcome attacks, such as the known session key attacks, unknown key share attacks, and reflection attacks, which we term the "repairable" attacks. We now check to determine if the revealed attack is in our pre-defined list of "repairable" attacks.

Attack 1 can be considered an unknown key share attack since A believes the key is being shared with B when in fact, the key is being shared with C (the adversary). Hence, Attack 1 is in our pre-defined list of "repairable" attacks. We now proceed to repair the

protocols. For readability, we present an informal overview of our repairing procedure (in plain english rather than in the unwieldy APA specifications). Note that we assume that sid is the concatenation of the protocol participants' identities and other unique messages.

1: Session Key Derivation Function Checking. Is the session identifier (sid) part of the keying material used in the session key derivation function?

If yes, proceed to Step 2,

Otherwise modify the key derivation function within the protocol specifications to a hash of the concatenation of the session identifier and existing keying materials using an independent hash function (i.e., in the case of Protocol 1, session key is now constructed as $\mathcal{H}_1(sid||K_{AB}) = \mathcal{H}_1(A||B||S||r_A||r_B||K_{AB})$ where $\mathcal{H}$ and $\mathcal{H}_1$ are independent secure cryptographic hash functions).

2: Encryption Scheme Checking Does the protocol uses any encryption scheme?

If yes, check if the identities of the sender and the intended recipient are included within the message to be encrypted?

If yes, proceed to Step 3.

Otherwise modify the encryption function within the protocol specifications to include the identity of the sender and the intended recipient so that these identities are included within the message to be encrypted.

Otherwise proceed to Step 3.

3: MAC Scheme Checking Does the protocol uses any MAC scheme?

If yes, check if the identities of the sender and the intended recipient are included within the MAC digest to be generated?

If yes, proceed to Step 4.

Otherwise modify the MAC function within the protocol specifications to include the identity of the sender and the intended recipient so that these identities are included within the MAC digest to be generated.

Otherwise proceed to Step 4.

4: Signature Scheme Checking Does the protocol uses any signature scheme?

If yes, check if the identities of the sender and the intended recipient are included within the signature to be generated?

If yes, proceed to Step 5.

Otherwise modify the signature function within the protocol specifications to include the identity of the sender and the intended recipient so that these identities are included within the signature to be generated.

Otherwise proceed to Step 5.

5: Protocol Analysis Restart the protocol analysis described in Section 3.3 with the repaired protocol. Check to see if any of the goal states in Definition 3 are violated.

If yes, output error message "Protocol cannot be repaired!".

Otherwise output message "Protocol successfully repaired!" and then terminated. Due to the current limitations of SHVT, we are unable to automatically output the specifications of the repaired protocol (in a nice and presentable diagram). However, the specifications of the repaired protocol will be written to another ($.vsp$) file, which can be viewed with any word editor.

Protocol 2[5] describes the resulting repaired Protocol 1. Let sid_U and $\mathcal{SK}_U$ denote the session identifier and session key of some participant respectively.

1. $A \longrightarrow B$: $A, r_A, \{A, B, ST_A\}_{PK_{A1}}$
2. $B \longrightarrow S$: $B, \{B, A, ST_B\}_{PK_{B1}}, A, r_A, \{A, B, ST_A\}_{PK_{A1}}$
3. $S \longrightarrow B$: $Sig_{SK_{B2}}(A, B, K_{AB}), Sig_{SK_{A2}}(A, B, K_{AB})$
4. $B \longrightarrow A$: $Sig_{SK_{A2}}(A, B, K_{AB}), MAC_{K_{AB}}(B, A, r_A), r_B$
5. $A \longrightarrow B$: $MAC_{K_{AB}}(A, B, r_B)$

$$sid_A = A||B||S||r_A||r_B = sid_B$$

$$\mathcal{SK}_A = \mathcal{H}_1(sid_A||K_{AB}) = \mathcal{H}_1(sid_B||K_{AB}) = \mathcal{SK}_B$$

Protocol 2. A repaired Protocol 1.

