Noname manuscript No.
(will be inserted by the editor)

Modular Proofs for Key Exchange: Rigorous Optimizations in the Canetti–Krawczyk Model*

Yvonne Hitchcock, Colin Boyd, Juan Manuel González Nieto

Information Security Research Centre, Queensland University of Technology,
GPO Box 2434, Brisbane Q 4001, Australia
e-mail: {y.hitchcock, c.boyd, j.gonzaleznieto}@qut.edu.au

The date of receipt and acceptance will be inserted by the editor

Abstract Various optimizations in the Canetti–Krawczyk model for secure protocol design are proven to preserve security. In particular it is shown that multiple authenticators may be safely used together; that certain message components generated by authenticators may be reordered (to be sent at a different time) or replaced with other values with certain precautions; and that protocols may be defined in the ideal world with session identifiers constructed during protocol runs. Consequently protocol designers now have a set of clear rules to optimize and customize their designs without fear of breaking the security proof. In order to obtain the required proofs, we find it necessary to slightly revise the authenticated links part of the Canetti–Krawczyk model.

1 Introduction

The computational approach to proofs of protocols for key establishment was pioneered by Bellare and Rogaway [5,2] and later extended to several other protocols in different scenarios by themselves and others [6,4,8,9]. Using *indistinguishability*, a familiar notion in cryptography, as the basis of their security definition, they showed that if an adversary can be successful then there must be an efficient way to break at least one of the underlying cryptographic algorithms. Proofs in this model are monolithic; they also tend to be long and complex thereby providing obstacles to practitioners who want to understand what may happen if a small change is made to the protocol.

In 1998 Bellare, Canetti and Krawczyk [3] introduced the idea of a modular approach to protocol proofs. Although the general structure of the model was quite similar to the earlier Bellare–Rogaway model the security definition used a simulatability approach rather than the earlier indistinguishability approach. Unfortunately it turned out that the simulatability definition for security was too strong to be practical, since many natural protocols that seem to be secure do not satisfy it. Therefore Canetti and Krawczyk recast that model a little later [11] with a new security definition based on indistinguishability. The new version retained the modular approach of the earlier paper by defining *authenticators* that can be used to transform protocols secure in an ideal world into protocols secure in the real world. We will refer to this new version as the CK approach throughout this paper. An authenticator can be regarded as a compiler that takes protocol messages from the world of ideal "magically authenticated" messages to the real world.

The CK approach forms the basis for reuse of protocol proofs since it is possible to match any protocol in the ideal world with any authenticator in order to form new proven secure protocols. The papers of Bellare, Canetti and Krawczyk [3,11] provided only a basic set of these building blocks. They showed how these may be used to prove the security of a small number of protocols, but did not produce any new designs. Recently

* Research funded by Australian Research Council through Discovery Project DP0345775

a number of additional building blocks have been provided. Because multiple new protocols result from each new component, this means that a large number of proven secure protocols can already be derived this way.

Most of the authenticators that exist today are built from a special kind of basic authenticator that can be applied to a single message in the ideal world independently of any other message. These so-called message transmission (MT-) authenticators transform one ideal-world message into multiple (typically three) real-world messages. This means that naïve application of the method to a protocol with multiple messages in the ideal world leads to a very inefficient protocol. For this reason real-world messages need to be reordered and combined if a protocol with acceptable efficiency is to be obtained. This process may include omission of certain elements and reuse (or overloading) of elements to take multiple roles. For example, nonces may often be replaced by other protocol values as long as they are known to be fresh. Use of MT-authenticators also allows a richer set of real-world protocols when different ideal-world messages are authenticated using different MT-authenticators.

An important element in the CK approach is the use of *session identifiers* (SIDs) which are used to ensure that messages are used in only one session. In the abstract model it is assumed that a unique SID is assigned and available to both participating parties before each protocol run commences. This is unrealistic in practice and so SIDs used in practical designs are typically formed from the combination of randomly chosen inputs from each participant.

Put together there are four types of protocol optimization that we have discussed:

1. mixing of different authenticators in the same protocol;
2. reordering of messages in real world protocols;
3. omission and re-use of certain protocol elements;
4. formation of SIDs from information exchanged during the protocol run.

The last three of these were suggested and illustrated in the original papers on the CK approach [3, 11] while the first one is very useful in extending the range of proven secure protocols that can be achieved. However, although there are heuristic arguments for why these actions are reasonable, up to now there has been no formal argument that these optimizations do not break the security proofs. This is not a very satisfactory situation since it means that in order to end up with a practical protocol with a claimed rigorous security proof, we need to take some steps without any guarantee that the proof remains valid. The purpose of this paper is to remedy this situation. In particular the contributions of this paper are:

- formal proofs of the first three optimization steps listed above provided certain conditions are met;
- extension of the model to realize these optimizations, and to enable session identifiers to be chosen after the protocol run has begun;
- illustration of how the resulting mechanical rules can be used to obtain practical protocols with fully rigorous security proofs.

The remainder of this paper proceeds as follows. Section 2 presents background on the CK approach sufficient to explain its use in the rest of the paper. This includes a list of the currently available ideal-world protocols and MT-authenticators. Section 3 describes how to alter the existing real world model in order to properly reorder messages and choose session identifiers after the protocol has begun. Section 4 proves that multiple authenticators may be used together in one protocol. The remaining optimizations are described in Section 5, namely omitting and moving various authenticator elements, and reusing values as nonces. We use a running example to illustrate use of the optimization rules, which applies two authenticators to the two messages of the standard ephemeral Diffie–Hellman protocol; one message is authenticated using a signature, and the other using encryption.

2 The Canetti–Krawczyk Model

In this section the CK approach is reviewed. Further details of the model can be found in the original papers [3, 11].

2.1 Key Establishment Protocols

In the CK model a 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. When first invoked within a party, a key exchange protocol π calls an initialization function that returns any information needed for the bootstrapping of the cryptographic authentication functions (e.g. private keys and authentic distribution of other parties' public keys). After this initialization stage, the party waits for activation. $\mathcal{A}$ may activate a party P_i in two ways:

1. by means of an $\mathsf{establish\text{-}session}(P_i, P_j, s, role)$ request, where P_j is another party with whom the key is to be established, s is a session-id string which uniquely identifies a session between the participants, and $role \in \{\mathsf{initiator}, \mathsf{responder}\}$. Note that the session-id is chosen by the adversary, with the restriction that it must 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 exists on how the adversary activates parties, depending on which of the following two adversarial models is being considered.

- The *authenticated-links model (AM)* defines an idealized adversary that is restricted to delivering messages faithfully between uncorrupted parties, if at all. That is, the adversary may only deliver messages where the specified (uncorrupted) sender actually sent that message to the specified recipient. The *AM–adversary* is not allowed to fabricate, modify, or replay messages of its choice except if the message is purported to come from a corrupted party. However, the adversary may decide not to deliver some messages from uncorrupted parties at all, or to deliver messages out of order.
- 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 perform some computations, update their internal state, and may output messages together with the identities of the intended receivers. Sessions may also add special messages to a local output to record the occurrence of important, security-related events. For example, when the run of a session finishes, a special key establishment event (P_i, P_j, s, κ) is recorded in the local output, to denote that a key κ has been established with party P_j. $\kappa = null$ denotes that the session has been $\mathsf{aborted}$. When $\kappa \neq null$ the session is said to be $\mathsf{completed}$. The adversary's view consists of all parties' public authentication information, output messages and local outputs, except for the established keys (κ-values) of completed sessions. Two sessions $(P_i, P_j, s, role)$ and $(P_i', P_j', s', role')$ are said to be *matching sessions* if $P_i = P_j'$, $P_j = P_i'$, and $s = s'$, i.e. if their 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 queries:

1. $\mathsf{corrupt}(P_i)$. With this query $\mathcal{A}$ learns 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. Whenever $\mathcal{A}$ corrupts a party, the event is recorded in the local output of that party, and that party is never activated again;
2. $\mathsf{session\text{-}key}(P_i, P_j, s)$. This query returns the session key (if any) accepted by P_i during a given session s with P_j. This event is recorded in the local output of that party;
3. $\mathsf{session\text{-}state}(P_i, P_j, s)$. This query returns all the internal state information of party P_i associated to a particular session s with P_j. This event is recorded in the local output of that party and is sometimes known as session corruption (as opposed to party corruption above);
4. $\mathsf{session\text{-}expiration}(P_i, P_j, s)$. This query can only be performed on a completed session. It is used for defining *forward secrecy* and ensures that the corresponding session key is erased from the party's memory. The session is thereafter said to be $\mathsf{expired}$. This event is recorded in the local output of that party;

5. test-session(P_i, P_j, s). To respond to this query, a random bit b is selected. If $b = 1$ then the session key is output. Otherwise, a random key is output 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*. A session is exposed if the adversary performs any of the following actions:
   - a session-state or session-key query to this session or to the matching session, or
   - a corrupt query to either partner before the session expires at that partner.

2.2 Definition of security

Security is defined based on a game played by the adversary. In this game $\mathcal{A}$ interacts with the protocol. In a first phase of the game, $\mathcal{A}$ is allowed to activate sessions and perform corrupt, session-key, session-state and session-expiration queries as described above. The adversary then performs a test-session query to a party and session of its choice. The adversary is not allowed to expose the test-session. $\mathcal{A}$ may then 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 to the test-session query was the session key or a random value, then halts. $\mathcal{A}$ wins the game if $b = b'$. The definition of security follows.

Definition 1 *A key establishment protocol π is called* session key (SK-) secure *with perfect forward secrecy (PFS) in the UM (resp. AM) if the following properties are satisfied for any UM (resp. 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 function in the security parameter.*

We define the advantage of $\mathcal{A}$ to be twice the probability that $\mathcal{A}$ wins, minus one. Hence the second requirement will be met if the advantage of $\mathcal{A}$ is negligible. Canetti and Krawczyk also provide a definition of SK-security *without PFS*. The only difference with respect to the above definition is that now the adversary is not allowed to expire sessions.

Protocol 1 shows the basic Diffie-Hellman protocol, which was proven SK-secure with PFS in the AM by Canetti and Krawcyzk [11]. The protocol assumes that primes p and q are known to each party before the protocol begins, where q is of length k bits (where k is a security parameter), q divides $p-1$ and g is of order q in $\mathbb{Z}_p^*$. Exponentiations are performed modulo p. This example will be used repeatedly later in this paper.

Shared Information: Parameters p, q, g, k.

A		B
Input: (A, B, s)		
$r_A \in_R \mathbb{Z}_q$		
$t_A = g^{r_A}$	$\xrightarrow{A, s, t_A}$	$r_B \in_R \mathbb{Z}_q$
		$t_B = g^{r_B}$
$K_{AB} = t_B^{r_A}$	$\xleftarrow{B, s, t_B}$	$K_{BA} = t_A^{r_B}$
Erase r_A		Erase r_B
'Established (A, B, s, K_{AB})'		'Established (B, A, s, K_{BA})'

Protocol 1. Basic Diffie-Hellman protocol

2.3 Authenticators

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 them. An authenticator is a compiler $\mathcal{C}$ 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 are in fact defined to achieve the stronger notion of *protocol emulation*. The authenticated protocol π' emulates protocol π in the UM, which informally means that whatever a UM adversary can do against π' can be done by an AM adversary against π. More formally, let $\mathcal{U}$ be an UM adversary interacting with protocol π', and let $\text{UNAUTH}_{\pi',\mathcal{U}}$ denote the concatenation of the cumulative local output of all parties running π' and the output of $\mathcal{U}$. Similarly, let $\mathcal{A}$ be an AM adversary interacting with protocol π, and $\text{AUTH}_{\pi,\mathcal{A}}$ the concatenation of the cumulative local output of all parties running π and the output of $\mathcal{A}$.

Definition 2 (Protocol Emulation) *Protocol* π' emulates *protocol* π *in the UM if for any UM adversary* $\mathcal{U}$ *there exists an AM adversary* $\mathcal{A}$ *such that* $UNAUTH_{\pi',\mathcal{U}}$ *and* $AUTH_{\pi,\mathcal{A}}$ *are computationally indistinguishable.*

Definition 3 (Authenticator) *An* authenticator *is a compiler* $\mathcal{C}$ *that given a protocol* π *outputs a protocol* $\pi' = \mathcal{C}(\pi)$ *such that* π' *emulates* π *in the UM.*

Authenticators can be constructed from *message transmission authenticators*. An MT-authenticator is a protocol that enables delivery of messages in the UM in an authenticated manner. However, each message is authenticated almost independently of all other messages. (Although the same long term keys may often be used by an authenticator to authenticate many messages, other data used by the authenticator such as a nonce or signature is generated separately for each message to be authenticated.) To translate an SK-secure protocol in the AM to an SK-secure protocol in the UM an MT-authenticator can be applied to each message and the resultant sub-protocols combined to form one overall SK-secure protocol in the UM. In order to define the security of an MT-authenticator, it is necessary to define the MT-protocol in the AM as follows.

