Key Agreement Using Statically Keyed Authenticators

Colin Boyd[1⋆], Wenbo Mao[2⋆⋆], and Kenneth G. Paterson[3⋆⋆⋆]

[1] Information Security Research Centre, Queensland University of Technology, Brisbane, Australia
[2] Hewlett-Packard Laboratories, Filton Road, Stoke Gifford, Bristol BS34 8QZ, UK
[3] Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK

Abstract. A family of authenticators based on static shared keys is identified and proven secure. The authenticators can be used in a variety of settings, including identity-based ones. Application of the authenticators to Diffie-Hellman variants in appropriate groups leads to authenticated key agreement protocols which have attractive properties in comparison with other proven-secure protocols. We explore two key agreement protocols that result.

1 Introduction

There is a vast range of protocols for key establishment. Historically such protocols have been regarded as difficult to design correctly and the literature is replete with broken examples. This has led to the realisation that a proof of security is an almost essential property of any new protocol. In recent years the number of key establishment protocols that carry a security proof has increased enormously. Most popular has been the model introduced by Bellare and Rogaway [3,4] and later refined by themselves and others.

In the modular approach to protocol design and proof [2], Bellare, Canetti and Krawczyk introduced the notion of an *authenticator* as a protocol translator. Protocols may be proven secure in an ideal model (the so-called authenticated links model, or simply the AM) in which the adversary is prevented from fabricating messages coming from uncorrupted principals. The role of the authenticator is to transform a protocol secure in the AM, into one that is secure in the more realistic unauthenticated links model (the UM). A major advantage of using this modular approach is that authenticators may be re-used with different AM protocols. This facilitates an engineering approach to protocol design, where components may be selected as appropriate to the application at hand. A

⋆ Work supported by the Australian Research Council through Discovery Project DP0345775.
⋆⋆ Research partially funded by the EU Fifth Framework Project IST-2001-324467 "CASENET".
⋆⋆⋆ Supported by the Nuffield Foundation, NUF-NAL 02.

M. Jakobsson, M. Yung, J. Zhou (Eds.): ACNS 2004, LNCS 3089, pp. 248–262, 2004.

potential disadvantage of the approach is that for some protocols there may not exist any efficient decomposition into an AM protocol and an authenticator. We remark that, despite the extensive theoretical framework that has been built up, there have been few new protocols proven secure as a result of this technique.

Bellare *et al.* [2] designed two general-purpose authenticators, one based on signatures and the other based on public key encryption. They showed how these authenticators can be used to generate efficient protocols with similar properties to some existing ones, but with the benefit of a formal security proof. In a later refinement of the technique, Canetti and Krawczyk [9] designed a MAC-based authenticator which uses a pre-existing shared secret as the MAC key.

In our earlier work [7], we focussed on deniability properties of protocols resulting from taking an identity-based approach to obtaining keys for the MAC-based authenticator of Canetti and Krawczyk [9]. In this paper, we provide a more detailed study of provable security aspects of MAC-based authenticators. We focus on two methods for obtaining the MAC key. The first uses static Diffie-Hellman keys (supported by certificates). The second uses an identity-based non-interactive key distribution protocol due to Sakai *et al.* [16]. We show that both authenticators have the security properties required to make them usable in the Canetti-Krawczyk methodology. By applying these authenticators to a basic Diffie-Hellman protocol that is secure in the AM, and using various optimisations, we obtain two concrete protocols that are provably secure in the UM. We compare our first protocol with the Unified Model protocol [5] and with the SIGMA protocol of Krawczyk [13]. We compare our second protocol to recent protocols of Chen and Kudla [10]. Analysis shows that our protocols are competitive with these existing protocols in terms of efficiency and security properties. Our protocols show that taking a systematic approach to the use of protocol components can bring new ideas to this heavily researched area.

The rest of this paper is structured as follows. In the next section we provide an overview of the modular approach to protocol proofs. Section 3 provides security proofs for our two MAC-based authenticators. In Sections 4 and 5 we develop and analyse the two key exchange protocols that result from using these two authenticators on the basic Diffie-Hellman protocol.

2 Authenticators and the Canetti–Krawczyk Model

In this section we describe the modular approach to protocol proofs [2] and the Canetti–Krawczyk (CK) model [9]. We aim to give an informal understanding of how the approach works, sufficient to follow the rest of the paper. However, we necessarily omit the formal details and refer the interested reader to the original papers [2,9].

The CK model is based on the idea of *message driven* protocols. In such protocols a set of principals $P_1, P_2, \ldots, P_n$ are activated either by:

- messages from the network;
- external requests to initiate a protocol run.

The output of a protocol consists of the cumulative output of all protocol principals as well as the output of the adversary.

In the AM, the adversary $\mathcal{A}$, a probabilistic polynomial time algorithm, controls the principals $P_1, \ldots, P_n$. The possible actions of $\mathcal{A}$ are:

activate a principal with incoming message m. This includes the ability to start a protocol session.
corrupt a principal and obtain its internal information.
session-key query to obtain the session key agreed in a completed session.
session-state reveal to obtain the internal state of a principal corresponding to an incomplete session. For example, this can include ephemeral parameters deleted after the session is complete.

