Provably Secure Key Exchange: An Engineering Approach

Yiu Shing Terry Tin **Colin Boyd** **Juanma Gonzalez Nieto**

Information Security Research Centre
Queensland University of Technology,
PO Box 2434, Brisbane, QLD 4001, Australia
Emails: {ttin,boyd,juanma}@isrc.qut.edu.au

Abstract

We promote an engineering approach to design of provably secure key exchange protocols. Using the model of Canetti and Krawczyk we present a systematic method to arrive at efficient and practical protocols with proven security and illustrate its use with existing building blocks. We further show a dual approach which allows protocols with known features to be 'reverse engineered', thereby allowing easier security proofs and providing new building blocks for future designs.

Keywords: Provable security, key exchange, secure protocols.

1 Introduction

Protocols for key exchange provide the basis for secure communications and so it is important that they are designed correctly. However, despite their relative simplicity as measured by the number and size of messages, experience has shown that informal approaches to protocol design frequently result in flawed outcomes. Moreover, the modern approach to cryptography demands more than an absence of known attacks; a proof that security is equivalent to some well understood problem is increasingly regarded as essential.

Proofs for key exchange protocols were first provided by Bellare & Rogaway (1993). Initially this only covered a two-party example, but later the same authors Bellare & Rogaway (1995) provided a proof in the three party (server based) scenario, and this was subsequently extended to a variety of other settings. A feature of all these proofs is that they are rather long and difficult to read for the non-specialist. Perhaps more of a problem is that a small change in the protocol may make the proof invalid and so it is very difficult to re-use proofs or design protocols in an incremental way.

Bellare, Canetti & Krawczyk (1998) introduced the idea of a modular approach to provably secure protocols. However, there were certain problems with their original security definitions and Canetti & Krawczyk (2001) re-used much of the earlier model with a new definition of security. The approach relies on proving protocols secure in an *ideal world* model, and then using a secure transformation to move them into a *real world* model. Although there are a few examples in these papers, they are mainly concerned with explaining the formal models and proving that the necessary building blocks are secure. One of the aims of this paper is to make the approach of Canetti and Krawczyk more accessible by giving more and detailed examples and explaining how it can be used in a systematic way. Because it is possible to use the approach without a detailed knowledge of the formal models and proofs, we believe that it is suitable for applications by practitioners.

The main contributions of this paper are the following.

- Detailed examples of use of Canetti-Krawczyk approach to design of practical and secure protocols.
- A new method of using the Canetti-Krawczyk model by inverting protocols with desirable properties, to allow a simplified proof.
- A new proven secure protocol in the ideal model which can be used as a new building block for protocol design.

The remainder of this paper is structured as follows. In section 2 we cover the background material on the Canetti-Krawczyk model. Section 3 gives a detailed example of use of the model with existing building blocks. Section 4 explains how to use an existing protocol to extract a simple protocol in the ideal world including an example which leads to a new proven secure protocol in the ideal world. Section 5 describes other existing building blocks which can be used for other protocol designs. The appendix contains a more formal description of the Canetti-Krawzyk model and the proof of security for our new protocol.

2 Background

This section provides a high level description of the modular approach to the design of AKE protocols Bellare et al. (1998) and Canetti & Krawczyk (2001). More details on some elements of the model appear in Appendix A.1.

2.1 Canetti-Krawczyk Approach to Protocol Construction

The model defines protocol principals who may run multiple sessions of the protocol. A powerful adversary attempts to break the protocol by interacting with the principals. In addition to controlling all the communications the adversary is able to *corrupt* any principal and chooses its long-term key (this models insider attacks). The adversary may also *reveal* any accepted session keys. The adversary must be efficient in the sense of being a probabilistic polynomial time algorithm.

Definition 1 (Informal) *An AKE protocol is called* SK-secure *if the following hold.*

1. *If two uncorrupted parties complete matching sessions, then they both accept the same key.*
2. *Suppose a session key is agreed between two uncorrupted parties and has not been revealed by the adversary. Then the adversary cannot distinguish the key from a random string with probability significantly more than 1/2.*

Two adversarial models are defined. The first can be considered as an ideal world in which messages are authenticated magically. The second can be considered as the real world in which we want our real protocols to be proven secure. However, in order to modularise the process protocols are first proven secure in the ideal world and then translated into the real world.

The authenticated-links adversarial model This model is also known as the AM. In this model the adversary is able to invoke protocol runs, masquerade as protocol principals, and find used session keys. Although the adversary is quite powerful it is unable to fabricate or replay messages which appear to come from uncorrupted parties.

The unauthenticated-links adversarial model This model is also known as the UM. In this model the adversary can do everything that it can do in the AM, but can also replay and fabricate messages using anything it can calculate.

