Protocols with Security Proofs for Mobile Applications†

Yiu Shing Terry Tin, Harikrishna Vasanta, Colin Boyd*, and Juan Manuel González Nieto

Information Security Research Centre,
Queensland University of Technology.
PO Box 2434, Brisbane, QLD 4001, Australia.
{t.tin,h.vasanta,c.boyd,j.gonzaleznieto}@qut.edu.au

Abstract. The Canetti-Krawczyk (CK) model is useful for building reusable components that lead to rapid development of secure protocols, especially for engineers working outside of the security community. We work in the CK model and obtain a new secure authenticated key transport protocol with three parties. This protocol is constructed with two newly developed components in the CK model, thus extending the power of the model.

Keywords: Provable security, secure key exchange, authenticator, mobile security.

1 Introduction

A proof of security has become an essential statement for structural correctness of new key establishment protocols. The first provably secure protocol was proposed by Bellare and Rogaway [7] in 1993 with a two-party example. In 1995 they extended their work to a three-party server-based key distribution protocol [8]. A feature of proofs in this model is that they are long and difficult to read for the non-specialist, making them more error-prone. Another shortcoming is that a minor change in the protocol requires a complete revision of the proof. There is no easy way to reuse fragments of security proof.

In 1998, Bellare, Canetti and Krawczyk [4] proposed a new model for provable security. The novelty of their work was its modular approach. This approach uses layers for separated treatments for key exchange and authentication. In comparison to the approach used in [8], this approach turns out to be easier to understand because of its modularity. Moreover, it provides reusable building blocks for construction of new provably secure protocols. More precisely, a module carrying a security proof can be reused for construction of new secure

† This is the complete version of a paper appearing in Proceedings of ACISP 2004, LNCS, Springer-Verlag.

* Supported by ARC Discovery Project DP0345775.

protocols with other modules in the model that have been proven secure independently. This work was later extended by Canetti and Krawczyk [9]; we refer to this model as the CK model in this paper.

Despite all advantages of the CK approach, there exists a limited number of modules in the different layers. With a limited number of reusable modules, the full power of the CK approach cannot be seen. By increasing the number of reusable modules, secure protocols with different requirements can easily be constructed.

1.1 Motivation

A typical mobile network involves a network operator and mobile users. With advances in mobile technologies, independent application service providers (ASPs) start providing services to the mobile users via their mobile devices. Mobile users make payments to their respective network operators where funds are later transferred to the appropriate ASP. These activities need cryptographic protection for security reasons. The typical methodology of achieving security for communications is to first establish a session key, then encrypt all subsequent messages using the session key. Establishment of secure session keys was never an easy task historically. With limited resources in mobile networking, it is even harder to perform the task appropriately.

When considering the interactions between the three parties, network operator, mobile user and application service provider, it is easy to see that neither a public key nor a symmetric key scheme alone is sufficient and efficient for establishing a session key between the parties. From the viewpoints of mobile users, it is preferable to choose from a list of ASPs, thus implying the passive status of the ASPs until a choice is made.

To solve this problem, let us assume that the network operator is trustworthy. Since users and the network operator share symmetric keys in modern mobile networks, symmetric key cryptography can be applied to this link. With sufficiently high computational power, public key cryptography can be used by the network operator and ASP. Lastly, the network operator generates the session key and sends it to the user. This is an important setting for the sake of simplicity and providing the choice of ASPs to mobile users. The choice of ASPs is important for protecting the privacy of the user with regards to privacy policy. Note that this setting implies that the session key is forwarded by the user to the ASP.

From our library of reusable modules, there exists no suitable modules in the CK approach to address this real life application. Furthermore, there exists no suitable protocol in the literature of secure protocols for this particular requirement. The lack of a suitable candidate is the main motivation for this paper.

1.2 Main Contributions

We regard the following as the main contributions of this paper:

– A new key distribution protocol for applications in mobile networks;
– A new non-interactive MT-authenticator which does not generate additional messages for authentication;
– Two new authenticated protocols with security proofs.

2 The Model

In the CK model the definition of security for key-exchange (KE) protocols follows the tradition of Bellare and Rogaway [6], and is based on a game played between the adversary and the parties $P_1, \ldots, P_n$. In this game, protocol π is modeled as a collection of n programs running at different parties $P_1, \ldots, P_n$. Each program is an interactive probabilistic polynomial-time (PPT) machine. Each invocation of π within a party is defined as a *session*, and each party may have multiple sessions running concurrently. The communications network is controlled by an adversary $\mathcal{A}$, also a PPT machine, which schedules and mediates all sessions between the parties. $\mathcal{A}$ may activate a party P_i in two ways.

1. By means of an $\mathsf{establish\text{-}session}(P_i, P_j, s)$ request, where P_j is another party with whom the key is to be established, and s is a session-id string which uniquely identifies a session between the participants. Note that session-id is chosen by the adversary, with the restriction that it has to be unique among all sessions between the two parties involved. This allows the delivery of messages to the right protocol instantiation within a party.
2. By means of an *incoming message* m with a specified sender P_j.

A restriction on how the adversary activates parties exists depending on which of the following two adversarial models is being considered.

– *Authenticated-links adversarial Model (AM)* defines an idealised adversary that is not allowed to generate, inject, modify, replay and deliver messages of its choice except if the message is purported to come from a corrupted party. Thus, an *AM–adversary* can only activate parties using incoming messages that were generated by other parties in π.
– *The Unauthenticated-links adversarial Model (UM)* is a more realistic model in which the adversary does not have the above restriction. Thus, a *UM–adversary* can fabricate messages and deliver any messages of its choice.