In the AM, the adversary $\mathcal{A}$ may only activate principals with incoming messages that have already been sent by another principal to that principal. A set M of undelivered messages is defined. When a principal P_i sends a message m to another principal P_j then m is stored in M. Later m can be used by $\mathcal{A}$ to activate P_j (and no other party) and then m is deleted from M. All messages in the protocol are different; this is enforced by appending a session identifier that is unique for each session.

In the UM an adversary $\mathcal{U}$ has the same capabilities as $\mathcal{A}$ except that any message calculated by $\mathcal{U}$ may be used to activate principals. An important part of every protocol is an *initialisation function*, I, that sets up the public keys and associated parameters. An adversary is allowed to perform a special *test session* query by identifying an uncorrupted session whose principals are uncorrupted. The adversary is then given either the correct session key for this session or a random string of the same length, each with probability 1/2. The definition of protocol security is essentially the same in both the AM and the UM and is based on indistinguishability of these two strings.

Definition 1. *A protocol is called* SK-secure *if:*

- *two uncorrupted parties that complete sessions with matching identifiers both accept the same session key;*
- *the probability that the adversary can distinguish between the correct key in a test session and a random string of the same length is no more than 1/2 plus a negligible function in the security parameter.*

Canetti and Krawczyk [9] show that a protocol that is SK-secure in the AM is transformed into an SK-secure protocol in the UM if an authenticator is used. In order to explain what an authenticator is, we must first define the concept of emulation.

Definition 2. *A protocol π' in the UM,* emulates *a protocol π in the AM if given any adversary $\mathcal{U}$ against protocol π', there exists an adversary $\mathcal{A}$ against π such that the output of π' with adversary $\mathcal{U}$ is indistinguishable from π with $\mathcal{A}$.*

Definition 3. *An* authenticator *is a mapping of protocols that transforms a protocol π in the AM to a protocol π' in the UM such that π' emulates π.*

In common with Bellare *et al.* [2], we require the indistinguishability in Defn. 2 to be computational. That is, there should be no efficient algorithm that can distinguish the output of the two protocols.

All the authenticators we talk about in this paper take the special form, known as *message transmission authenticators*, or simply MT-authenticators. The message transmission protocol, MT, is the protocol in the AM that simply transmits a message between two principals. Formally, any party P_i may be activated with (P_j, m) in order to send (P_i, P_j, m) to P_j and then has output 'P_i sent m to P_j'. If party P_j is activated with (P_i, P_j, m) from P_i then P_j has output 'P_j received m from P_i'. Note that sending and receiving of a message m entails it being first stored in, and then removed from, the message store M.

A UM protocol is an MT-authenticator if it emulates the AM protocol MT. Bellare *et al.* [2] showed that the mapping of protocols obtained by replacing each message M in an AM protocol by an MT-authenticator corresponding to M is an authenticator. Therefore, given an SK-secure protocol in the AM, we can convert it to an SK-secure protocol in the UM simply by replacing each separate message of the AM protocol by the MT-authenticator for that message. It is often desirable to optimise the protocol that results from the naive use of this approach when there is more than one message in the basic AM protocol. This optimisation typically consists of piggy-backing flows from one authenticated message onto flows from another and reordering independent protocol messages. Further optimisation is often possible, and can be argued heuristically not to disturb the protocol security.

3 Two Authenticators

Our MT-authenticators can be viewed as variants of the MT-authenticator based on MACs that was proposed by Canetti and Krawczyk [9]. The format of our authenticators is shown in Fig. 1. On successful completion of the protocol, B will output 'B received m from A'. In Fig. 1 k is a security parameter, m is the message to be transmitted and H is a hash function with a k bit output which we replace by a random oracle in our security proofs. In the MAC-based MT-authenticator of Canetti and Krawczyk it is assumed that the key F_{AB} is already shared between A and B during the initialisation phase. In our authenticators F_{AB} is generated by A and B from the long-term keying material which is established in the initialisation phase. We consider two different methods for achieving this: static Diffie-Hellman and an identity-based approach. The main purpose of this section is to prove that our two methods of generating F_{AB} still produce authenticators (in the sense of Defn. 3). A second difference between our approach and that of Canetti and Krawczyk [9] is that we replace the MAC by a hash function where the static shared key is included as an input to the hash. The reason for this is that it makes both the authenticator and the proof a little simpler. We can only do this by modelling the hash function as a random

oracle, but in any case we would need to use the random oracle assumption in our proofs even if a MAC were used.

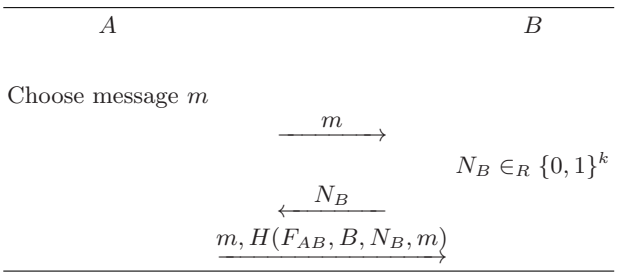

Fig. 1. Statically keyed MT-authenticator

3.1 Authenticator Using Static Diffie-Hellman

Here, the initialisation function I, on input security parameter k, generates for each party P_i a long-term key pair consisting of private key x_i and public key $y_i = g^{x_i}$, where g generates a group $\mathbb{G}$ of prime order q. The shared secret between principals P_i and P_j is $F_{P_iP_j} = g^{x_ix_j}$. When this key is used in the protocol of Figure 1, we name the resulting protocol λ_{SDH}. The fact that λ_{SDH} is an MT-authenticator relies on the difficulty of the following well known problem.