2.2 Authenticators

An MT-authenticator (message transmission authenticator) is an important mechanism in the modularisation of the process. It is used to obtain SK-secure protocols in the UM by emulating individual messages of AM SK-secure protocols in the UM and thus transforming AM SK-secure protocols to SK-secure protocols in the UM. The following gives an informal definition:

Definition 2 (Informal) *An* authenticator *is a protocol translator that takes an SK-secure protocol in the AM to a SK-secure protocol in the UM. An* MT-authenticator *is a protocol translator that is applied to each separate message sent in the AM.*

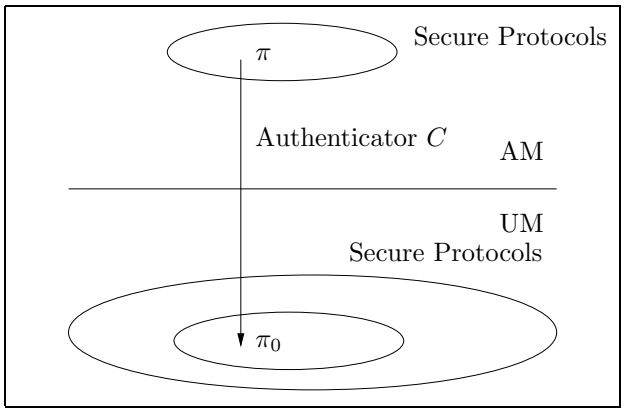

Figure 1: Applying authenticator in Canetti-Krawczyk model

Figure 1 illustrates how an authenticator provides a connection between a protocol π in the AM and a protocol π_0 in the UM. Each flow of the protocol in the AM will become a multi-flow sub-protocol in the UM. The resultant secure protocol can be simply a concatenation of all sub-protocols. However such a simple approach generates an inefficient protocol and is of limited interest to us despite its proven security. Fortunately we can collapse flows of the sub-protocols with only one assumption: *the proof does not rely on the sequence of the internal flows of a particular MT-authenticator.* The process is straightforward. It simply combines flows which are sent to the same direction.

2.3 Optimisation

Suppose that a 3-move MT-authenticator is applied to a 2-move protocol in the AM, then the resultant protocol in the UM is the concatenation of 2 sub-protocols which are denoted as λ_I and λ_R respectively. Let message m be any message and m_I and m_R be the first and second protocol message of the 2-move protocol in the AM respectively. The message m is formed to include at least a session identifier (ID) together with other information complying to the protocol specification. Each flow (λ_{Ii} and λ_{Ri} where $i \in \{1,2,3\}$) of λ_I and λ_R contains a message $m \in \{m_I, m_R\}$. Figures 2 and 3 show the direction of messages sent in the sub-protocols.

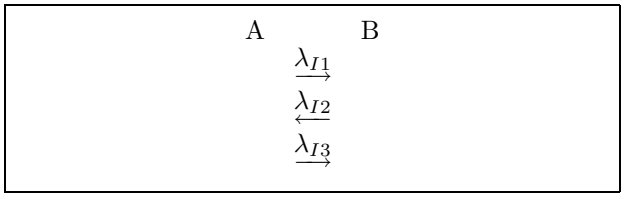

Figure 2: Sub-protocol One (λ_I)

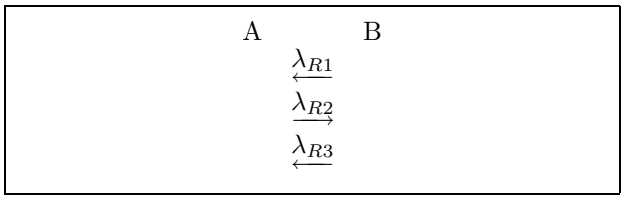

Figure 3: Sub-protocol Two (λ_R)

The final protocol λ_{IR} is illustrated in Figure 4. The first message of λ_{IR} is a combination of λ_{I1} together with λ_{R2}. The second message of λ_{IR} is formed by λ_{I2} together with λ_{R3}. The third message of λ_{IR} is copied from λ_{I3}. The missing message λ_{R1} is implied in the second message of λ_{IR} so that can be ignored. As all messages of λ_I and λ_R include either m_I or m_R, the missing message $\lambda_{R1} = m_R$ is implied in λ_{R2} and λ_{R3} and is sent to A in the second message of λ_{IR} by B.

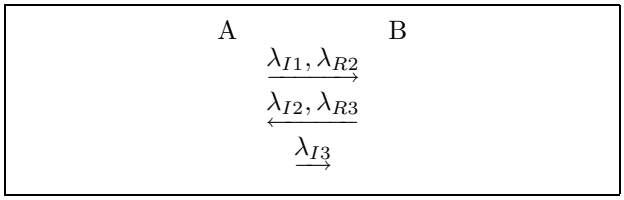

Figure 4: Final Protocol (λ_{IR})

Remark. Sub-protocols cannot be collapsed using the "mechanism" described above without any analysis as the process is critical to security. It is important to understand the type of message flows of any MT-authenticator. In particular, the elements for forming

the message flows such as nonce, identity and secret data. The final protocol must satisfy the format requirements of the MT-authenticators, otherwise it is likely to be insecure even if the assumption stands. More precisely, it can be done via the following steps:

1. Apply the mechanism without analysis:
   This step involves mechanical work which simply collapses message flows to generate a 3-move protocol from 2 3-move sub-protocols;
2. Identify redundant and replaceable elements of message flows:
   This step is critical and requires understanding and analysis of the authenticator. If A chooses x at random, it can be used as a nonce generated by A and as a challenge to B. Some information is not required and can be marked as redundant and removed later;
3. Substitute elements of message flows:
   This step replaces redundant elements which are often the nonces in the protocol used for freshness. It is likely that some elements will become redundant although they were necessary in the previous step;
4. Remove redundant elements of message flows to finalise the protocol:
   This step simply removes the redundant elements marked in the previous two steps;
5. Inspect the final protocol against the format requirements of the MT-authenticator:
   This step is a verification process which checks the correctness of the final protocol.

2.4 Model Summary

We can now summarise the Canetti-Krawczyk approach in the following three steps.

1. Design a basic protocol and prove that it is SK-secure in the AM.
2. Design an MT-authenticator and prove that it is a valid authenticator.
3. Apply the MT-authenticator to the basic protocol to produce a protocol that is automatically secure in the UM.
4. As necessary, re-order and re-use message components to optimise the resulting protocol.

3 Usage of MT-authenticator

In this section we discuss the importance of the session identifier in m and then demonstrate the usage of MT-authenticators in the development of provably secure protocols in detail. In particular, we apply the signature based MT-authenticator of Bellare et al. (1998) to the protocol named ENC of Canetti & Krawczyk (2001) which is proven secure in the AM.

3.1 Importance of Session Identifier

Although the session identifier may not be explicitly visible in a MT-authenticator, it is included as part of the message m. We stress that the session identifier is fundamentally critical to the security of the model. Its duty is to assist two parties to maintain a particular session among all concurrent sessions. The value of the session identifier must be unique to the parties where communications take place.

The choice of the session identifier can be determined by initiators or responders solely, however this approach allows replay of messages. A countermeasure is that each party keeps track of all the messages received in the past as well as maintains protocol states across sessions in order to accept only new messages. Clearly, impracticality is indicated.

A secure yet practical implementation for uniqueness of the session identifier is to enforce contributions from both parties. Each party then knows that the session identifier is fresh and unique as their individual inputs form part of it. For generality we do not specify how it is achieved. In other words whether it is a hash or simply a concatenation of the inputs does not matter.

3.2 A Signature Based MT-authenticator

The signature based MT-authenticator is shown in Figure 5: it is assumed that each party has its private key and knows the authentic public keys of other parties. The signature scheme in use is assumed to be secure against chosen message attacks. Let m be any message, N_B be a nonce chosen by B and $Sig_A(\cdot)$ be the signature of some arbitrary values generated by A. Although the message m is shown in every flow, it may actually occur in the first flow or even in the last flow only. If more than one copy of the message m exists in the authenticated version, some copies can be eliminated but a unique session identifier must be used to assure that the communication takes place in the same session. The security proof of the authenticator will fail if any of the three elements signed for generating the signature in the last message is omitted.

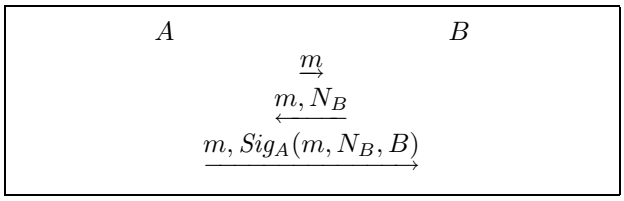

Figure 5: Signature Based MT-authenticator

The protocol ENC is provably secure in the AM with the assumption that the encryption scheme is secure against chosen ciphertext attack and that the hash function acts as a pseudorandom function. Its security proof is presented by Canetti & Krawczyk (2001). Each party A possesses a pair (PK_A, SK_A) of encryption and decryption keys where PK_A is public and all parties have the public keys of other parties. The pseudorandom function is denoted by $\{f_a\}_{a\in\{0,1\}^b}$ where b denotes the security parameter.

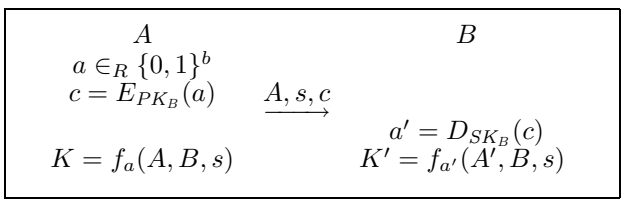

Figure 6: The Protocol ENC

3.3 The Authenticated Protocol ENCUM

Since the protocol ENC is a single message protocol in the AM, applying the signature based MT-authenticator to the protocol is straightforward and no collapsing needs to be done. Simply substitute m in the authenticator by the message (A, s, c), then the protocol is authenticated and is provably secure in

the UM. This is related to step 1 of the optimisation process and requires no analysis.