We remark that in our generic patches, we ensure that the identities of the protocol participants are explicitly exchanged in every encryption, MAC digest, and signature; and also include the session identifier in the key derivation function. In so doing, we provide a binding between the messages and the protocol participants, and between the session key and the session identifier. However, our approach has the drawback of making the protocol scheme non-symmetric as the participants will have to be ordered.

Although one might observe that our generic patches are not that new – it has been pointed out in our earlier work (Choo *et al.*, 2005b; Choo *et al.*, 2005a) that similar fixes can prevent certain attacks – having an automated tool to analyse and repair "broken" protocols is. Moreover, we hope that by having access to such an automated tool will help to prevent future cases of old attacks making their way into new protocols (e.g., our case study protocol was recently published in *Security Protocols Workshop 2006* (Lim and Paterson, 2006)).

4. Conclusions

We proposed a formal framework in which we integrate an adversarial model from the computational complexity paradigm (i.e., the BR93 adversarial model (Bellare and Rogaway, 1993)) and an automated tool from the computer security paradigm (i.e., APA / SHVT) in the setting of the planning problem, and an artificial intelligence problem, to analyse protocols. As a case study, we specified and analysed the three-party identity-based secret public key protocol (3P-ID-SPK) protocol of Lim and Paterson (2006). We then refuted its existing heuristic security arguments by revealing a previously unpublished flaw in the protocol with SHVT. More importantly, we describe how our formal

[5] A further discussion on Protocol 2 and the basis for our repair procedures is presented in Appendix B.

framework can be used to repair protocols that were found to be insecure against certain attacks.

Analysis and repair of another case study protocol, the conference key agreement protocol of Boyd and González Nieto (2003), is presented in Appendix C. The same flaw revealed in Asiacrypt 2005 on the protocol (Choo *et al.*, 2005b) was also found by our tool. Using the repair technique described in Section 3.4, the same fix proposed in our earlier work (Choo *et al.*, 2005b) is also derived.

Appendix A. Examples of Some APA Specifications

Fig. 4 describes examples of some basic types used in our specifications. Note that we only present sufficient background on the syntax of SHVT to specify and analyse the case study protocol and refer interested reader to the SHVT manual (SHVTManual, 2004) for a more comprehensive read. Let A, B, and C denote clients and S denote a sever.

The initial state of both Protocol 1 is shown in Fig. 5 of Appendix A. For simplicity, we restrict the analyses to only four players, A, B, C, S, and a malicious adversary.

Appendix B. Informal Discussions on Protocol 2

We remark that Protocol 2 is not proven secure, as this is not the purpose of our work. Our work here is to show that we are able to reveal flaws, in particularly previously unknown flaws, in protocols and repair the revealed flaws using an automated model checker tool.

We now present an informal discussion to provide a better insight into the protocol failures and the basis for our repair procedures.

In Protocol 1, the identities of the protocol participants are not included within the encrypted messages, the signatures generated, and the MAC digests generated. This effectively allows a malicious adversary, $\mathcal{A}$, to intercept messages and cause confusion on who are the actual partipants of this protocol session. Hence, by including the identities

Examples of some basic types	
Agents	::= set of all the principals denoted as A, B, S and $\mathcal{A}$ (denoted as E)
Keywords	::= set of all the data items in the messages (e.g., ephemeral secrets, session keys)
A_State	::= A's internal state
A_Keys	::= set of A's public and private keys
B_State	::= B's internal state
B_Keys	::= set of B's public and private keys
C_State	::= C's internal state
C_Keys	::= set of C's public and private keys
E_State	::= E's internal state
S_State	::= S's internal state
S_Keys	::= set of S's public and private keys
Accepted	::= set of all oracles who had accepted and this information is visible to the adversary

Fig. 4. Examples of basic types.