Computational Diffie-Hellman Problem (CDHP): Let $\mathbb{G}$, g and q be as above. The CDHP in $\mathbb{G}$ is as follows: Given g, g^x, g^y with $x, y \in \mathbb{Z}_q$, compute $g^{xy} \in \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the CDHP if

$$\Pr\left[\mathcal{A}(g, g^x, g^y) = g^{xy}\right] = \epsilon.$$

Here the probability is measured over random choices of x, y in $\mathbb{Z}_q$ and the random operations of $\mathcal{A}$. We will use the Computational Diffie-Hellman (CDH) Assumption, which states that ϵ is negligible in the security parameter k for all efficient algorithms $\mathcal{A}$.

Theorem 1. *Suppose that H is a random oracle. Then protocol λ_{SDH} emulates* MT *if the CDH Assumption holds.*

Proof. We follow [2] in our proof structure. Our aim is to take any adversary $\mathcal{U}$ against the protocol λ_{SDH} and construct $\mathcal{A}$ in the AM against MT such that the outputs of the two are indistinguishable. The first step in the proof is to show that most of the actions of $\mathcal{U}$ can be emulated by $\mathcal{A}$ in the 'obvious' way, then leaving the bulk of the proof to show that the exceptional case (which prevents completion of the obvious emulation) happens with only negligible probability in the security parameter.

The scenario is that $\mathcal{A}$ runs $\mathcal{U}$ and attempts to emulate the protocol output. For any party P in the AM we denote the corresponding party in the UM as P'. Note that $\mathcal{A}$ simulates the actions of all parties in the UM. Firstly $\mathcal{A}$ chooses and distributes all the long-term keys for all parties in the protocol using function I. When $\mathcal{U}$ activates party A' in the UM to send message m to party B' then $\mathcal{A}$ activates A to send m to B in the AM. (Recall that this entails m being put into the message store M.) Similarly, when B' in the UM outputs 'B' received m from A'', $\mathcal{A}$ activates B with message m from A in the AM. When $\mathcal{U}$ corrupts a party in the UM, $\mathcal{A}$ corrupts the corresponding party in the AM and hands the information (including the long-term key which $\mathcal{A}$ has) to $\mathcal{U}$. Finally $\mathcal{A}$ outputs whatever $\mathcal{U}$ outputs.

The only obstacle occurs if $\mathcal{A}$ wants to activate a party in the AM with a message which is not in the set M of stored messages. Let $\mathcal{B}$ be the event that, for uncorrupted parties A' and B', B' outputs 'B' received m from A'' and either A was not activated by $\mathcal{A}$ to send m to B, or B previously output 'B received m from A'. Suppose that $\mathcal{B}$ occurs with non-negligible probability $\epsilon(k)$. We will show that if this is the case then it is possible to solve the CDHP with non-negligible probability. This will contradict our CDH Assumption.

From now we assume the existence of an efficient algorithm $\mathcal{U}$ that runs the protocol in the UM such that event $\mathcal{B}$ occurs with non-negligible probability $\epsilon(k)$. We also assume that $\mathcal{U}$ will complete in finite time $\mathcal{T}(k)$. We construct an algorithm $\mathcal{V}$ that interacts with $\mathcal{U}$ in order to solve the CDHP. $\mathcal{V}$ simulates the actions of all parties $P_1, P_2, \ldots, P_n$ and must be able to respond properly to all actions of $\mathcal{U}$. $\mathcal{V}$ also mediates calls to the random oracle H.

$\mathcal{V}$ is given as input a tuple $(\mathbb{G}, g, q, g^x, g^y)$ and is tasked with the problem of finding g^{xy}. Let $n(k)$ be a polynomial bound (in the security parameter k) on the number of principals that might be activated by $\mathcal{U}$. Firstly $\mathcal{V}$ chooses two parties P_f, P_g with $f \neq g$ randomly from the set of all parties $P_1, \ldots, P_n$. $\mathcal{V}$ then generates long-term secret keys x_i chosen randomly from $\mathbb{Z}_q$ for all parties P_i except P_f and P_g. The protocol parameters given to $\mathcal{U}$ are the group parameters $(\mathbb{G}, g, q)$, the public keys $y_i = g^{x_i}$ for parties P_i different from P_f and P_g, and g^x, g^y for parties P_f and P_g respectively.

When $\mathcal{U}$ activates any party, $\mathcal{V}$ follows the protocol specification on behalf of that party, choosing a random value for the output of all queries made to H in that process. $\mathcal{V}$ stores all the hash values output, together with the inputs to H, in a list L and uses L to consistently reply to H queries. Note that for H queries resulting from protocol runs between P_f and P_g, $\mathcal{V}$ does not know the value g^{xy} and so cannot calculate the correct entry to place on the list L. Instead, $\mathcal{V}$ places the symbolic string "$F_{P_f P_g}$" on L along with a record of the other values that should have been input to H in that query. $\mathcal{V}$ can then use these entries to recognize subsequent queries of the same type. $\mathcal{U}$ may also make H queries not associated with any particular protocol run; $\mathcal{V}$ responds to these as above, using the list L to ensure that queries are answered consistently. If $\mathcal{U}$ corrupts any party P_i apart from P_f and P_g then $\mathcal{V}$ can answer with x_i. If $\mathcal{U}$ corrupts P_f or P_g, then $\mathcal{V}$ terminates with failure.