Definition 4 (MT-Protocol) *The* MT*-protocol in the AM has the following properties:*

- *upon activation within party* A *on external request* (B, m)*, party* A *sends the message* (A, B, m) *to party* B *and outputs '*A *sent* m *to* B*.'*
- *upon receipt of a message* (A, B, m)*,* B *outputs '*B *received* m *from* A*.'*

Definition 5 (MT-Authenticator) *An* MT-authenticator *is a protocol that emulates the* MT*-protocol in the UM.*

We note that a protocol having no output is in fact a secure authenticator, because there exists an AM adversary (which delivers no messages) with computationally indistinguishable output to the authenticator. However, such an authenticator is trivial and of no practical use. Some authors have introduced non-triviality requirements to eliminate such useless protocols [10, Section 6], [12, Section 3.1.1], [1]. However, no such requirement currently exists in the CK model. Indeed, we consider that since trivial authenticators are unlikely to be examined by users of the model, such a requirement is of limited practical value.

Figures 1 and 2 show two MT-authenticators proposed by Bellare *et al.* [3]. They authenticate the sending of a message m from party A to party B. In the first one, party A signs the message together with a random challenge sent by B and B's identity. In the second one, B encrypts a random challenge using A's public key. A then uses the random challenge as the key in a message authentication code (MAC) of the message and the recipient's identity. Note that in both MT-authenticators m is sent in all protocol exchanges.

Let λ be an MT-authenticator and let $\mathcal{C}_\lambda$ be a compiler that on input a protocol π outputs another protocol π', which is identical to π except that all outgoing and incoming messages are processed through λ. Thus, whenever π specifies sending a message m to a party, λ is activated to send the same message to the same party. Whenever π' is activated on an incoming message, the message is handed to λ.

Theorem 1 ([3]) $\mathcal{C}_\lambda$ *as described above is an authenticator.*

Finally, we state the result that allows us to turn SK-secure protocols in the AM into SK-secure protocols in the UM by application of authenticators.

Theorem 2 ([11]) *Let protocol* π *be SK-secure in the AM, and let* $\mathcal{C}$ *be an authenticator. Then, the protocol* $\pi' = \mathcal{C}(\pi)$ *is SK-secure in the UM.*

A		B
Choose message m		
'A sent m to B'	$\xrightarrow{m}$	
	$\xleftarrow{m, N_B}$	$N_B \in_R \{0,1\}^k$
$\sigma = \mathrm{Sig}_A(m, B, N_B)$	$\xrightarrow{m, \sigma}$	
		Check $\sigma = \mathrm{Sig}_A(m, B, N_B)$; if not then abort.
		'B received m from A'

Fig. 1 Signature-based MT-authenticator [3]

A		B
Choose message m		
'A sent m to B'	$\xrightarrow{m}$	$N_B \in_R \{0,1\}^k$
		$c = \mathcal{E}_{PK_A}(N_B)$
	$\xleftarrow{m, c}$	
$c = \mathcal{D}_{SK_A}(c)$		
$\mu = \mathrm{MAC}_{N_B}(m, B)$	$\xrightarrow{m, \mu}$	
		Check $\mu = \mathrm{MAC}_{N_B}(m, B)$; if not then abort.
		'B received m from A'

Fig. 2 Encryption-based MT-authenticator [3]

Protocol 2 shows the result of the direct application of a compiler built from the signature authenticator of Figure 1 to the basic Diffie-Hellman protocol (Protocol 1). As noted just after the proof of Theorem 4 later in this paper, the identity of the sender of each message included in each message in Protocol 1 is not necessary to the security of the protocol, and was omitted before applying the authenticator to produce Protocol 2. (It is assumed that if they are necessary for practical purposes, the identity of sender and recipient are provided automatically, for example in a header field of the message.) The new protocol, which by virtue of Theorem 2 is SK-secure in the UM, consists of 6 message flows. This increase of message exchanges is clearly unsatisfactory. Canetti and Krawczyk [11] argued that Protocol 2 can be turned into a three-pass protocol (Protocol 3) by:

- sending message 5 at the same time as message 1 but omitting t_B,
- sending messages 4 and 6 together with message 2, and
- setting $N_A = t_A$ and $N_B = t_B$.

However, the arguments given for such optimization steps were heuristic. (The inclusion of the identity of the sender in each message has been retained from Canetti and Krawczyk's original description in Protocol 3.) As we show in subsequent sections, the formal justification of the optimization steps is not trivial, and we have only achieved it by modifying the underlying model.

2.4 A Library of Building Blocks

Table 1 summarizes the existing AM protocols and MT-authenticators that can be used as building blocks in the CK model. We give a descriptive name for each, as well as indicating the computational assumption on which it depends. The first two authenticators listed were proposed and proven secure in the original paper defining the modular approach [3] while the first two AM protocols as well as the third authenticator listed come from the later paper of Canetti and Krawczyk [11]. The remaining components have been proposed and proven secure in different papers published recently [15,16,13,7].

In principle, every AM protocol in the table can be used with every authenticator. Moreover, when the AM protocol has two messages, each of these messages can use a different authenticator, as we shall show in Section 4. This is the case for Diffie-Hellman and server-based key transport (the other AM protocols shown

A		B
$r_A \in_R \mathbb{Z}_q$ $t_A = g^{r_A}$		
'A sent (s, t_A) to B'	$\xrightarrow{s, t_A}$	$N_B \in_R \{0,1\}^k$
$\sigma_A = \mathrm{Sig}(s, t_A, B, N_B)$	$\xleftarrow{s, t_A, N_B}$	
	$\xrightarrow{s, t_A, \sigma_A}$	
		Check $\sigma_A = \mathrm{Sig}_A(s, t_A, B, N_B)$; if not then abort. '$B$ received (s, t_A) from A' $r_B \in_R \mathbb{Z}_q$ $t_B = g^{r_B}$
	$\xleftarrow{s, t_B}$	'B sent (s, t_B) to A'
$N_A \in_R \{0,1\}^k$	$\xrightarrow{s, t_B, N_A}$	$\sigma_B = \mathrm{Sig}(s, t_B, A, N_A)$
	$\xleftarrow{s, t_B, \sigma_B}$	
Check $\sigma_B = \mathrm{Sig}_B(s, t_B, A, N_A)$; if not then abort. '$A$ received (s, t_B) from A' $K_{AB} = t_B^{r_A}$ Erase r_A 'Established (A, B, s, K_{AB}))'		$K_{BA} = t_A^{r_B}$ Erase r_B 'Established (B, A, s, K_{BA})'

Protocol 2. Unoptimized Diffie-Hellman protocol in the UM (one MT-authenticator)

A		B
Input: (A, B, s) $r_A \in_R \mathbb{Z}_q$ $t_A = g^{r_A}$	$\xrightarrow{A, s, t_A}$	$r_B \in_R \mathbb{Z}_q$ $t_B = g^{r_B}$ $K_{BA} = t_A^{r_B}$ Erase r_B
Check $\sigma_B = \mathrm{Sig}_B(B, s, t_B, t_A, A)$; if not then abort. $K_{AB} = t_B^{r_A}$ Erase r_A 'Established (A, B, s, K_{AB})'	$\xleftarrow{B, s, t_B, \sigma_B}$	$\sigma_B = \mathrm{Sig}_B(B, s, t_B, t_A, A)$
$\sigma_A = \mathrm{Sig}_A(A, s, t_A, t_B, B)$	$\xrightarrow{A, s, \sigma_A}$	Check $\sigma_A = \mathrm{Sig}_A(A, s, t_A, t_B, B)$; if not then abort. 'Established (B, A, s, K_{BA})'

Protocol 3. Original optimized Diffie-Hellman protocol using signatures for authentication

have only one message). This means that in total there are 60 proven secure protocols that can be derived from the components in Table 1. Not all of these protocols may be of practical significance but a great many of them correspond closely to published protocols in the literature. For example, Bellare and Rogaway's 3PKD protocol [2] can be derived by applying the authenticator using MACs with existing shared keys to the server-based key transport AM protocol. Another example is a standardised key agreement protocol [14] that can be derived by applying the signature authenticator to the basic Diffie-Hellman protocol. In addition to providing new proofs for existing protocols, the components have also been used to construct new proven secure protocols [15,16,13,7].

AM protocols	Computational assumption
Diffie-Hellman	DDH
Key transport	CCA-secure encryption
ElGamal type	Gap-DH
Server-based key transport	CPA-secure encryption

Authenticators	Computational assumption
Signature-based	Secure signature
Encryption-based	Secure MAC and CCA-secure encryption
MAC with existing shared key	Secure MAC
Password-based	CCA-secure encryption
Statically keyed	CDH and random hash function

Table 1 Existing proven secure components

3 The Authenticated Links Model Revisited

This section explains a variation of the original AM definition, which is required in order to make the subsequent optimizations formally valid. We first explain what can go wrong if the current AM definition is used, and then explain the details of the revised model.

3.1 Potential Problems of Reordering AM Messages

Although Protocol 3 was constructed using the signature-based authenticator of Fig. 1, certain outputs that the authenticator generates are omitted, such as the output 'A sent A, s, t_A to B' by A, or the output 'B received A, s, t_A from A' by B. Outputs of this type are hereafter referred to as sent and received outputs. Although these outputs do not appear to affect the security of the final key exchange protocol directly, they are important for the analysis of the protocol's security. Attempts to insert the sent and received outputs into the optimized UM protocol reveal that they cannot occur in the same order as in the AM protocol. For example, A's output 'A sent A, s, t_A to B' could be inserted before either the first or third protocol message. If it is inserted before the first message, B must output 'B sent B, s, t_B to A' before it outputs 'B received A, s, t_A from A', which is contrary to the AM protocol (since B must receive the message from A before it sends its own message to A according to the AM protocol). If A's output is inserted before the third message, then t_A becomes known to the adversary and to B in the UM before it would have become known in the AM. Although this prior knowledge may not cause a problem for the Diffie-Hellman protocol, it is conceivable that other protocols may exist where early release of a message (even though unauthenticated) may compromise security. It is clear that the optimized Protocol 3 does not emulate Protocol 1 from the AM due to the different order of the outputs.

In addition to the differences in placement of sent and received messages, there is another reason why the optimized UM protocol does not emulate the AM protocol. In the original AM protocol, the responder receives the initiator's message, chooses its secret exponent, calculates and outputs the secret key, erases the secret exponent and sends its Diffie-Hellman value to the initiator all in one activation. However, in the optimized UM protocol, the responder does not output the secret key until it receives the initiator's authentic message in its second activation. A UM adversary that never delivers the second message to the responder never enables the responder to output the initiator's message or the secret key. Therefore emulation is not achieved in the UM because there is no AM adversary that can force the responder in the AM protocol to split its operations between two activations, instead of completing them together. Although the emulation between AM and UM is broken in the above example, this does not imply that the UM key exchange described by Canetti and Krawczyk is insecure. However, it would be quite feasible to destroy the security of the UM protocol when splitting the activities from one activation in the AM into two activations in the UM, for example by outputting the secret key before receiving the initiator's authentic message. Although it may seem obvious that this would break the security of the protocol, restrictions on other steps, such as the erasure of the secret exponent are not so clear.

There are two possible strategies for proving that optimized UM protocols are secure realizations of AM protocols. One is to start with an AM protocol suitable for emulation by an optimized UM protocol. The other is to prove that even though the UM protocol may not emulate the AM protocol, it still satisfies the definition of SK-security. The latter approach seems attractive, but does not seem feasible unless a new proof of security is generated for each optimized UM protocol. It seems impossible to prove a generic theorem to enable AM protocol components to be moved during the optimization process in the UM. Furthermore, to generate a new proof for each UM protocol defeats the main advantage of the CK-model, namely its modular proof approach. Therefore, the option of starting with an AM protocol suitable for emulation by an optimized UM protocol is now examined.

