An abridged version of this paper appears in *Advances in Cryptology – Crypto '93 Proceedings*, Springer-Verlag. This is the full version.

Entity Authentication and Key Distribution

Mihir Bellare* Phillip Rogaway†

August 1993

Abstract

Entity authentication and key distribution are central cryptographic problems in distributed computing—but up until now, they have lacked even a meaningful definition. One consequence is that incorrect and inefficient protocols have proliferated. This paper provides the first treatment of these problems in the complexity-theoretic framework of modern cryptography. Addressed in detail are two problems of the symmetric, two-party setting: mutual authentication and authenticated key exchange. For each we present a definition, protocol, and proof that the protocol meets its goal, assuming the (minimal) assumption of pseudorandom function. When this assumption is appropriately instantiated, the protocols given are practical and efficient.

Key words: authentication, computer security, cryptographic protocols, cryptography, distributed systems, key distribution.

*Department of Computer Science & Engineering, Mail Code 0114, University of California at San Diego, 9500 Gilman Drive, La Jolla, CA 92093. E-mail: mihir@cs.ucsd.edu

† Department of Computer Science, University of California at Davis, Davis, CA 95616, USA. E-mail: rogaway@cs.davis.edu

1 Introduction

1.1 Context

For centuries, cryptographic protocols were designed by trial and error. A scheme was proposed, and people tried to break it. If they didn't succeed then, after a while, the scheme was assumed to be adequate. History has shown that the success rate of this method is not too impressive: proposed protocols were often broken, sometimes years after they were first put forward.

The early 1980s saw the proposal of a fundamental and radical idea to get beyond iterative, attack-responsive design of cryptographic protocols: Goldwasser and Micali [21], followed by Blum-Micali [7] and Yao [39], suggested that security could be *proved* under "standard" and well-believed complexity theoretic assumptions (e.g., the assumed intractability of factoring). The methodology which they identified has come to be known as "provable security." Achieving it for a given problem of interest entails providing (i) a definition of the goal, (ii) a protocol, and (iii) a proof that the protocol meets its goal, assuming some standard complexity-theoretic assumption holds true.

By 1985 provable security had been achieved for probabilistic encryption [21], pseudorandom number generation [7, 39] and digital signatures [23]. It was then the opinion of many researchers that provable security was in hand for all of the "basic" cryptographic primitives. Attention turned to other issues, such as reducing the complexity assumptions needed to achieve provable security or increasing the efficiency of the constructions.

As it happens, the belief that provable security had been achieved for all of cryptography's basic primitives was wrong; there remained a crucial set of unformalized goals which were well-known to applied cryptographers and those who had tried to build secure distributed system. These goals involve *entity authentication* and associated problems *key distribution.* We now turn towards describing these objectives.

1.2 Informal problem statement

Entity authentication is the process by which an agent in a distributed system gains confidence in the identity of a communication partner. More often than not, the entity authentication process is coupled with the distribution of a "session key" which the partners can later use for message confidentiality, integrity, or whatever else. These are central problems in computing practice, for without their resolution distributed computing cannot realistically get off the ground. This importance is reflected in the enormous amount of attention that these problems have received in the literature; literally hundreds of papers have been written and protocols proposed and implemented.

Yet entity authentication for the distributed environment rests on no satisfactory formal foundations. This is more than an academic complaint. We are speaking of an area in which an informal approach has often lead to work which is at worst wrong, and at best only partially analyzable. In particular, an alarming fraction of proposed authentication protocols have subsequently been found to be flawed. It is therefore desirable that confidence in an authentication protocol should stem from more than a few people's inability to break it. In fact, in the tradition of provable security discussed above, each significant entity authentication goal should be formally defined and any candidate protocol should be proven to meet its goal under a standard cryptographic assumption. Of course the definition must properly model the real-world characteristics of the problem at hand, the protocols must be practical, and the proofs must be "meaningful" for practice.

Problems in authentication and authenticated key distribution come in various flavors: there may be two parties involved or more; the authentication may be unilateral or mutual; parties might (the symmetric case) or might not (the asymmetric case) share a secret key.

This paper focuses on two versions of the the two-party, mutual, symmetric case. In the *mutual authentication* problem the parties engage in a conversation in which each gains confidence that it is the other with whom he speaks. In the *authenticated key exchange* problem the parties also want to distribute a "fresh" and "secret" *session key.*[1] Despite their significance and long histories, both problems lack any modern, complexity-theoretic treatment. In particular, the primitives and tools formalized and understood in the theoretical community today (e.g. encryption, signatures, zero-knowledge [22], proofs of knowledge, identification) don't seem adequate to treat these problems.

1.3 Provable security for entity authentication

We provide entity authentication and key distribution with provable security, raising these goals to the same level as primitives such as encryption, pseudorandom generators, or digital signatures. In particular, we offer protocols whose security can be guaranteed from weak complexity-theoretic assumptions, bringing assurance to an area which has been fraught with uncertainty.

The definitional ideas needed to treat these problems are novel. Notions like indistinguishability [21, 22] and simulatability [22] which were so successful in formalizing other cryptographic primitives are not enough for this setting. In fact, we have to begin by re-defining the very *model* which underlies the communication.

MODEL. It has been pointed out in many places that one difficulty in laying foundations for entity authentication has been the lack of an appropriate *model* for authentication in the distributed environment. We specify an appropriate model. To be fully general here, we assume that all communication among interacting parties is under the adversary's control. In particular, the adversary can read the messages produced by the parties, provide messages of her own to them, modify messages before they reach their destination, and delay messages or replay them. Most importantly, the adversary can start up entirely new "instances" of any of the parties, modeling the ability of communicating agents to simultaneously engage in many *sessions* at once; this gives us the ability to model the kinds of attacks that were suggested by [6]. Formally, each party will be modeled by an infinite collection of oracles which the adversary may run. These oracles only interact with the adversary, they never directly interact with one another. See Section 3.

Note how this differs from the models underlying notions such as interactive proofs [22] or secure function evaluation [38, 20]. In the former case the communication is trusted and it is one of the parties who may be adversarial; in the later case, individual parties may be good or bad, but their communication proceeds in a simple and orderly manner. In neither case is there an analogue to the notion of sessions.

DEFINITIONS. In the presence of an adversary as powerful as the one we define, it is unclear what it could possibly mean to be convinced that one has engaged in a conversation with a specified

[1] At first glance it might seem unnecessary for two parties who already share a key a to come up with another key α. One reason a new key is useful is the necessity of avoiding cross-session "replay attacks" —messages copied from one session being deemed authentic in another— coupled with an insistence on *not* attempting to carry "state" information (e.g., a message counter) across distinct sessions.

partner; after all, every bit communicated has really been communicated to the the adversary, instead. We deal with this problem as follows.