We then need to follow steps 2 to 5 of the optimisation process to obtain the resultant protocol. The only replaceable element of the protocol is s. The only erasable element is A in the third message as we have already presented the identity of A in the first message. We now replace the session id s with inputs from both parties and is replaced by (c, N_B) where c and N_B denote contributions from A and B respectively. Furthermore, we remove the identity of A in the third message and finally, we inspect the correctness of the protocol against the format requirements of the signature based MT-authenticator. The authenticated and simplified protocol ENCUM is shown in Figure 7 which is an optimised version of the result after applying the MT-authenticator to the protocol ENC.

4 Extraction of SK-secure Protocol

Up until now, we have demonstrated the modular approach for developing protocols with provable security by applying MT-authenticators to SK-secure protocols in the AM to obtain SK-secure protocols in the UM. One way to be more productive (in terms of number of SK-secure protocols in the UM) is by increasing the number of SK-secure protocols in the AM in our collection.

On the one hand, we can design our own protocols which are SK-secure in the AM from scratch. On the other hand, we can extract SK-secure protocols in the AM from some existing protocols which are preferably proven secure in the real world. One of the major advantages of the latter method is that we actually know the outcome protocol in the UM. It is ideal in scenarios where a protocol is found to be well suited for some task, then one can extract the SK-secure protocol in the AM by 'reverse engineering' the authenticator, and put it in the collection of building blocks for new provably secure protocols in the UM.

We demonstrate the extraction methodology by providing an example where we gain a new protocol which is SK-secure in the AM from the protocol named 3PKDUM which was proven secure by Bellare & Rogaway (1995). For the sake of simplicity, the message routing of the first two messages has been modified from $A \rightarrow B \rightarrow S$ to $A \rightarrow S \leftarrow B$ as illustrated in Figure 8. Bellare & Rogaway (1995) suggested that "*protocols for alternative connectivity model ought differ only in their message routing*", but not their security.

The protocol involves three parties (see Bellare & Rogaway (1993) for a two-party protocol), a trusted server S and two players A and B. It is a key distribution protocol where the session key is generated by the server S using a session key generator Sn with input 1^k where k denotes the security parameter. Party A shares a pair of symmetric keys (SK_{AS}, MK_{AS}) [resp. (SK_{BS}, MK_{BS}) for party B] with the server S for encryption and message authentication. These keys are generated by a long term key generator $LKG(1^k, r)$ which returns a uniformly distributed $2k$-bit string value where r is a random number. Let w be the length in bits where random numbers r_A and r_B are chosen. Note that r_A and r_B are used to provide semantic security (Bellare & Rogaway 1995) and we preserve them here so that our result matches exactly as Figure 8.

4.1 Extraction Mechanism

The first step towards extraction of AM protocols out of existing protocols is to understand the protocol properly in terms of service provision. More precisely, we want to identify the part of the protocol which is responsible for confidentiality as well as the part which provides authentication. We recall that the model (Canetti & Krawczyk 2001) separates the treatment for confidentiality and authentication.

After identifying the type of services provided by the protocol, we also need to be familiar with the ways that how those services are carried out, namely using a secure encryption scheme for session key protection. This step intends to be more directive in identifying the type of authenticators that may be used for emulating AM SK-secure protocols in the UM.

The last step is to remove unnecessary elements of the protocol such as random nonces and elements providing authentication which is not dealt with in the AM.

As an example, we identify that the protocol 3PKDUM provides both confidentiality and authentication services. In the flows from the server S to A and B, $E_{SK_{AS}}(\cdot)$ and $E_{SK_{BS}}(\cdot)$ provide confidentiality services while $MAC_{MK_{AS}}(\cdot)$ and $MAC_{MK_{BS}}(\cdot)$ provide authentication services. The former aims to protect secrecy of the session key σ and prevent leakage of session key information to the adversary and the latter intends to provide assurance of party identity as well as message authentication.

We then notice that the protocol uses symmetric key encryption scheme which requires a long term secret key to be distributed securely before the protocol execution. The MAC function serves as an ordinary keyed MAC. The use of the MAC function suggests the use of a MAC based authenticator. We describe such an authenticator of Canetti & Krawczyk (2001) in Figure 9.

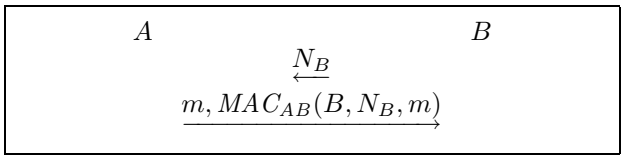

Figure 9: MAC Based MT-authenticator

The removal of redundant elements is easy and straightforward. Almost at all times, random nonces can be safely deleted as random nonces are likely to be part of the authenticators. We then remove operations related to MAC as authentication is assumed in the AM.

4.2 New SK-secure Protocol in the AM

Following the approach described above, we extract a protocol which is a candidate to be SK-secure in the AM and is illustrated in Figure 10. Without proving its security, we verify the correctness of the protocol structure by applying the MAC based authenticator to the protocol 3PKD. The resultant protocol should be similar to protocol 3PKDUM. If it is not the case, the protocol may need to be fine tuned, or it may serve just as well as the original depending on the requirements.