3.2 Formulating AM Protocols Suitable for Optimized Emulation

The two main problems raised in the example above are the splitting of steps carried out in one AM activation across two UM activations, and the use of an AM value in the UM before it has been sent in the AM or received authentically. A third problem is that it may be desired to use the protocol with $s = (t_A, t_B)$. However, this is not possible in the current formulation because s must be chosen before t_A and t_B are chosen. To overcome the problem of using AM values in the UM before they have been sent in the AM, the AM message could be shifted to an earlier position. However, this requires waiting for that AM message to be sent authentically before carrying out further actions specified in the AM protocol. This is likely to cause the resultant UM protocol to be non-optimal. The only other alternative is to specify in the AM protocol that the AM message may be released at an earlier time than it is sent authentically. This could be done in a number of ways, such as addressing the message to the AM adversary, $\mathcal{A}$, instead of the other protocol participant, and having $\mathcal{A}$ forward the message to the desired recipient. Another method would be to add the value to the party's local output where $\mathcal{A}$ can read it, and have $\mathcal{A}$ input the value to the other party. However, the method with the most clarity seems to be to allow unauthentic messages in the AM. To avoid confusion, we specify a new model, the *hybrid model* or HM, which captures this concept. HM protocols must indicate which messages are authenticated, and which are not. Specifically, the HM adversary, $\mathcal{H}$, can perform any action originally allowed in the AM, and is still restricted to faithfully delivering authentic HM messages at most once and to the specified recipient with the actual sender correctly specified (if the messages are delivered at all). $\mathcal{H}$ may not deliver a fabricated or modified message as an authentic HM message. However, $\mathcal{H}$ may fabricate any message of her choosing at any time and deliver it as an unauthentic message in the HM to any recipient with any party specified as the sender. $\mathcal{H}$ may use any information at her disposal to fabricate unauthentic HM messages, including the contents of previously sent authenticated HM messages that may or may not have been delivered.

In addition, HM protocols may begin before being assigned a session identifier, and may specify at what point a session identifier is to be assigned (by the adversary or the party running the protocol). In this case, instead of the adversary inputting $\mathsf{establish\text{-}session}(P_i, P_j, s, \mathsf{role})$ to begin a session at party P_i to exchange a secret key with party P_j with session identifier s and with P_i acting as an initiator if $\mathsf{role} = \mathsf{initiator}$ or a responder if $\mathsf{role} = \mathsf{responder}$, the value s may be replaced with ϕ to indicate no session identifier has yet been chosen. The adversary may input the value of the session identifier at a later point in the protocol, or the protocol may specify how the party running the protocol is to construct the session identifier. Any session that has not yet been assigned a session identifier is not considered to have any matching session. However, once a session identifier has been assigned, the definition of matching remains basically unchanged from the AM and UM (i.e. two sessions $(P_i, P_j, \phi, \mathsf{role})$ and $(P'_i, P'_j, \phi, \mathsf{role}')$ match if they have been assigned the same session identifier and $P_i = P'_j$ and $P_j = P'_i$). As in the AM and UM, the adversary is restricted to choose a session identifier for a session of a party that has never been used as a session identifier by another session of that party. The corrupt, session-key, session-state, session-expiration and test-session queries remain unchanged, as does the definition of exposure.

We observe that running a protocol π' that never sends any authentic messages in the HM and has a session identifier assigned at the very beginning is equivalent to running the same protocol in the UM. Also, running a protocol π in the HM that never sends any unauthentic messages and has a session identifier assigned at the very beginning with an adversary that never delivers any unauthentic messages is equivalent to running the same protocol in the AM. When a protocol from the HM is converted to a UM protocol,

authenticators need not be applied to unauthentic HM messages, only authentic ones. This is captured by the following definitions and theorem.

Definition 6 (HM-emulation)
Let $\mathcal{H}$ be an HM adversary interacting with protocol π, and let $HYBRID_{\pi,\mathcal{H}}$ denote the concatenation of the cumulative local output of all parties running π and the output of $\mathcal{H}$.

Protocol π' HM-emulates *protocol π in the HM if for any HM adversary $\mathcal{U}$ there exists an HM adversary $\mathcal{H}$ such that $HYBRID_{\pi,\mathcal{H}}$ and $HYBRID_{\pi',\mathcal{U}}$ are computationally indistinguishable.*

Definition 7 (HM-Authenticator) *An* HM-authenticator *is a compiler $\mathcal{C}$ that given a protocol π in the HM outputs a protocol $\pi' = \mathcal{C}(\pi)$ such that π' HM-emulates π in the HM.*

Of course, with respect to the previous two definitions we are ultimately interested in protocols where π' is in the UM. However, since the UM is a special case of the HM, to facilitate future proofs we allow π' to be in the HM in the above definitions.

Definition 8 *Let λ be an* MT*-authenticator and let $\chi : \mathbb{N} \to \{\lambda, \phi\}$ be a mapping from message number to either the authenticator or ϕ (where ϕ indicates no authenticator), and message numbers of unauthentic messages are required to be mapped to ϕ. $\mathcal{C}_\chi$ is defined to be a HM-compiler that given a protocol π in the HM produces a protocol π' in the HM which is the same as π except that it uses the authenticator λ to deliver some of the authentic messages. Which messages are to be sent using λ is specified by χ, where a mapping of a message to λ indicates that the authenticator should be used and a mapping to ϕ indicates no authenticator is to be used and the message is to be sent in π' as an authentic message in the HM if it was authentic in π, or as an unauthentic message if it was unauthentic in π.*

Theorem 3 *If λ is an* MT*-authenticator (i.e. in the UM it emulates the* MT*-protocol from the AM) then:*

1. *the HM-compiler $\mathcal{C}_\chi$ given by Definition 8 is an HM-authenticator.*
2. *If $\pi' = \mathcal{C}_\chi(\pi)$ and π is SK-secure, then π' is also SK-secure.*

Proof The proof is similar to that of Theorem 3 by Bellare et al. [3], except for a slight modification in the simulation to cater for the HM and unauthentic messages, as well as queries specific to key exchange protocols such as the test session query and session key queries. It proceeds as follows:

Let π be a protocol. We begin by showing that $\pi' = \mathcal{C}_\chi(\pi)$ HM-emulates π in the HM. For clarity, we denote the model in which π runs HM_1 and the model in which π' runs HM_2. That is, let $\mathcal{H}_2$ be an HM_2 adversary that works against π'. We construct an HM_1 adversary $\mathcal{H}_1$ such that $\mathrm{HYBRID}_{\pi,\mathcal{H}_1}$ and $\mathrm{HYBRID}_{\pi',\mathcal{H}_2}$ have negligible statistical distance.

Adversary $\mathcal{H}_1$ operates in HM_1, with (real) parties $P_1, \ldots, P_n$ running π. $\mathcal{H}_1$ also simulates the parties $P'_1, \ldots, P'_n$ in HM_2 for $\mathcal{H}_2$, the HM_2 adversary. Parties $P'_1, \ldots, P'_n$ do not actually exist, and neither do they run protocol π'. $\mathcal{H}_1$ simply generates output and messages on behalf of these imaginary parties for the benefit of $\mathcal{H}_2$, since $\mathcal{H}_2$ interacts with HM_2 parties. $\mathcal{H}_1$ runs λ on behalf of the imaginary parties $P'_1, \ldots, P'_n$, and orchestrates an interaction between $\mathcal{H}_2$ and λ, while using the parties $P_1 \ldots P_n$ in HM_1 to play the (upper-layer) protocol π to generate the messages sent and received via λ.

First $\mathcal{H}_1$ invokes λ. Next, $\mathcal{H}_1$ proceeds according to the following rules:

1. Whenever $\mathcal{H}_2$ activates P'_i with an external request, $\mathcal{H}_1$ activates (in its HM_1 model) P_i with the same request. For each outgoing authentic message that P_i generates in this activation, say for P_j, that π' specifies is to be delivered via λ, $\mathcal{H}_1$ activates protocol λ with external request for sending that message from P'_i to P'_j. Next $\mathcal{H}_1$ hands $\mathcal{H}_2$ all the outgoing messages generated by λ on behalf of P'_i, as well as any outgoing unauthentic messages generated by P_i, any outgoing authentic messages generated by P_i that are not to be delivered by λ (and must therefore be treated by $\mathcal{H}_2$ as authentic HM_2 messages awaiting delivery), and any outgoing requests and outputs that P_i may have generated.
2. Whenever $\mathcal{H}_2$ activates P'_i with an incoming authentic message m purportedly from P'_j, (so not generated by the MT-authenticator), $\mathcal{H}_1$ delivers the same message to P_i as authentic with sender P_j. Any outgoing messages and external requests generated by P_i are handled as above.

3. Whenever $\mathcal{H}_2$ activates P'_i with an incoming unauthentic message m purportedly from P'_j, and according to π', the next message that P'_i would receive is one not generated by the MT-authenticator, $\mathcal{H}_1$ delivers the same message to P_i as unauthentic with sender P_j. Any outgoing messages and external requests generated by P_i are handled as above.
4. Whenever $\mathcal{H}_2$ activates P'_i with an incoming unauthentic message m, if (according to π') the next message P'_i should receive is one via the authenticator, $\mathcal{H}_1$ first activates λ with this incoming message. Any outgoing messages and external requests generated by λ on behalf of P'_i are handed by $\mathcal{H}_1$ to $\mathcal{H}_2$.
5. If any activation of λ outputs 'P'_i received m from P'_j' on behalf of P'_i, then $\mathcal{H}_1$ activates, in HM_1, party P_i with incoming authentic message m from party P_j. Each outgoing message that P_i generates as a result of this activation is handled as above.
6. Whenever $\mathcal{H}_2$ corrupts party P'_i, $\mathcal{H}_1$ corrupts P_i in HM_1. $\mathcal{H}_1$ then hands $\mathcal{H}_2$ the internal data of P_i together with the information regarding all activations of λ by (the imaginary) P'_i.
7. Whenever $\mathcal{H}_2$ expires a session at party P'_i, $\mathcal{H}_1$ expires the same session at P_i in HM_1. $\mathcal{H}_1$ will no longer return any local output (i.e. the session key) from this session to $\mathcal{H}_2$ upon corruption of P'_i. $\mathcal{H}_2$ may no longer perform a session key query on this session at party P'_i, and $\mathcal{H}_1$ may no longer perform a session key query on this session at party P_i.
8. Whenever $\mathcal{H}_2$ makes a session key query on session s at party P'_i, $\mathcal{H}_1$ makes a session key query on session s at party P_i, and returns to $\mathcal{H}_2$ the key it received from its own query.
9. Whenever $\mathcal{H}_2$ chooses a test session s belonging to P'_i, $\mathcal{H}_1$ chooses session s belonging to P_i as its test session and returns to P'_i the same key that it received in answer to its own test session query.
10. If $\mathcal{H}_2$ corrupts a session within some party P'_i then $\mathcal{H}_1$ corrupts the same session within P_i and hands the corresponding information back to $\mathcal{H}_2$ together with the information regarding activations of λ by (the imaginary) P'_i for that session.
11. $\mathcal{H}_1$ outputs whatever $\mathcal{H}_2$ outputs.

We first need to show that the above description of the behaviour of $\mathcal{H}_1$ is a legitimate behaviour of an HM_1 adversary. The above steps are easy to verify as legal moves for $\mathcal{H}_1$, except for step 5. In this case, it could be possible that the triple (P_j, P_i, m) is not currently in the set of authentic undelivered messages in HM_1, and P_j (and the originating session within P_j) is uncorrupted. It is easy to see that if this is not the case, namely if we assume that step 5 can always be carried out, then the above construction satisfies the requirement for $\mathrm{HYBRID}_{\pi,\mathcal{H}_1}$ and $\mathrm{HYBRID}_{\pi',\mathcal{H}_2}$ to have negligible statistical distance. This holds since the simulated run with $\mathcal{H}_2$, together with the activations of the parties in HM_1 is an exact imitation of a run of $\mathcal{H}_2$ in HM_2 with parties running π'.

Thus, it remains to show that the probability that $\mathcal{H}_1$ cannot carry out step 5 is negligible. Assume that protocol λ outputs 'P'_i received m from P'_j' on behalf of an uncorrupted party P'_i. We show that, except with negligible probability, the following events occur in HM_1:

1. The triple (P_i, P_j, m) was added to the set of authentic undelivered messages.
2. This triple was not yet deleted from the set of authentic undelivered messages (i.e. had not previously been delivered as an authentic message).

To see (1), notice first that if, in the simulated run of $\mathcal{H}_2$, λ outputs 'P'_i received m from P'_j' on behalf of party P'_i, then λ had been invoked on behalf of P'_j for sending m to P'_i.