As has often been observed, an adversary in our setting can always make the parties accept by faithfully relaying messages among the communication partners. But this behavior does not constitute a damaging attack; indeed, the adversary has functioned just like a wire, and may as well not have been there. The idea of our definition of a mutual authentication is then simple but strong: we say that a protocol is secure if the *only* way that an adversary can get a party to accept is by faithfully relaying messages in this manner. In other words, any adversary effectively behaves as a trusted wire, if not a broken one. Formalizing this simple idea is not so simple; the main tool will be a notion of *matching conversations*.

To define authenticated key exchange it is necessary to capture a protocol's robustness against the loss of a session key: even if the adversary gets hold of one, this should not compromise anything but the session which that key protects. We model this requirement by allowing the adversary to obtain session keys just by asking for them. When this inquiry is made, the key is no longer *fresh*, and any partner's key is declared unfresh, too. Fresh keys must remain "protected." We formalize the adversary's inability to gain any helpful information about them along the lines of formalizations of security for probabilistic encryption [21, 16, 17].

PROTOCOLS. Having defined mutual authentication and authenticated key exchange, we provide protocols to achieve these ends. Four protocols are specifically discussed in this paper. Protocol MAP1, which is an extension of the protocol 2PP of [6], is a mutual authentication protocol for an arbitrary set I of players. Protocol MAP2 is an extension of MAP1, allowing arbitrary text strings to be authenticated along with its flows. Protocol AKEP1 is a simple authenticated key exchange which uses MAP2 to do the key distribution. Protocol AKEP2 is a particularly efficient authenticated key exchange which introduces the idea of "implicitly" distributing a key; its flows are identical to MAP1, but it accomplishes a key distribution all the same. The primitive required for all of these protocols is a pseudorandom function.

Our protocols are *simpler* than previous ones. Our ability to attain provable security while simultaneously simplifying the solutions illustrates an advantage of having a clear definition of what one is trying to achieve; lacking such a definition, previous solutions encumbered their protocols with unnecessary features.

PROOFS. Assuming that pseudorandom functions exist, each protocol that we give is proven to meet the definition for the task which this protocol is claimed to carry out. The proofs for MAP1 and AKEP1 are given in this paper; the proofs for MAP2 and AKEP2 are omitted because they are essentially identical.

1.4 Design for practice

Provably secure protocols are not usually efficient. Ours are an exception to this rule. Every protocol presented in this paper is efficient in terms of rounds, communication, and computation. This efficiency was *designed* into our protocols in part through the choice of the underlying primitive—namely, a pseudorandom function.

In theory, pseudorandom functions and other important cryptographic primitives (one-way functions, pseudorandom generators, digital signatures) are equivalent [26, 24, 18, 33], since the

existence of any one of these implies the existence of the others.[2] In practice, pseudorandom functions (with the right domain and range) are a highly desirable starting point for efficient protocols in the symmetric setting. The reason is that beginning with primitives like DES and MD5 one can construct efficient pseudorandom functions with arbitrary domain and range lengths, and these constructions are themselves provably secure given plausible assumptions about DES and MD5. See Section 6 for discussion of these issues.

Our proofs are not so bad as to render the reductions meaningless for cryptographic practice. In other words, if one had a practical method to defeat the entity authentication, this would translate into a practical method to defeat the underlying pseudorandom function.

1.5 History and related work

Unsatisfactory protocols for entity authentication and key distribution have been leading researchers steadily towards establishing firm foundations. An important step in the process was that of Bird, Gopal, Herzberg, Janson, Kutten, Molva and Yung [6]. They drew attention to this area by pointing to new classes of attacks, called "interleaving attacks," which they used to break existing protocols, and they suggested a protocol (2PP) defeated by none of the interleaving attacks they considered. The recognition of interleaving attacks helped lead us to the formal model of Section 3, and our MAP1 protocol is an extension of 2PP. However, while an analysis such as theirs is useful as a way to spot errors in a protocol, resistance to interleaving attacks does not make a satisfactory notion of security; in particular, it is easy to construct protocols which are insecure but defeated by no attack from the enumeration. When our work was announced, the authors of [6] told us that they understood this limitation and had themselves been planning to work on general definitions; they also told us that the CBC assumption of their paper [6, Definition 2.1] was intended for proving security under a general definition.

Mentioned in the introduction of [6] is an idea of "matching histories." Diffie, van Oorschot and Wiener [11] expand on this to introduce a notion of "matching protocol runs." They refine this idea to a level of precision adequate to help them separate out what are and what are not "meaningful" attacks on the protocols they consider. Although [11] stops short of providing any formal definition or proof, the basic notion these authors describe is the same as ours and is the basis of a definition of entity authentication. Thus there is a clear refinement of definitional ideas first from [6] to [11], and then from [11] to our work.

The failure of the informal approach to designing correct authentication protocols has led to widespread recognition of the need for better foundations. By now the continued absence of a formal definition is a recognized deficiency. See, for example, [11] and [5, p. 59].

A different line of work aimed at improving the design and analysis of entity authentication protocols begins with the paper of Burrows, Abadi and Needham [8]. This "logic-based" approach attempts to "reason" that an authentication protocol is correct as it evolves the set of "beliefs" of its participants. This idea is useful and appealing, but it has a serious defect: a correctness proof does not guarantee that the protocol is right, only that it lacks the flaws in reasoning captured by the underlying logic. Thus while a negative result implies that something is wrong, a positive

[2] We remark that the existence of a secure mutual authentication protocol implies the existence of a one-way function, as can be shown using techniques of [25]; thus mutual authentication also exists if and only if one-way functions do.

result gives no assurance that everything is all right.

The notion of a zero-knowledge "proof of knowledge" [22, 13, 35, 9, 14, 3] has underlied identification protocols in the smart card model. But the definition of an interactive proof does not attempt to model attacks in which responses of entities are played off against one another, as is required for the distributed setting. Furthermore, (unilateral) authentication is not "proving knowledge" of a secret insofar as it is is fundamentally irrelevant that an agent A "knows" a in the sense that it can be extracted by a simulator: all that is important is that the good party can prove his identity and a bad party can't.

More closely related to the approach we adopt is the idea of a *non-transferable proof,* a notion for (asymmetric, unilateral) authentication due to Feige, Fiat and Shamir [13]. Here an (honest) claimant P interacts with a (cheating) verifier $\tilde{V}$, and then a ($\tilde{V}$-conspiring cheating) prover $\tilde{P}$ tries to convince an (honest) verifier V that she ($\tilde{P}$) is really P. This appealing definition models a world of smart-card claimants and untrusted verifiers—but, again, not a distributed system of always-running processes.