Upon activation, the parties do some computations, update their internal state, and may output messages which include the identity of the intended receiver. Two activated parties P_i and P_j are said to have a *matching session* if they have sessions whose session-ids are identical and they recognised each other as their respective communicating partner for the session. In addition to the activation of parties, $\mathcal{A}$ can perform the following actions:

1. $\mathcal{A}$ may *corrupt* a party P_i at will, by issuing the query $\mathsf{corrupt}(P_i)$, and learn the entire current state of P_i including long-term secrets, session internal states and session keys. From this point on, $\mathcal{A}$ may issue any message in which P_i is specified as the sender and play the role of P_i;

2. $\mathcal{A}$ may issue the query session-key(P_i, s), which returns the session key (if any) accepted by P_i during a given session s;
3. $\mathcal{A}$ may issue the query session-state(P_i, s), which returns all the internal state information of party P_i associated to a particular session s;
4. $\mathcal{A}$ may issue the query test-session(P_i, s). To respond to this query, a random bit $b \in_R \{0, 1\}$ is selected. If $b = 1$ then the session key is returned. Otherwise, return a random key chosen from the probability distribution of keys generated by the protocol. 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.

During the game, the adversary performs a test-session query to a party and session of its choice. After that, the adversary is not allowed to expose the test-session. $\mathcal{A}$ may continue with its regular actions with the exception that no more test-session queries can be issued. Eventually $\mathcal{A}$ outputs a bit b' as its guess on whether the returned value is the session key or a random number, then halts. $\mathcal{A}$ wins the game if $b = b'$. The definition of security follows.

Definition 1. *A KE protocol π is called* SK-secure *without perfect forward secrecy in the AM if the following properties are satisfied for any AM-adversary $\mathcal{A}$.*

1. *If two uncorrupted parties complete matching sessions then they both output the same key;*
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.*

The definition of SK-secure protocols in the UM is done analogously. The SK-security is defined with or without forward secrecy. By distinguishing between the AM and the UM, Canetti and Krawczyk allow for a modular approach to the design of SK-secure protocols. Protocols that are SK-secure in the AM can be converted into SK-secure protocols in the UM by applying an *authenticator* to it. An authenticator is a protocol translator $\mathcal{C}$ that takes as input a protocol π and outputs another protocol $\pi' = \mathcal{C}(\pi)$, with the property that if π is SK-secure in the AM, then π' is SK-secure in the UM. Authenticators can be constructed by applying a *message transmission (MT) authenticator* to each of the messages of the input protocol. Bellare *et al.* [4] and Canetti and Krawczyk [9] provided three examples of MT-authenticators.

3 Definition of Security of Bi-encryption Scheme

In this section, we define the notion of security to deal with the case where symmetric and asymmetric encryption schemes are used for separate encryptions of an identical message. We describe security in terms of indistinguishability [11] under chosen plaintext attacks (IND-CPA) [5]. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme which consists of a key generation algorithm $\mathcal{K}$, a probabilistic encryption algorithm $\mathcal{E}$ and a decryption algorithm $\mathcal{D}$. Let m_0 be a message and $m_1 \in_R \{0,1\}^{|m_0|}$. Let $\mathcal{A}_{\Pi}^{cpa,\mathcal{E}(\cdot)}$ be a polynomial time adversary against Π given access to the encryption oracle $\mathcal{E}(\cdot)$. Let κ be a security parameter. Security in the sense of IND-CPA entails that an adversary cannot distinguish a ciphertext $C \leftarrow \mathcal{E}(m_b)$ where $b \in_R \{0,1\}$ with probability "significantly" more than $\frac{1}{2}$.

Let $\Pi_1 = \{\mathcal{K}_1, \mathcal{E}_1, \mathcal{D}_1\}$ and $\Pi_2 = \{\mathcal{K}_2, \mathcal{E}_2, \mathcal{D}_2\}$ be respectively a symmetric and an asymmetric encryption scheme that satisfy indistinguishability under CPA. We define the bi-encryption algorithm $\mathcal{E}^*$ to be the execution of $\mathcal{E}_1$ and $\mathcal{E}_2$ such that on input a message m, it outputs two ciphertexts $C_1 \leftarrow \mathcal{E}_1(m), C_2 \leftarrow \mathcal{E}_2(m)$. We write $\{C_1, C_2\} \leftarrow \mathcal{E}^*(m)$ and refer to this particular setting as "Bi-encryption".

We denote the bi-encryption scheme $\Pi^* = \{\Pi_1, \Pi_2\}$. Let $\mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}$ be an adversary attacking Π^* with access to the bi-encryption oracle $\mathcal{E}^*(\cdot)$.

Definition 2 (Security Notion of Bi-encryption in IND-CPA). *We define the experiment of the attacking mode in the sense of IND-CPA:*

$$\begin{gathered}
\text{Experiment } \mathbf{Exp}_{\Pi^*}^{cpa}(\mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}, b) \\
k \leftarrow \mathcal{K}_1(\kappa_1), (e,d) \leftarrow \mathcal{K}_2(\kappa_2) \\
(m_0, m_1, s) \leftarrow \mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}(find, \kappa_1, \kappa_2, e) \\
\{C_1, C_2\} \leftarrow \mathcal{E}^*(m_b) \\
z \leftarrow \mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}(guess, \kappa_1, \kappa_2, C_1, C_2, s) \\
\textbf{Return } z
\end{gathered}$$

The advantage of the adversary is defined as follows:

$$\mathbf{Adv}_{\Pi^*}^{cpa}(\mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}) = \Pr[\mathbf{Exp}_{\Pi^*}^{cpa}(\mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}, 1) = 1] - \Pr[\mathbf{Exp}_{\Pi^*}^{cpa}(\mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}, 0) = 1]$$

The encryption scheme Π^* is said to be secure under CPA if the function $\mathbf{Adv}_{\Pi^*}^{cpa}(\mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)})$ is negligible for every polynomial time adversary $\mathcal{A}_1$ in κ_1 and $\mathcal{A}_2$ in κ_2.