A_State:=	{(B,agent), (start,B), (S,server)};	A starts a protocol session run with B (indicated by the keyword $start$).
B_State:=	{(A,agent), (respond,A), (S,server)};	B knows that that A is an agent and can respond to a protocol run initiated by A (indicated by the keyword $respond$).
C_State:=	{(A,agent), (B,agent), (S,server)};	C knows that both A and B are agents.
E_State:=	{(A,agent), (B,agent), (C,agent), (S,server)};	The adversary, $\mathcal{A}$, (denoted by E) knows A, B, and C are agents of this protocol.
A_Keys:=	(S,pwdA);	A has a secret password that is being shared with the server, S.
B_Keys:=	(S,pwdB);	B has a secret password that is being shared with the server, S.
C_Keys:=	(S,pwdC);	C has a secret password that is being shared with the server, S.
S_Keys:=	(A,pwdA), (B,pwdB), (C,pwdC);	S has secret passwords shared with A, B, and C respectively.
Network:=	{};	Network is initially empty, hence, denoted by an empty set.
Transcript:=	{};	Transcript is initially empty, hence, denoted by an empty set.

Fig. 5. Initial state of Protocol 1.

of the protocol participants within the encrypted messages, the signatures generated, and the MAC digests generated, we should be able to thawrt Attack 1.

Moreover, as we pointed out in an earlier work (Choo *et al.*, 2005b), including the unique session identifiers (SIDs) comprising the identities of the participants and their roles within the session key deriviation function ensures that session keys will be fresh and 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.

Appendix C. Another Case Study – Boyd–González Nieto Protocol

The conference key agreement protocol (Boyd *et al.*, 2003) described in Protocol 3 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. 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 cryptographic hash function; and SK_U denotes the session key accepted by U. Protocol 3 involves a set of p users, $\mathcal{U} = \{U_1, U_2, \ldots, U_p\}$. The session identifier (SID) in Protocol 3 is defined to be the concatenation of messages received and sent.

1 and 2. The initiator, U_1, randomly selects a k-bit challenge N_1, where k is the security parameter. U_1 then 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 of Protocol 3.

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 $1 < i \leqslant p$
3. $U_i \rightarrow *$: U_i, N_i

The session key is $SK_{U_i} = \mathcal{H}(N_1||N_2||\ldots||N_p)$.

Protocol 3. Boyd–González Nieto conference key agreement protocol.

3. 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 $SK_{U_i} = \mathcal{H}(N_1||N_2||\ldots||N_p)$.

C.1. *Protocol Specification*

For simplicity and run-time efficiency in our specification and analysis, we modified Protocol 3 to a two-party protocol. Although Protocol 3 is group-based, it is easily modified to a two-party setting as explained by Jeong, Katz, and Lee (Jeong *et al.*, 2004). The modified protocol is described in Protocol 4.

1. $U_1 \rightarrow U_2$: $\mathcal{U} = \{U_1, U_2\}, \sigma_{d_{U_1}}(\mathcal{U}, \{N_1\}_{e_{U_2}})$
2. $U_1 \rightarrow U_2$: $\{N_1\}_{e_{U_2}}$
3. $U_2 \rightarrow U_1$: U_2, N_2

The session key is $SK_{U_1} = \mathcal{H}(N_1||N_2) = SK_{U_2}$.

Protocol 4. Boyd–González Nieto conference key agreement protocol in a two-party setting.

C.2. *Protocol Analysis*

State space analysis in SHVT reveals that both goal state 1 of Definition 3 are violated. The attack sequence is described in Attack 2[6]. Let $\mathcal{U} = \{U_1, U_2\}$ and $\mathcal{U}_{\mathcal{A}} = \{\mathcal{A}, U_2\}$ denote two different sessions and $\mathcal{A}_U$ denotes $\mathcal{A}$ impersonating some entity U.

The adversary, $\mathcal{A}$, asks a Corrupt query to C where $C \neq U_1, U_2$ – *static corruption*. $\mathcal{A}$ now runs as C.

