Modular Security Proofs for Key Agreement Protocols

Caroline Kudla* and Kenneth G. Paterson

Information Security Group
Royal Holloway, University of London, UK
{c.j.kudla,kenny.paterson}@rhul.ac.uk

Abstract. The security of key agreement protocols has traditionally been notoriously hard to establish. In this paper we present a modular approach to the construction of proofs of security for a large class of key agreement protocols. By following a modular approach to proof construction, we hope to enable simpler and less error-prone analysis and proof generation for such key agreement protocols. The technique is compatible with Bellare-Rogaway style models as well as the more recent models of Bellare *et al.* and Canetti and Krawczyk. In particular, we show how the use of a decisional oracle can aid the construction of proofs of security for this class of protocols and how the security of these protocols commonly reduces to some form of Gap assumption.

1 Introduction

Background

The first works formalizing the notion of security for key agreement were those of Bellare and Rogaway [7, 8]. Extensions have been made to these models, most notably by Blake-Wilson *et al.* [9] and later Bellare *et al.* [6]. Although these models are generally accepted as being reasonable approaches to modelling the security of key agreement protocols, in general it appears to be rather difficult to prove key agreement protocols secure in such models and only relatively few protocols have full proofs of security in these models.

A more "modular" approach to constructing key agreement protocols was advocated by Bellare, Canetti and Krawczyk [5]. This approach entails constructing a secure protocol for ideally "authenticated links", and then applying "authenticators" to all the protocol flows to obtain a protocol secure in the standard "unauthenticated links" model. A library of basic protocols and authenticators may be built up, from which many different secure key agreement protocols may be constructed.

* This author is funded by Hewlett-Packard Laboratories.

The disadvantage of using this modular approach is that it says nothing about the security of certain very efficient protocols that are not constructed in this modular way. In addition, cryptographic primitives such as encryption, signatures or MACs are usually required to build these "authenticators" and the application of these "authenticators" often increases the round complexity of a protocol. Therefore the resulting protocols are also often less efficient than protocols designed without the modular approach in mind. Of course protocols constructed using this modular approach may be modified to be more efficient using various techniques, but then the security proof may no longer be valid.

However, due to the ease of designing secure protocols using this modular approach, it has subsequently been advocated in later models such as [15, 16] and has been used in the design of various key agreement protocols such as [12, 13, 11, 22]. Although the security models of [5, 15, 16] do not mandate a modular approach in that their definitions of security apply directly in the standard unauthenticated network model, they do not consider protocols that are not constructed in this modular fashion. Direct proofs for non-modular protocols in the standard unauthenticated network models of [5, 15, 16] seem to be difficult to construct.

In many environments, the benefits of being able to easily design secure protocols outweigh the possible disadvantages. However there exist environments in which efficiency is of utmost importance, and most key agreement protocols optimized for efficiency are not constructed in a modular way. Indeed we can find several efficient key agreement protocols in the literature which do not have formal proofs of security (such as protocols in [9, 19, 23, 24, 27]) or have only proofs of security in weakened models (such as protocols in [2, 3, 17]. Since the structure of these protocols is not compatible with the modular approach in [5], complete proofs of security for such protocols appear to be difficult to construct.

Contributions

In this paper, we consider protocols which are not designed in a modular way but which we nevertheless wish to prove secure. Since such protocols are not designed in a modular way, the proofs of security are often complicated and error-prone. We present a technique by which the proof process of a large class of key agreement protocols can be simplified.

Informally, our technique for proving the security of a protocol Π works as follows. The first step is to prove that protocol Π has a property that we call "strong partnering" (which is defined in Section 4.1). The second step is to prove that a related protocol π is secure in a highly reduced security model. Finally, as the main result of the paper, we show

how the proof of security of π in the reduced model can be translated into a proof of security for Π in the full security model using a Gap assumption.

Each step above is far simpler than a single proof of security in the full security model. The result is a modular technique for constructing proofs of security for a large class of key agreement protocols which are not constructed using the modular approach presented in [5].

We then use this technique to consider various key agreement protocols in the literature previously without proofs or with incomplete proofs of security. It is possible, using our techniques, to provide full proofs of security for protocols such as [2, 3, 9, 17, 27] (possibly after slight modifications to the protocols if necessary). Due to lack of space, we focus in detail only on the long-standing Protocol 4 in [9] which was previously without proof.

We also hope that our methods will aid future designers of lightweight key agreement protocols in the formal analysis of their protocols in simplifying their task by breaking it up into components.

Related Work

Since the pioneering work of Bellare and Rogaway [7, 8], many extensions and modifications have been made to the definition of secure key agreement [6, 5, 9, 15, 16, 26]. The model of security in which work is a slightly modified version of the model of Bellare *et al.* [6], although analogous versions of our results also hold in the models of [7, 9, 15].

Our technique also makes use of Gap assumptions, as defined by Okamoto and Pointcheval [25]. Informally, a Gap problem is the problem of solving some computational problem (e.g. computational Diffie-Hellman) with the help of a corresponding decisional oracle (in this case a decisional Diffie-Hellman oracle). The decisional problem may be easy or hard; irrespective of this a Gap problem may still be defined.

Gap assumptions have recently found several applications in cryptography. In particular, Gap assumptions have been used in [1, 14, 20] to prove the security of certain key agreement protocols.