Entity authentication is not to be confused with message authentication or signing [23]; here the goal is to authenticate a document rather than an entity, and the model is different. We will see, however, that message authentication is a useful tool for entity authentication.

Much discussed in the literature is the "mafia fraud" (or "grandmaster chess problem"), in which an adversary faithfully relays messages between communication partners; in some settings (cf. [10]) this constitutes a damaging attack. Protection against such attacks is addressed in [12]. As we discussed, however, the faithful relaying of messages by an adversary does not, in our setting, constitute an attack; indeed, this is the basis of our definition of mutual authentication.

1.6 Future directions

The communication model we have introduced in this paper is captures attacks that are realistic threats in practice but not addressed by other models. It would make sense to return to known primitives, such as zero-knowledge proofs, and investigate their security in the more stringent setting we suggest.

2 Preliminaries

NOTATION. Let $\{0,1\}^*$ denote the set of finite binary strings, $\{0,1\}^\infty$ the set of infinite ones, and $\{0,1\}^{\leq L}$ the set of binary strings of length at most L. The empty string is written λ. When a, b, c, ... are strings used in some context, by $a\,.\,b\,.\,c\,.\,\cdots$ we denote an encoding of these strings such that each constituent string is efficiently recoverable given the encoding and the context of the string's receipt. In our protocols, concatenation will usually be adequate for this purpose. A function is *efficiently computable* if it can be computed in time polynomial in its first argument. A real-valued function $\epsilon(k)$ is *negligible* if for every $c > 0$ there exists a $k_c > 0$ such that $\epsilon(k) < k^{-c}$ for all $k > k_c$.

PROTOCOLS. The protocols we consider are two party ones, formally specified by an efficiently computable function Π on the following inputs:

1^k — the "security parameter" — $k \in \mathsf{N}$.

i — the "identity of the sender" — $i \in I \subseteq \{0,1\}^k$.
j — the "identity of the (intended) partner" — $j \in I \subseteq \{0,1\}^k$.
a — the "secret information of the sender" — $a \in \{0,1\}^*$.
κ — the "conversation so far" — $\kappa \in \{0,1\}^*$.
r — the "random coin flips of the sender" — $r \in \{0,1\}^\infty$.

The value of $\Pi(1^k, i, j, a, \kappa, r) = (m, \delta, \alpha)$ specifies:
m — the "next message to send out" — $m \in \{0,1\}^* \cup \{*\}$.
δ — the "decision" — $\delta \in \{\mathsf{A}, \mathsf{R}, *\}$.
α — the "private output" — $\alpha \in \{0,1\}^* \cup \{*\}$.

EXPLANATION. I is a set of *identities* which defines the *players* who can participate in the protocol. Although our protocols involve only two parties, the set of players I could be larger, to handle the possibility (for example) of an arbitrary pool of players who share a secret key. Elements of I will sometimes be denoted A or B (Alice and Bob), rather than i, j; we will switch back and forth irrationally between these notations. We stress that A, B (and i, j) are variables ranging over I (not fixed members of I), so $A = B$ (or $i = j$) is quite possible. Note that the adversary is *not* a player in our formalization.

The value a that a player sees is the private information provided to him. This string is sometimes called the *long-lived key* (or LL-key) of a player. In the case of (pure) symmetric authentication, all players $i \in I$ will get the same LL-key, and the adversary will be denied this key. In general, a LL-key generator $\mathcal{G}$ associated to a protocol will determine who gets what initial LL-key (see below).

The value "$*$" is supposed to suggest, for m, that the "the player sends no message." For δ, it means that "the player has not yet reached a decision." For α, it means "the player does not currently have any private output." The values A and R, for δ, are supposed to suggest "accept" and "reject," respectively. We denote the t-th component of Π (for $t \in \{1,2,3\}$) by Π_t.

Acceptance usually does not occur until the end of the protocol, although rejection may occur at any time. Some protocol problems, such as mutual authentication, do not make use of the private output; these protocol are concerned only with acceptance or rejection. For others, including key exchange protocols, the private output of a party will be what this party thinks is the key which has been exchanged. It is convenient to assume that once a player has accepted or rejected, this output cannot change.

To each protocol is associated its number of moves, R. In general this is a polynomially bounded, polynomial time computable function of the security parameter; in all our protocols, however, it is a constant.

THE LL-KEY GENERATOR. Associated to a protocol is a *long-lived key generator* (LL-key generator) $\mathcal{G}(1^k, \iota, r_G)$. This is a polynomial time algorithm which takes as input a security parameter 1^k, the identity of a party $\iota \in I \cup \{\mathsf{E}\}$, and an infinite string $r_G \in \{0,1\}^\infty$ (coin flips of the generator).

For all of the protocols of this paper, the associated LL-key generator will be a *symmetric* one, where for each $i, j \in I$ we have that $\mathcal{G}(1^k, i, r_G) = \mathcal{G}(1^k, j, r_G)$; while, on the other hand, $\mathcal{G}(1^k, \mathsf{E}, r_G) = \lambda$. The value of $\mathcal{G}(1^k, i, r_G)$ will just be a prefix of r_G (that is, a random string). The length of this prefix will vary according to the protocol we consider.

The presence of the LL-key generator allows us to consider protocols which we could not consider if we built a particular generator into our definition. For example, we can consider symmetric protocols which require that there be public information known to all parties. In this manner, the formalization of a protocol does not need to change depending on the particular setting. Although we will not need this extra generality in this paper, it seems useful to present.

3 A Communication Model for Distributed Security

We will now formulate a model appropriate for defining authentication and key distribution goals in the distributed environment. The situation we address is one where communication between players is entirely controlled by the adversary. The adversary can deliver messages out of order and to unintended recipients, and she can concoct messages of her own choosing. What is more, the adversary can conduct as many *sessions* as she pleases amongst the players, and she can control, for each, who is attempting to authenticate to whom.

Formally the adversary E is a probabilistic machine[3] $E(1^k, a_E, r_E)$ equipped with an infinite collection of oracles $\Pi^s_{i,j}$, for $i, j \in I$ and $s \in \mathsf{N}$. Oracle $\Pi^s_{i,j}$ models player i attempting to authenticate player j in "session" s. Adversary E communicates with the oracles via queries of the form (i, j, s, x) written on a special tape. The query is intended to mean that E is sending message x to i, claiming it is from j in session s. The query will, in our model, be answered by $\Pi^s_{i,j}$; the manner in which this response is computed is given by the following "experiment."