Otherwise, we construct from $\mathcal{H}_2$ a UM adversary $\mathcal{U}$ that contradicts the assumption that λ emulates MT. $\mathcal{U}$ will simply run $\mathcal{H}_2$ (and complete $\mathcal{H}_2$'s missing information by running a simulated copy of π). Whenever $\mathcal{H}_2$ activates an imaginary party simulated by $\mathcal{U}$, say P'_i, and during that activation that party ought to send a message authenticated using λ, $\mathcal{U}$ activates the corresponding party with whom it interacts, P''_i to send the same message to the same recipient using λ. $\mathcal{U}$ passes messages and outputs pertaining to the authenticator λ between the parties in its model and $\mathcal{H}_2$. $\mathcal{U}$ simulates the rest of the messages and outputs pertaining to π' for $\mathcal{H}_2$ itself. $\mathcal{U}$ also simulates for $\mathcal{H}_2$ the other possible queries, such as party corruption and session corruption, and completes any missing information it needs for the authenticator by corrupting the corresponding party or session in its own model. Observe that the proofs of security of the authenticators already existing in the literature do not consider expiry, session key and test session queries. However, this is not a problem, since the only information $\mathcal{H}_2$ can receive from such queries is the session key. $\mathcal{U}$ is able to answer these queries without interacting with the parties $P''_1, P''_2, \ldots, P''_n$ in its own model, since the session

key cannot depend on any information generated by the authenticator λ (otherwise the session key would not be able to be specified in π). Now, in the global output of the parties running λ with $\mathcal{U}$ the event that some party accepts a message that was never sent occurs with non-negligible probability. Yet, in the AM, with *any* adversary, this event never occurs. It now follows from the construction that P_j has sent m to P_i in HM_1. Thus the triple (P_i, P_j, m) was added to the set of authentic undelivered messages.

To see (2), notice that, in the simulated run of $\mathcal{H}_2$, λ has not previously output 'P'_i received m from P'_j' on behalf of P'_i. Otherwise it would be the case that λ outputs 'P'_i received m from P'_j' twice on behalf of P'_i, and as above one can construct from $\mathcal{H}_2$ a UM adversary $\mathcal{U}$ that contradicts the assumption that λ emulates MT. It now follows from the construction that P_j was not previously activated in HM_1 with incoming authentic message m from P_j. Thus the triple (P_i, P_j, m) was not deleted from the set of authentic undelivered messages. Hence the HM-compiler $\mathcal{C}_\chi$ given by Definition 8 is an HM-authenticator.

We now show that if $\pi' = \mathcal{C}_\chi(\pi)$ and π is SK-secure, then π' is also SK-secure. Observe that since $\mathcal{C}_\chi$ is an HM-authenticator, if π satisfies requirement 1 of Definition 1, so must π'. Otherwise, the global outputs from running π and π' would be easily distinguishable.

In addition, if π' violates requirement 2 of Definition 1, then so must π. This is true since if there exists an adversary $\mathcal{H}_2$ against π' that has an advantage in outputting the correct bit b, and there does not exist an adversary $\mathcal{H}_1$ with the same advantage, then $\mathrm{HYBRID}_{\pi,\mathcal{H}_1}$ and $\mathrm{HYBRID}_{\pi',\mathcal{H}_2}$ will not have negligible statistical distance, since the output of $\mathcal{H}_2$ will correctly indicate whether the answer to the test session was real or random more often than the output of $\mathcal{H}_1$. □

Now that the necessary extension to the model has been described, it remains to specify how to change the existing AM protocols to take advantage of it. Firstly, it is observed that changes to AM protocols into HM protocols require new proofs of security. However, different variations of an HM protocol may be possible, with different versions being better for use with different authenticators. The use of an HM template is therefore proposed. An HM template shows the possible actions for each party, and specifies which actions are prerequisites for others, and which may be performed in parallel. It also specifies which actions must be performed in a single activation. Figure 3 shows the HM template for the Diffie-Hellman protocol at party A. All messages are assumed to be authentic, unless otherwise specified, and actions that must be performed in the same activation are shown in the one box and joined with "AND." An arrow from one step to another indicates that the first (prerequisite) step must be completed before the second is begun. Several arrows to the one step mean that all of the prerequisite steps must be completed before that step is begun, unless the arrows are combined with "or," in which case only one of the prerequisite steps need be completed. Optional steps are shown using dashed boxes. The session identifier s received from the adversary must be the same as that received in the authentic message from B, otherwise the protocol halts without outputting a secret key. The template for party B is identical, except for the renaming of A to B, B to A, x to y and y to x.

Assumption 1 (Decisional Diffie-Hellman (DDH) [11]) *Let k be a security parameter. Let primes p and q be such that q is of length k bits and divides $p-1$, and let g be of order q in $\mathbb{Z}_p^*$. Then the probability distributions of quintuples $Q_0 = \{\langle p, g, g^x, g^y, g^{xy}\rangle\} : x, y, \stackrel{R}{\leftarrow} \mathbb{Z}_q\}$ and $Q_1 = \{\langle p, g, g^x, g^y, g^z\rangle\} : x, y, z \stackrel{R}{\leftarrow} \mathbb{Z}_q\}$ are computationally indistinguishable.*

Theorem 4 *Any Diffie-Hellman based HM protocol where the actions of each party satisfy the requirements specified in Figure 3 is SK-secure, provided no deadlock conditions occur (i.e. provided both parties are not waiting for a message from the other party simultaneously) and the decisional Diffie-Hellman assumption is true.*

Proof The proof is similar to Canetti and Krawczyk's Theorem 8 [11] as follows:

To see that if two uncorrupted parties complete matching sessions then they both output the same key (requirement 1 of Definition 1), note that if both P_i and P_j are uncorrupted during the exchange of the key and both complete the protocol with matching sessions, then they both establish the same key, which is g^{xy}. This is because the session identifier s uniquely binds the values of g^x and g^y to these particular matching sessions and differentiates them from other exponentials that the parties may exchange in other (possibly simultaneous) sessions.

We now show that requirement 2 of Definition 1 is met. Assume to the contrary that there is an adversary $\mathcal{H}$ in the HM against the protocol that has a non-negligible advantage in guessing correctly whether the

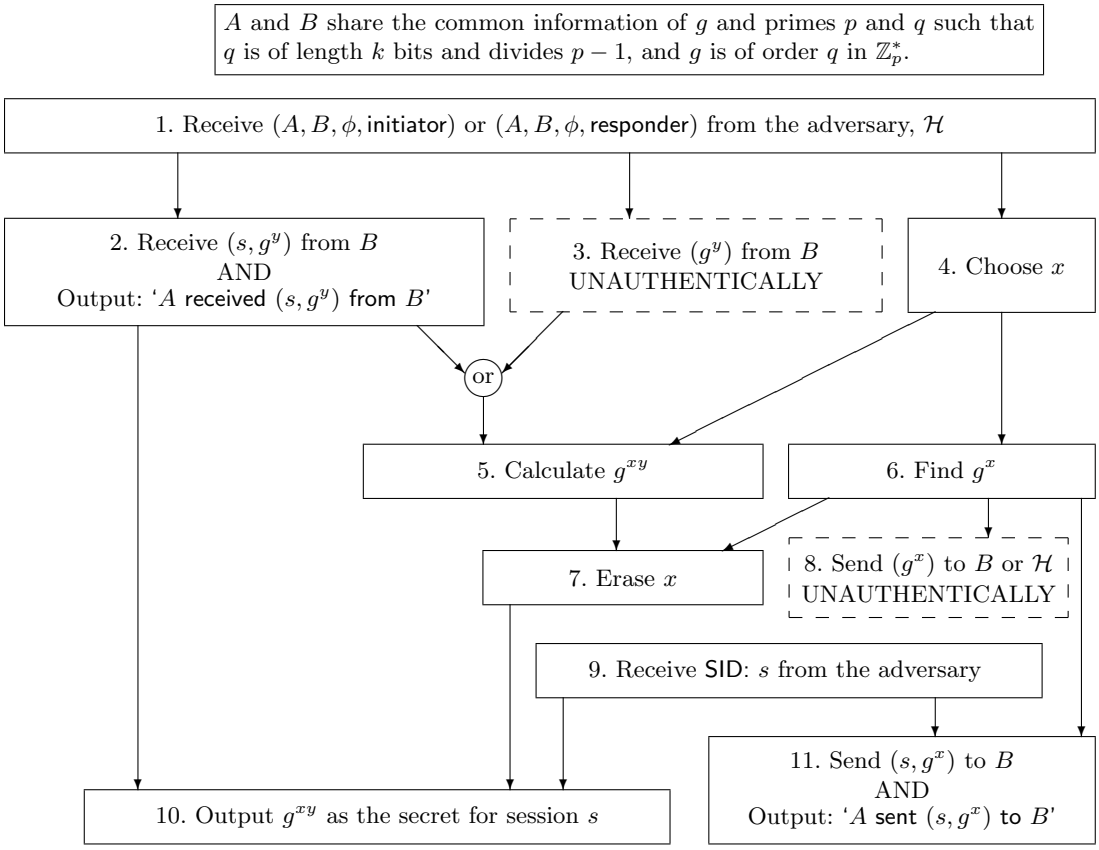

Fig. 3 Possible step order for the Diffie-Hellman protocol in the AM, with optional steps in dashed boxes

response to a test-query is real or random. Out of this attacker $\mathcal{H}$, we construct an algorithm $\mathcal{D}$ that distinguishes between the distributions Q_0 and Q_1 with non-negligible probability, thus reaching a contradiction with Assumption 1. The input to $\mathcal{D}$ is denoted by $(p, g, \alpha^*, \beta^*, \gamma^*)$ and is chosen from Q_0 or Q_1 each with probability 1/2. Let l be an upper bound on the number of sessions invoked by $\mathcal{H}$ for a particular party in any interaction. $\mathcal{D}$ uses $\mathcal{H}$ as a subroutine and proceeds as follows:

1. Choose $A, B \xleftarrow{R} \{P_1, P_2, \ldots, P_n\}$ and $r_A, r_B \xleftarrow{R} \{1, 2, \ldots, l\}$. We will refer to A's ${r_A}^{\text{th}}$ session and B's ${r_B}^{\text{th}}$ session the *target sessions*.
2. Invoke $\mathcal{H}$ on a simulated interaction in the HM with parties $P_1, \ldots, P_n$ running the DH protocol. Hand $\mathcal{H}$ the values p and g as the public parameters for the protocol execution.
3. Whenever $\mathcal{H}$ activates a party to establish a new session (except for the target sessions) or to receive a message, follow the instructions of the DH protocol on behalf of that party. When a session is expired at a player erase the corresponding session key from that player's memory. When a party is corrupted or a session (other than a target session) is exposed, hand $\mathcal{H}$ all the information corresponding to that party or session as in a real interaction.
4. When A's ${r_A}^{\text{th}}$ session is invoked within A to exchange a key with P_j, if $B \neq P_j$, then $\mathcal{D}$ outputs $b' \xleftarrow{R} \{0, 1\}$ and halts. Otherwise, whenever A would send the message (g^x) or (s, g^x) to B (either authentically or unauthentically) let A send the message (α^*) or (s, α^*) to B instead.
5. When B's ${r_B}^{\text{th}}$ session is invoked to exchange a key with P_i, if $A \neq P_i$, then $\mathcal{D}$ outputs $b' \xleftarrow{R} \{0, 1\}$ and halts. Otherwise, whenever B would send the message (g^y) or (s', g^y) to A (either authentically or unauthentically) let B send the message (β^*) or (s', β^*) to A instead.
6. If a target session is chosen by $\mathcal{H}$ as the test-session, and both the target sessions have the same session identifier s, then provide $\mathcal{H}$ with γ^* as the answer to this query. (Whether the sessions have the same session identifier can be deduced from the outputs of the session. A session matching the test session must

exist, since completion of the test session means that it has received an authentic message containing the session identifier from the other party. Furthermore, that session identifier must be in the other party's output. If either of the target sessions has not produced any output containing its session identifier, then it is not a partner of the test session.)

7. If a target session is ever exposed, or a session other than a target session is chosen as the test-session, or if the test session does not match either of the target sessions, or if $\mathcal{H}$ halts without choosing a test-session then $\mathcal{D}$ outputs $b' \stackrel{R}{\leftarrow} \{0,1\}$ and halts.
8. If $\mathcal{H}$ halts and outputs a bit b', then $\mathcal{D}$ halts and outputs b' too.

First note that the run of $\mathcal{H}$ by $\mathcal{D}$ up to the point where $\mathcal{H}$ stops (or $\mathcal{D}$ aborts $\mathcal{H}$'s run) is identical to a normal run of $\mathcal{H}$ against the DH protocol.