1. $U_1 \rightarrow U_2$: $\mathcal{U} = \{U_1, U_2\}, \sigma_{d_{U_1}}(\mathcal{U}, \{N_1\}_{e_{U_2}})$
2. $U_1 \rightarrow U_2$: $\{N_1\}_{e_{U_2}}$

Both messages intercepted by $\mathcal{A}$ (i.e., C).

1(C). $C \rightarrow U_2$: $\mathcal{U}_C = \{C, U_2\}, \sigma_{d_C}(\mathcal{U}_C, \{N_1\}_{e_{U_2}})$
2(C). $C \rightarrow U_2$: $\{N_1\}_{e_{U_2}}$
3. $U_2 \rightarrow C$: U_2, N_2
3(C). $C_{U_2} \rightarrow U_1$: U_2, N_2

The session key is $SK_{U_1} = \mathcal{H}(N_1||N_2) = SK_{U_2}$.

Attack 2. Unknown key share attack on Protocol 4.

[6]We remark that Attack 2 is similar to that revealed in an earlier work (Choo *et al.*, 2005b) except that the attack we revealed in the earlier work is in a group-based setting.

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

Static Corruption. The adversary, $\mathcal{A}$, asks a Corrupt query to C where $C \neq U_1, U_2$ – *static corruption*. $\mathcal{A}$ now runs as C.

1 and 2. The initiator, U_1, encrypts N_1 under the public key of U_2 (i.e., $\mathcal{U} \setminus U_1$), signs the encrypted nonces $\{N_1\}_{e_{U_2}}$ together with $\mathcal{U}$, and broadcasts these messages in protocol flows 1 and 2.

1(C) and 2(C). C intercepts the broadcast messages sent by U_1; that is, the broadcast messages sent by U_1 never reach U_2.

- C then signs the intercepted encrypted nonces $\{N_1\}_{e_{U_2}}$ together with $\mathcal{U}_{\mathcal{A}}$ (instead of $\mathcal{U}$) under $\mathcal{A}$'s signing key.
- C now acts as the **initiator** in a **different session** and broadcasts these messages in protocol flows 1 and 2.

3. U_2 upon receiving the broadcast messages, replies to $\mathcal{A}$ with the identity and random nonce.

C impersonates U_2 and forwards the messages from U_2 to U_1.

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 1 describes the internal states of players U_1 and U_2 at the end of Attack 2. We observe that U_1 is not partnered with either U_2 according to Definition 1, since U_1 does not have matching SIDs or agreeing PIDs. Such an attack is also termed the key replicating attack as described in Definition 4 – goal state 1 of Definition 3. 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_1 believes that the session key SK_{U_1} is being shared with U_2, but U_2 believes the key $SK_{U_2} = \mathcal{H}(N_1||N_2) = SK_{U_1}$ is being shared with C, when in fact, the key is being shared between U_1 and U_2. However, $SK_{U_1} = SK_{U_2} = SK_{U_3} = \mathcal{H}(N_1||N_2)$. Although the adversary does not know the value of the session key as $\mathcal{A}$ does not know the value of N_1, $\mathcal{A}$ is able to send a Reveal query to the session associated with U_2 and obtain $SK_{U_2} = \mathcal{H}(N_1||N_2)$, which has the same value as SK_{U_1}. Protocol 4 (and therefore, Protocol 3) is not secure in the BR93 model since $\mathcal{A}$ is able to obtain the fresh session key of the initiator U_1 by revealing non-partner oracle of U_1, namely U_2.

Table 1

Internal states of players U_1, U_2, and U_3

U	sid_U	pid_U
U_1	$\mathcal{U}, \sigma_{d_{U_1}}(\mathcal{U}, \{N_1\}_{e_{U_2}}), \{N_1\}_{e_{U_2}}, U_2, N_2$	U_2
U_2	$\mathcal{U}_C, \sigma_{d_C}(\mathcal{U}_C, \{N_1\}_{e_{U_2}}), \{N_1\}_{e_{U_2}}, U_2, N_2,$	C

C.3. *A Repaired Protocol*

Using our approach outlined in Section 3.4, we obtained the repaired protocol described by Protocol 5.