We note that the resultant protocol after applying the MAC based authenticator to protocol 3PKD needs to be adjusted so that the message routing matches to the one in protocol 3PKDUM. The partner identity is added to get the identical result comparing with 3PKDUM in the flows from S to A and B.

Once the structure is confirmed, we need to prove that it is SK-secure in the AM. This critical step requires knowledge regarding the operation of the Canetti-Krawczyk model. However, even though this

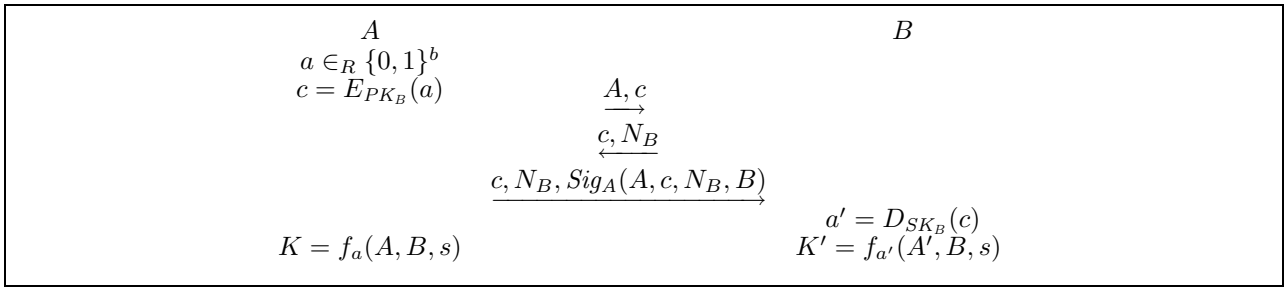

Figure 7: The Authenticated Protocol ENCUM

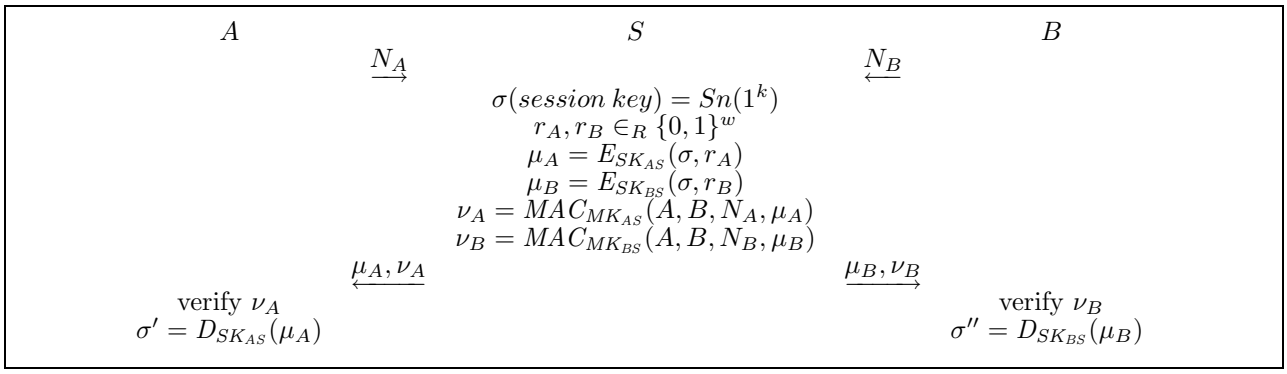

Figure 8: The Protocol 3PKDUM

step may not be straightforward for many practitioners, two important factors make the effort more attractive. Firstly, the effort involved is greatly reduced in comparison with the full proof in the UM, such as the one that was carried out by Bellare and Rogaway. Secondly, once proven secure in the AM the protocol may be re-used with other authenticators, thereby helping to build up the library of proven secure protocols.

The security of the protocol 3PKD can be reduced to the security of the encryption scheme which is used to protect the session key. Formally, we state it in Theorem 1. The proof of the theorem appears in Appendix A.2.

Theorem 1 *If the long and short term keys (SK_i, σ) are uniformly distributed, the symmetric encryption scheme (LKG, E, D) is semantically secure and the server S leaks no information regarding the keys SK_i, then the protocol 3PKD is SK-secure in the authenticated links model (AM).*

5 Additional Building Blocks

As a practical solution for secure protocol design and development using the Canetti-Krawczyk model, one needs as many building blocks as possible. We collect a SK-secure protocol in the AM and a proven secure authenticator and present them here as additional building blocks to be used in practical circumstances. These building blocks are not used throughout this paper, but they are as useful as those that are used. More importantly, one does not need to know the proof technique for building secure protocols as all building blocks presented here have already been proven secure.