Running The Protocol. Running a protocol Π (with LL-key generator $\mathcal{G}$) in the presence of an adversary E, using security parameter k, means performing the following experiment:

(1) Choose a random string $r_G \in \{0,1\}^\infty$ and set $a_i = \mathcal{G}(1^k, i, r_G)$, for $i \in I$, and set $a_E = (1^k, \mathsf{E}, r_G)$.

(2) Choose a random string $r_E \in \{0,1\}^\infty$ and, for each $i, j \in I$, $s \in \mathsf{N}$, a random string $r^s_{i,j} \in \{0,1\}^\infty$.

(3) Let $\kappa^s_{i,j} = \lambda$ for all $i, j \in I$ and $u \in \mathsf{N}$. (The variable $\kappa^s_{i,j}$ will keep track of the conversation that $\Pi^s_{i,j}$ engages in.)

(4) Run adversary E on input $(1^k, a_E, r_E)$, answering oracle calls as follows. When E asks a query (i, j, s, x), oracle $\Pi^s_{i,j}$ computes $(m, \delta, \alpha) = \Pi(1^k, i, j, a_i, \kappa^s_{i,j} \,.\, x, r^s_{i,j})$ and answers with (m, δ). Then $\kappa^s_{i,j}$ gets replaced by $\kappa^s_{i,j} \,.\, x$.

We point out that in response to an oracle call, E learns not only the outgoing message but also whether or not the oracle has accepted or rejected. (For convenience of discourse, we often omit mention of the latter.) According to the above, E doesn't learn the oracle's private output. For some problems (such as authenticated key exchange) we will need to give the adversary the power to sometimes learn these private outputs. Such an extension is handled by specifying a new kind of oracle query and then indicating how the experiment is extended with responses to the new class of queries.

The Benign Adversary. It is useful for some of our definitions to consider a certain particularly

[3] Adversaries can be uniform or non-uniform, and the results of this paper hold in both cases, with uniform adversaries requiring a uniform complexity assumptions and non-uniform adversaries requiring non-uniform ones.

friendly kind of adversary. An adversary is called *benign* if it is deterministic and restricts its action to choosing a pair of oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ and then faithfully conveying each flow from one oracle to the other, with $\Pi^s_{i,j}$ beginning first. In other words, the first query E makes is (i, j, s, λ), generating response α_1; the second query E makes is (j, i, t, α_1), generating response β_1; and so forth. While the choice of i, j, s, t is up to the adversary, this choice is the same in all executions with security parameter k.

TIME. In a particular execution of a protocol, the adversary's i-th query to an oracle is said to occur at time $\tau = \tau_i \in \mathbf{R}$. We intentionally do not specify $\{\tau_i\}$, except to demand that $\tau_i < \tau_j$ when $i < j$. Conforming notions of time include "abstract time," where $\tau_i = i$, and "Turing machine time," where $\tau_i =$ the i-th step in E's computation, when parties are realized by interacting Turing machines. Another conforming notion of time (but a harder one to formalize) is "real time," where τ_i the exact time when the i-th query is made, when parties are realized by interacting computers.

A SINGLE MODEL FOR MANY GOALS. We have not yet defined any particular goal; we have only specified the adversarial model in which these goals are formulated. Indeed, this same model underlies a large collection of authentication and key-distribution goals. In the following, we define mutual authentication and authenticated key exchange, but we stress that these definitions can be easily extended to ones for related problems, including unilateral and three-party authentication.

4 Entity Authentication

A central notion in formalizing entity authentication goals is that of a matching conversation.

4.1 Matching Conversations

We will define mutual authentication (MA) by an experiment involving the running of adversary E with security parameter k. When E terminates, each oracle $\Pi^s_{i,j}$ has had a certain conversation $\kappa^s_{i,j}$ with E, and it has reached a certain decision $\delta \in \{\mathsf{A}, \mathsf{R}, *\}$. Whether or not Π is a secure MA protocol is defined in terms of the distribution on these conversations and decisions. One way to think of it is that each execution will be classified as either "good" or "bad," depending on whether or not the adversary managed to subvert this particular run. We now turn towards distinguishing the good runs from the bad.

DEFINITION. Fix an execution of an adversary E (that is, fix the coins of the LL-key generator, the oracles, and the adversary). For any oracle $\Pi^s_{i,j}$ we can capture its *conversation* (for this execution) by a sequence

$$K = (\tau_1, \alpha_1, \beta_1),\ (\tau_2, \alpha_2, \beta_2),\ \ldots,\ (\tau_m, \alpha_m, \beta_m).$$

This sequence encodes that at time τ_1 oracle $\Pi^s_{i,j}$ was asked α_1 and responded with β_1; and then, at some later time $\tau_2 > \tau_1$, the oracle was asked α_2 and answered β_2; and so forth, until, finally, at time τ_m it was asked α_m and answered β_m. Adversary E terminates without asking oracle $\Pi^s_{i,j}$ any more questions.

Suppose oracle $\Pi^s_{i,j}$ has conversation prefixed by $(\tau_1, \alpha_1, \beta_1)$. Then if $\alpha_1 = \lambda$ we call $\Pi^s_{i,j}$ an *initiator* oracle; if α_1 is any other string we call $\Pi^s_{i,j}$ a *responder* oracle.

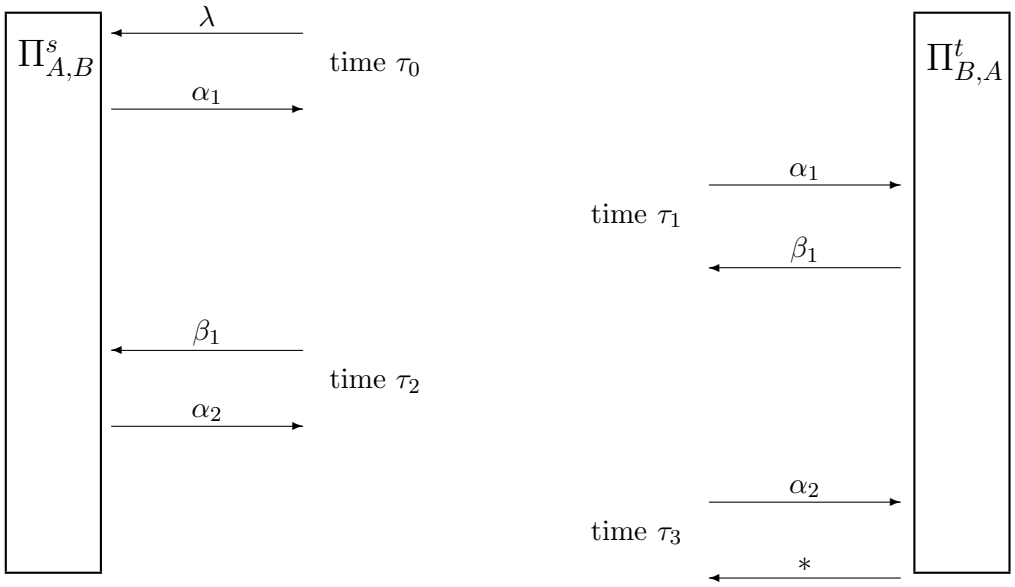

Figure 1: *Anatomy of a matching conversation for a 3-move protocol. The left-hand conversation matches the right-hand one. Omitting arrows associated to τ_3 (lower right), the right-hand conversation matches the left-hand one.*

We now define matching conversations. For simplicity we focus on the case where R is odd; the case of even R is analogous and is left to the reader. Explanations follow the formal definition.