Secondly, note that in step 3, it is always possible for $\mathcal{D}$ to answer the corrupt and exposure queries of $\mathcal{H}$, unless step 7 specifies that $\mathcal{D}$ should output a random bit and halt. This is so since if $\mathcal{D}$ is not to output a random bit and halt:

- if any session is exposed, it must not be a target session. Therefore the discrete logarithms of α^* and β^* will not be part of the exposed session's state in any way. Also γ^* will not be part of the session's state.
- if any session is corrupted other than a target session, the discrete logarithms of α^* and β^* will not be part of the corrupted session's state in any way. Also, γ^* will not be part of the session's state.
- if a target session is corrupted, the corrupted session must first have been expired. However, if the session was expired, it must also have been completed, and if it was completed, then the discrete logarithm of the exponential generated by the session ought to have been erased. Therefore, the discrete logarithms of α^* and β^* will not be part of the corrupted session's state in any way. γ^* will not be part of the state either, since the expiry erased the session key.

Consider the case in which the test session chosen by $\mathcal{H}$ coincides with a target session chosen at random by $\mathcal{D}$. In this case, the response to the test-query by $\mathcal{H}$ is γ^*. Thus, if the input to $\mathcal{D}$ came from Q_0 then the response was the actual value of the key exchanged between P_i and P_j during the test-session (since, by construction, the session key exchanged in Steps 4 and 5 of the actions of $\mathcal{D}$ is $\gamma^* = g^{xy}$). On the other hand, if the input to $\mathcal{D}$ came from Q_1 then the response to the test query was a random exponentiation, i.e. a random value from the distribution of keys generated by the protocol. In addition, the input to $\mathcal{D}$ was chosen with probability $1/2$ from Q_0 and with probability $1/2$ from Q_1, so the distribution of responses provided by $\mathcal{D}$ to the test query of $\mathcal{H}$ is the same as specified by Definition 1. In this case, the probability that $\mathcal{H}$ guesses correctly whether the test value was "real" or "random" is $1/2 + \epsilon$ for non-negligible ϵ. By the above argument this is equivalent to guessing whether the input to the distinguisher $\mathcal{D}$ came from Q_0 or Q_1, respectively. Thus, by outputting the same bit as $\mathcal{H}$ we get that the distinguisher $\mathcal{D}$ guesses correctly the input distribution Q_0 or Q_1 with the same probability $1/2 + \epsilon$ as $\mathcal{H}$ did.

Now consider the case in which a target session is not chosen as a test-session. In this case $\mathcal{D}$ always ends outputting a random bit, and thus its probability to guess correctly the input distribution is $1/2$.

Since the first case (in which the test-session and a target session coincide) happens with probability $\frac{1}{n^2 l^2}$ while the other case happens with probability $1 - \frac{1}{n^2 l^2}$, the overall probability of $\mathcal{D}$ to guess correctly is $1/2 + \frac{\epsilon}{n^2 l^2}$, and thus $\mathcal{D}$ succeeds in distinguishing Q_0 from Q_1 with non-negligible advantage. □

Although the original 2DH protocol of Canetti and Krawczyk [11] included the identity of the sender in the protocol messages, it has not been explicitly included in messages here. It is assumed that the identities of the sender and recipient of the message are provided automatically. For authenticated HM messages, such information must be provided in any case for the message to be authentic. For unauthenticated messages, if there is no other source for the information (such as a header field on the message) it may be necessary to add the information to the message in practice. However, it is omitted here for clarity. As can be seen above, this has not affected the proof.

An examination of Figure 3 shows that quite a number of steps can be performed if one party has possession of the other's unauthentic Diffie-Hellman value. Therefore, it seems logical to specify an HM protocol such as the one shown by Protocol 4 that allows a party to receive an unauthentic Diffie-Hellman value and then carry out as many actions as possible. The steps carried out by the other party are then fairly straightforward since the other party's Diffie-Hellman value is required before most steps can be carried out.

Messages that are not labelled "unauthentic" are authentic. In this version of the protocol, the adversary chooses the Diffie-Hellman values, (t_A, t_B) to be the session identifier. The unauthentic message t_B from B is addressed to the adversary and is never delivered to A. It is used to allow the adversary to create the session identifier.

A		B
$r_A \in_R \mathbb{Z}_q$ $t_A = g^{r_A}$		
	Unauthentic: t_A $\longrightarrow$	$r_B \in_R \mathbb{Z}_q$ $t_B = g^{r_B}$ $K_{BA} = t_A^{r_B}$ Erase r_B
	$\mathcal{H}$ $\longleftarrow$ Unauthentic: t_B	
	$\mathcal{H}$ $\xrightarrow{s = (t_A, t_B)}$	
		'B sent (s, t_B) to A'
	$\longleftarrow$ s, t_B	
'A rec. (s, t_B) from B' $K_{AB} = t_B^{r_A}$ Erase r_A		
	$s = (t_A, t_B)$ $\longleftarrow$ $\mathcal{H}$	
'Established (A, B, s, K_{AB})'		
'A sent (s, t_A) to B'	$\longrightarrow$ s, t_A	'B rec. (s, t_A) from A' 'Established (B, A, s, K_{BA})'

Protocol 4. Secure Diffie-Hellman protocol in the HM

Protocol 5 shows another version of the Diffie-Hellman protocol where the adversary no longer inputs the session identifier to the parties. (Note that t_A and t_B are unique to each session except with negligible probability.) In addition, messages containing the same term twice have had the second term removed, and the unauthentic message to the adversary has been removed. However, it is still SK-secure, by the following theorem.

Theorem 5 *If Protocol 4 is secure then so is Protocol 5.*

Proof For convenience, let π denote Protocol 4 and π' denote Protocol 5 throughout the proof. Protocol π' satisfies the first requirement of Definition 1 since the same key is calculated in π' and π. Therefore, it remains to show that the second requirement is satisfied. Suppose to the contrary that there exists an adversary $\mathcal{H}'$ against π' with a non-negligible advantage. Then we construct an adversary $\mathcal{H}$ against π that has a non-negligible advantage. $\mathcal{H}$ runs $\mathcal{H}'$ and simulates an interaction of $\mathcal{H}'$ with imaginary parties $P'_1, P'_2, \ldots, P'_n$ running Protocol 5. The simulation proceeds as follows:

- Whenever $\mathcal{H}'$ activates P'_i to start the protocol with P'_j as an initiator (or responder respectively), $\mathcal{H}$ activates the corresponding party, P_i, in its own model to start a protocol run with P_j as the initiator (or responder respectively).
- Whenever a party P_i outputs an unauthentic message, $\mathcal{H}$ causes P'_i to output the same unauthentic message to $\mathcal{H}'$, except in the case where the unauthentic message output by P_i was from the responder and addressed to $\mathcal{H}$.
- Whenever a party P_i outputs an authentic message (s, t_A) to P_j where $s = (t_A, t_B)$, $\mathcal{H}$ causes P'_i to pass (t_A, t_B) to $\mathcal{H}'$ for delivery as an authentic message to P'_j.
- Whenever a party P_j outputs an authentic message (s, t_B) to P_i where $s = (t_A, t_B)$, $\mathcal{H}$ causes P'_j to pass (t_A, t_B) to $\mathcal{H}'$ for delivery as an authentic message to P'_i.
- Whenever an unauthentic message is delivered to a party P'_j by $\mathcal{H}'$, $\mathcal{H}$ delivers the same unauthentic message to P_j.

A		B
$r_A \in_R \mathbb{Z}_q$ $t_A = g^{r_A}$		
	Unauthentic: t_A $\longrightarrow$	$r_B \in_R \mathbb{Z}_q$ $t_B = g^{r_B}$ $K_{BA} = t_A^{r_B}$ Erase r_B Set SID $= s = (t_A, t_B)$ 'B sent (t_A, t_B) to A'
	$\longleftarrow$ t_A, t_B	
'A rec. (t_A, t_B) from B' $K_{AB} = t_B^{r_A}$ Erase r_A Set SID $= s = (t_A, t_B)$ 'Established (A, B, s, K_{AB})'		
'A sent (t_A, t_B) to B'	t_A, t_B $\longrightarrow$	'B rec. (t_A, t_B) from A' 'Established (B, A, s, K_{BA})'

Protocol 5. Secure Diffie-Hellman protocol in the HM suitable for optimization

- Whenever an authentic message (t_A, t_B) is delivered to a party P_j' from P_i' by $\mathcal{H}'$, if P_j is a responder, $\mathcal{H}$ delivers $((t_A, t_B), t_A)$ as an authentic message from P_i to P_j. Otherwise, since P_j is an initiator, $\mathcal{H}$ delivers $((t_A, t_B), t_B)$ as an authentic message from P_i to P_j.
- Whenever a party P_i is waiting for a session identifier to be input, $\mathcal{H}$ calculates the session identifier in the same way as in π and inputs it to P_i.
- Whenever a party P_i outputs that it has established a session key, $\mathcal{H}$ causes P_i' to output the same thing.
- Whenever $\mathcal{H}'$ chooses a session as the test session, $\mathcal{H}$ chooses the corresponding session (i.e. the one with the same session identifier and run by the corresponding party) in its own model and passes the value it receives in response to its test session query to $\mathcal{H}'$.
- Whenever $\mathcal{H}'$ corrupts a party or a session, or performs a session key query, $\mathcal{H}$ corrupts the corresponding party or session or performs a session key query on the corresponding session in its own model and passes whatever it receives to $\mathcal{H}'$.
- Whenever $\mathcal{H}'$ expires a session, $\mathcal{H}$ expires the corresponding session in its own model.
- $\mathcal{H}$ outputs whatever $\mathcal{H}'$ outputs.

Observe that any session s' with which $\mathcal{H}'$ interacts will have the same session identifier and secret key (if they have been set) as the corresponding session s with which $\mathcal{H}$ interacts. Furthermore, the matching session to s will correspond to the matching session to s' (if it exists). In addition, no two sessions belonging to one party will have the same session identifier (this is a requirement of the HM). Hence, $\mathcal{H}$ will have the same advantage against Protocol 4 as $\mathcal{H}'$ has against Protocol 5. □

Although the above theorem and proof are for Protocols 4 and 5, other protocols may be changed in a similar way so that the session identifier may be specified by the protocol, instead of being chosen by the adversary. A proof would then be constructed in a similar way.

Of the remaining key exchange protocols in the AM, the key transport [11] and ElGamal type [16] protocols only contain one message and hence can be compiled using only one MT-authenticator, which is likely to eliminate the need to alter the order of AM steps. Therefore, no templates of possible step order are provided for these protocols. The server-based key transport protocol [15] may be placed in the same category if the server sends its encryptions of the session key directly to each of the two other participants. However, if the session identifier is to be constructed from values generated within the protocol, a proof of the protocol with this modification in the HM will need to be provided, similarly to Theorems 4 and 5.

4 Using Two or More Authenticators

In some situations it may be desirable to use more than one MT-authenticator in the one protocol, for example when the two parties in the protocol do not have the same level of computational power. Let $\lambda_1, \lambda_2, \ldots, \lambda_l$ be l MT-authenticators to be used at once. (For all current AM protocols $l \leq 2$, but the general case seems just as easy to prove.) Let $\Lambda : \mathbb{N} \to \{\lambda_1, \lambda_2, \ldots, \lambda_l, \phi\}$ be a mapping from message number to authenticator (where a mapping of a message to ϕ indicates no authenticator is to be used and the message is to be sent in the compiled protocol as an authentic message in the HM if it was an authentic message in the original protocol, and otherwise the message is to be sent as an unauthentic message in the compiled protocol). Let $\mathcal{C}_\Lambda(\pi)$ be the HM-compiler that uses MT-authenticators $\lambda_1, \lambda_2, \ldots, \lambda_l$ to transform protocol π to protocol π', where Λ specifies which MT-authenticator to apply to which message of π, and π' proceeds as follows:

- π' first invokes the initialization procedure for each authenticator, $\lambda_1, \lambda_2, \ldots, \lambda_l$ (note that each authenticator has independent state),
- each message of π is converted to one or more messages in π' as follows (with the original message ordering of π maintained in π'):
  - an unauthentic message sent (or received) in π is converted to exactly the same message to be sent (or received) in π' with the same sender and recipient. (Of course, since no modifications to provide any authentication are made for unauthentic messages, it is possible that in an execution of π', the adversary will change the recipient, message contents or purported sender. However, this could also have happened to the corresponding message in an execution of π.)
  - an authentic message m sent (or received) in π is converted to a message m sent (or received) using authenticator λ_k with the same sender and recipient, where $k \in \{1, 2, \ldots, l\}$, and λ_k was specified by Λ for this message. Hence, the sending (or receiving) of message m in π may be converted to the sending and receiving of several messages in π' via the authenticator λ_k. However, an exception is an authentic message mapped by Λ to ϕ. In this case, no authenticator is to be used and the message is to be sent in π' as an authentic message in the HM.
- any output specified by π is also specified by π' and occurs in π' in the same order and in connection with the same message(s) as in π. I.e. output before an authentic message is sent in π occurs in π' *before* the authenticator is activated to send that message. Output after an authentic message is received in π occurs in π' *after* the authenticator has received the message. Any other actions (e.g. choosing random values, exponentiation, conditional actions, etc.) specified by π are converted to π' in a similar way.