5.1 A SK-secure Protocol in the AM

The well-known Diffie-Hellman (DH) protocol, Figure 11, is SK-secure in the AM under the decision Diffie-Hellman (DDH) assumption (Boneh 1998). We use the common notation for the description of the protocol as follows: $\mathbb{Z}_p^*$ denotes the multiplicative group of integers modulo a prime p, G denotes a cyclic subgroup of prime order q and a generator g. All arithmetic is performed in $\mathbb{Z}_p^*$ unless indicated otherwise.

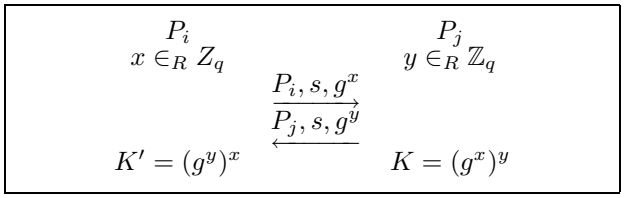

Figure 11: The Protocol DH

This protocol features some desirable properties such as forward secrecy and has been extended to be SK-secure in the UM (Canetti & Krawczyk 2001) using the signature based authenticator. An important consideration when choosing an appropriate SK-secure protocol in the AM as a building block is to consider the demands of additional protocol properties apart from security.

5.2 An Encryption Based MT-authenticator

The encryption based authenticator of Bellare et al. (1998) uses public key encryption and a MAC scheme. A fundamental assumption is that the MAC is secure and the public key encryption scheme is semantically secure against chosen ciphertext attack (Cramer & Shoup 1998, Rackoff & Simon 1991). The authenticator is presented in Figure 12.

We may use this authenticator in the same way as the two other MT-authenticators in this paper, namely the signature (Figure 5) and MAC based MT-authenticator (Figure 9). When choosing which authenticator should be used, one needs to understand the assumptions being made behind the security and the type of services that need to be provided.

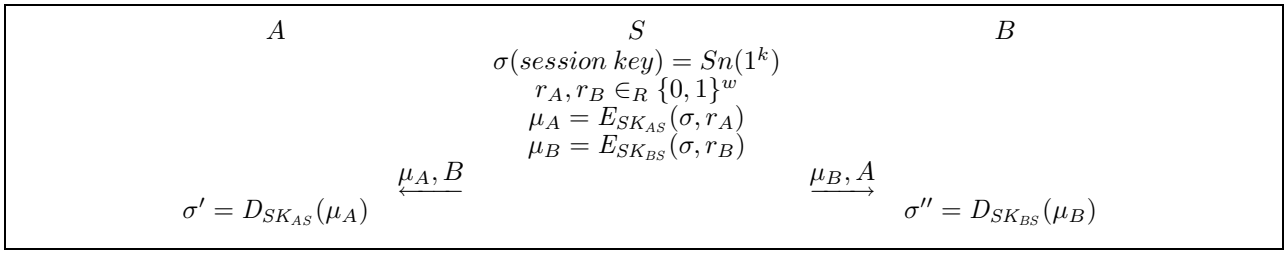

Figure 10: The Protocol 3PKD

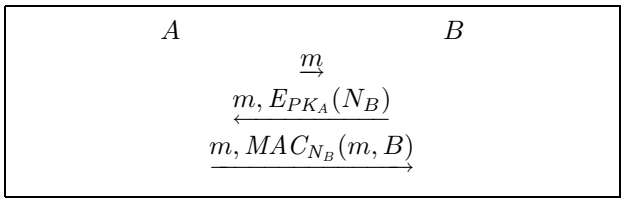

Figure 12: Encryption Based MT-authenticator

5.3 Mixed Use of MT-authenticators

In order to extend the usage of authenticators, we argue that it is valid to use more than one authenticator in the same protocol. For example, we may apply the signature based MT-authenticator to the first message of the protocol DH while using the encryption based MT-authenticator to emulate the second message in the UM. Of course to do this the protocol in the AM must consist of at least two message flows.

The proofs of the MT-authenticators are focused on the emulation of an AM message in the UM. More precisely, a SK-secure protocol in the AM is formed by one or more messages which are individually converted to messages in the UM. Such an approach preserves the possibility of applying different treatments to messages of protocols.

The steps for applying different MT-authenticators to a SK-secure protocol in the AM are the same as those for using a single MT-authenticator. This technique facilitates an increase in the number of protocols which are SK-secure in the UM.

6 Conclusion

We have demonstrated the usefulness of a formal modular approach for secure protocol design in a practical way without the need to understand the detailed proof technique. The idea is to collect SK-secure protocols in the AM and MT-authenticators, then apply the MT-authenticators to the SK-secure protocols in the AM to obtain SK-secure protocols in the UM. We note that it is possible to apply different MT-authenticators to different message flows of SK-secure protocols in the AM.

We have also demonstrated the technique for extraction of SK-secure protocols in the AM from a protocol, in particular a three-party key distribution protocol. The number of SK-secure protocols in the AM may be greatly increased as there are a large number of protocols which have been published. Of course, this technique demands the ability to prove the SK-security of the newly extracted protocol.