In this paper, we show that, if a protocol satisfies some weakened notion of security and has a specific form, then using the Gap assumption, a full proof of security can be constructed. This result holds for protocols analyzed in the Bellare-Rogaway model [7] (or its extensions [6, 9]) or in the Canetti-Krawczyk model of SK-security [15].

2 Preliminaries

Following the notation of Okamoto and Pointcheval [25], we informally define a family of Gap problems.

Let $f : X \times Y \to \{0, 1\}$ be any relation on sets X and Y. The *computational* problem (or *inverting* problem in the language of [25]) of f is, given $x \in X$, to compute any $y \in Y$ such that $f(x, y) = 1$ if such a y exists, or to return `Fail` otherwise.

The *decisional* problem of f is, given $(x, y) \in X \times Y$, to decide whether $f(x, y) = 1$ or not.

Definition 1. *The Gap problem of f is to solve the computational problem of f using an oracle which solves the decisional problem of f.*

As an example, we define the computational, decisional and Gap Diffie-Hellman problems.

Let p and q be primes where $q|p-1$. Let G be a multiplicative subgroup of $\mathbb{Z}_p^*$, of order q, and let $g \in G$ generate G. We denote by $DL(g, h) \in \mathbb{Z}_q$ the discrete logarithm of $h \in G$ with respect to base g. So $g^{DL(g,h)} = h \bmod p$.

Given $a, b, c \in \mathbb{Z}_q$, we define the Diffie-Hellman relation f_{DH} as follows:

$$f_{DH} : (G \times G) \times G \to \{0, 1\}, \text{ where } f_{DH}(g^a, g^b, g^c) = \begin{cases} 1 & \text{if } g^{ab} = g^c \\ 0 & \text{otherwise} \end{cases}$$

We can now define the computational, decisional and Gap problems of f_{DH}, better known as the computational, decisional and Gap Diffie-Hellman problems.

Computational Diffie-Hellman (CDH) Problem: Given $g^a, g^b \in G$, where $a, b \in_R \mathbb{Z}_q$, compute $g^c \in G$, such that $f_{DH}(g^a, g^b, g^c) = 1$. That is, compute $g^c = g^{ab} \bmod p$.

Decisional Diffie-Hellman (DDH) Problem: Given $g^a, g^b, g^c \in G$, where $a, b \in_R \mathbb{Z}_q$, determine whether $f_{DH}(g^a, g^b, g^c) = 1$ or not. That is, determine whether $c = ab \bmod q$ or not.

Gap Diffie-Hellman (GDH) Problem: Given $g^a, g^b \in G$ where $a, b \in_R \mathbb{Z}_q$, as well as an oracle that solves the DDH problem on G, compute $g^{ab} \bmod p$.

The corresponding assumptions are that the above problems are *hard*, that is, they are infeasible to solve in polynomial time in a security parameter used to define the problem instances.

3 The modified Bellare-Rogaway model

We start by defining a modified Bellare-Rogaway (mBR) model for authenticated key agreement protocols. The model follows closely the model of Bellare *et al.* [6] which extends the original Bellare-Rogaway model [7]. However we present our model in the public key setting as in the model of Blake-Wilson *et al.* [9].

The model includes a set of participant IDs $\{U\}$, where each participant has a distinct ID U, a long-term public key P_U and a long-term private key S_U. We use Π_U^i to denote the oracle modelling the ith instance of participant U. An oracle Π_U^i may accept at any time, and once accepted it should hold a role $role \in \{initiator, responder\}$, a partner ID pid (the ID of the oracle with which it assumes it is communicating), a session ID sid and a session key sk. We note that the value i is not the sid but rather an internal session counter for each oracle. This may act as an internal identifier for the session until the sid is established.

Oracles follow the rules of the protocol, responding to input messages (from the adversary). Each oracle maintains a public transcript $T_{\Pi_U^i}$ which records all messages they have sent or received as a result of queries they have answered.

3.1 The mBR Game

The security of a key agreement protocol is modelled via the following game between a challenger C and an adversary E.

C runs a `Setup` algorithm on a security parameter k to create the public parameters, a set of participants $\{U\}$ and oracles Π_U^i to model instances of each participant U, and to distribute long-term keys to each participant. C also randomly selects a bit b.

The model also includes an adversary E who is given all the participants' public keys and has access to all the participants' oracles as well as any random oracles in the game. E can make the following queries:

`Send`(U, i, M): E can send the oracle Π_U^i a message M. If oracle Π_U^i has $pid = U'$, then Π_U^i assumes that M has come from U' and responds according to the protocol. E may also make a special `Send` query λ to an oracle Π_U^i which instructs U to initiate a protocol run with its partner U'. An oracle Π_U^i sets $role_U = initiator$ and is called an *initiator oracle* if the first message it has received is λ. If Π_U^i did not receive a message λ as its first message, then it sets $role_U = responder$ and is called a *responder oracle*.

`Reveal(`U, i`):` this allows E to ask the oracle Π_U^i to reveal the session key (if any) it currently holds to E.
`Corrupt(`U`):` this allows E to ask participant U to reveal its long-term private key.

Oracle States An oracle exists in one of the following possible states:

`Accepted:` an oracle has accepted if it decides to accept, holding a session key, after receipt of properly formulated messages.
`Rejected:` an oracle has rejected if it decides not to establish a session key and to abort the protocol.
`State *:` an oracle is in state * if it has not made any decision to accept or reject.
`Revealed:` an oracle is revealed if it has answered a reveal query.
`Corrupted:` an oracle is corrupted if it has answered a corrupt query.