The simulation continues until $\mathcal{U}$ halts, or time $\mathcal{T}(k)$ has passed. In the latter cases $\mathcal{V}$ simply aborts $\mathcal{U}$. Finally, $\mathcal{V}$ chooses α randomly from the prefixes of all oracle queries in L, and returns α as its guess for g^{xy}.

Let $\mathcal{J}$ denote the event that $\mathcal{V}$ is successful. We must now evaluate $\Pr(\mathcal{J})$.

It is easy to see that because H is a random oracle, the simulation provided by $\mathcal{V}$ is indistinguishable from what $\mathcal{U}$ would see in a real attack, unless $\mathcal{U}$ makes an oracle query prefixed by the value g^{xy} at some point in the simulation, or unless $\mathcal{U}$ corrupts P_f or P_g. In the former case, $\mathcal{V}$ may not consistently reply to oracle queries since $\mathcal{V}$ cannot "recognise" the input g^{xy}. Then $\mathcal{U}$'s behaviour is strictly speaking undefined, but this is catered for in our simulation because $\mathcal{V}$ will eventually halt $\mathcal{U}$.

Let $\mathcal{E}$ denote the event that $\mathcal{U}$ makes an oracle query prefixed by the value g^{xy}. Let $\mathcal{F}$ denote the event that $\mathcal{B}$ occurs with $\{A', B'\} = \{P_f, P_g\}$. In other words, $\mathcal{F}$ is the event that a message was passed between P_f and P_g that was not previously sent or already received by one of these parties. Since f and g were chosen at random from $\{1, \ldots, n(k)\}$, we have $\Pr(\mathcal{F}) = \epsilon(k)/\binom{n(k)}{2}$. Let $\mathcal{G}$ denote the event that $\mathcal{U}$ corrupts P_f or P_g. Notice that if event $\mathcal{F}$ occurs, then P_f and P_g must be uncorrupted. Hence if $\mathcal{F}$ occurs, then so must the event $\neg\mathcal{G}$.

After event $\mathcal{E}$, $\mathcal{V}$'s simulation may no longer be correct, but up until the point that $\mathcal{E}$ occurs, it is, so long as $\mathcal{G}$ does not occur. Therefore the probability that $\mathcal{E}$ occurs in the simulation is the same as the probability that it does in $\mathcal{U}$'s real attack, provided $\mathcal{G}$ does not occur. Note too that $\mathcal{V}$'s probability of success $\Pr(\mathcal{J})$ is equal to $\Pr(\mathcal{E} \wedge \neg\mathcal{G})/q_H(k)$ where $q_H(k)$ is a polynomial bound on the number of hash queries made during the simulation. This is because of the way that $\mathcal{V}$ selects an entry at random from the list L and because $\mathcal{V}$ aborts if event $\mathcal{G}$ occurs.

Now suppose that event $\mathcal{F}$ occurs. This means that either P_f or P_g has accepted a value which equals the output of a query to H prefixed with g^{xy}. Then either $\mathcal{U}$ has successfully guessed an output of H without making the relevant query to H, an event of probability 2^{-k}, or event $\mathcal{E}$ occurs. Notice too that if $\mathcal{F}$ occurs, then so does $\neg\mathcal{G}$. Hence we have $\Pr(\mathcal{F}) \leq 2^{-k} + \Pr(\mathcal{E} \wedge \neg\mathcal{G})$, from which we deduce $\Pr(\mathcal{E} \wedge \neg\mathcal{G}) \geq \Pr(\mathcal{F}) - 2^{-k} := \epsilon_1(k)$, a non-negligible quantity. Hence we have

$$\Pr(\mathcal{J}) = \Pr(\mathcal{E} \wedge \neg\mathcal{G})/q_H(k) \geq \epsilon_1(k)/q_H(k).$$

We see that $\Pr(\mathcal{J})$, the probability that $\mathcal{V}$ is successful, is non-negligible in the security parameter k. This completes the proof. □

Remark. The reduction in the proof may be tightened if we assume, instead of the CDH Assumption, that the gap-DH problem is hard. With this assumption we may allow $\mathcal{V}$ access to an oracle that will distinguish between Diffie-Hellman triples and random triples in $\mathbb{G}$. The gap-DH assumption [15] is that CDHP is still hard even given access to this oracle. In this case $\mathcal{V}$ can test if $\mathcal{U}$ has asked a critical query (one involving g^{xy}) of the random oracle and can abort the protocol run with certainty of the correct answer at that point. Therefore we can improve the success probability to the following:

$$\Pr(\mathcal{J}) \geq \epsilon_1(k).$$

3.2 Authenticator Using Identity-Based Static Keys

Using the notation of Boneh and Franklin [6], we let $\mathbb{G}_1$ be an additive group of prime order q and $\mathbb{G}_2$ be a multiplicative group of the same order q. We assume the existence of an efficiently computable, non-degenerate, bilinear map $\hat{e}$ from $\mathbb{G}_1 \times \mathbb{G}_1$ to $\mathbb{G}_2$. Typically, $\mathbb{G}_1$ will be a subgroup of the group of points on an elliptic curve over a finite field, $\mathbb{G}_2$ will be a subgroup of the multiplicative group of a related finite field and the map $\hat{e}$ will be derived from either the Weil or Tate pairing on the elliptic curve. By $\hat{e}$ being bilinear, we mean that for $Q, W, Z \in \mathbb{G}_1$, both

$$\hat{e}(Q, W + Z) = \hat{e}(Q, W) \cdot \hat{e}(Q, Z) \quad \text{and} \quad \hat{e}(Q + W, Z) = \hat{e}(Q, Z) \cdot \hat{e}(W, Z).$$

By $\hat{e}$ being non-degenerate, we mean that for some element $P \in \mathbb{G}_1$, we have $\hat{e}(P, P) \neq 1_{\mathbb{G}_2}$.