Definition 4.1 (matching conversations) *Fix a number of moves $R = 2\rho - 1$ and an R-move protocol Π. Run Π in the presence of an adversary E and consider two oracles, $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$, that engage in conversations K and K', respectively.*

(1) *We say that K' is a* matching conversation *to K if there exist $\tau_0 < \tau_1 < \ldots < \tau_R$ and $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ such that K is prefixed by*

$$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), (\tau_4, \beta_2, \alpha_3), \ldots, (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-1}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$$

and K' is prefixed by

$$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), (\tau_5, \alpha_3, \beta_3), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}) \ .$$

(2) *We say that K is a* matching conversation *to K' if there exist $\tau_0 < \tau_1 < \ldots < \tau_R$ and $\alpha_1, \beta_1, \ldots, \alpha_\rho, \beta_\rho$ such that K' is prefixed by*

$$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), (\tau_5, \alpha_3, \beta_3), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}), (\tau_{2\rho-1}, \alpha_\rho, *) \ .$$

and K is prefixed by

$$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), (\tau_4, \beta_2, \alpha_3), \ldots, (\tau_{2\rho-4}, \beta_{\rho-2}, \alpha_{\rho-2}), (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho)$$

Explanation. Case (1) defines when the conversation of a responder oracle matches the conversation of an initiator oracle. Case (2) defines when the conversation of an initiator oracle matches the conversation of a responder oracle.

Let us paraphrase our definition. Consider an execution in which $\Pi^s_{A,B}$ is an initiator oracle and $\Pi^t_{B,A}$ is a responder oracle. If every message that $\Pi^s_{A,B}$ sends out, except possibly the last, is subsequently delivered to $\Pi^t_{B,A}$, with the response to this message being returned to $\Pi^s_{A,B}$ as its own next message, then we say that the conversation of $\Pi^t_{B,A}$ matches that of $\Pi^s_{A,B}$. Similarly, if every message that $\Pi^t_{B,A}$ receives was previously generated by $\Pi^s_{A,B}$, and each message that $\Pi^t_{B,A}$ sends out is subsequently delivered to $\Pi^s_{A,B}$, with the response that this message generates being returned to $\Pi^t_{B,A}$ as its own next message, then we say that the conversation of $\Pi^s_{A,B}$ matches the one of $\Pi^t_{B,A}$. Note that this second condition is easily seen to imply the first one.

We comment that the party who sends the last flow ($\Pi^s_{A,B}$, above) can't "know" whether or not its last message was received by its partner, so when this oracle accepts accepts, it cannot "know" (assuming this last message to be relevant) whether or not its partner will accept. This asymmetry is an inherent aspect of authentication protocols with a fixed number of moves, giving a certain information benefit to the party who refrains from putting in the last word.

We will say that oracle $\Pi^t_{j,i}$ has a matching conversation with oracle $\Pi^s_{i,j}$ if the first has conversation K', the second has conversation K, and K' matches K. Either party, here, may be the initiator.

4.2 Mutual Authentication

We now define mutual authentication, provide a simple protocol for it, and, assuming the existence of pseudo-random function, prove this protocol meets our definition.

Definition. We require that any mutual authentication protocol have $R \geq 3$ rounds. We implicitly make this assumption in our definition and throughout the remainder of this paper.

In a mutual authentication protocol, when a party accepts with a certain conversation K, that party believes that there is some other party who engaged in a matching conversation K'. Just saying this wouldn't be enough: we also need that parties accept in the absence of an adversary.

Let $\mathsf{No\text{-}Matching}^E(k)$ be the event that there exist i, j, s such that $\Pi^s_{i,j}$ accepted and there is no oracle $\Pi^t_{j,i}$ which engaged in a matching conversation. The definition of a mutual authentication is as follows:

Definition 4.2 (secure mutual authentication) *We say that* Π *is a* secure mutual authentication *protocol if for any polynomial time adversary* E*,*

(1) (Matching conversations $\Rightarrow$ acceptance.) *If oracles* $\Pi^s_{A,B}$ *and* $\Pi^t_{B,A}$ *have matching conversations, then both oracles accept.*

(2) (Acceptance $\Rightarrow$ matching conversations.) *The probability of* $\mathsf{No\text{-}Matching}^E(k)$ *is negligible.*

Restating this may make it clearer. The first condition says that if each party's messages are faithfully relayed to one another, than the parties accept the authentication of one another. The second condition calls an execution *good* if for each accepting conversation K by an oracle $\Pi^s_{i,j}$ there exists a matching conversation K' by some oracle $\Pi^t_{j,i}$, and *bad* otherwise. We require that the probability of a bad execution be negligible.

Uniqueness Of Matching Partner. One consequence of the definition worth stating is that an oracle's matching partner is unique. More formally, let $\mathsf{Multiple\text{-}Match}^E(k)$ be the event that

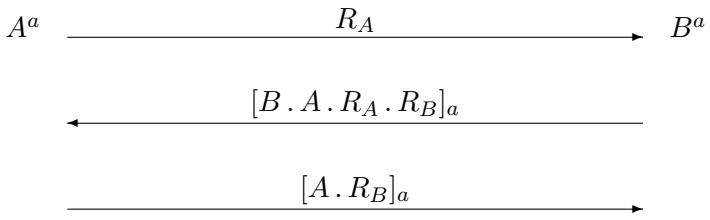

Figure 2: *Protocol MAP1: a mutual authentication of any two principals, A and B, among a set of principals I who share a key a.*

some $\Pi^s_{i,j}$ accepts, and there are at least two distinct oracles $\Pi^t_{j,i}$ and $\Pi^{t'}_{j,i}$ which have had matching conversations with $\Pi^s_{i,j}$. We can show the following.