Partners When running the protocol, if oracles Π_U^i holding (sk, sid, pid) and $\Pi_{U'}^j$ holding (sk', sid', pid') have both accepted and the following conditions hold:

1. $sid = sid'$, $sk = sk'$, $pid = U'$ and $pid' = U$,
2. $role_U = initiator$ and $role_{U'} = responder$ or vice versa,
3. No oracle in E's game besides Π_U^i or $\Pi_{U'}^j$ accepts with session ID equal to sid,

then Π_U^i and $\Pi_{U'}^j$ are said to be *partners*.

Freshness An oracle Π_U^i is called *unfresh* if it is revealed, or it has a revealed partner, or if its partner $\Pi_{U'}^j$ was corrupted. If an oracle is not unfresh, then the oracle is *fresh*.

Test query E may make a polynomial number of queries in k. Then at some point E makes a special `Test` query to an oracle Π_U^i. This oracle must be accepted and fresh, and it answers as follows. If $b = 0$, then Π_U^i randomly chooses a session key sk and outputs it, otherwise if $b = 1$ it outputs its own session key sk_U^i.

After this point E can continue querying the oracles except that E cannot reveal or corrupt the test oracle or its partner (if it exists). Finally E outputs a guess b' for b.

E's advantage, denoted $advantage^E(k)$, is the probability that E outputs a bit b' such that $b = b'$.

3.2 Definition of security

We define a *benign adversary* as in [7]. Informally, a benign adversary is one who simply relays messages between parties without modification. We then define secure authenticated key agreement (AKE) protocols as follows:

Definition 2. *A protocol is an mBR-secure AKE protocol if:*

1. *In the presence of the benign adversary, two oracles running the protocol both accept holding the same session key and session ID, and the session key is distributed uniformly at random on* $\{0,1\}^k$*; and*
2. *For any adversary* E*,* $Advantage^E(k)$ *is negligible.*

We say that protocol Π is *mBR-insecure* if it is not mBR-secure. That is, there exists an adversary E which, with non-negligible probability (in k), wins the game against challenger C. We say that such an adversary E can successfully *mBR-attack* protocol Π.

3.3 Notes on the Security Model

Our model of security is closely related to that of Bellare *et al.* [6]. However we do not explicitly distinguish between acceptance and termination as is done in [6], and we do not model perfect forward secrecy. Both of these properties can be added as in [6]. We omit them for simplicity of presentation, but our results still hold if these properties are included.

Notice that corruption in our model is simply a query to an oracle which reveals the long-term secret key held by the oracle. The adversary does not learn other internal state of the oracle and does not gain control of the oracle. Therefore a corrupted oracle may still be considered to be fresh and can therefore still be chosen as a Test oracle. This is important in order to model key compromise impersonation attacks as defined in [9], since these attacks involve oracles whose long-term private keys have been compromised but which are not under adversarial control.

The main differences between our model and the original model of Bellare and Rogaway [7] are that our model is adaptive (that is, the adversary may continue making queries after the `Test` query), it is in the public key setting (as in [9]), and we define partnering via session IDs and partner IDs (as in [6]) rather than by matching conversations. We also include the possibility for corrupted oracles to be considered fresh, allowing us to model key compromise impersonation attacks. As

mentioned before, our model can easily be extended to model perfect forward secrecy as well.

We direct the reader to [6] for further details of the model presented here and to [5–7, 9, 15] for details of other models illustrating different methods for dealing with partnering, corruptions and freshness.

4 Modular Construction of Security Proofs

From now on, we assume that we are only dealing with key agreement protocols that produce a *hashed session key* on completion of the protocol. By this we mean that the key agreement protocol Π specifies that the session key be computed as the hash H of some string which we call the *session string* ss_Π. We define the session string for a particular oracle $\Pi^i_{U'}$ to be $ss_{\Pi^i_U}$. We will model H as a random oracle in our security analysis.

This reliance on hashing to produce a session key does not seem to be too strong a restriction since it is fairly common to use a key derivation function to obtain a session key from a secret value established during a key agreement protocol, and this key derivation function is usually implemented via a hash function.

4.1 Protocol Partnering

When trying to establish that a protocol Π is secure in the BR-style model, we need to ensure that an adversary cannot trivially win the game defined in Section 3.1 by an attack on the partnering properties of Π.

Definition 3. *Suppose* Π *is a key agreement protocol. If there exists an adversary* E*, which when attacking* Π *in an mBR game defined in Section 3.1 and with non-negligible probability in the security parameter* k*, can make any two oracles* Π^i_U *and* $\Pi^j_{U'}$ *accept holding the same session key when they are not partners, then we say that* Π *has* weak partnering*. If* Π *does not have weak partnering, then we say that* Π *has* strong partnering*.*

If a protocol Π had weak partnering against an adversary E, then E could make oracles Π^i_U and $\Pi^j_{U'}$ accept holding the same session key but without being partners. The rules of the mBR game would then allow the adversary to reveal the session key held by Π^i_U, and then choose $\Pi^j_{U'}$ as the test session, allowing E to can trivially win the game.

Therefore, for Π to be a secure key agreement protocol as defined in Definition 2, Π must have strong partnering.