When $a \in \mathbb{Z}_q$ and $Q \in \mathbb{G}_1$, we write aQ for Q added to itself a times, also called scalar multiplication of Q by a. As a consequence of bilinearity, we have that, for any $Q, W \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_q$:

$$\hat{e}(aQ, bW) = \hat{e}(Q, W)^{ab} = \hat{e}(abQ, W).$$

We refer to [1,6,11] for a more comprehensive description of how these groups, pairings and other parameters should be selected in practice for efficiency and security.

In this setting, the initialisation function I on input a security parameter k selects suitable groups $\mathbb{G}_1$, $\mathbb{G}_2$ and map $\hat{e}$. Then I generates a random key $s \in \mathbb{Z}_q$. This key will play the role of the master secret of the Trusted Authority in the ID-based system. Then I distributes to each party P_i with identity ID_i a long-term key pair consisting of public key $Q_i = H_1(ID_i)$ and private key $S_i = sQ_i$. Here H_1 is a hash function mapping identities $ID_i \in \{0,1\}^*$ onto $\mathbb{G}_1$.

With this initialisation, any two principals P_i, P_j with identities ID_i, ID_j can efficiently calculate the shared key $F_{ij} = \hat{e}(Q_i, Q_j)^s = \hat{e}(S_i, Q_j) = \hat{e}(S_j, Q_i)$. This method of *identity-based, non-interactive key distribution* is due to Sakai *et al.* [16]. When this key is used in the protocol of Figure 1, we call the resulting protocol λ_{SBDH}. The fact that λ_{SBDH} is an MT-authenticator relies on the difficulty of the following problem.

Bilinear Diffie-Hellman Problem (BDHP): Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\hat{e}$ be as above. The BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ is as follows: Given $\langle P, xP, yP, zP \rangle$ with $P \in \mathbb{G}_1$ and $x, y, z \in \mathbb{Z}_q$, compute $\hat{e}(P, P)^{xyz} \in \mathbb{G}_2$. An algorithm $\mathcal{A}$ has advantage ϵ in solving the BDHP in $\langle \mathbb{G}_1, \mathbb{G}_2, e \rangle$ if

$$\Pr\left[\mathcal{A}(\langle P, xP, yP, zP \rangle) = \hat{e}(P, P)^{xyz}\right] = \epsilon.$$

Here the probability is measured over random choices of $P \in \mathbb{G}_1$, $x, y, z \in \mathbb{Z}_q$ and the random operations of $\mathcal{A}$. We will use the Bilinear Diffie-Hellman Assumption, which states that, for all efficient algorithms $\mathcal{A}$, the advantage ϵ is negligible as a function of the security parameter k used in generating $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$.

Theorem 2. *Suppose that H and H_1 are random oracles. Then protocol λ_{SBDH} emulates* MT *if the Bilinear Diffie-Hellman Assumption holds.*

Proof. The proof is the same as for Theorem 1 until we assume the existence of an efficient algorithm $\mathcal{U}$ which runs the protocol in the UM such that event $\mathcal{B}$ occurs with non-negligible probability ϵ. We construct an algorithm $\mathcal{V}$ that interacts with $\mathcal{U}$ in order to solve the BDHP.

This time $\mathcal{V}$ is given as input $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ and a tuple $\langle P, xP, yP, zP \rangle$ with the aim of finding $\hat{e}(P,P)^{xyz}$. The idea is that z will take the role of s and xP and yP will be the public keys of two entities, P_f and P_g, where f and g are selected randomly from $\{0, 1, \ldots, n(k)\}$. Protocol parameters $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ are given to $\mathcal{U}$At any point, $\mathcal{U}$ generates the public key for entity P_i with identity ID_i by making an H_1 query on ID_i. $\mathcal{V}$ handles these H_1 queries as follows. When $i \neq f, g$, $\mathcal{V}$ responds with $H_1(ID_i) = h_i P$ where $h_i \in \mathbb{Z}_q$ is selected at random. $\mathcal{V}$ then sets the private key for entity P_i to $S_i = h_i \cdot zP$. When $i = f$, $\mathcal{V}$ responds with xP and when $i = g$, $\mathcal{V}$ responds with yP.