Proposition 4.3 *Suppose Π is a secure MA protocol. Let E be any polynomial time adversary. Then the probability of* $\mathsf{Multiple\text{-}Match}^E(k)$ *is negligible.*

The proof is given in Appendix C. We comment that in all of our protocols the probability of $\mathsf{Multiple\text{-}Match}^E(k)$ is not just negligible but exponentially small, being at most $T_E(k)^2 \cdot 2^{-k}$ where $T_E(k)$ is the (polynomial) number of oracle calls of the adversary.

Authenticating Messages. Message authentication via pseudo-random functions [18, 19] is a tool in our entity authentication protocols. Let f be a pseudorandom function family [18]. Denote by f_a: $\{0,1\}^{\leq L(k)} \to \{0,1\}^{l(k)}$ the function specified by key a. In general, the length of the key, the length L of the input to f_a, and the length l of the output, are all functions of the security parameter. Here we assume the key length is just k, and, for our first protocol (MAP1) it suffices to assume $L(k) = 4k$ and $l(k) = k$.

For any string $x \in \{0,1\}^{\leq L(k)}$ define $[x]_a = (x, f_a(x))$; this will serve as an authentication of message x [18, 19]. For any $i \in I$, $[i\,.\,x]_a$ will serve as i's authentication of message x.

A Protocol For Mutual Authentication. Our first protocol (called "MAP1," for "mutual authentication protocol one") is represented by Figure 2. Alice (A) begins by sending Bob (B) a random challenge R_A of length k. Bob responds by making up a random challenge R_B of length k and returning $[B\,.\,A\,.\,R_A\,.\,R_B]_a$. Alice checks that this message is of the right form and is correctly tagged as coming from B. If it is, Alice sends Bob the message $[A\,.\,R_B]_a$ and accepts. Bob checks that this message is of the right form and is correctly tagged as coming from A, and, if it is, he accepts. We stress that checking the message is of the right form, for A in the second flow, includes checking that the nonce present in the message is indeed the same nonce she sent in the first flow; similarly for B with respect to checking the third flow. We comment that $A = B$ is permitted; these are any two identities in the set I.

Theorem 4.4 (MAP1 is a secure MA) *Suppose f is a pseudorandom function family. Then protocol MAP1 described above and based on f is a secure mutual authentication.*

The proof of Theorem 4.4 appears in Appendix A.

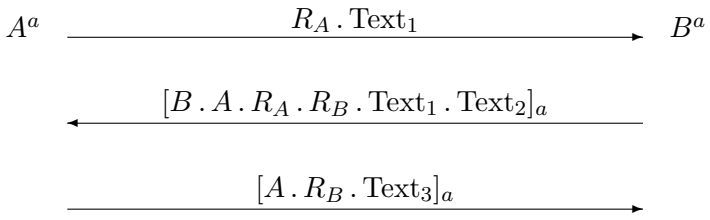

Figure 3: *Protocol MAP2: Flows for an authenticated exchange of three text strings.*

Unpredictability Of Challenges. We comment that MAP1 (and every other protocol of this paper) does not meet its definition if its random challenges (R_A and R_B) are taken to be arbitrary nonces. (A *nonce* is a value used at most once.) The challenges must be unpredictable; a predictable nonce (like a sequence number) won't work.

Out-Of-Band Data. Notice that A is not present in the first flow, even though, "in practice," B may need this information to select the "right" shared key a. Formally, this identifier is irrelevant; in practice, it might be communicated "out of band," or it might be authenticated along with the other exchanged messages, as we now describe.

Authenticated Exchange Of Text. For most applications, it is useful to combine a mutual authentication with an exchange of authenticated data, so that an accepting party accepts not only the identity of a partner, but also has confidence that data associated to protocol flows originates with this partner. A protocol to accomplish this, derived from MAP1, is shown in Figure 3. Here the Text_i strings are authenticated along with the rest of the exchanged messages. As illustrated in the next section, one use of these strings is to carry an encryption of a fresh session key, thereby accomplishing an authenticated key exchange.

We will not, in this paper, give any formal definition of the authenticated exchange of text goal. We give the MAP2 only to serve as intuition for the derivation of the AKEP1. However the definitions of authenticated key exchange and formal proof of correctness of AKEP1 that we will provide later will be independent of discussions of authenticated exchange of text.

Security Of 2PP. Combining ideas from our proof of Theorem 4.4 with a lemma from [2], we can show that a special case of the protocol 2PP of [6] meets our definition of a secure mutual authentication. Specifically, assume that the "encryption" function E being used in 2PP is a PRF; assume nonces are instantiated with random values; and assume $|I| = 2$ and authenticating parties are guaranteed to have distinct identities. The CBC Lemma of [2] (stated here informally as Lemma D.1) says that the function being used in their protocol, namely the CBC of E, will also be a PRF. Given this, one can trace through our proof as given in Appendix A and check that it extends.

Discussion. Our definition is very strong—perhaps the strongest possible natural definition. In asking that flows match *exactly* it may be criticized as too strong; as pointed out by [11], certain parts of protocol flows might be "irrelevant" for the authentication, and perhaps one ought to allow them to be ignored in matching. Such extensions seem valuable, especially for the asymmetric case.

The case where all protocol flows (except authenticated text strings, if present) are ignored may be particularly interesting.

5 Authenticated Key Exchange

We specify a formal definition for authenticated key exchange (AKE). As mentioned in the introduction, mutual authentication protocols can usually be extended to distribute keys. We will appropriately extend MAP1 to AKEP1 and prove that it meets our definition. We will also specify AKEP2, the "implicit" key distribution protocol associated to MAP1; this protocol manages to distribute a key with no added communication overhead.

Session Keys. We wish to say what it means for a MA protocol Π to be a secure AKE. Fix $S = \{S_k\}_{k \in \mathsf{N}}$ with each S_k a distribution over $\{0,1\}^{\sigma(k)}$, for some polynomial $\sigma(k)$. The intent of an AKE will be both to authenticate entities *and* to distribute a "session key" sampled from S_k. When a player accepts, his private output will be interpreted as the session key which he has computed. Formally, the session key α will be defined by Π_3. For simplicity, we assume that an accepting player always has a string-valued private output of the right length (that is, if $\Pi_2 = \mathsf{A}$ then $\Pi_3 \in \{0,1\}^{\sigma(k)}$), while a non-accepting player has a session key of $*$ (that is, if $\Pi_2 \in \{\mathsf{R}, *\}$ then $\Pi_3 = *$).