1. $U_1 \rightarrow U_2$: $\mathcal{U} = \{U_1, U_2\}, \sigma_{d_{U1}}(\mathcal{U}, \{N_1, U_1, U_2\}_{K_{U_2}})$
2. $U_1 \rightarrow U_2$: $\{N_1, U_1, U_2\}_{e_{U_2}}$
3. $U_2 \rightarrow U_1$: U_2, N_2
$sid = U_1 \|\| U_2 \|\| \mathcal{U} = \{U_1, U_2\} \|\| \sigma_{d_{U1}}(\mathcal{U}, \{N_1, U_1, U_2\}_{K_{U_2}}) \|\| \{N_1, U_1, U_2\}_{e_{U_2}} \|\| N_2$ The session key is $SK_{U_i} = \mathcal{H}(sid \|\| N_1 \|\| N_2)$.

Protocol 5. An improved Protocol 4.

There are some minor differences between Protocol 5 and the fix proposed in our earlier work (Choo *et al.*, 2005b). In the latter, the identity of the recipient is not included in the encrypted message.

Acknowledgments

The author would like to thank the anonymous reviewers for their constructive feedback. Despite their invaluable assistance, any errors remaining in this paper are solely attributed to the author.

References

SHVT Manual (2004). *Technical Report*. Fraunhofer Institute for Secure Telecooperation, Darmstadt, Germany.

Abadi, M., and P. Rogaway (2002). Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, **15**(2), 103–127.

Abdalla, M., E. Bresson, O. Chevassut and D. Pointcheval (2006). Password-based group key exchange in a constant number of rounds. In *PKC 2006*, *LNCS*, vol. 3958. Springer. pp. 427–442.

Backes, M. (2004a). A cryptographically sound dolev-yao style security proof of the Needham–Schroeder–Lowe public-key protocol. *IEEE Journal on Selected Areas in Communications*, **22**(10), 2075–2086.

Backes, M. (2004b). A cryptographically sound Dolev–Yao style security proof of the Otway–Rees protocol. In *ESORICS 2004*, *LNCS*, vol. 3193. Springer. pp. 89–108.

Backes, M., and C. Jacobi (2003). Cryptographically sound and machine-assisted verification of security protocols. In *STACS 2003*, *LNCS*, vol. 2607. Springer. pp. 310–329.

Bellare, M., D. Pointcheval and P. Rogaway (2000). Authenticated key exchange secure against dictionary attacks. In *EUROCRYPT 2000*, *LNCS*, vol. 1807. Springer. pp. 139–155.

Bellare, M., and P. Rogaway (1993). Entity authentication and key distribution. In *CRYPTO 1993*, *LNCS*, vol. 773. Springer. pp. 110–125.

Blanchet, B. (2006). A computationally sound mechanized prover for security protocols. In *IEEE S&P 2006*. IEEE Computer Society Press. To appear.
(Extended version available from `http://eprint.iacr.org/2005/401`).

Blanchet, B., and D. Pointcheval (2006). Automated security proofs with sequences of games. In *CRYPTO 2006*, *LNCS*. Springer. To appear.
(Extended version available from `http://eprint.iacr.org/2006/069`).

Boyd, C., and J.M. González Nieto (2003). Round-optimal contributory conference key agreement. In *PKC 2003*, *LNCS*, vol. 2567. Springer. pp. 161–174.

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

Canetti, R. (2000). *Universally Composable Security*: *A New Paradigm for Cryptographic Protocols*. Cryptology ePrint Archive, Report 2000/067. http://eprint.iacr.org/2000/067/.