The observations above apply equally to our BR-style model as they do to the Canetti-Krawczyk model [15], even though the concept of partners are slightly different in the two models. In our security model, partnership is defined via session keys, session IDs and partner IDs. For oracles Π_U^i and $\Pi_{U'}^j$ to accept holding the same session key but without being partners, they must have different *sid*s and/or *pid*s. To ensure that the protocol Π has strong partnering, we must ensure that (with overwhelming probability) $sk_U^i = sk_{U'}^j$ only if $role_U^i \neq role_{U'}^j$, $sid_U^i = sid_{U'}^j$ and $pid_U^i = pid_{U'}^j$. This can be ensured by including $role_U^i$, sid_U^i and pid_U^i in the session string $ss_{\Pi_U^i}$ (and therefore in the computation of the session key sk_U^i).

This idea of including the "partnering information" in the session string ensures strong partnering in other models as well. For example, in the models of [7–9], partnering is defined via matching conversations, or transcripts. Therefore a key agreement protocol secure in these models can never allow two oracles to share the same key without having identical transcripts. Strong partnering in these models can therefore be ensured by including the protocol transcript in the session string of each oracle.

4.2 Reduced Games

We now consider two reduced mBR games. The first game is identical to the mBR game defined in Section 3.1 except that the adversary E is not allowed to make any `Reveal` queries. We call this reduce game a No-Reveals mBR (NR-mBR) game. The second game is identical to the NR-mBR game, except that the adversary no longer makes a `Test` query. Instead, to win the game, the adversary must select an accepted and fresh Test oracle at the end of its computation and output the session key for this oracle. Since the adversary in this game must actually compute the session key of an oracle (instead of having to decide between a session key and a random value from the key space), we call this game a computational NR-mBR (cNR-mBR) game. We define E's advantage, denoted $Advantage^E(k)$, in the cNR-mBR game to be the probability that E outputs a session key sk such that $sk = sk_{\Pi_U^i}$ where Π_U^i is the Test oracle selected by the adversary.

Definition 4. *A protocol Π is a (c)NR-mBR-secure key agreement protocol if:*

1. *In the presence of the benign adversary, two oracles running the protocol both accept holding the same session key and session ID, and the session key is distributed uniformly at random on $\{0,1\}^k$; and*

2. *For any adversary E, Advantage$^E(k)$ in the (c)NR-mBR game is negligible.*

We say that protocol Π is *(c)NR-mBR-insecure* if it is not (c)NR-mBR-secure. That is, there exists an adversary E which, with non-negligible probability (in k), wins the (c)NR-mBR game against challenger C. We say that such an E can successfully *(c)NR-mBR-attack* protocol Π.

As part of our proof process for a given protocol Π which produces hashed session keys on completion of the protocol, we will consider a related protocol π. Protocol π is defined in the same way as Π except that the session key generated by π will be the session string of Π. That is, $sk_{\pi_U^i} = ss_{\Pi_U^i}$. It will then be necessary to prove that protocol π is cNR-mBR secure. Since the cNR-mBR game is a highly reduced game, it is usually fairly easy to establish a protocol's security in this model. Although it may not be obvious how a proof of security in this reduced model may be helpful, in Section 4.3 we present a theorem which shows how a proof of cNR-mBR security for π can be transformed into a proof of mBR security for Π using a Gap assumption, provided that Π has strong partnering.

The reason we defined NR-mBR security when cNR-mBR security is our focus is that, although it is a more complex game than the cNR-mBR game, a number of recent papers presenting new key agreement protocols prove that their protocols meet such a weakened definition of security [2, 9, 17, 3]. That is, they take an appropriate security model, and prove the security of their protocols in the No-Reveals (NR) variant of the security model.

It is trivial to see that if protocol Π is NR-mBR secure, then it is also cNR-mBR secure. We also have the following result relating the NR-mBR security of Π and the cNR-mBR security of the related protocol π.

Theorem 1. *If a protocol Π produces a hashed session key via hash function H and is NR-mBR secure, then the related protocol π is cNR-mBR secure.*

A sketch of the proof of this theorem is in Appendix A. We note that in the proof of the above theorem, no assumption is required concerning the properties of H.

4.3 Handling Reveal Queries using Gap Assumptions

We now consider a protocol Π which has strong partnering and for which the related protocol π is cNR-mBR secure. In order to translate these

results into a proof of mBR security for Π, we need to be able to construct a challenger C in an mBR game for Π which is able to answer an adversary E's `Reveal` queries.

At first glance, it seems that C needs to be able to compute the session key sk_U for any oracle Π_U^i that E may wish to reveal during the mBR game. However this is not the case if Π produces a hashed session key (via hash function H) and if H is modelled as a random oracle. We will see below in Theorem 2 that in this case, C only needs to be able to solve the following decisional problem:

> Given the transcript T_U^i of oracle Π_U^i in an mBR game, as well as the P_U and $P_{U'}$ (the public keys of U and U' where $pid_U^i = U'$) and s, where s is a string, decide whether $s = ss_{\Pi_U^i}$, where $ss_{\Pi_U^i}$ is the session string of oracle Π_U^i.

We call this decisional problem the *session string decisional problem for protocol* Π.

We now present our main result.