When $\mathcal{U}$ activates any party, $\mathcal{V}$ follows the protocol specification on behalf of that party, choosing a random value for the output of all queries made to H in that process. $\mathcal{V}$ stores all the hash values output, together with the inputs to the H, in a list L and uses L to consistently reply to H queries. Note that for H queries resulting from protocol runs between P_f and P_g, $\mathcal{V}$ does not know the value $\hat{e}(P,P)^{xyz}$ and so cannot calculate the correct entry to place on the list L. Instead, $\mathcal{V}$ places the symbolic string "$F_{P_f P_g}$" on L along with a record of the other values that should have been input to H in that query. $\mathcal{V}$ can then use these entries to recognize subsequent queries of the same type. $\mathcal{U}$ may also make H queries not associated with any particular protocol run; $\mathcal{V}$ responds to these as above, using the list L to ensure that queries are answered consistently. If $\mathcal{U}$ corrupts any party P_i apart from P_f and P_g then $\mathcal{V}$ can answer with S_i. If $\mathcal{U}$ corrupts P_f or P_g then $\mathcal{V}$ terminates with failure.

The simulation continues until $\mathcal{U}$ halts or time $\mathcal{T}(k)$ has passed. In the latter case $\mathcal{V}$ simply aborts $\mathcal{U}$. Finally, $\mathcal{V}$ chooses α randomly from the prefixes of all oracle queries in L, and returns α as its guess for $\hat{e}(P,P)^{xyz}$.

The rest of the proof is the same as that of Theorem 1, with the value $\hat{e}(P,P)^{xyz}$ replacing the value g^{xy}. □

4 An Authenticated Key Agreement Protocol

In this and the following section we will examine the effect of applying our authenticators to the basic Diffie-Hellman protocol in the AM in order to derive SK-secure protocols in the UM. We present optimised versions of the protocols rather than the protocols that result by naively applying the authenticators to each AM protocol message. The latter protocols automatically carry a security proof but are not very efficient. In contrast, our optimised protocols rely on heuristic arguments that the proofs are preserved in making the various optimisations.

Canetti and Krawczyk [9] proved that Diffie-Hellman with ephemeral keys is an SK-secure protocol in the AM as long as the Decision Diffie-Hellman (DDH)

Assumption holds in the group in which it is executed.[1] Although we could use groups for which the DDH Assumption is believed hard, we can instead use the weaker CDH Assumption and hash the Diffie-Hellman key with a hash function modelled as a random oracle. So for our *basic Diffie-Hellman protocol*, A and B choose random values r_A and r_B respectively, exchange the values g^{r_A} and g^{r_B}, and calculated the shared secret $Z_{AB} = H_2(g^{r_A r_B})$ where H_2 is a random oracle. Since we are already using the random oracle model for the authenticators, it seems logical to use it here too. This also allows us to be more flexible in our choice of groups.

The following theorem may be proven in a standard way. We emphasise that although Canetti and Krawczyk use the stonger DDH Assumption they do not use the random oracle assumption, so their result on the AM security of Diffie-Hellman is not weaker.

Theorem 3. *The basic Diffie-Hellman protocol is SK-secure in the AM given the CDH Assumption and that H_2 is a random oracle.*

Here, we leave the details of the ephemeral Diffie-Hellman group flexible, assuming only that it is a cyclic group of prime order in which the CDH Assumption holds. Natural choices for this group will arise in the context of each of our two authenticators.

4.1 Key Agreement from Static Diffie-Hellman

We have shown in Section 3 how an MT-authenticator can be obtained from static Diffie-Hellman key exchange. The general result of Canetti and Krawczyk [9] and Theorem 3 ensure that using this MT-authenticator to replace each message of the basic Diffie-Hellman protocol results in a protocol that is SK-secure in the UM. We present an optimised version of this protocol as Protocol 1. The group $\mathbb{G}$ already needed for the authenticator is a natural choice for the basic Diffie-Hellman exchange, and we have selected it here. In Protocol 1, parties A and B generate ephemeral secrets r_A and r_B, and exchange values $t_A = g^{r_A}$ and $t_B = g^{r_B}$. In the optimisation process, messages have been piggy-backed upon one another and values t_A and t_B play dual roles as both messages and the random nonces of A and B respectively. In addition the pair (t_A, t_B) form a unique session identifier which is required for all protocols. We have also added the identities of the sender in the first two messages in plaintext. This will be a functional necessity in case either party does not initially know with which partner it is running the protocol. Both A and B can compute the shared secret Z_{AB} which is then used to calculate a session key using a suitable key derivation function.

[1] The DDH Assumption says that Diffie-Hellman triples (g^x, g^y, g^{xy}) cannot be efficiently distinguished from triples (g^x, g^y, g^z), where x, y, z are random exponents.

Shared Information: Static Diffie-Hellman key: $F_{AB} = g^{x_A x_B}$.

A		B
$r_A \in_R \mathbb{Z}_q$		
$t_A = g^{r_A}$	$\xrightarrow{A, t_A}$	$r_B \in_R \mathbb{Z}_q$
		$t_B = g^{r_B}$
	$\xleftarrow{B, t_B, H(F_{AB}, B, t_A, t_B)}$	
Verify hash		
	$\xrightarrow{H(F_{AB}, A, t_B, t_A)}$	Verify hash
$Z_{AB} = H_2(t_B^{r_A})$		$Z_{AB} = H_2(t_A^{r_B})$

Protocol 1: Key agreement protocol based on static Diffie-Hellman

4.2 Comparison with Related Protocols

The most similar proven secure protocol to Protocol 1 is the Unified Model Protocol (UMP) analysed by Blake-Wilson and Menezes [5] in the Bellare–Rogaway model. In the variant of the UMP that we consider, the shared secret Z_{AB} is equal to $g^{r_A r_B} || F_{AB}$, the concatenation of ephemeral and static Diffie-Hellman keys. A MAC key (for confirmation) and session key are both derived from Z_{AB} using independent hash functions.