Canetti, R., and M. Fischlin (2001). Universally composable commitments. In *CRYPTO 2001*, *LNCS*, vol. 2139. Springer. pp. 19–40.

Canetti, R., and H. Krawczyk (2001). Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT 2001*, *LNCS*, vol. 2045. Springer. pp. 453–474. (Extended version available from http://eprint.iacr.org/2001/040/).

Cervesato, I., N. Durgin, P.D. Lincoln, J.C. Mitchell and A. Scedrov (1999). A meta-notation for protocol analysis. In *CSFW 1999*. IEEE Computer Society Press. pp. 55–71.

Choo, K.-K.R. (2006a). Refuting security proofs for tripartite key exchange with model checker in planning problem setting. In *CSFW 2006*. IEEE Computer Society Press. pp. 297–308.

Choo, K.-K.R. (2006b). *Key Establishment: Proofs and Refutations*. Ph.D. Thesis. Information Security Institute, Queensland University of Technology.

Choo, K.-K.R., C. Boyd, Y. Hitchcock and G. Maitland (2004). Complementing computational protocol analysis with formal specifications. In *IFIP FAST 2004*, *IFIP Series*, vol. 173. Springer. pp. 129–144.

Choo, K.-K.R., C. Boyd and Y. Hitchcock (2005a). On session key construction in provably secure protocols. In E. Dawson and S. Vaudenay (Eds.), *MYCRYPT 2005*, *LNCS*, vol. 3715. Springer. pp. 116–131.

Choo, K.-K.R., C. Boyd and Y. Hitchcock (2005b). Errors in computational complexity proofs for protocols. In *ASIACRYPT 2005*, *LNCS*, vol. 3788. Springer. pp. 624–643.

hoo, K.-K.R. (2007). Proof of revised Yahalom protocol in the Bellare and Rogaway (1993) model. *The Computer Journal*, **50**(5), 591–601.

Diffie, W., and M. Hellman (1976). New directions in cryptography. *IEEE Transaction on Information Theory*, **22**, 644–654.

Dolev, D., and A.C. Yao (1983). On the security of public key protocols. *IEEE Transaction of Information Technology*, **29**(2), 198–208.

Goldwasser, S., and S. Micali (1984). Probabilisitic encryption. *Journal of Computer and System Sciences*, **28**(3), 270–299. (Available from http://theory.lcs.mit.edu/ joanne/shafi-pubs.html).

Jakobsson, M., and D. Pointcheval (2001). Mutual authentication and key exchange protocol for low power devices. In *Financial Cryptography*, *LNCS*, vol. 2339. Springer. pp. 169–186.

Jeong, I.R., J. Katz and D.H. Lee (2004). One-round protocols for two-party authenticated key exchange. In *ACNS 2004*, *LNCS*, vol. 3089. Springer. pp. 220–232.

Krawczyk, H. (2005). HMQV: A high-performance secure Diffie–Hellman protocol. In *CRYPTO 2005*, *LNCS*, vol. 3621. Springer. pp. 546–566.
(Extended version available from http://eprint.iacr.org/2005/176/).

Kudla, C., and K.G. Paterson (2005). Modular security proofs for key agreement protocols. In *ASIACRYPT 2005*, *LNCS*, vol. 3788. Springer. pp. 549–569.

Lifschitz, V. (1999). Answer set planning. In *LPNMR 1999*, *LNCS*, vol. 1730. Springer. pp. 373–374. (Full version available from http://www.cs.utexas.edu/ vl/).

Lim, H.W., and K.G. Paterson (2006). Secret public key protocols revisited. In *Security Protocols Workshop 2006*, LNCS. Springer. To appear. (Available from http://www.isg.rhul.ac.uk/ hwlim/).

Lowe, G. (1996). Breaking and fixing the Needham–Schroeder public key protocol using FDR. In *TACAS 1996*, *LNCS*, vol. 1055. Springer. pp. 147–166.