Session Key Freshness. An important property that we want of a protocol that distributes session keys is that the compromise of one of these keys should have minimal consequences overall. For example, its revelation should not allow one to subvert subsequent authentication, nor should it leak information about other (as yet uncompromised) session keys. To capture this requirement, we extend the interaction of the adversary with its oracles by adding a new type of query, as follows: we say that the adversary can learn a session key $\alpha_{i,j}^s$ of an oracle $\Pi_{i,j}^s$ by issuing to the oracle a distinguished reveal query, which takes the form $(i, j, s, \mathsf{reveal})$. The answer returned by the oracle is $\alpha_{i,j}^s$.

To quantify the power of an adversary who can perform this new type of query, we require some additional definitions. Initially, each oracle $\Pi_{i,j}^s$ is declared *unopened*, and so it remains until the adversary generates a reveal query $(i, j, s, \mathsf{reveal})$. At this point, the oracle is declared *opened.* We say that an oracle $\Pi_{i,j}^s$ is *fresh* if the following three conditions hold: First, $\Pi_{i,j}^s$ has accepted. Second, $\Pi_{i,j}^s$ is unopened. Third, there is no opened oracle $\Pi_{j,i}^t$ which engaged in a matching conversation with $\Pi_{i,j}^s$. When oracle $\Pi_{i,j}^s$ is fresh, we will also say that "the oracle holds a fresh session key." Intuitively, an oracle holds a fresh session key if that key is unavailable to the adversary by trivial means.

Protecting Fresh Session Keys. We want that the adversary should be unable to understand anything interesting about a fresh session key. This can be formalized along the lines of security of probabilistic encryption; the particular formalization we will adapt is that of (polynomial) indistinguishability of encryptions [21, 16, 17]. We demand that at the end of a secure AKE the adversary should be unable to distinguish a fresh session key α from a random element of S_k. We proceed as follows.

After the adversary has asked all the (i, j, s, x) and $(i, j, s, \mathsf{reveal})$ queries that she wishes to ask, the adversary asks of a fresh oracle $\Pi_{i,j}^s$ a single query (i, j, s, test). The query is answered

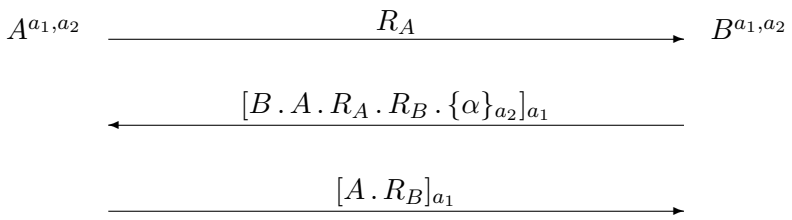

Figure 4: *Protocol AKEP1: The value α is the session key distributed.*

by flipping a fair coin $b \leftarrow \{0,1\}$ and returning $\alpha^s_{i,j}$ if $b = 0$, or else a random sample from S_k if $b = 1$. The adversary's job is to guess b. To this end, she outputs a bit Guess, and then terminates. Let $\mathsf{Good\text{-}Guess}^E(k)$ be the event that $\mathsf{Guess} = b$, when the protocol is executed with security parameter k; in other words, this is the probability the adversary has correctly identified whether she was given the real session key or just a sample from S_k. Let

$$advantage^E(k) = \max\left\{\, 0,\ \Pr\left[\mathsf{Good\text{-}Guess}^E(k)\right] - \tfrac{1}{2} \,\right\}\ .$$

Secure AKE. We are now ready to give the main definition of this section. We refer the reader to Section 3 for the definition of a benign adversary.

Definition 5.1 (Authenticated Key Exchange (AKE)) *Protocol Π is a* secure AKE *over $S = \{S_k\}_{k\in\mathsf{N}}$ if Π is a secure mutual authentication protocol, and, in addition, the following are true:*

(1) (Benign adversary $\Rightarrow$ keys according to S_k) *Let B be any benign adversary and let $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ be its chosen oracles in the experiment with security parameter k. Then both oracles always accept, $\alpha^s_{i,j} = \alpha^t_{j,i}$, and moreover this random variable is distributed according to S_k.*

(2) (Session key is protected) *Let E be any polynomial time adversary. Then $advantage^E(k)$ is negligible.*

The first condition says that if flows are honestly conveyed then a session key is agreed upon, and this key is properly distributed. The second condition says that the adversary can't tell this session key from a random string of the same distribution.

Comments. Since the protocol is assumed to be a secure mutual authentication, we know that if oracles $\Pi^s_{i,j}$ and $\Pi^t_{j,i}$ have matching conversations then they both accept. From the first condition it follows that they will also have the same session key.

It is now easier to see why it is only fair to talk about the adversary guessing session key / random key for an oracle hiding a fresh session key. If the oracle pointed to by E is not fresh then E already knows the key: If the oracle has not accepted then the key is $*$ (and E sees whether or not an oracle has accepted); if the oracle has been opened, then the adversary was provided the session key; and if a matching partner has had its oracle opened, then once again E knows the session key, as E can tell that this oracle engaged in a matching conversation. Of course our assumption that E always points to a fresh oracle is just a convenient simplification, in the sense that we cannot stop an arbitrary adversary from pointing to an unfresh oracle (and there may well

be no fresh oracle to point to at all). But this is clearly not a situation which there is any need to address.

PROTOCOL. Let $S = \{S_k\}$ be a family of samplable distributions on $\{0,1\}^{\sigma(k)}$. A protocol for AKE over distribution S is derived from MAP2 by using the second text string to communicate an encrypted key, as follows.

The parties share a $2k$ bit LL-key which we denote a_1, a_2. The first part, a_1, is taken as the key to the pseudo-random function family f, yielding a PRF $f_{a_1}\colon \{0,1\}^{\leq L(k)} \to \{0,1\}^k$ to be used for message authentication; this time, $L(k) = 5k + \sigma(k)$ will suffice. The second part, a_2, is used as a key to another pseudo-random family f' with the property that $f'_{a_2}\colon \{0,1\}^k \to \{0,1\}^{\sigma(k)}$. A probabilistic encryption of string $\alpha \in \{0,1\}^{\sigma(k)}$ is defined by $\{\alpha\}_{a_2} \stackrel{\text{def}}{=} (r, f'_{a_2}(r) \oplus \alpha)$, with r selected at random [18]. Party B chooses the session key α from S_k and sets Text_2 to be $\{\alpha\}_{a_2}$. The strings Text_1 and Text_3 of MAP2 are set to λ. This protocol, which we call AKEP1, is shown in Figure 4.