Theorem 2. *Suppose that key agreement protocol* Π *produces a hashed session key on completion of the protocol (via hash function* H*) and that* Π *has strong partnering. If the cNR-mBR security of the related protocol* π *is probabilistic polynomial time reducible to the hardness of the computational problem of some relation* f*, and the session string decisional problem for* Π *is polynomial time reducible to the decisional problem of* f*, then the mBR security of* Π *is probabilistic polynomial time reducible to the hardness of the Gap problem of* f*, assuming that* H *is a random oracle.*

Proof. Since the cNR-mBR security of π is probabilistic polynomial time reducible (in security parameter k) to the hardness of the computational problem of some relation f, there exists an algorithm A that, on input a problem instance of the computational problem of f and interacting with an adversary E which has non-negligible probability η of winning the cNR-mBR game for π in time τ, is able to solve the computational problem of f with some non-negligible probability $g(\eta)$ and in time $h(\tau)$, where g and h are polynomial functions.

We now define an algorithm B which, given an adversary D which has non-negligible probability η' of winning the mBR game for Π in time τ', is able to solve the Gap problem of f with some non-negligible probability $g'(\eta')$ and in time $h'(\tau')$ where g' and h' are polynomial functions. B will

act as a challenger for D. B will also run algorithm A and will simulate an adversary for A. Since B attempts to solve the Gap problem of f, B will also have access to a decisional oracle for f.

Since Π has strong partnering, we know that if two oracles share the same session key, then they must be partners (with overwhelming probability). We therefore know that D will never reveal a session key sk where sk is equal to the Test oracle Π_T^i's session key $sk_{\Pi_T^i}$. This is because D is not permitted to reveal the session key of the Test oracle or its partner (if it exists).

We also assumed that the session string decisional problem for Π is polynomial time reducible to the decisional problem of f. That is, there exists some algorithm C which, given a decisional oracle for f, is able to solve the session string decisional problem for Π in polynomial time τ''.

B runs A on the problem instance of the computational problem of f and simulates an adversary for A. A sets up a cNR-mBR game for B and gives all the public parameters to B. B in turn passes these public parameters to adversary D. B now answers all of D's queries as follows.

B passes all D's queries besides `Reveal` and H queries to A. Since, in any session, protocol π is identical to protocol Π until the session is completed and the session key is computed, these queries will all be answerable by A. B passes A's responses back to D.

In order for B to answer D's `Reveal` queries, B maintains a Guess session key list (G-List). Each element on the G-List is a tuple of the form $(T_V^j, P_V, P_{V'}, R_V^j)$ where T_V^j is the transcript of oracle Π_V^j, P_V is the public key of V, $P_{V'}$ is the public key of V' where $pid_{\Pi_V^j} = V'$, and R_V^j is a random guess for the session key sk_V^j of oracle Π_V^j. Initially the G-List is empty.

In order for B to answer E's H queries, B maintains an (initially empty) H-List containing tuples of the form (s_i, sk_i, str). For each H query on string s that D makes, B checks whether s is on the H-List as the first component in some tuple (s_i, sk_i, str). If it is, then B outputs sk_i. If s is not on the H-List then B uses the algorithm C to determine whether s is a valid session string for any oracle Π_V^j on the G-List. If $s = ss_{\Pi_V^j}$ is the session string for some oracle Π_V^j on the G-List, then B outputs R_V^j and adds the tuple (s, R_V^j, str) where str="V,j" to the H-List. Otherwise B selects a random sk from the session key space, adds the tuple (s, sk, str) (where str is the empty string λ) to the H-List, and outputs sk.

When D makes a `Reveal` query on any oracle Π_U^i which has accepted, B proceeds as follows. If Π_U^i has an entry on the G-List of the form $(T_U^i, P_U, P_{U'}, R_U^i)$, B outputs the value R_U^i. Otherwise B checks whether any entry on the H-List of the form (s_i, sk_i, str) where $str = \lambda$ has $s_i = ss_{\Pi_U^i}$ using algorithm C. If such an entry (s_i, sk_i, str) exists, then str is set to "U,i" on the H-List and the entry $(T_U^i, P_U, P_{U'}, R_U^i)$ is added to the G-List, where $R_U^i = s_i$, T_U^i is the transcript of Π_U^i, P_U is the public key of U and $P_{U'}$ is the public key of U' where $pid_{\Pi_U^i} = U'$. Otherwise a random session key R_U^i is selected and the entry $(T_U^i, P_U, P_{U'}, R_U^i)$ is added to the G-List. To answer the `Reveal` query, B outputs the value R_U^i in every case.

In this way, B can consistently answer D's `Reveal` and H queries. At some point D selects a Test oracle Π_T^i. B selects a random element sk from the session key space and gives this to D.

If D does not query H on the Test oracle's session string $ss_{\Pi_T^i}$, then D can only win with probability $1/S_H$ where S_H is the size of the output space of H, which we assume is negligible in security parameter k. So with overwhelming probability $1 - 1/S_H$, D queries H on $ss_{\Pi_T^i}$. B can detect this value by checking which of the tuples (s_i, sk_i, str) on the H-List with $str = \lambda$ has $s_i = ss_{\Pi_T^i}$ using algorithm C. B gives this s_i to A.

Since $ss_{\Pi_T^i} = sk_{\pi_T^i}$, B has simulated a valid adversary E for A with non-negligible probability $\eta = \eta' \cdot (1 - 1/S_H)$ and in polynomial time $\tau = \tau' + \tau'' \cdot N_H \cdot (N_R + 1)$, where N_H and N_R are the number of H and `Reveal` queries that D makes respectively. So A outputs the solution to the instance of the computational problem of f with non-negligible probability $g(\eta)$ and in time $h(\tau)$.