In conclusion, three SK-secure protocols in the AM have been demonstrated, one of which is new and proven to be SK-secure in the AM. Three authenticators have been collected, one of which is used for generating a new SK-secure protocol in the UM.

References

Bellare, M., Boldyreva, A. & Micali, S. (2000), Public-key encryption in a multi-user setting: Security proofs and improvements, *in* B. Preneel, ed., 'Advances in Cryptology – Eurocrypt 2000', Vol. 1807 of *LNCS*, Springer-Verlag. Full verision at `http://www-cse.ucsd.edu/users/mihir/papers/key-distribution.html`.

Bellare, M., Canetti, R. & Krawczyk, H. (1998), A modular approach to the design and analysis of authentication and key exchange protocols, *in* 'Proceedings of the 30th Annual Symposium on the Theory of Computing, ACM'.

Bellare, M. & Rogaway, P. (1993), Entity authentication and key distribution, *in* 'Advances in Cryptology - Crypto '93 Proceedings', Springer-Verlag. Full version.

Bellare, M. & Rogaway, P. (1995), Provably secure session key distribution - the three party case, *in* 'Proceedings of the 27th ACM Symposium on the Theory of Computing'.

Boneh, D. (1998), The decision Diffie-Hellman problem, *in* 'Proceedings of the Third Algorithmic Number Theory Symposium', Vol. LNCS 1423, Springer-Verlag, pp. 48–63.

Canetti, R. & Krawczyk, H. (2001), Analysis of key-exchange protocols and their use for building secure channels, *in* 'Proceedings of Eurocrypt', Vol. LNCS 2045.

Cramer, R. & Shoup, V. (1998), 'A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack', *in* 'Advances in Cryptology - Crypto 98, LNCS 1462', pp. 13–25.

Rackoff, C. & Simon, D. (1991), Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack, *in* 'Advances in Cryptology - Crypto 91 Proceedings, LNCS', Vol. 576.

A Security Proof

A.1 Model Description

Before we prove Theorem 1, we describe in more detail some elements of Canetti & Krawczyk's (2001) model, which are needed to understand the proof.

The execution of a key-exchange (KE) protocol is modelled as a collection of n programs each running at a different party. Each program can fork multiple sub-processes to handle multiple session of a protocol run. All communications between parties occur through the adversary $\mathcal{A}$, which has different capabilities depending on which of the two adversarial models (AM or UM) we are considering. In what follows we only concern ourselves with AM-adversaries. Furthermore, we do not consider forward secrecy, since it is not needed in our proof.

A KE protocol π can be activated at each party P_i in two ways:

1. By means of an establish–session(P_i,P_j,s, *role*) request, where P_j is other party with whom the key is to be established, s is a session-id string which uniquely identifies a session , and *role* can be either initiator or responder.

2. By means of an incoming message m with a specified sender P_j

We say that two sessions are *matching*, if in an execution of the protocol, party P_i has a session with input $(P_i, P_j, s, role)$ and party P_j has a session with input $(P_j, P_i, s', role')$ and $s = s'$.

In addition to the activation of parties, the adversary can performed the following actions:

1. $\mathcal{A}$ may issue a session–key query, which returns the session key (if any) accepted during a given session.

2. $\mathcal{A}$ may issue a session–state query, which returns all the internal state information associated to a particular session.

3. $\mathcal{A}$ may issue a test–session query. To respond to this query, a random bit $b \in_R \{0,1\}$ is selected. If $b = 0$ then the session key is returned. Otherwise, a random key chosen from the probability distribution of keys generated by the protocol is outputted. This query can only be issued to a session that has not been *exposed*, i.e that has not been the subject of a session–state or session–key queries, and whose involved parties have not been corrupted.

Once the adversary performs a test–session query, the adversary is not allowed to expose the test-session. Eventually $\mathcal{A}$ outputs a bit b' and halts.

Definition 3 *A KE protocol π is called* SK-secure *without perfect forward secrecy in the AM if the following two properties are satisfied for any AM-adversary $\mathcal{A}$.*

1. *Protocol π satisfies the property that if two uncorrupted parties complete matching sessions then they both output the same key; and*

2. *the probability that $\mathcal{A}$ guesses correctly the bit b is no more than $\frac{1}{2}$ plus a negligible fraction in the security parameter.*

A.2 Proof of Theorem 1

Since the protocol 3PKD is a key transport protocol, it is easy to see that both parties A and B are in possession of the same session key upon the completion of the protocol execution and satisfies the first condition of SK-security. This proof mainly concentrates on proving that the second condition of SK-security can be achieved.

In order to prove the satisfaction of the second condition, we will define a couple of algorithms. Let $\mathcal{A}$ be an adversary against the protocol 3PKD and $\mathcal{X}$ an experimental machine simulating the execution of the protocol.

We assume that the advantage of the adversary $\mathcal{A}$ in distinguishing between a session key and a random value is with a non-negligible probability ϵ and show that if $\mathcal{A}$ breaks the SK-security of the protocol, then the $\mathcal{X}$ can break the semantic security of the encryption scheme (LKG, E, D) with non-negligible probability, thus reaching contradiction.