Protocol 1 and the UMP have similar properties and efficiency which we will summarise below. Another proven secure protocol that is worth examining is the SIGMA protocol of Krawczyk [13] which was used as the basis of the Internet Key Exchange (IKE) protocol and which has also been proposed as the basis for its replacement. It is interesting to note that when it comes to additional properties SIGMA and Protocol 1 are in some senses complementary. We will give further details below.

Efficiency. In Protocol 1, each principal has to complete three exponentiations in total. The computational requirements of Protocol 1 are almost identical to those of the UMP. However, one difference is that both parties can complete Protocol 1 before computing the ephemeral secret Z_{AB}. This means that Protocol 1 can be completed more quickly than the UMP.

Protocol 1 has better computational and bandwidth efficiency than SIGMA. An exact comparison relies on the details of the signature scheme used in SIGMA and the size of various parameters. The MQV protocol [14] has slightly smaller computational requirements in total but currently has no published security proof. In terms of bandwidth, Protocol 1 and the UMP also seem to be optimal. The only components included are the ephemeral key and the MAC or hash. In contrast, the SIGMA protocol includes both a MAC and a digital signature sent by each party, in addition to the Diffie-Hellman ephemeral key.

Identity Protection and Knowledge of Peer Entity. One of the main distinctive properties of SIGMA, which motivated its design, is strong identity protection. This property allows the protocol principals to hide their identities from adversaries. In contrast to SIGMA, Protocol 1 and the UMP do not seem well suited to provide identity protection, since each party must know the identity of the other in order to calculate F_{AB} and authenticate to the other.

As a consequence of its strong identity protection, a property that is missing from the SIGMA protocol, but which is held by Protocol 1 and the UMP, is what is often called 'knowledge of the peer entity'. This means that party A in SIGMA can complete the protocol without any indication that B is prepared to communicate. By simply deleting the final message, A will accept the session key, apparently shared with B, even though B may never have received any indication of the existence of A. Krawcyzk discusses the possibility of adding an extra acknowledgement message from B to add knowledge of the peer entity to SIGMA, but this has obvious drawbacks.

Key Compromise Impersonation and Deniability. Although SIGMA is more flexible in providing identity protection, Protocol 1 fares much better than SIGMA when it comes to another desirable feature: deniability. This is the property that each party should be able to deny having taken part in the protocol run. Krawczyk pointed out that although SIGMA provides deniability when both parties cooperate with each other, if one party defects by revealing its random input, then the other cannot deny taking part.

Following the definition of *deniable encryption* by Canetti *et al.* [8] we may say that a two-party protocol is *deniable* for party A, if a legitimate party B could have simulated the protocol without the presence of A. Since all protocols using our authenticators can be simulated perfectly by either party the protocols can be seen to provide very strong deniability: there is no situation in which one party can prevent the other from denying having been involved.

Another property that may be useful is resistance to key compromise impersonation (KCI). It is obvious that in the situation when the adversary obtains the long-term key of a party A, the adversary can masquerade as A; KCI is possible if the adversary can masquerade as other parties to A in this event. Such an attack could potentially allow compromise of A's private key to go undetected.

The SIGMA protocols provide protection against KCI since any partner of A needs to demonstrate knowledge of its own private key through generation of a new signature. However, Protocol 1 and the UMP do not protect against KCI. This is because knowledge of either party's private key is enough to complete the protocol. The mechanisms used to prevent KCI and allow undeniability are in conflict. In SIGMA, signatures protect against KCI but prevent deniability, while in Protocol 1 and the UMP, the mechanism used to provide deniability allows KCI.

5 Identity-Based Key Agreement

There has been considerable recent interest in identity-based key agreement based on pairings. In this section we give more detail of how to apply our identity-based authenticator to derive an identity-based key agreement protocol. We compare this protocol with a protocol proven secure in [10].

We assume the same algebraic setting as in Section 3.2. In principle we can use Diffie-Hellman in any group for the AM protocol. However, it is practical to choose a group that is already implemented for the authenticator. The group $\mathbb{G}_1$ is an obvious choice if we make the natural assumption that the CDHP is hard in $\mathbb{G}_1$: we are already making the stronger assumption that the BDHP is hard for $\langle \mathbb{G}_1, \mathbb{G}_2, \hat{e} \rangle$ to obtain a secure authenticator anyway. We can use any non-zero element $P \in \mathbb{G}_1$ as the base for the Diffie-Hellman protocol. The result of applying our identity-based MT-authenticator to the basic Diffie-Hellman protocol and then optimising is the identity-based key agreement protocol shown as Protocol 2. Its properties are explored in detail in [7], in particular its strong deniability feature.

Shared Information:
Fixed key F_{AB} derived from static Diffie-Hellman key: $F_{AB} = H_1(\hat{e}(Q_A, Q_B)^s)$;
A generator P of $\mathbb{G}_1$.