Therefore B solves the Gap problem of f with non-negligible probability $g(\eta)$ and in time $h(\tau)$.

□

4.4 Different Security Models

Analogous results to Theorem 2 can be obtained for the security models of [6–9, 15].

For each of these models, an equivalent definition of strong partnering can be made. In the models of [7–9] partnering is defined via the concept of matching conversations, so strong partnering would be defined in this context as well.

For each of these models, NR and cNR versions can be defined in the same way as for our mBR model. The definition of the related protocol π is independent of the model used.

It is then possible to prove analogous versions of Theorem 2 for these models. These in turn illustrate how proofs in these models can be constructed in a modular way.

We notice that analogous versions of Theorem 1 for alternative security models are also easy to formulate and prove.

Further details will be provided in the full paper.

5 Applying the Technique to Existing Protocols

We are now able to apply our results to key agreement protocols in the literature. We find numerous protocols [2, 3, 9, 17] which use a hash function to derive a session key and which have proofs of security reducing to some computational assumption but only in the NR version of the security model used[1]. For each such protocol Π, full proofs of security in the relevant model can be obtained as follows.

1. It must be shown that the protocol Π has strong partnering. If Π does not have strong partnering, this can be achieved by modifying the protocol to include the appropriate partnering information (for the security model used) in the session string. It should be checked that such modifications do not affect the existing proof of security.
2. The appropriate version of Theorem 1 can now be applied to Π to guarantee that the related protocol π is secure in the cNR version of the security model used.
3. It must be shown that the appropriate decisional oracle can be used to solve the session string decisional problem of Π. In general this is a trivial reduction.
4. The appropriate version of Theorem 2 may now be used to obtain a complete security proof for Π in the full version of the security model used.

For example, the proof of security for Protocol 3 of [9] can be completed in the manner described above, although the protocol does require some modifications to achieve strong partnering. A suitably modified version of this protocol is in fact presented in [21] together with a proof of

[1] A proof for the protocol of [17] appearing in [10] allows the adversary to make some but not all Reveal queries

security. Interestingly, Protocol 3 of [9] and the modified version in [21] are vulnerable to a key compromise impersonation attack. However this does not affect the proof of security since the model of [9] does not capture security against these attacks.

5.1 A Concrete Example

We now consider Protocol 4 in [9], which was conjectured to be secure in [9] but has never been proven secure. We modify the protocol slightly to guarantee strong partnering and then prove this protocol secure in our mBR model. It is possible to prove the unmodified protocol secure in the model of [9] using the method described above, but the proof of strong partnering is more complicated.

We now present our modified version of Protocol 4 of [9].

Protocol 1

The *Setup* algorithm generates primes p and q where $q|p-1$. It then chooses G to be a multiplicative subgroup of $\mathbb{Z}_p^*$, where G has order q, and element $g \in G$ generates G. It also sets the session ID space $S = G^4$ and selects a hash function $H : G^2 \times S \rightarrow \{0,1\}^k$. Each participant I selects a private key x_I randomly from Z_q and sets their public key to be $X_I = g^{x_I} \bmod p$.

Suppose that A and B are participants with public keys $X_A = g^{x_A} \bmod p$ and $X_B = g^{x_B} \bmod p$ respectively. A and B run the protocol as follows:

A, as initiator will receive some input $(X_B, initiator)$ and initiates session Π_A^i, setting $pid_A = X_B$ and $role_A = initiator$.

A randomly picks a value $a \in \mathbb{Z}_q$, computes $T_A = g^a \bmod p$ and sends the following to B:

$$A \rightarrow B : T_A, X_A, X_B.$$

On receipt of the message from A, B initiates session Π_B^j with $pid_B = X_A$ and $role_B = responder$. B randomly picks a value $b \in \mathbb{Z}_q$ and computes $T_B = g^b \bmod p$. B then sends the following to A:

$$B \rightarrow A : T_B, T_A, X_B, X_A.$$

B computes $sid_B = X_A, X_B, T_A, T_B$ and $K_B = H(T_A^{x_B} \bmod p, X_A^b \bmod p, sid_B)$ and accepts with session key $sk_B = K_B$.

On receipt of the message from B, A computes $sid_A = X_A, X_B, T_A, T_B$ and $K_A = H(X_B^a \bmod p, T_B^{x_A} \bmod p, sid_A)$ and accepts with session key $sk_A = K_A$.

If the protocol completes correctly, it is easy to see that $K_A = K_B$.

The modified version of Protocol 1 in which the session key is equal to the session string of Protocol 1 is denoted by Protocol $1'$.

Theorem 3. *The cNR-mBR security of Protocol $1'$ is probabilistic polynomial time reducible to the hardness of the CDH problem in G.*

This is proved in Appendix A. It is interesting to note how short the proof of this theorem is; this is due to the simplicity of the cNR-mBR model.

We note that a common error when proving that a protocol Π is mBR-secure (or even NR-mBR or cNR-mBR secure) is to make the assumption that the Test oracle Π_U^i has a partner, and that the input to Π_U^i comes from this partner. In fact the challenger has no control over the input to Π_U^i since the adversary controls all communications between oracles. This error can be seen in papers such as [4, 18] where proofs of security were attempted in the full security model. Their corrected versions [3, 17] provide proofs in the NR versions of the original models.