Meadows, C. (1999). Analysis of the internet key exchange using the NRL analyzer. In *IEEE S&P 1999*. IEEE Computer Society Press. pp. 216–231.

Meadows, C. (2003). Formal methods for cryptographic protocol analysis: emerging issues and trends. *IEEE Journal on Selected Area in Communications*, **21**(1), 44–54.

Menezes, A.J., P.C. van Oorschot and S.A. Vanstone (1997). *Handbook of Applied Cryptography*. The CRC Press Series On Discrete Mathematics And its Applications. CRC Press.

Merkle, R.C. (1978). Secure communications over insecure channels. *Communications of the ACM*, **21**(4), 294–299.

NIST (2006). *Special Publication* (SP 800-56A) – *Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography – of National Institute of Standards and Technology, Computer Security Division*. (Available from http://csrc.nist.gov/publications/nistpubs/index.html). Last updated on 03 May 2006.

Ochsenschläger, P., J. Repp, R. Rieke and U. Nitsche (1998). The SH-verification tool – Abstraction-based verification of co-operating systems. *Journal of Formal Aspects of Computing*, 381–404.

Phan, R.C.-W., and B.-M. Goi (2006). Cryptanalysis of two provably secure cross-realm C2C-PAKE protocols. In R. Barua and T. Lange (Eds.), *INDOCRYPT 2006*, *LNCS*, vol. 4329. Springer. pp. 104–117.

Russell, S., and P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Shoup, V. (2001). OAEP reconsidered. In *CRYPTO 2001*, *LNCS*, vol. 2139. Springer. pp. 239–259.

Song, D. (1999). Athena – An automatic checker for security protocol analysis. In *CSFW 1999*. IEEE Computer Society Press.

Song, D., S. Berezin and A. Perrig (2001). Athena – A novel approach to efficient automatic security protocol analysis. *Journal of Computer Security*, **9**(1/2), 47–74.

Weld, D.S. (1999). Recent advances in AI planning. *AI Magazine*, **20**(2), 93–123.

Wolf, S. (1999). *Information-Theoretically and Computationally Secure Key Agreement in Cryptography*. Ph.D. Thesis. ETH Zurich, Swiss Federal Institute of Technology Zurich. Available from `http://www.iro.umontreal.ca/ wolf/papers.html`.

Wong, D.S., and A.H. Chan (2001). Efficient and mutually authenticated key exchange for low power computing devices. In *ASIACRYPT 2001*, *LNCS*, vol. 2248. Springer. pp. 172–289.

K.-K.R. Choo, currently a research analyst in high tech crime and anti-money laundering with the Australian Institute of Criminology, received his PhD in information security from Queensland University of Technology. His research has been widely cited, including a citation in a special publication of the U.S. National Institute of Standards and Technology (NIST). He was awarded the "Best Student Paper Award" at the 10th Australasian Conference on Information Security and Privacy in 2005, and the "2006 FIT Executive Dean's outstanding thesis commendation" in 2007. He has also served on the program committee for several international conferences and as a reviewer for several international conferences and journals including *ASIACRYPT 2005 & 2006*, *ACM Computing Reviews*, *IET Information Security and Asian Journal of Criminology*.

Integruota protokolo analizės ir taisymo schema: Bellare-Rogaway modelis + planavimas + modelis tikrintojas

Kim-Kwang Raymond CHOO

Modifikuota Bellare ir Rogaway (1993) konkuruojančio modelio versija yra užkoduota asinchroninio rezultato automatu. Modelio tikrinimo įrankis "Simple Homomorphism Verification Tool" yra panaudotas automato būsenų erdvės analizei. Trišalis tapatybe paremtas slapto viešo rakto protokolas, pasiūlytas Lim ir Paterson (2006), yra pasirinktas atvejo analizei. Mes paneigiame jo saugumą atskleisdami anksčiau nepublikuotą trūkumą. Tada parodoma, kaip pasiūlytas būdas gali ištaisyti protokolą.