It is important to confirm that the notion of indistinguishability in encryption algorithms is strong enough to also imply security in the bi-encryption. This problem was addressed by Baudron, Pointcheval and Stern [2] and independently by Bellare, Boldyreva and Micali [3]. Note that the proof by the latter is based on a single public key encryption scheme using different keys.

Theorem 1 ([2]). *If the encryption schemes Π_1 and Π_2 are secure under CPA, then the encryption scheme Π^* is secure under CPA. More precisely:*

$$\mathbf{Adv}_{\Pi^*}^{cpa}(\mathcal{A}_{\Pi^*}^{cpa,\mathcal{E}^*(\cdot)}) \leq \mathbf{Adv}_{\Pi_1}^{cpa}(\mathcal{A}_{\Pi_1}^{cpa,\mathcal{E}_1(\cdot)}) + \mathbf{Adv}_{\Pi_2}^{cpa}(\mathcal{A}_{\Pi_2}^{cpa,\mathcal{E}_2(\cdot)}) \tag{1}$$

The actual result in [2] considers only asymmetric encryption schemes. However, their result is also applicable to any combination of symmetric and asymmetric encryption schemes as long as the equivalent notion of security is satisfied.

4 A New Protocol Secure in the AM

We propose a key distribution protocol in the AM where two parties use a trusted server for session key generation. This protocol uses both symmetric and asymmetric encryption. There is a need for this type of setting, particularly for business in mobile networks. We explain why.

The typical setting in mobile networks involves a network operator and users with a mobile device. A third player in mobile networks is the ASP which provides services to users and gets payment from users via the network operator. Typically, they do not share a long term secret key. Thus it is required to setup a secure communication channel between them before any sensitive data could be exchanged. The secure communication channel is constructed with a session key generated by the network operator given that both the users and the ASPs trust the former. The trust on the network operator is reasonable because the basic telecommunications services are provided by the network operator.

When determining the mechanism for setting up the session key, we consider two links: the link between the users and the network operator and the link between the network operator and the ASPs. Since users in mobile networks share a long term secret key with the network operator for basic telecommunications services, it is natural to use a symmetric key encryption scheme on this link. It is also the best choice for the users due to the constraints in computation of mobile devices. On the link between the network operator and the ASPs, we use an asymmetric key encryption scheme. We have two reasons for this decision. Firstly, it is reasonable to assume that both the network operator and the ASPs are capable of running computationally intensive algorithms. Secondly, it is difficult to securely distribute long term secret keys to all ASPs, thus limiting the use of symmetric encryption schemes on this link.

Let S be the network operator, U be the user and A be the ASP. The sequence of the communications link is: $U \to S \to U \to A$. Note that the link between the network operator and the ASPs has been broken down to two links where U relays messages from S to A. This setting allows the users to choose different ASPs based on their preferences.

Assume that U shares a long term secret key k_{US} with S. Let (e_X, d_X) be the public and private key of party X. The session key is encrypted with the long term secret key shared between U and S. On the link between A and S, public key cryptography is employed. Let $\sigma \leftarrow Sn(1^\kappa)$ be a session key generated by the key generator $Sn(1^\kappa)$ with security parameter κ. We denote $\mathcal{E}_k(m)$ be encryption of some message m under key k. Similarly, $\mathcal{D}_k(c)$ denotes decryption of some ciphertext c with key k. The protocol is shown in Figure 1.

Remark 1. We note the importance of the session identifier s. It is used to match concurrent sessions running by protocol participants, thus its value must be unique such that probability of the appearance of the same value twice is negligible. A practical implementation for uniqueness of the session identifier is to enforce contributions s_1, s_2 from both parties. Each party then knows that the session identifier is fresh and unique.

$$
\begin{array}{ccccc}
\hline
S & & U & & A \\
\sigma \leftarrow Sn(1^{\kappa}) & & & & \\
\mu_U \leftarrow \mathcal{E}_{k_{US}}(\sigma) & & & & \\
\mu_A \leftarrow \mathcal{E}_{e_A}(\sigma) & & & & \\
 & \xrightarrow{A, s, \mu_U} & & & \\
 & \xrightarrow{U, s, \mu_A} & & & \\
 & & & \xrightarrow{U, s, \mu_A} & \\
 & & \sigma' \leftarrow \mathcal{D}_{k_{US}}(\mu_U) & & \sigma'' \leftarrow \mathcal{D}_{d_A}(\mu_A) \\
\hline
\end{array}
$$

Fig. 1. The Protocol 3PKD

Theorem 2. *Let Π_1 be the symmetric encryption scheme and Π_2 be the asymmetric encryption scheme used in protocol 3PKD. The protocol 3PKD is SK-secure in the authenticated links model (AM) if the encryption schemes Π_1 and Π_2 are secure under chosen plaintext attacks.*

Proof. Since the protocol 3PKD is a key distribution protocol, it is easy to see that both parties U and A are in possession of the same session key upon the completion of the protocol execution and therefore satisfies the first condition of SK-security in Definition 1. Note that uncorrupted parties behave according to the protocol specification. This proof concentrates on proving the second condition of the SK-security.