In addition to the assistance of the algorithms described above, we need the following result to capture the fact that an adversary can get extra information by observation of α and β where the same session key is encrypted using different long term keys. This is a special case of a more general result of Bellare, Boldyreva & Micali (2000).

Lemma 1 *(Bellare et al. 2000) If (LKG, E, D) is a secure encryption scheme, then any multiple eavesdropper has negligible advantage.*

The inputs to $\mathcal{X}$ include strings σ_0, σ_1, one of which is the session key chosen according to Sn while the other is a random value of the same length of the session key. $\mathcal{X}$ also takes inputs $\alpha = E_{SK_{AS}}(\sigma_x, r_A), \beta = E_{SK_{BS}}(\sigma_x, r_B)$ and has access to the encryption oracles $E_{SK_{AS}}(\cdot)$ and $E_{SK_{BS}}(\cdot)$.

We note that the keys SK_{AS}, SK_{BS} are generated by the server S using LKG, $x \in_R \{0,1\}$ and $r_A, r_B \in_R \{0,1\}^w$.

We then proceed as follows:

1. Machine $\mathcal{X}$ sets up a virtual scenario for the run of protocol 3PKD and activates $\mathcal{A}$ against the virtual run. The adversary $\mathcal{A}$ has full control of the communication channels and is responsible to schedule all operations. The scheduled operations are perform by $\mathcal{X}$ that works on behalf of all virtual players (including S) in the virtual scenario for the run of protocol 3PKD;

2. At setup stage, $\mathcal{X}$ chooses randomly a pair of players $(A, B) \in_R \{1, ..., n\}$ where n denotes the maximum number of players that can be invoked and picks $u \in_R \{1, ..., l\}$ where l denotes the maximum number of sessions. We assume that the chosen session is (A, B, S, u). That is, the involved players are parties A, B and the server S which is used for session key generation and distribution amongst all parties;

3. When $\mathcal{A}$ invokes a new party $P_i \in \{1, ..., n\} \setminus \{A, B\}$, then $\mathcal{X}$ selects at random a shared encryption key SK_i between the party and the server by using the long term key generator LKG. If the new party is A (resp. B), its long term encryption key is SK_{AS} (resp. SK_{BS}). Note that neither $\mathcal{X}$ nor $\mathcal{A}$ knows the keys, but $\mathcal{X}$ has access to the encryption oracles $E_{SK_{AS}}(\cdot)$ and $E_{SK_{BS}}(\cdot)$;

4. Whenever $\mathcal{A}$ activates parties P_i and P_j, $\mathcal{X}$ generates μ_i and μ_j according to the protocol specification except for the case where $P_i = A$, $P_j = B$ and the session $s = u$. Then $\mathcal{X}$ generates a session key σ using Sn, picks two random nonces r_i and r_j, computes $\mu_i = E_{SK_i}(\sigma, r_i)$ and $\mu_j = E_{SK_j}(\sigma, r_j)$ and sends μ_i, μ_j to P_i and P_J respectively. We note that if the participants are different from A or B, then $\mathcal{X}$ knows their long-term secret keys and hence can compute the encryptions directly. Otherwise, $\mathcal{X}$ has to query the encryption oracles $E_{SK_{AS}}(\cdot)$ and $E_{SK_{BS}}(\cdot)$;

5. If $\mathcal{A}$ decides to activate A and B at session u, then $\mathcal{X}$ sends the test encryptions α and β to A and B, respectively;

6. During the experiment, all attacks against the session $(P_i, P_j, S, s) \neq (A, B, S, u)$ and its participants are answered by $\mathcal{X}$ using its knowledge of keys. If a session reveal is scheduled against the session u, a corruption against A or B, or a different test session $(P_i, P_j, S, s) \neq (A, B, S, u)$ is chosen, $\mathcal{X}$ aborts the run of $\mathcal{A}$ and outputs a bit $x' \in_R \{0,1\}$;

7. If $\mathcal{A}$ queries the test session u, $\mathcal{X}$ flips a coin for the value of $x' \in_R \{0,1\}$. If $x' = 0$, $\mathcal{X}$ returns σ_0, otherwise σ_1;

8. If $\mathcal{A}$ halts and outputs a bit x', then $\mathcal{X}$ halts and outputs a bit x' too;

9. Now, we calculate the overall probability with which $\mathcal{X}$ can succeed against the virtual protocol run in distinguishing a session key from a random nonce. Firstly, the probability is $1/2$ in the case where $\mathcal{A}$ is aborted. Secondly, the probability that $\mathcal{A}$ selects the correct test session is $1/[ln(n-1)]$. Thirdly, the probability that $\mathcal{A}$ breaks the SK-security of the protocol is non-negligible (by assumption) over $1/2$, thus the overall advantage of $\mathcal{X}$ for breaking the semantic security of the encryption scheme is $1/2 + \epsilon/[ln(n-1)]$ which is non-negligible.