A		B
$r_A \in_R \mathbb{Z}_q$		
$T_A = r_A P$	$\xrightarrow{A, T_A}$	$r_B \in_R \mathbb{Z}_q$
		$T_B = r_B P$
	$\xleftarrow{B, T_B, H(F_{AB}, B, T_A, T_B)}$	
Verify hash		
	$\xrightarrow{H(F_{AB}, A, T_B, T_A)}$	Verify hash
$Z_{AB} = H_2(r_A T_B)$		$Z_{AB} = H_2(r_B T_A)$

Protocol 2: Identity-based key agreement protocol

The most efficient identity-based key agreement protocol with confirmation and enjoying a security proof currently seems to be that due to Chen and Kudla [10, Protocol 4]. Their proof is in the Bellare–Rogaway model but makes the strong restriction that the adversary reveals no session keys in the course of its attack. In [10, Protocol 4], the parties exchange ephemeral values $r_A Q_A$ and $r_B Q_B$ and calculate the shared secret $Z_{AB} = \hat{e}(Q_A, Q_B)^{s(r_A + r_B)}$. MACs are used in a standard way to provide key confirmation.

Protocol 4 of [10] requires each party to compute one elliptic curve pairing and two elliptic curve multiplications (equivalent to exponentiations in a mul-

tiplicative group). Therefore the computational effort required is the same as that in Protocol 2. However, if many protocol instances take place between the same parties, Protocol 2 can cache and re-use the same F_{AB} value and therefore requires only one pairing for all of these protocols. In contrast Protocol 4 of [10] requires one pairing for every protocol run, even between the same parties.

Although Protocol 4 of [10] does not provide forward secrecy, Chen and Kudla do provide alternatives with this property. With regard to additional properties discussed in Section 4, Protocol 4 of [10] seems to share some similarity with our protocol and some similarity with the SIGMA protocol. Chen and Kudla establish that their protocol protects against KCI attacks, though only in a security model where no reveal queries are made by the adversary. Their protocol also provides deniability: it is easy to see that either party can simulate a protocol run. Identity protection is not discussed by Chen and Kudla, but their protocols require each party to know the identity of the other party in order to derive the session key. Therefore it seems that, like our protocols, identity protection against active adversaries cannot be efficiently achieved. Protocol 4 of [10] does provide confirmation of knowledge of the peer entity.

6 Conclusion

We have shown that the CK model can be profitably used to design novel, provably secure key exchange protocols. We obtain protocols that have not been proven secure before; subsequent optimisation yields protocols with efficiency properties equal to or better than all known similar protocols. We have also examined the additional properties that our protocols possess. Our work illustrates that a systematic approach to design of provable security can have other benefits apart from the proofs themselves. It may be profitable to augment our authenticators with more examples by using different ways of deriving static keys. The self-certified keys of Girault [12] are one promising example. In addition we reiterate that our authenticators can also be used with any other proven secure AM protocol to provide new SK-secure protocols. One AM protocol that could be used is the key transport protocol proven SK-secure by Canetti and Krawcyzk [9]. A particularly interesting key transport protocol results if only identity-based components are used in the construction.

References

1. P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology - CRYPTO 2002*, LNCS. Springer-Verlag, 2002.
2. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the thirtieth annual ACM symposium on theory of computing*, pages 419–428. ACM Press, 1998.
3. M. Bellare and P. Rogaway. Entity authentication and key distribution. In D.R. Stinson, editor, *Advances in Cryptology – CRYPTO'93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1994.

4. M. Bellare and P. Rogaway. Provably secure session key distribution – the three party case. In *27th ACM Symposium on Theory of Computing*, pages 57–66. ACM Press, 1995.
5. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *Crypography and Coding - 6th IMA Conference*, pages 30–45. Springer-Verlag, 1997. LNCS Vol. 1355.
6. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):585–615, 2003.
7. C. Boyd, W. Mao, and K.G. Paterson. Deniable authenticated key establishment for Internet protocols. In *Proceedings of 11th International Workshop on Security Protocols*, LNCS. Springer-Verlag, to appear.
8. R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky. Deniable encryption. In B. Kaliski, editor, *Advances in Cryptology – Crypto'97*, volume 1294 of *LNCS*, pages 90–104. Springer-Verlag, 1997.
9. R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In B. Pfitzmann, editor, *Advances in Cryptology – Eurocrypt 2001*, volume 2045 of *LNCS*, pages 453–474. Springer-Verlag, 2001.
10. L. Chen and C. Kudla. Identity based authenticated key agreement protocols from pairings. In *IEEE Computer Security Foundations Workshop*, pages 219–233. IEEE Computer Society Press, 2003. Updated version at Cryptology ePrint Archive, Report 2002/184.
11. S.D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate pairing. In C. Fieker and D.R. Kohel, editors, *Algorithmic Number Theory – ANTS-V*, volume 2369 of *LNCS*, pages 324–337. Springer-Verlag, 2002.
12. M. Girault. Self-certified public keys. In D. W. Davies, editor, *Advances in Cryptology – EUROCRYPT 1991*, volume 547 of *LNCS*, pages 490–497. Springer-Verlag, 1992.
13. H. Krawczyk. SIGMA: The SIGn and MAc approach to authenticated Diffie-Hellman and its use in the IKE protocols. In D. Boneh, editor, *Advances in Cryptology – Crypto 2003*, volume 2729 of *LNCS*, pages 400–425. Springer-Verlag, 2003.
14. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, March 2003.
15. T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In K. Kim, editor, *Public Key Cryptography (PKC'01)*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, 2001.
16. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. In *The 2000 Sympoium on Cryptography and Information Security*, Okinawa, Japan, January 2000.