Let Π^* be the bi-encryption scheme constructed from Π_1 and Π_2. Let $\mathcal{A}$ be an adversary against the protocol 3PKD. Let ϵ be the advantage of $\mathcal{A}$ in distinguishing between a session key and a random value of the same length. We show that if ϵ is non-negligible, then at least one of Π_1, Π_2 can be broken, thus reaching a contradiction.

For simplicity, message routes are modified such that A has a direct link with S. That is, the routes are now: $S \rightarrow U, S \rightarrow A$. We stress that this alteration does not affect the security of the protocol, as messages from S to A can always be forwarded by U if necessary.

We construct an algorithm $\mathcal{X}$ to break Π^*. It runs $\mathcal{A}$ as a subroutine. Let $k^* \leftarrow \mathcal{K}_1(\kappa_1)$ be the symmetric key and $(e^*, d^*) \leftarrow \mathcal{K}_2(\kappa_2)$ be the pair of asymmetric keys used in 3PKD. Following Definition 2, $\mathcal{X}$ first computes $\sigma_0 \leftarrow Sn(1^{\kappa})$ and $\sigma_1 \in_R \{0,1\}^{|\sigma_0|}$. The challenges that $\mathcal{X}$ receives are a pair of ciphertexts $(\mu_U^* \leftarrow \mathcal{E}_{k^*}(\sigma_b), \mu_A^* \leftarrow \mathcal{E}_{e^*}(\sigma_b))$ where $b \in_R \{0,1\}$. Resources available to $\mathcal{X}$ are the public key e^*, access to the bi-encryption oracle $\mathcal{E}^*(\cdot) = (\mathcal{E}_{k^*}(\cdot), \mathcal{E}_{e^*}(\cdot))$ and the session key generator $Sn(\cdot)$. $\mathcal{X}$ then proceeds 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 control of the communication channels and schedule of all operations. The scheduled operations are performed by $\mathcal{X}$ on behalf of all virtual players in the virtual scenario for the run of protocol 3PKD. Virtual players include server, user and service provider. The simulation takes into consideration of the possession of keys of different parties.

2. At setup stage of the virtual protocol run, $\mathcal{X}$ chooses randomly a server S^* from all possible servers $(S_1, \ldots, S_n)$ in the virtual run of the protocol. Similarly $\mathcal{X}$ chooses randomly a user U^* and a service provider A^* from a list of imitated users $(U_1, \ldots, U_u)$ and service providers $(A_1, \ldots, A_a)$ respectively. We use n (resp. u and a) to denote the maximum number of S (resp. U and A) that can be invoked in the virtual protocol run. Since multiple sessions are possible amongst the chosen parties (S^*, U^*, A^*), $\mathcal{X}$ picks $s^* \in_R \{1, \ldots, l\}$, where l denotes the maximum number of sessions between the chosen parties, as its guess on which $\mathcal{A}$ will choose as the test session. That is, the chosen session by $\mathcal{X}$ is (S^*, U^*, A^*, s^*).
3. All long-term symmetric keys and asymmetric key pairs are chosen by $\mathcal{X}$ except for the case of U^* and A^*. $\mathcal{X}$ provides $\mathcal{A}$ with all public keys for all service providers $A_i \neq A^*$. The input public key to $\mathcal{X}$, e^*, is used as the public key of A^*.
4. All session establishments are executed by $\mathcal{X}$ according to the protocol specification except the chosen session. Whenever a session $s \neq s^*$ is invoked by $\mathcal{A}$, $\mathcal{X}$ runs $Sn(1^\kappa)$ to obtain a session key σ_i. If U^* is not involved in the session s, encryptions of the session key can easily be computed because of the complete knowledge of the long term keys. If U^* is involved in the session s, $\mathcal{X}$ accesses the encryption oracle $(C_1, C_2) \leftarrow \mathcal{E}^*(\sigma_i)$ and returns C_1 as the encryption of σ_i for U^*. This simulation is perfect from $\mathcal{A}$'s point of view. If S^* is activated for establishing a session between U^* and A^* at session s^*, $\mathcal{X}$ sends μ^*_U and μ^*_A to U^* on behalf of S^* and sends μ^*_A to A^* on behalf of U^*.
5. All exposures against session $s \neq s^*$ can be answered by $\mathcal{X}$ with its knowledge of keys except for exposures against S^*, U^* or A^*. If a session reveal is scheduled against the session s^*, a corruption against U^*, A^* or S^*, or a different test session $(S_i, U_j, A_k, s) \neq (S^*, U^*, A^*, s^*)$ is chosen by $\mathcal{A}$, $\mathcal{X}$ aborts the run of $\mathcal{A}$ and outputs a bit $b' \in_R \{0, 1\}$ as its guess on b.
6. If $\mathcal{A}$ queries the test session s^*, $\mathcal{X}$ returns (μ^*_A, μ^*_U) to $\mathcal{A}$. We stress that no exposure against the test session (S^*, U^*, A^*, s^*) can be performed after the test key is returned.
7. If $\mathcal{A}$ halts and outputs a bit b', then $\mathcal{X}$ halts and outputs the same bit.