Theorem 6 *The compiler $\mathcal{C}_\Lambda$ as described above is an HM-authenticator.*

Proof We must show that π' HM-emulates π. This can be seen by constructing intermediate HM-authenticators $\mathcal{C}_{\chi_1}, \mathcal{C}_{\chi_2}, \ldots, \mathcal{C}_{\chi_l}$ where we let $\pi_i = \mathcal{C}_{\chi_i}(\pi_{i-1})$ for $i \in \{1, 2, \ldots, l\}$ and $\pi_0 = \pi$. χ_i is defined such that if $\Lambda(j) = \lambda_i$ then $\chi_i(j) = \lambda_i$. Otherwise, $\chi_i(j) = \phi$. That $\mathcal{C}_{\chi_1}, \mathcal{C}_{\chi_2}, \ldots, \mathcal{C}_{\chi_l}$ are HM-authenticators can be seen by Theorem 3. Therefore, for any adversary $\mathcal{H}_i$ against π_i for $i \in \{1, 2, \ldots, l\}$, there exists an adversary $\mathcal{H}_{i-1}$ against π_{i-1} such that $\text{HYBRID}_{\pi_{i-1}, \mathcal{H}_{i-1}}$ and $\text{HYBRID}_{\pi_i, \mathcal{H}_i}$ are computationally indistinguishable. Therefore, we can deduce that for any adversary $\mathcal{H}_l$ against $\pi_l = \pi'$, there exists an adversary $\mathcal{H}_0$ against $\pi_0 = \pi$ such that $\text{HYBRID}_{\pi_0, \mathcal{H}_0}$ and $\text{HYBRID}_{\pi_l, \mathcal{H}_l}$ are computationally indistinguishable and therefore π' HM-emulates π. □

As an example, Protocol 6 shows the Diffie-Hellman Protocol 5 from the HM authenticated with two different authenticators, the signature-based authenticator of Fig. 1 and the encryption-based authenticator of Fig. 2. The public and private keys used for the signature authenticator must be separate from those used for the encryption authenticator for the proof to hold. This example protocol will be further optimized in the following sections.

5 Optimizations for Existing Authenticators

Now that an HM protocol suitable for emulation has been described and the use of multiple authenticators has been justified, we wish to use various optimizations to derive an efficient protocol in the UM. We first clarify the definitions of existing authenticators to make them as flexible as possible.

A		B
$r_A \in_R \mathbb{Z}_q$ $t_A = g^{r_A}$		
	$\xrightarrow{t_A}$	$r_B \in_R \mathbb{Z}_q$ $t_B = g^{r_B}$ $K_{BA} = t_A^{r_B}$ Erase r_B Set SID $= s = (t_A, t_B)$ 'B sent (t_A, t_B) to A'
	$\xleftarrow{t_A, t_B}$	
$N_A \in_R \{0,1\}^k$	$\xrightarrow{t_A, t_B, N_A}$	$\sigma = \mathrm{Sig}_B(t_A, t_B, N_A, A)$
	$\xleftarrow{t_A, t_B, \sigma}$	
Check $\sigma = \mathrm{Sig}_B(t_A, t_B, N_A, A)$; if not then abort. '$A$ rec. (t_A, t_B) from B' $K_{AB} = t_B^{r_A}$ Erase r_A Set SID $= s = (t_A, t_B)$ 'Established (A, B, s, K_{AB})'		
'A sent (t_A, t_B) to B'	$\xrightarrow{t_A, t_B}$	$N_B \in_R \{0,1\}^k$ $c = \mathcal{E}_{\mathrm{PK}_A}(N_B)$
	$\xleftarrow{t_A, t_B, c}$	
$N_B = \mathcal{D}_{\mathrm{SK}_A}(c)$		
$\mu = \mathrm{MAC}_{N_B}(t_A, t_B, B)$	$\xrightarrow{t_A, t_B, \mu}$	Check $\mu = \mathrm{MAC}_{N_B}(t_A, t_B, B)$; if not then abort. '$B$ rec. (t_A, t_B) from A' 'Established (B, A, s, K_{BA})'

Protocol 6. Unoptimized Diffie-Hellman protocol in the UM (two authenticators)

5.1 Flexibility of Existing Authenticators

Figures 1, 2, and 4 to 6 give a description of the various authenticators for the CK model that currently exist in the literature. (More precise descriptions including details of the initialization process appear in the original proposals, apart from that of the MAC-based authenticator, which is very sketchy.) The notation used in the authenticators is as follows:

m: unique message to be transmitted authentically,
κ_{AB}: shared fixed key, for example derived from static Diffie-Hellman keys,
π: password of A, shared with B,
γ: threshold for number of failed logons per party,
e_P: public encryption key of party P,
s_P: secret signature key of party P,
$\mathcal{M}_K(x)$: MAC of x under key K,
$\mathcal{E}_K(x)$: encryption of x under key K,
$\sigma_K(x)$: signature of x under key K,
$H(x)$: hash of x.

The following conditions pertain to authenticators $\lambda_{\text{P-ENC}}$, λ_{SIG}, λ_{ENC}, λ_{MAC}, and λ_{STATIC}. They provide slightly more flexibility that the original definitions of these authenticators but their original security proofs remain essentially unchanged. The notation used is that the authenticator is to deliver some message m from some party A to some other party B.

1. As soon as A is activated (e.g. by $\mathcal{U}$ or a higher layer protocol) to send m to B, A must output 'A sent m to B'. Hereafter, we call this activation the *send activation*.

A (Client)		(Server) B
'A sent m to B'	$\underrightarrow{m}$	$N_B \in_R \{0,1\}^k$
	$\underleftarrow{m, N_B}$	
	$\underrightarrow{m, \mathcal{E}_{e_B}(m, N_B, A, \pi)}$	Decrypt $\mathcal{E}_{e_B}(m, N_B, A, \pi)$
		If m, A, π are correct, N_B is outstanding, and failed logons for $A \leq \gamma$
		Then output 'B received m from A' and remove N_B from the list of outstanding nonces.
		Else if A a valid user, increase the number of failed logons for A by 1.

Fig. 4 Password Based Authenticator, $\lambda_{\text{P-ENC}}$ [13]

A		B
'A sent m to B'	$\underrightarrow{m}$	
		$N_B \in_R \{0,1\}^{2k}$
	$\underleftarrow{m, N_B}$	
$\mu = \mathcal{M}_{\kappa_{AB}}(B, N_B, m)$	$\underrightarrow{m, \mu}$	Check $\mu = \mathcal{M}_{\kappa_{AB}}(B, N_B, m)$; if not, then abort.
		'B received m from A'

Fig. 5 MAC-based MT-authenticator, λ_{MAC} [11]

A		B
'A sent m to B'	$\xrightarrow{m}$	
		$N_B \in_R \{0,1\}^k$
	$\xleftarrow{m, N_B}$	
$h = H(\kappa_{AB}, B, N_B, m)$	$\xrightarrow{m, h}$	Check $h = H(\kappa_{AB}, B, N_B, m)$; if not, then abort.
		'B received m from A'

Fig. 6 Authenticator using static key agreement, λ_{STATIC} [7]

2. A may send m or part thereof, by itself or as part of another message, unauthentically any number of times (including zero) to B at any time after the send activation. All of the authenticators except for $\lambda_{\text{P-ENC}}$ require B to know m (by this or some other means) before B can receive m authentically. However, it is possible that if the authenticator is being used as part of a larger protocol, that protocol may have delivered m to B before the send activation. In practice, if m is removed from a message of the output protocol, a "hint" (such as a session identifier) to enable the communicating parties to establish to which session the message belongs may need to be added in place of m, although the use of such a "hint" does not affect the security of the authenticator.
3. A may receive $\mathcal{E}_{e_A}(N_B)$ in the case of λ_{ENC}, or N_B in other cases, (alone or as part of another message) at any time before the send activation, at the same time as the send activation, or after the send activation. However, in the case of λ_{ENC}, N_B must be erased by A in the same activation as it is decrypted.
4. After A has received $\mathcal{E}_{e_A}(N_B)$ in the case of λ_{ENC}, or N_B in other cases, and after outputting 'A sent m to B' in the send activation, A can construct and send to B:
   - $\mathcal{E}_{e_B}(m, N_B, A, \pi)$ for $\lambda_{\text{P-ENC}}$,
   - $\sigma_{s_A}(m, N_B, B)$ for λ_{SIG},
   - $\mathcal{M}_{N_B}(m, B)$ for λ_{ENC}
   - $\mathcal{M}_{\kappa_{AB}}(B, N_B, m)$ for λ_{MAC}, or
   - $H(\kappa_{AB}, B, N_B, m)$ for λ_{STATIC}.
5. B may choose to send $\mathcal{E}_{e_A}(N_B)$ in the case of λ_{ENC}, or N_B in other cases, to A at any time. (B need not wait for a message from A to do this, and N_B may not need to be sent at all if A is able to derive its value

in some other way, e.g. from the session identifier.) Once N_B is chosen, it is recorded as an outstanding nonce.

6. In $\lambda_{\text{P-ENC}}$, λ_{SIG}, λ_{MAC}, and λ_{STATIC}, N_B may be any value of B's choice (including a value chosen by the adversary), that has not previously been used as a nonce in the same authenticator by B, except with negligible probability.
7. Suppose N_B is an outstanding nonce previously output or chosen by this invocation of a certain authenticator in B. When B receives or knows the value of the last message in the authenticator it accepts m (i.e. outputs 'B received m from A') and notes that N_B is no longer an outstanding nonce as long as:
   - in the case of $\lambda_{\text{P-ENC}}$, it can decrypt $\mathcal{E}_{e_B}(m, N_B, A, \pi)$ into m, N_B, A, π, π is a password shared between A and B, and the number of unsuccessful attempts to complete the protocol with A is less than or equal to the previously chosen threshold, γ. Otherwise, if A is a valid client, B increases the number of unsuccessful attempts to complete the protocol with A by one.
   - in the case of λ_{SIG}, if it can verify the signature successfully.
   - in the case of λ_{ENC}, if it can verify the MAC. Otherwise, B terminates this invocation of λ_{ENC} without accepting any message and notes that N_B is no longer an outstanding nonce.
   - in the case of λ_{MAC}, if it can verify the MAC, κ_{AB} is the long term shared key between A and B.
   - in the case of λ_{STATIC}, if it can verify the hash, κ_{AB} is the long term shared key between A and B.

Definition 9 *An activation of an authenticator in which a party P_i outputs 'P_i sent m to P_j' for some message m and party P_j is called a* send activation.

Definition 10 *All messages in an authenticator that may be sent and received before the send activation are called* preamble messages.

Note that specification of which messages may be sent at various times is part of the authenticator definition. We observe that the message $\mathcal{E}_{e_A}(N_B)$ in the case of λ_{ENC}, or N_B in the case of $\lambda_{\text{P-ENC}}$, λ_{SIG}, λ_{MAC}, and λ_{STATIC} is a preamble message.

In Protocol 6, $m = (s, t_A)$ for the first authenticator. In accordance with the above authenticator definitions, Protocol 7 shows this example protocol with the first message of the authenticator removed, since the message only consists of $m = (s, t_A)$. In addition, the value $m = (s, t_A)$ was omitted in the second authenticator message (containing the nonce). However, it is still possible for the recipient to obtain $m = (s, t_A)$ from the last authenticator message. The same changes were also made for the second authenticator, but with $m = (s, t_B)$ in that case.

5.2 Some Parts of Authenticator Messages Can Be Shifted and Protocol Values Used for Nonces

As discussed in Section 3, it seems that AM messages and actions cannot be reordered without providing a new proof of security for the AM protocol. However, it is often desirable to shift some authenticator messages to occur at an earlier point in the protocol. For example, it seems desirable to combine the first and second messages and third and fourth messages of Protocol 7. This involves invoking the authenticator at the recipient of the authentic message and transmitting the authenticator message containing the nonce or encrypted nonce in an earlier activation than previously. It remains to be proven that this can be done without affecting the SK-security of the protocol.

Another attractive optimization is to replace the nonce N_A in Protocol 7 with the protocol value t_A. It also remains to be proven that this can be done without affecting the SK-security of the protocol.