Theorem 4. *Protocol 1 has strong partnering in the random oracle model.*

The simple proof of this theorem is left to the reader.

Corollary 1. *Protocol 1 is secure in the random oracle model assuming the hardness of the Gap Diffie-Hellman problem.*

Proof. This result comes immediately from Theorems 2, 3 and 4 and the observation that a decisional Diffie-Hellman oracle can be used to solve the session string decisional problem for Protocol 1. Therefore the session string decisional problem for Protocol 1 is reducible to the decisional Diffie-Hellman problem (in constant time). □

We note that Protocol 1 can easily be extended to have perfect forward security by including the value $T_A^b \bmod p$ into the computation of the hash function H. This extended Protocol 1 can then be proven secure in an extended mBR model which models perfect forward secrecy.

The protocol of [27], after slight modifications to ensure strong partnering, can also be proven secure in the random oracle model in a similar way to our Protocol 1.

6 Special Gap Groups

The Gap assumptions may not be acceptable to all, since in developing security proofs, one must assume the use of an oracle which is not known

to exist: a decisional oracle. For instance, for Protocol 1, the proof of security ultimately requires an oracle which solves DDH in the group G. This is thought to be a hard problem, so there is no known method of constructing such an efficient oracle.

However there do exist groups in which the computational problem is thought to be hard but where the decisional problem is known to be easy. For instance, groups of points on an elliptic curve on which an efficient bilinear map (or pairing operation) is defined. In such groups, the pairing operation can be used to construct an efficient DDH oracle, and the Gap problem is in fact equivalent to the computational problem. Therefore if Protocol 1 had been defined over such a group, then its security would in fact reduce to the CDH problem.

7 Conclusions

We have presented a modular technique that makes use of Gap assumptions for simplifying proofs of security for key agreement protocols which are not built using the modular approach of [5]. Protocols of this type have traditionally been notoriously hard to prove secure, and we have indicated how the proofs of security of many such protocols in the literature may be constructed or completed using our technique. Our technique works not only with the model presented in this paper, but also with the models of [7–9, 15].

We considered in detail a long-standing protocol presented in [9] which previously lacked a proof of security. We then provided a full proof of security for a slightly modified version of this protocol using the techniques introduced in this paper. We also indicated how full proofs of security for protocols in [2, 3, 9, 17, 27] may be constructed using our techniques.

References

1. M. Abdalla, O. Chevassut, and D. Pointcheval. One-time verifier-based encrypted key exchange. In S. Vaudenay, editor, *Public Key Cryptography - PKC 2005*, volume 3386 of *LNCS*, pages 47–64. Springer-Verlag, 2005.
2. S.S. Al-Riyami and K.G. Paterson. Authenticated three party key agreement protocols from pairings. In K.G. Paterson, editor, *Proceedings of 9th IMA International Conference on Cryptography and Coding*, volume 2898 of *Lecture Notes in Computer Science*, pages 332–359. Springer-Verlag, 2003.
3. P.S.L.M. Barreto and N. McCullagh. A new two-party identity-based authenticated key agreement. Cryptology ePrint Archive, Report 2004/122, 2005. http://eprint.iacr.org/.

4. P.S.L.M. Barreto and N. McCullagh. A new two-party identity-based authenticated key agreement. In *Topics in Cryptology – CT-RSA'2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 262–274. Springer-Verlag, 2005.
5. M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. In *Proceedings of the 30th Annual Symposium on the Theory of Computing*, pages 419–428. ACM, 1998.
6. M. Bellare, D. Pointcheval, and P. 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.
7. M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology - CRYPTO '93*, volume 773 of *LNCS*, pages 232–249. Springer-Verlag, 1994.
8. M. Bellare and P. Rogaway. Provably secure session key distribution: The three party case. In *Proceedings of the 27th Annual ACM Symposium on Theory of Computing STOC*, pages 57–66. ACM, 1995.
9. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In *Cryptography and Coding*, volume 1355 of *LNCS*, pages 30–45. Springer-Verlag, 1997.
10. C. Boyd, K.R. Choo, and Y. Hitchcock. On session key construction in provably-secure key establishment protocols. To appear, Mycrypt 2005. http://eprint.iacr.org/2002/184.
11. C. Boyd, W. Mao, and K. Paterson. Key agreement using statically keyed authenticators. In *Applied Cryptography and Network Security: Second International Conference, ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 388–401. Springer-Verlag, 2004.
12. C. Boyd, J.M. González Nieto, and Y. Hitchcock. Tripartite key exchange in the Canetti-Krawczyk proof model. In *Proceedings of 5th International Conference on Cryptology in India INDOCRYPT 2004*, volume 3348 of *Lecture Notes in Computer Science*, pages 388–401. Springer-Verlag, 2004.
13. C. Boyd, J.M. González Nieto, Y. Hitchcock, P. Montague, and Y.S.T. Tin. A password-based authenticator: Security proof and applications. In *Proceedings of 4th International Conference on Cryptology in India INDOCRYPT 2003*, volume 2904 of *Lecture Notes in Computer Science*, pages 388–401. Springer-Verlag, 2003.
14. C. Boyd, J.M. González Nieto, and Y.S.T. Tin. Provably secure mobile key exchange: Applying the Canetti-Krawczyk approach. In *Information Security and Privacy, 8th Australasian Conference, ACISP 2003*, volume 2727 of *Lecture Notes in Computer Science*, pages 166–179. Springer-Verlag, 2003.
15. 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.
16. R. Canetti and H. Krawczyk. Universally composable notions of key exchange and secure channels. In L.R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag, 2002.
17. L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. Cryptology ePrint Archive, Report 2002/184, 2002. http://eprint.iacr.org/.
18. L. Chen and C. Kudla. Identity based authenticated key agreement from pairings. In *IEEE Computer Security Foundations Workshop – CSFW-16 2003*, pages 219–233. IEEE Computer Society Press, 2003.