By running $\mathcal{A}$ as a subroutine, $\mathcal{X}$ can break the bi-encryption scheme Π^* with overall probability $\frac{1}{2} + \frac{\epsilon}{l \cdot n \cdot u \cdot a}$. This is derived from the probability $(l \cdot n \cdot u \cdot a)^{-1}$ that $\mathcal{A}$ chooses the correct test session and the probability of guessing correctly the bit b when $\mathcal{A}$ is aborted. The advantage $\frac{\epsilon}{l \cdot n \cdot u \cdot a}$ is non-negligible.

By Theorem 1, the maximum advantage of the adversary attacking the bi-encryption scheme Π^* under CPA is the sum of the advantages of the adversaries attacking Π_1 and Π_2. If Π^* can be broken under CPA with non-negligible advantage, then at least one of Π_1, Π_2 can be broken with non-negligible advantage. This contradicts our assumptions in Theorem 2.

5 New Authenticator

5.1 A Non-interactive Signature Based MT-authenticator

We propose a non-interactive signature-based MT-authenticator λ_{Sig}^{Time} that uses time stamps to provide freshness. In contrast to the original signature–based MT–authenticator of Bellare *et al.* [4], our new MT-authenticator results in only one message in the UM for every message to be authenticated.

Using time stamps for freshness requires synchronisation in time. In practice, maintaining perfect synchronisation is extremely expensive and is not commonly used for communications between clients and servers. Thus, we need to accept "loosely synchronised" times for the scheme to be practical. In addition, the time stamp is required to be unique. The uniqueness is maintained by the procedures for signature verification.

Time stamps could be in different formats [14, 13], but we provide an abstract formulation for our needs. We assume that there exists a secure time server $\mathcal{TS}$ which we model as a universal time oracle available to all parties. All parties access this oracle whenever a time stamp is required. We also assume that there exists a boolean function $\mathcal{V}$ which returns *TRUE* or *FALSE* on input a time stamp Υ. If it returns *TRUE*, then the time stamp Υ is fresh. Otherwise, Υ is expired. Note that this function may make decisions based on some "looseness" in time if necessary.

Let $\mathcal{I}$ be an initialisation function which invokes, once for each party, the key generation algorithm of a signature scheme secure under chosen message attacks with security parameter κ. Let *Sig* and *Ver* denote the signing and verification algorithms. Let sk_i and pk_i be the private and public keys of a party P_i where all public keys pk_i are known by all other parties. Whenever P_i needs a time stamp, it requests the time oracle $\mathcal{TS}(\cdot)$ for a time value.

The MT-authenticator λ_{Sig}^{Time} is activated within some party P_i with an external request to send a unique message m to party P_j. Then λ_{Sig}^{Time} invokes a two-party protocol λ_S^T that proceeds as follows. Since λ_S^T involves only two parties, we use A and B for their identities. Party A sends 'signature:$Sig_A(m, \Upsilon_A, B)$' to B and outputs 'A sent message m to B'. When 'signature:$Sig_A(m, \Upsilon_A, B)$' arrives at party B, B accepts m if the signature is *successfully verified* as described below. If m is accepted by B, B outputs 'B received m from A'. This type of transmission is captured in Figure 2.

$$\begin{tikzcd}
A \arrow[r, "Sig_A(m, \Upsilon_A, B)"] & B
\end{tikzcd}$$

Fig. 2. Signature Based MT-authenticator using Time stamps

Remark 2. Our representation of the signature makes no assumption on its nature. In the case of a signature scheme with message recovery, only the signature is sent. In the case of a signature scheme with appendix, the messages signed and the signature are sent together.

For ensuring the security of λ_{Sig}^{Time}, B needs to maintain a list $\mathcal{L}$ of received and accepted messages by all instances run by B. Party B maintains this list with two operations, namely insertion and deletion of records. Let n be the maximum number of records in $\mathcal{L}$. A record $l_{i\in\{1,\ldots,n\}} = (m', \Upsilon'_A)$ is added to $\mathcal{L}$ when m' is accepted by B such that B outputs 'B received m' from A'. The record l_i is removed from $\mathcal{L}$ when its associated time stamp Υ'_A has become invalid.

Upon receiving the message $Sig_A(m, \Upsilon_A, B)$ from A, party B proceeds to execute the signature verification procedures as follows. Firstly B recovers the data (m, Υ_A, B) from the signature and searches for an identical time stamp from its list of records $\mathcal{L}$. If an entry is found, then m is rejected. If not, B verifies the signature using the signature verification algorithm $Ver_{pk_A}(m, \Upsilon_A)$. If the signature is invalid, then m is rejected. Otherwise, B runs $\mathcal{V}$ to determine whether the time stamp is freshly generated. Party B only accepts m if $\mathcal{V}$ returns "TRUE".

Remark 3. Counters could be used to replace time stamps for freshness for our new MT-authenticator. Replacing time stamps with counters is straightforward [1] and the resultant MT-authenticator is also non-interactive. The security proof of λ_{Sig}^{Time} can easily be modified to suit the MT-authenticator that uses counters. The main difference in the proof is the mechanism used to verify freshness of messages.

Security Proof We now proceed to show that the protocol λ_{Sig}^{Time} is a valid MT-authenticator which emulates an MT-protocol in the UM. An MT-protocol refers to the transmission of a single message flow while emulation follows the convention of Ballere, Canetti and Krawczyk [4]. Assume that the signature scheme is secure under adaptive chosen message attack in the convention of Goldwasser, Micali and Rivest [12] and is thus existentially non-forgeable.