Theorem 7 *Suppose that an HM-authenticator, $\mathcal{C}_\Lambda$, built as described in Section 4 from* MT*-authenticators $\lambda_1, \lambda_2, \ldots, \lambda_l$, has been used to compile an SK-secure HM protocol, π, into $\pi' = \mathcal{C}_\Lambda(\pi)$. Furthermore, suppose that another protocol, π'' has been constructed from π' by moving some messages to an earlier point in the protocol and/or by replacing some values with values from other parts of π'. Suppose these changes are subject to the following conditions:*

- *When some message, g, from P_i to P_j, is moved to an earlier point in π'' than in π', g must be produced by λ_i for $i \in \{1, \ldots, l\}$ to deliver some message m in π. Furthermore, g must be a preamble message in λ_i.*

A		B
$r_A \in_R \mathbb{Z}_q$ $t_A = g^{r_A}$		
	$\xrightarrow{t_A}$	$r_B \in_R \mathbb{Z}_q$ $t_B = g^{r_B}$ $K_{BA} = t_A^{r_B}$ Erase r_B Set SID $= s = (t_A, t_B)$ 'B sent (t_A, t_B) to A'
$N_A \in_R \{0,1\}^k$	$\xrightarrow{N_A}$	$\sigma = \mathrm{Sig}_B(t_A, t_B, N_A, A)$
	$\xleftarrow{t_A, t_B, \sigma}$	
Check $\sigma = \mathrm{Sig}_B(t_A, t_B, N_A, A)$; if not then abort. '$A$ rec. (t_A, t_B) from B' $K_{AB} = t_B^{r_A}$ Erase r_A Set SID $= s = (t_A, t_B)$ 'Established (A, B, s, K_{AB})' 'A sent (t_A, t_B) to B'		
		$N_B \in_R \{0,1\}^k$ $c = \mathcal{E}_{\mathrm{PK}_A}(N_B)$
	$\xleftarrow{c}$	
$N_B = \mathcal{D}_{\mathrm{SK}_A}(c)$		
$\mu = \mathrm{MAC}_{N_B}(t_A, t_B, B)$	$\xrightarrow{t_A, t_B, \mu}$	Check $\mu = \mathrm{MAC}_{N_B}(t_A, t_B, B)$; if not then abort. '$B$ rec. (t_A, t_B) from A' 'Established (B, A, s, K_{BA})'

Protocol 7. Partially optimized Diffie-Hellman protocol in the UM (first message of authenticators removed)

– *When some value, h, generated by some party P_j in actions and messages produced by some authenticator, λ_i, for $i \in \{1, \ldots, l\}$, is replaced by some other value, h', from π', λ_i must allow h to be any value of P_j's choice (including a value chosen by the adversary), that has not previously been used in the same authenticator by P_j, except with negligible probability. Furthermore, h' must be a value which has not been used in the same authenticator by P_j, except with negligible probability. Also, by the time a term equal to or derived from h' is generated or used in a message in place of h in π'', h' must already be known to the adversary or be able to be derived by the adversary.*

Then protocol π'' is SK-secure.

Proof Suppose that the theorem is not true and that π'' is not SK-secure. π'' meets the first requirement of Definition 1 since the values used to calculate the secret key remain unchanged from π. Therefore, we assume that the second requirement is violated and that an adversary $\mathcal{U}$ against π'' exists having a non-negligible advantage.

We now construct several protocols, $\pi''_0, \ldots, \pi''_l$ where $\pi''_0 = \pi$ and $\pi''_l = \pi''$ and π''_k for $k \in \{0, \ldots, l-1\}$ is the same as π''_{k+1} except that all messages in π''_{k+1} that correspond to the use of λ_{k+1} to deliver a message m from π (including shifted messages and messages with replaced values) are removed from π''_k and instead the message m is delivered authentically in the HM in π''_k. If messages in π''_k now require the use of values whose generation has been deleted (because they were generated by λ_{k+1} or input by the adversary to λ_{k+1}), each value is marked as being required to be input by the adversary the first time it is used, but the adversary is subject to the condition that it choose a value that has not been used previously by the authenticator(s) that use the value. (Observe that the value had not previously been used by the authenticator(s) in question in π''_{k+1} due to the requirements of the theorem being proven.)

Furthermore, we construct adversaries $\mathcal{H}_0, \ldots, \mathcal{H}_l$ where $\mathcal{H}_l = \mathcal{U}$ such that $\mathrm{HYBRID}_{\pi''_k, \mathcal{H}_k}$ and $\mathrm{HYBRID}_{\pi''_{k+1}, \mathcal{H}_{k+1}}$ have negligible statistical distance for all $k \in \{0, \ldots, l-1\}$.

Adversary $\mathcal{H}_k$ for $k \in \{0, \ldots, l-1\}$ operates in the HM, with (real) parties $P_1, \ldots, P_n$ running π''_k. $\mathcal{H}_k$ also simulates the parties $P'_1, \ldots, P'_n$ for $\mathcal{H}_{k+1}$, the adversary against π''_{k+1}. Parties $P'_1, \ldots, P'_n$ do not actually exist, and neither do they run protocol π''_{k+1}. $\mathcal{H}_k$ simply generates output and messages on behalf of these imaginary parties for the benefit of $\mathcal{H}_{k+1}$, since $\mathcal{H}_{k+1}$ interacts with parties running π''_{k+1}. $\mathcal{H}_k$ runs λ_{k+1} on behalf of the imaginary parties $P'_1, \ldots, P'_n$, and orchestrates an interaction between $\mathcal{H}_{k+1}$ and λ_{k+1}, while using the parties $P_1 \ldots P_n$ to play the protocol π''_k to generate the messages sent and received via λ_{k+1}.

First $\mathcal{H}_k$ invokes λ_{k+1}. Next, $\mathcal{H}_k$ proceeds according to the following rules:

1. Whenever $\mathcal{H}_{k+1}$ activates P'_i with an external request, $\mathcal{H}_k$ activates P_i with the same request. For each outgoing authentic message that P_i generates in this activation, say for P_j, that π''_{k+1} specifies is to be delivered via λ_{k+1}, $\mathcal{H}_k$ activates protocol λ_{k+1} with external request for sending that message from P'_i to P'_j.
   Furthermore, if λ_{k+1} on behalf of P'_i must choose a value in this activation that has been replaced with another value in π''_{k+1} (e.g. from π or from another authenticator), $\mathcal{H}_k$ inputs this value to λ_{k+1}. (Note that the theorem stipulates that this value must be known to $\mathcal{H}_{k+1}$, and so it is also known to $\mathcal{H}_k$ since it is running an internal copy of $\mathcal{H}_{k+1}$.)
   Next $\mathcal{H}_k$ hands $\mathcal{H}_{k+1}$ all the outgoing messages generated by λ_{k+1} on behalf of P'_i, as well as any outgoing unauthentic messages generated by P_i, any outgoing authentic messages generated by P_i that are not to be delivered by λ_{k+1} (and must therefore be treated by $\mathcal{H}_{k+1}$ as authentic messages awaiting delivery), and any outgoing requests and outputs that P_i may have generated.
2. Whenever $\mathcal{H}_{k+1}$ activates P'_i with an incoming authentic message m purportedly from P'_j, (not generated by the MT-authenticator), $\mathcal{H}_k$ delivers the same message to P_i as authentic with sender P_j. Any outgoing messages and external requests generated by P_i are handled as above.
3. Whenever $\mathcal{H}_{k+1}$ activates P'_i with an incoming unauthentic message m purportedly from P'_j, and according to π''_{k+1}, the next message that P'_i would receive is one not generated by the MT-authenticator, $\mathcal{H}_k$ delivers the same message to P_i as unauthentic with sender P_j. Any outgoing messages and external requests generated by P_i are handled as above.
4. Whenever $\mathcal{H}_{k+1}$ activates P'_i with an incoming unauthentic message m, if (according to π''_{k+1}) the next message P'_i should receive is one via the authenticator, $\mathcal{H}_k$ first activates λ_{k+1} with this incoming message. Furthermore, if λ_{k+1} on behalf of P'_i must choose a value in this activation that has been replaced with another value in π''_{k+1} (e.g. from π or from another authenticator), $\mathcal{H}_k$ inputs this value to λ_{k+1}. (Note that the theorem stipulates that this value must be known to $\mathcal{H}_{k+1}$, and so it is also known to $\mathcal{H}_k$ since it is running an internal copy of $\mathcal{H}_{k+1}$.)
   Any outgoing messages and external requests generated by λ_{k+1} on behalf of P'_i are handed by $\mathcal{H}_k$ to $\mathcal{H}_{k+1}$.
5. If any activation of λ_{k+1} outputs 'P'_i received m from P'_j' on behalf of P'_i, then $\mathcal{H}_k$ activates party P_i with incoming authentic message m from party P_j. Each outgoing message that P_i generates as a result of this activation is handled as above.
6. Whenever π''_k requires a value to be input by the adversary for use in an authenticator, $\mathcal{H}_k$ inputs the same value as would have been used in π''_{k+1}, from either λ_{k+1} or $\mathcal{H}_{k+1}$, whichever is appropriate.
7. Whenever $\mathcal{H}_{k+1}$ corrupts party P'_i, $\mathcal{H}_k$ corrupts P_i. $\mathcal{H}_k$ then hands $\mathcal{H}_{k+1}$ the internal data of P_i together with the information regarding all activations of λ_{k+1} by (the imaginary) P'_i.
8. Whenever $\mathcal{H}_{k+1}$ expires a session at party P'_i, $\mathcal{H}_k$ expires the same session at P_i. $\mathcal{H}_k$ will no longer return any local output (i.e. the session key) from this session to $\mathcal{H}_{k+1}$ upon corruption of P'_i. $\mathcal{H}_{k+1}$ may no longer perform a session key query on this session at party P'_i, and $\mathcal{H}_k$ may no no longer perform a session key query on this session at party P_i.
9. Whenever $\mathcal{H}_{k+1}$ makes a session key query on session s at party P'_i, $\mathcal{H}_k$ makes a session key query on session s at party P_i, and returns to $\mathcal{H}_{k+1}$ the key it received from its own query.
10. Whenever $\mathcal{H}_{k+1}$ chooses a test session s belonging to P'_i, $\mathcal{H}_k$ chooses session s belonging to P_i as its test session and returns to P'_i the same key that it received in answer to its own test session query.
11. If $\mathcal{H}_{k+1}$ corrupts a session within some party P'_i then $\mathcal{H}_k$ corrupts the same session within P_i and hands the corresponding information back to $\mathcal{H}_{k+1}$ together with the information regarding activations of λ_{k+1} by (the imaginary) P'_i for that session.
12. $\mathcal{H}_k$ outputs whatever $\mathcal{H}_{k+1}$ outputs.

We first need to show that the above description of the behaviour of $\mathcal{H}_k$ is a legitimate behaviour of an adversary against π''_k. Observe that the simulation above does not have any problem with messages shifted to an earlier point in π'' than π', since the only messages shifted are preamble messages and therefore the authenticator λ_{k+1} can be activated to send and receive these messages when required by π''_{k+1}. The above steps are easy to verify as legal moves for $\mathcal{H}_k$, except for step 5. In this case, it could be possible that the triple (P_j, P_i, m) is not currently in the set of authentic undelivered messages, and P_j (and the originating session within P_j) is uncorrupted. It is easy to see that if this is not the case, namely if we assume that step 5 can always be carried out, then the above construction satisfies the requirement for $\text{HYBRID}_{\pi''_k,\mathcal{H}_k}$ and $\text{HYBRID}_{\pi''_{k+1},\mathcal{H}_{k+1}}$ to have negligible statistical distance. This holds since the simulated run with $\mathcal{H}_{k+1}$, together with the activations of the parties $P_1, P_2, \ldots, P_n$ is an exact imitation of a run of $\mathcal{H}_{k+1}$ with parties running π''_{k+1}.

Thus, it remains to show that the probability that $\mathcal{H}_k$ cannot carry out step 5 is negligible. Assume that protocol λ_{k+1} outputs 'P'_i received m from P'_j' on behalf of an uncorrupted party P'_i. We show that, except with negligible probability, the following events occur in the HM_1:

1. The triple (P_i, P_j, m) was added to the set of authentic undelivered messages.
2. This triple was not yet deleted from the set of authentic undelivered messages (i.e. had not previously been delivered as an authentic message).

To see (1), notice first that if, in the simulated run of $\mathcal{H}_{k+1}$, λ_{k+1} outputs 'P'_i received m from P'_j' on behalf of party P'_i, then λ_{k+1} had been invoked on behalf of P'_j for sending m to P'_i.