19. W. Diffie, P. C. van Oorschot, and M. J. Weiner. Authentication and authenticated key exchange. *Designs, Codes and Cryptography*, 2:107–125, 1992.
20. M. Jakobsson and D. Pointcheval. Mutual authentication and key exchange protocol for low power devices. In *Financial Cryptography, 5th International Conference, FC 2001*, volume 2339 of *Lecture Notes in Computer Science*, page 178195. Springer-Verlag, 2002.
21. I.R. Jeong, J. Katz, and D.H. Lee. One-round protocols for two-party authenticated key exchange. In *Applied Cryptography and Network Security: the Second International Conference, ACNS 2004*, volume 3089 of *Lecture Notes in Computer Science*, pages 220 – 232. Springer-Verlag, 2004.
22. J. Katz and M. Yung. Scalable protocols for authenticated group key exchange. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *LNCS*, pages 110–125. Springer-Verlag, 2003.
23. L. Law, A. Menezes, M. Qu, J. Solinas, and S.A. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28(2):119–134, 2003.
24. T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key-distribution systems. *Electronics Letters*, E69(2):99–106, 1986.
25. 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 2001*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, 2001.
26. V. Shoup. On formal models for secure key exchange. IBM Technical Report RZ 3120, 1999. http://shoup.net/papers.
27. N.P. Smart. An identity based authenticated key agreement protocol based on the Weil pairing. *Electronics Letters*, 38(13):630–632, 2002.

Appendix A

Proof of Theorem 1. We provide a sketch of the proof of this theorem. The details are left to the reader. We show that if there exists an adversary E that can cNR-mBR attack π, then we can build an adversary A that can NR-mBR attack Π.

Suppose that an adversary E wins the cNR-mBR game for protocol π with non-negligible probability η. Suppose also that A runs an NR-mBR game with challenger C. A in turn acts as a challenger for E in a cNR-mBR game. A passes E's queries to C and returns C's outputs to E. Finally E will output the session key $sk_{\pi_U^i}$ of some fresh oracle π_U^i. Recall however that $sk_{\pi_U^i} = ss_{\Pi_U^i}$.

A then chooses Π_U^i as the Test oracle and receives a challenge key sk. If $sk = H(sk_{\pi_U^i})$ then A outputs 1, otherwise it outputs 0. A wins the NR-mBR game with probability η. □

Proof of Theorem 3. We assume that for security parameter k there exists an adversary E for Protocol $1'$ who can win the cNR-mBR game with advantage η which is non-negligible in k and in polynomial time τ of

k. Suppose that the number of participants is n_P and the number of sessions each participant may be involved in is n_S, where n_P and n_S are polynomial functions of k.

We now construct from E algorithm F which solves the CDH problem in G with non-negligible probability. That is, given as input elements $g^x, g^y \in G$, F's task is to compute and output the value $g^{xy} \bmod p$.

F simulates a challenger in a cNR-mBR game with E. F sets up the game with the group G and generator $g \in G$. F generates a set of participants of size n_P. For each participant I, F sets I's private key to be a randomly chosen $x_I \in \mathbb{Z}_q$ and sets their public key to be $X_I = g^{x_I} \bmod p$. However for some participant P, F sets P's public key to be $X_P = g^x$. F also picks a random participant $Q \neq P$, a session number $t \in \{1, .., n_S\}$ and a number $l \in \{1, .., n_H\}$. F starts E and answers E's queries as follows.

`Send:` E may make a special `Send` query Π_I^s which sets $pid_I = X_{I'}$ and instructs I to initiate a protocol run with its partner I'. E can also send any oracle Π_I^s a message M, and the oracle responds according to the protocol. However if E initializes or sends a message to oracle Π_Q^t, then Π_Q^t outputs g^y.

`Corrupt`(U): If E corrupts participant P, then F aborts. Otherwise F gives E the long-term private key of the participant.

`Test:` When E asks a `Test` query to an oracle Π_I^s, F outputs a random element from the key space $G^2 \times S = G^6$.

The probability that E queries oracle Π_Q^t for the Test session and that $pid_Q = X_P$ is $\frac{1}{n_P^2.n_S}$. In this case, we note that E could not have corrupted participant P, and so F would not have aborted.

E finally outputs a session key of the form (a, b, c) where $a, b \in G$ and $c \in G^4$. If Π_I^s was an initiator, then F outputs b as its guess for the value $g^{xy} \bmod p$, otherwise F outputs a as its guess. It is now easy to see that F solves the CDH problem on input (g^x, g^y) with probability $\eta' = \eta.(\frac{1}{n_P^2.n_S})$ which is non-negligible in k, and in time τ. □