Theorem 3. *Assume that the signature scheme in use is secure under adaptive chosen message attacks. Then protocol λ_{Sig}^{Time} emulates protocol MT.*

Proof. The proof follows the general technique in the security proof of an MT-authenticator using random nonces [4] and appears in Appendix A.

5.2 From MT-authenticator to Authenticator

Our new MT-authenticator λ_{Sig}^{Time} can be made into a full-fledged authenticator using Theorem 4 as follows.

Theorem 4 ([4]). *Let λ be an MT-authenticator such that λ emulates message transmission MT in unauthenticated networks, and let $\mathcal{C}_\lambda$ be a compiler constructed based on λ as described above. Then $\mathcal{C}_\lambda$ is an authenticator.*

Depending on the number of message flows in the AM, a full-fledged authenticator can be a combination of MT-authenticators. If there is a single message flow in the AM, a MT-authenticator is sufficient to be used as an authenticator. Otherwise, two or more MT-authenticators need to be combined for transforming different message flows in the AM. We stress that Theorem 4 implies authentication of a single message and puts no restriction on how the other messages of the same AM protocol are authenticated, thus is independent of the number of messages or participants.

A New Authenticator for Three Party Protocol As an example for generating a full-fledged authenticator to transform a two-message AM protocol to a protocol secure in the UM, we recall a MAC-based MT-authenticator [9] and combine it with our signature-based MT-authenticator using time λ_{Sig}^{Time}. Other types of MT-authenticators exist, but the choice of λ_{MAC} is necessary to satisfy the unique requirements of the AM protocol of Figure 1. The MAC-based MT-authenticator is shown in Figure 3 and is referred to as λ_{MAC} hereafter.

$$\begin{array}{ccc} A & & B \\ & \xrightarrow{m} & \\ & \xleftarrow{m, N_B} & \\ & \xrightarrow{m, MAC_{k_{AB}}(m, N_B, B)} & \end{array}$$

Fig. 3. MAC Based MT-authenticator

We combine the MT-authenticators λ_{MAC} and λ_{Sig}^{Time} for a secure transformation of protocol 3PKD. It is also applicable to other AM protocols with two message flows with the respective computational requirements. The full-fledged authenticator $\mathcal{C}_{\lambda_{Sig}^{Time}}^{\lambda_{MAC}}$ is illustrated in Figure 4.

$$\begin{array}{ccccc} S & & U & & A \\ & \xleftarrow{r_U \in_R \{0,1\}^\kappa} & & & \\ & \xrightarrow{\substack{m_U, MAC_{k_{US}}(m_U, r_U, U), \\ Sig_S(m_A, \Upsilon_S, A)}} & & & \\ & & & \xrightarrow{Sig_S(m_A, \Upsilon_S, A)} & \end{array}$$

Fig. 4. A Full-fledged Authenticator derived from λ_{MAC} and λ_{Sig}^{Time}

Note the disappearance of the first message in $\mathcal{C}_{\lambda_{Sig}^{Time}}^{\lambda_{MAC}}$ from S to U. This message is not required as the MT-authenticator λ_{MAC} can be simplified to two message exchanges since the first message m from A to B is redundant.

6 Secure Protocols in the UM

In this section, we demonstrate the applications of our new modules in the CK model. First we show a secure three-party key transport protocol in the UM. Then we show an authenticated Diffie-Hellman key exchange.

6.1 An Authenticated Key Transport Protocol

Using $\mathcal{C}^{\lambda_{MAC}}_{\lambda^{Time}_{Sig}}$, we show that protocol 3PKD of Figure 1 can be compiled into a secure protocol in the UM. More precisely, we compile our new three-party key distribution protocol 3PKD of Figure 1 with $\mathcal{C}^{\lambda_{MAC}}_{\lambda^{Time}_{Sig}}$ to obtain a secure key distribution protocol in the UM. We denote by $\Upsilon_S \leftarrow \mathcal{TS}$ an action that S obtains a time from the time server to generate the time stamp. Let $a||b$ be the concatenation of a and b. The resultant secure protocol following the above procedures is shown in Figure 5 and is named 3PKDUM.

$$
\begin{array}{cccc}
S & & U & A \\
\hline
 & \xleftarrow{r_U \in_R \{0,1\}^\kappa} & & \\
\sigma \leftarrow Sn(1^k) & & & \\
\mu_U \leftarrow \mathcal{E}_{k_{US}}(\sigma) & & & \\
\mu_A \leftarrow \mathcal{E}_{e_A}(\sigma) & & & \\
\Upsilon_S \leftarrow \mathcal{TS} & & & \\
s = r_U || \Upsilon_S & & & \\
 & \xrightarrow{A, s, \mu_U, MAC_{k_{US}}(A, s, \mu_U, r_U, U)} & & \\
 & \xrightarrow{Sig_S(U, s, \mu_A, \Upsilon_S, A)} & & \\
 & & \xrightarrow{Sig_S(U, s, \mu_A, \Upsilon_S, A)} & \\
\hline
\end{array}
$$

Fig. 5. A Key Transport Protocol Secure in the UM

The session identifier s in 3PKDUM is a simple concatenation of r_U and Υ_S. The ASP A gets assurance of freshness because the checking process ensures that no time stamp is accepted twice. Even though U does not check the freshness of Υ_S, and A cannot check the freshness of r_U, the combination of these values is unique. In practice, of course, only one copy of each value is required.