Otherwise, we construct from $\mathcal{H}_{k+1}$ a UM adversary $\mathcal{U}$ interacting with parties $P''_1, P''_2, \ldots, P''_n$ that contradicts the assumption that λ_{k+1} emulates MT. $\mathcal{U}$ will simply run $\mathcal{H}_{k+1}$ (and complete $\mathcal{H}_{k+1}$'s missing information by running a simulated copy of π''_k). Whenever $\mathcal{H}_{k+1}$ activates an imaginary party simulated by $\mathcal{U}$, say P'_i, and during that activation that party ought to send a message authenticated using λ_{k+1}, $\mathcal{U}$ activates the corresponding party with whom it interacts, P''_i to send the same message to the same recipient using λ_{k+1}. $\mathcal{U}$ passes messages and outputs pertaining to the authenticator λ_{k+1} between the parties in its model and $\mathcal{H}_{k+1}$.

Furthermore, if λ_{k+1} must choose a value in this activation that has been replaced with another value in π''_{k+1} (e.g. from π or from another authenticator), $\mathcal{U}$ inputs this value to λ_{k+1}. (Note that the theorem stipulates that this value must be known to $\mathcal{H}_{k+1}$, and so it is also known to $\mathcal{U}$ since it is running an internal copy of $\mathcal{H}_{k+1}$.)

$\mathcal{U}$ simulates the rest of the messages and outputs pertaining to π''_{k+1} for $\mathcal{H}_{k+1}$ itself. $\mathcal{U}$ also simulates for $\mathcal{H}_{k+1}$ the other possible queries, such as party corruption and session corruption, and completes any missing information it needs for the authenticator by corrupting the corresponding party or session in its own model. Observe that expiry, session key and test session queries are not considered in the proofs of existing authenticators. However, this is not a problem, since the only information $\mathcal{H}_{k+1}$ can receive from such queries is the session key. $\mathcal{U}$ is able to answer these queries without interacting with the parties $P''_1, P''_2, \ldots, P''_n$ in its own model, since the session key cannot depend on any information generated by the authenticator λ_{k+1} (otherwise the session key would not be able to be specified in π''_k). Now, in the global output of the parties running λ_{k+1} with $\mathcal{U}$ the event that some party accepts a message that was never sent occurs with non-negligible probability. Yet, in the AM, with *any* adversary, this event never occurs. It now follows from the construction that P_j has sent m to P_i. Thus the triple (P_i, P_j, m) was added to the set of authentic undelivered messages.

To see (2), notice that, in the simulated run of $\mathcal{H}_{k+1}$, λ_{k+1} has not previously output 'P'_i received m from P'_j' on behalf of P'_i. Otherwise it would be the case that λ_{k+1} outputs 'P'_i received m from P'_j' twice on behalf of P'_i, and as above one can construct from $\mathcal{H}_{k+1}$ a UM adversary $\mathcal{U}$ that contradicts the assumption that λ_{k+1} emulates MT. It now follows from the construction that P_j was not previously activated with incoming authentic message m from P_j. Thus the triple (P_i, P_j, m) was not deleted from the set of authentic undelivered messages.

Let $\mathcal{C}_{k+1}$ be the compiler required to generate π''_{k+1} from π''_k (i.e. $\pi''_{k+1} = \mathcal{C}_{k+1}(\pi''_k)$). Then we have shown that $\mathcal{C}_{k+1}$ is an HM-authenticator.

We now show that if π''_k is SK-secure, then π''_{k+1} is also SK-secure. Observe that since $\mathcal{C}_{k+1}$ is an HM-authenticator, if π''_k satisfies requirement 1 of Definition 1, so must π''_{k+1}. Otherwise, the global outputs from running π''_k and π''_{k+1} would be easily distinguishable.

In addition, if π''_{k+1} violates requirement 2 of Definition 1, then so must π''_k. This is true since if there exists an adversary $\mathcal{H}_{k+1}$ against π''_{k+1} that has an advantage in outputting the correct bit b, and there does not exist an adversary $\mathcal{H}_k$ with the same advantage, then $\text{HYBRID}_{\pi''_k,\mathcal{H}_k}$ and $\text{HYBRID}_{\pi''_{k+1},\mathcal{H}_{k+1}}$ will not have negligible statistical distance, since the output of $\mathcal{H}_{k+1}$ will correctly indicate whether the answer to the test session was real or random more often than the output of $\mathcal{H}_k$.

Since $\pi''_0 = \pi$ is SK-secure, $\pi''_1, \pi''_2, \ldots, \pi''_{l-1}, \pi''_l = \pi''$ are all also SK-secure. □

Protocol 8 shows Protocol 7 modified to replace N_A of λ_{SIG} with t_A and to move the messages N_A and c to earlier parts of the protocol. Messages sent by the same party immediately after one another are combined into one message with duplicate terms removed. By Theorem 7, Protocol 8 is SK-secure.

A		B
$r_A \in_R \mathbb{Z}_q$ $t_A = g^{r_A}$	$\xrightarrow{t_A}$	
		$r_B \in_R \mathbb{Z}_q$ $t_B = g^{r_B}$ $K_{BA} = t_A^{r_B}$ Erase r_B Set SID $= s = (t_A, t_B)$ 'B sent (t_A, t_B) to A' $\sigma = \text{Sig}_B(t_A, t_B, t_A, A)$ $N_B \in_R \{0,1\}^k$ $c = \mathcal{E}_{\text{PK}_A}(N_B)$
	$\xleftarrow{s, \sigma, c}$	
Check $\sigma = \text{Sig}_B(t_A, t_B, t_A, A)$; if not then abort. '$A$ rec. (t_A, t_B) from B' $K_{AB} = t_B^{r_A}$ Erase r_A Set SID $= s = (t_A, t_B)$ 'Established (A, B, s, K_{AB})' 'A sent (t_A, t_B) to B'$N_B = \mathcal{D}_{\text{SK}_A}(c)$		
$\mu = \text{MAC}_{N_B}(t_A, t_B, B)$	$\xrightarrow{s, \mu}$	Check $\mu = \text{MAC}_{N_B}(t_A, t_B, B)$; if not then abort. '$B$ rec. (t_A, t_B) from A' 'Established (B, A, s, K_{BA})'

Protocol 8. Optimized Diffie-Hellman protocol in the UM (messages combined)

6 Conclusion

The Canetti-Krawczyk proof model is very attractive due to its modular nature. It enables an engineering approach to the provable security of protocols, by separating the proofs of authentication mechanisms from the proofs of protocols providing the basic required functionality such as key exchange. This enables a wide variety of proven secure protocols in the real world to be generated, by mixing and matching authentication mechanisms with basic key exchange protocols, yet with only a minimal number of security proofs.

However, until now, this engineering approach meant that proven secure protocols in the real world were often very inefficient in terms of the number of messages required and the number of values generated. Although optimization of such protocols was proposed in the original CK-model specification, the optimization process was not formally described or proven to preserve security. Only heuristic arguments were available to justify the process.

In this paper, the optimization process has been described, and theorems have been proven stating that the various optimization steps are valid and preserve security. A summary of these optimization steps is provided in Table 2. In addition, it has been proven that two or more authenticators may be used in conjunction with one another whilst preserving security. This result greatly increases the number and variety of available real-world protocols, and enables the use of different authentication mechanisms tailored to the needs of each participating party. As a consequence of our results, a number of optimized protocols from earlier papers [15, 16, 13, 7] now have full security proofs.

Optimization	Reason
Use of more than one authenticator with one AM protocol	Useful for protocols where capabilities of participants are not symmetrical, and increases the number and variety of possible secure protocols.
Omission of AM messages from the cleartext of messages generated by authenticators.	The message has often been sent elsewhere; this enables removal of redundancy.
Moving nonces or encrypted nonces generated by authenticators.	Enables the number of messages to be reduced by piggy-backing messages.
Substitution of any value not previously used as a nonce in that authenticator by the party for the nonce in most authenticators.	Saves on the number of values generated by the protocol and hence the message size.
Use of pre-existing protocol values for most nonces (provided they were previously unused).	”
Allowing the protocol to begin before the session identifier has been defined.	”
Use of pre-existing protocol values to create the session identifier.	”

Table 2 Summary of optimizations available

In the future, the number of basic proven secure components available in the CK-model is expected to increase. When new components in the AM or HM are proposed, a template specifying:

- the requirements if actions are to be reordered
- any additional unauthentic messages that may be sent containing values from authentic messages

will enable users of the component to easily create a concrete version of the protocol suitable for an optimization that preserves emulation between the real and ideal worlds. Similarly, when new authenticators are proposed, if the proposer indicates which parts of the authenticator may be omitted, which parts may be moved to an earlier or later position, and which parts may be replaced with other values provided they satisfy certain requirements, without invalidating the security proof, then users of these authenticators can easily optimize real world protocols created using these authenticators.

References

1. Michael Backes, Birgit Pfitzmann, Michael Steiner, and Michael Waidner. Polynomial fairness and liveness. In *15th IEEE Computer Security Foundations Workshop—CSFW 2002*, pages 160–174. IEEE Computer Society Press, June 2002.
2. 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. ACM Press, 1995.
3. Mihir Bellare, Ran Canetti, and Hugo Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 419–428. ACM Press, 1998. Full version at `http://www-cse.ucsd.edu/users/mihir/papers/key-distribution.html`.
4. Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 139–155. Springer-Verlag, 2000.

5. Mihir Bellare and Phillip Rogaway. Entity authentication and key distribution. In *Advances in Cryptology – CRYPTO'93*, volume 773 of *Lecture Notes in Computer Science*, pages 232–249. Springer-Verlag, 1993. Full version at `www-cse.ucsd.edu/users/mihir`.
6. Simon Blake-Wilson and Alfred Menezes. Authenticated Diffie-Hellman key agreement protocols. In S. Tavares et al., editors, *Selected Areas in Cryptography, 5th International Workshop*, volume 1556 of *Lecture Notes in Computer Science*, pages 339–361. Springer-Verlag, 1999.
7. Colin Boyd, Wenbo Mao, and Kenny Paterson. Key agreement using statically keyed authenticators. In *Applied Cryptography and Newwork Security (ACNS'04)*, volume 3089 of *Lecture Notes in Computer Science*, pages 248–262. Springer-Verlag, 2004. Corrected version at `http://sky.fit.qut.edu.au/~boydc/papers/acns04-corrected.pdf`.
8. Victor Boyko, Phillip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. In B. Preneel, editor, *Advances in Cryptology – Eurocrypt 2000*, volume 1807 of *Lecture Notes in Computer Science*, pages 156–171. Springer-Verlag, 2000.
9. Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Dynamic group Diffie-Hellman key exchange under standard assumptions. In *Advances in Cryptology – Eurocrypt 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 321–336. Springer-Verlag, 2002.
10. Ran Canetti. Universally composable security: a new paradigm for cryptographic protocols (extended abstract). In IEEE, editor, *42nd IEEE Symposium on Foundations of Computer Science*, pages 136–145. IEEE Computer Society Press, 2001. Full version available at: `http://eprint.iacr.org/2000/067`.
11. Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *Advances in Cryptology – Eurocrypt 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 453–474. Springer-Verlag, 2001. `http://eprint.iacr.org/2001/040.ps.gz`.
12. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the 34th Annual Symposium on Theory of Computing (STOC)*, pages 494–503. ACM Press, 2002. Full paper appeared as Cryptology ePrint Archive: Report 2002/140, `http://eprint.iacr.org/2002/140/`.
13. Yvonne Hitchcock, Yiu Shing Terry Tin, Colin Boyd, Juan Manuel González Nieto, and Paul Montague. A password-based authenticator: Security proof and applications. In *4th International Conference on Cryptology in India – INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*. Springer-Verlag, 2003.
14. ISO. *Information Technology – Security Techniques – Key Management – Part 3: Mechanisms Using Asymmetric Techniques ISO/IEC 11770-3*, 1999. International Standard.
15. Yiu Shing Terry Tin, Colin Boyd, and Juan Manuel González Nieto. Provably secure key exchange: An engineering approach. In Chris Johnson, Paul Montague, and Chris Steketee, editors, *Australasian Information Security Workshop*, volume 21 of *Conferences in Research and Practice in Information Technology*, pages 97–104, Adelaide, Australia, 2003. Australian Computer Society.
16. Yiu Shing Terry Tin, Colin Boyd, and Juan Manuel González Nieto. Provably secure mobile key exchange: Applying the Canetti-Krawczyk approach. In *Information Security and Privacy – ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 166–179. Springer-Verlag, 2003.