6.2 An Authenticated Diffie-Hellman Key Exchange using Time Stamp

To further demonstrate the advantage of our new MT-authenticator, we construct an authenticator based solely on λ^{Time}_{Sig}. This authenticator is then applied to the ordinary Diffie-Hellman key exchange [10] to obtain an authenticated Diffie-Hellman key exchange in the UM. Assume that the session identifier s is known by A and B before the run of the protocol. Then we simply apply λ_{Sig}^{Time} to each of the messages of the Diffie-Hellman key exchange, then combine the results of these applications. Note that the ordinary Diffie-Hellman key exchange was proven SK-secure in the AM [9]. The authenticated Diffie-Hellman key exchange using time stamps is illustrated in Figure 6 and is named 2DHTUM.

$$
\begin{array}{lcr}
\hline
A & & B \\
\varUpsilon_A \leftarrow \mathcal{TS} & & \\
x \in_R \mathbb{Z}_q & & y \in_R \mathbb{Z}_q \\
 & \xrightarrow{Sig_{sk_A}(A, s, g^x, \varUpsilon_A, B)} & \\
 & & \varUpsilon_B \leftarrow \mathcal{TS} \\
 & \xleftarrow{Sig_{sk_B}(B, s, g^y, \varUpsilon_B, A)} & \\
K' = (g^y)^x & & K = (g^x)^y \\
\hline
\end{array}
$$

Fig. 6. Authenticated Diffie-Hellman Key Exchange using Time Stamps

It is worth noting that the message flows are not required to be sequential. That is, the protocol 2DHTUM is a one-round protocol. In comparison, the authenticated Diffie-Hellman key exchange [9] by applying the original signature based MT-authenticator in the CK model requires three rounds to complete.

7 Conclusion

In this paper, we defined a notion of security for encryptions of an identical message using one symmetric and one asymmetric encryption scheme. We then constructed a new three-party key transport protocol satisfying this notion of security in the AM.

We designed a new non-interactive MT-authenticator using time stamps from which a new authenticator was generated. This authenticator is used to translate AM protocols to secure protocols in the UM. We also discussed a variant of the MT-authenticator using counters.

From the new authenticator presented in this paper, we constructed a secure protocol 3PKDUM of Figure 5 as well as an authenticated Diffie-Hellman exchange of Figure 6.

References

1. Technical Committee ISO/IEC JTC 1. ISO/IEC DIS 11770-3 information technology - security techniques - key management - part 3: Mechanisms using asymmetric techniques. International Standard, 1996.
2. Olivier Baudron, David Pointcheval, and Jacques Stern. Extended notions of security for multicast public key cryptosystems. In *Proceedings of the 27th international Colloquium on Automata, Languages and Programming (ICALP '2000)*, volume 1853 of *LNCS*, pages 499–511. Springer-Verlag, July 2000.

3. M. Bellare, A. Boldyreva, and S. Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *LNCS*, pages 259–274. Springer-Verlag, 2000. Full version at http://www-cse.ucsd.edu/users/mihir/papers/key-distribution.html.
4. M. Bellare, R. Canetti, and H. Krawczyk. 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*, pages 412–428, 1998. Full version at http://www-cse.ucsd.edu/users/mihir/papers/modular.pdf.
5. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in Cryptology – Crypto 1998*, volume 1446 of *LNCS*, pages 26–45. Springer-Verlag, 1998.
6. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the First ACM Conference on Computer and Communications Security*, pages 62–73, 1993.
7. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology - Crypto 1993*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1994. Full version at http://www-cse.ucsd.edu/users/mihir/papers/eakd.pdf.
8. M. Bellare and P. Rogaway. Provably secure session key distribution - the three party case. In *Proceedings of the 27th ACM Symposium on the Theory of Computing*, pages 57–66, May 1995.
9. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – Eurocrypt 2001*, volume 2045 of *LNCS*, pages 453–474. Springer-Verlag, 2001. Full version at http://eprint.iacr.org/2001/040.ps.
10. W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.
11. S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28(2):270–299, 1984.
12. S. Goldwasser, M. Sipser, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM Journal of Computing*, 17(2):281–308, 1988.
13. G. Klyne and C. Newman. *Date and Time on the Internet: Timestamps.* RFC 3339, July 2003. http://www.faqs.org/rfcs/rfc3339.html, accessed on 19th Nov.,2003.
14. J. Kohl and C. Neuman. *The Kerberos Network Authentication Service (V5).* RFC 1510, September 1993. http://www.faqs.org/rfcs/rfc1510.html, accessed on 11th Nov.,2003.

A Proof of Theorem 3

Proof. Let $\mathcal{U}$ be a UM-adversary that interacts with λ_{Sig}^{Time}. We construct an AM-adversary $\mathcal{A}$ such that the outputs of running $\mathcal{U}$ in the UM and running $\mathcal{A}$ in the AM are equally distributed except with negligible probability. Adversary $\mathcal{A}$ runs $\mathcal{U}$ as a subroutine on a simulated interaction with a set of parties running λ_{Sig}^{Time}. Note that adversary $\mathcal{U}$ interacts with some imitated party A' in the unauthenticated networks while adversary $\mathcal{A}$ interacts with some imitated party A in the authenticated networks.

In the imitated run of the protocol, $\mathcal{A}$ first chooses and distributes keys for the imitated parties according to function $\mathcal{I}$. When setup is completed, the interaction runs according to the following rules:

1. Whenever $\mathcal{U}$ activates some imitated party A' for sending a message m to imitated party B', adversary $\mathcal{A}$ activates party A in the authenticated network to send m to B.
2. When some imitated party B' outputs 'B' received m̂ from A'', adversary $\mathcal{A}$ activates party B in the authenticated-links model with incoming message m̂ from A.
3. Whenever $\mathcal{U}$ corrupts a party, $\mathcal{A}$ corrupts the same party in the authenticated network and hands the corresponding information (from the simulated run) to $\mathcal{U}$.
4. Finally, $\mathcal{A}$ outputs whatever $\mathcal{U}$ outputs.

It is easy to verify that the behaviour of $\mathcal{A}$ is legitimate model except for step two. In step two, it is possible that party A is uncorrupted and the message $(\hat{m}, A, B)$ is not currently in the set of undelivered messages $\mathcal{M}$. More precisely, it is the case where B' outputs 'B' received $\hat{m}$ from A'', but either A was not activated for sending $\hat{m}$ to B, or B has already had the same output before $(\hat{m}, A, B)$ expires. This case indicates that $\mathcal{U}$ broke party A'. We denote this event by $\mathcal{B}$.

If step two can always be carried out such that $\mathcal{B}$ does not occur, then it is clear that the simulation run by $\mathcal{A}$ is perfect and the output of $\mathcal{A}$ and $\mathcal{U}$ is equally distributed. Thus, we need to show that event $\mathcal{B}$ occurs only with negligible probability $\epsilon(\kappa)$ where κ is the security parameter.

To prove the negligible probability of occurrence of event $\mathcal{B}$, we construct a forger $\mathcal{F}$ that breaks the signature scheme with probability $\epsilon(\kappa)/p$ where p is the number of parties. The forger $\mathcal{F}$ runs $\mathcal{U}$ on a simulated interaction with a set of parties running λ_{Sig}^{Time} with input a public key pk^*. Access to a signing oracle $\mathcal{O}(Sig_{sk^*}(\cdot))$ is allowed for modelling adaptive chosen message attacks. Note that the private key sk^* is not known by $\mathcal{F}$ and that $\mathcal{F}$ runs $\mathcal{U}$ in the same way as $\mathcal{A}$ runs $\mathcal{U}$. It then proceeds as follows:

1. $\mathcal{F}$ chooses and distributes key pairs (pk_i, sk_i) where $i \in \{1, \ldots, p\}$ for the imitated parties according to function $\mathcal{I}$, with the exception that the public key with a randomly chosen party P^* is replaced with the input key pk^*.
2. If $\mathcal{U}$ corrupts party P^*, then $\mathcal{F}$ fails and aborts the simulation.
3. If $\mathcal{F}$ is required to sign some value l on behalf of some imitated party $P_i \neq P^*$, $\mathcal{F}$ generates the signature using its knowledge of the private key sk_i. When party P^* is required to sign some value l, $\mathcal{F}$ asks its signing oracle $\mathcal{O}(Sig_{sk^*}(l))$ to obtain a valid signature.
4. If some party Q outputs 'Q received m from P^*', where P^* was not activated to send m to Q or Q has already output this value before, $\mathcal{F}$ is successful in breaking the signature scheme and outputs the signature in the last incoming message that Q received.

First note that $\mathcal{U}$'s view of the interaction with $\mathcal{F}$ is identically distributed to $\mathcal{U}$'s view of a real interaction in an unauthenticated network unless $\mathcal{F}$ fails and aborts the simulation. Let $\mathcal{B}^*$ be the event where $\mathcal{B}$ occurs in the simulated run of $\mathcal{U}$ within $\mathcal{F}$, and that the party broken by $\mathcal{U}$ is P^*. Note that if event $\mathcal{B}^*$ occurs, $\mathcal{F}$ does not fail. If $\mathcal{F}$ fails, event $\mathcal{B}^*$ does not happen. Thus the probability of failure in the simulation run by $\mathcal{F}$ is at least $\epsilon(\kappa)/p$ because of the random choice of P^*.

Assume that event $\mathcal{B}^*$ occurs. The occurrence of event $\mathcal{B}^*$ implies that the signature that Q received in its last incoming message is a valid signature of the value (m,Υ_{P^*}, Q), but P^* never generated this particular signature.

Two cases were identified that might have happened corresponding to event $\mathcal{B}^*$. First, P^* was not activated to send m to Q. Second, Q has already had the same output before. In the first case it is clear that if P^* was not activated to send m to Q, P^* never signed the message.

If Q outputs the same value twice before the time stamp associated with the first m has expired, then the signature that Q received in its last incoming message is valid and fresh such that it was signed with sk^* and on input the signature $\mathcal{V}$ returns 'TRUE'. However, recall that Q maintains $\mathcal{L}$ where accepted messages and their respective time stamps are stored, no messages together with the same time stamp are repeated in $\mathcal{L}$. It follows that P^* only sent m once and thus only signed the signature once where the time stamp was different to the time stamp in the last signature that Q received. Recall again that P^* generates a new time stamp with each new message, the time stamp in the last signature that Q received was not generated by P^*. Therefore $\mathcal{F}$ (on behalf of P^* for signatures) has only asked its signing oracle once. As a result, $\mathcal{F}$ never asked its oracle for the last signature.

Consequently, $\mathcal{F}$ has successfully broken the signature scheme.

Remark 4. In fact it is possible that Q outputs the same value twice or more times referring to records that are not in the current set of records in $\mathcal{L}$. If the sender sends the same message m many times, Q will accept m provided the time stamp is fresh and unique, and the signature is valid. Note that time stamps from the same sender are different for different messages.