It is important that a_2 (the key used for encryption) be distinct from a_1 (the shared key used for the message authentication). Formally, the LL-key generator $\mathcal{G}$ provides the parties $i \in I$ with a $2k$-bit shared key. The two keys need not be independent, however; the generator could set $a_i = f_a(i)$ $(i = 1, 2)$ where a is a random k bit key and f_a is a pseudo-random function.

Theorem 5.2 *Let $S = \{S_k\}$ be samplable, and suppose f, f' are pseudo-random function families with the parameters specified above. Then the protocol AKEP1 based on f, f' is a secure AKE over S.*

The proof of this theorem is given in Appendix B.

IMPLICIT KEY DISTRIBUTION. A more efficient (in terms of communication complexity) AKE protocol may be devised by using what we call an "implicit" key distribution. In this case, the flows between A and B are the same as in MAP1, and one (or more) of the parameters already present in the flows of MAP1 (say R_B) is used to define the session key. Specifically, let $S = \{S_k\}$ be a family of distributions given by $S_k = g(U_k)$, for some deterministic, polynomial-time computable function g, where U_k is the uniform distribution on k-bit strings; for example $S_k = U_k$ and g the identity, the most useful choice in practice. Again the parties share a $2k$ bit LL-key a_1, a_2, with a_1 being used as the key in MAP1 (so $L(k) = 4k$). Let f' be a pseudorandom *permutation* family [28]; f'_{a_2} specifies a permutation on $\{0,1\}^k$. Define the protocol AKEP2 by having its flows be identical to MAP1, with a_1 being used for message authentication. Each accepting party outputs session key $\alpha = g(f'_{a_2}(R_B))$. This protocol, which we call AKEP2, is shown in Figure 5. Modifying the proof of Theorem 5.2 we can show the following:

Theorem 5.3 *Let $S = \{S_k\}$ be given by $S_k = g(U_k)$, for some polynomial time g. Suppose f, f' are a pseudo-random function family and a pseudo-random permutation family, with the parameters specified above. Then the protocol AKEP2 based on f, f' is a secure AKE over S.*

The assumption that f' is a pseudo-random permutation can be relaxed to f' being a pseudo-random function at the cost of distribution on session keys being $S_k = g(V_k)$, where V_k is pseudo-randomly distributed.

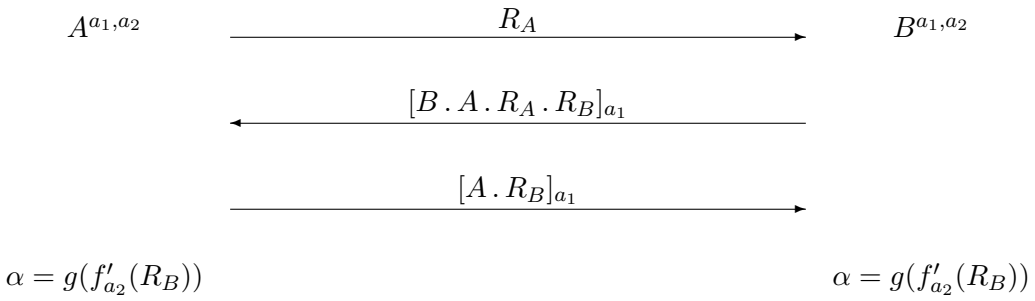

Figure 5: *Protocol AKEP2: The Implicit Key Exchange Protocol. The value α is the session key "implicitly" distributed.*

Discussion. A potential concern about the definition is whether or not hiding information about *individual* session keys is good enough. For example, is it possible that an adversary for a secure AKE might be able to point to a *pair* of fresh oracles, holding keys α and β, and then be able to distinguish (α, β) from a pair of random keys (s_1, s_2) from S_k? We can show (proof omitted) that this cannot happen under our definition of a secure AKE.

The formalization of the adversary's being unable to learn anything about a fresh session key could also have been made by adapting the notion of "semantic security" of encryptions [21, 16, 17] to our setting. Roughly, we would say the following. Let E be any polynomial time adversary and let Q_ξ^k be any collection of functions indexed by the security parameter k and possible views ξ of the adversary. Then there exists an adversary E' such that the following is true. If E could correctly predict Q_ξ^k on a fresh session key with probability $p(k)$ then E' could predict Q_ξ^k on a random, hidden sample of S_k with probability negligibly different from $p(k)$. Properly formalized, this can be shown to be equivalent to the second condition in Definition 5.1.

Our definition has been designed to ensure the following: after the key distribution protocol, it should be possible to use the session key for any purpose for which an out-of band distributed key could have been used. That is, the session key distributed by an AKE protocol may be used as safely as one handed privately to each party by a trusted third party.

One consequence of the strength of our definition is to exclude some protocols which seem to have traditionally been considered "secure." For example, let AKEP1$'$ work just like AKEP1 except that, in the third move, the message authentication is computed under the newly distributed session key α, instead of under the long-lived key a_1. One can check that this protocol is not a secure AKE under our definition. To see, intuitively, that our definition is right to exclude it, note that it is possible to construct a protocol that is secure if α had been distributed out-of-band but not if α is distributed by AKEP1$'$.

Nonetheless weaker notions for protecting the security of a distributed key may be desirable in some settings. For example, in some situations the intended usage of the session key is specific and known, and a definition could be designed which ensured that the distributed session key was secure enough to suffice for the particular intended application (although it may not suffice for other applications). We believe such investigations would be useful and constitute interesting open questions.

6 From Theory to Practice

The provably secure protocols of the previous sections lead to efficient and secure-in-practice protocols when the pseudorandom functions in terms of which these protocols are described are correctly and efficiently instantiated. The purpose of this section is to illustrate some options for doing this. We then move on to discuss some further implementation specifics.

6.1 Constructions of Practical Pseudo-Random Functions

We will suggest constructions of PRFs suitable for our purposes based on DES and secure hash functions such as MD5. Let's begin by discussing the primitives.

PRIMITIVES. The algorithm of the DES specifies for each 64 bit key a a permutation DES_a from $\{0,1\}^{64}$ to $\{0,1\}^{64}$. The viewpoint adopted here —suggested by Luby and Rackoff [28, 29]— is to regard DES as a pseudo-random permutation, with respect to practical computation.

The MD5 function [32] maps an arbitrary string x into a 128-bit string $\mathrm{MD5}(x)$. It is intended that this function be a collision-free hash function, with respect to practical computation.

THE PROBLEM AND OUR DESIGN PHILOSOPHY. Cryptographic practice provides good PRFs on particular input lengths l (for example, DES for $l = 64$). In contrast, our protocols need PRFs for arbitrary input lengths. In devising such PRFs, we prefer not to rely purely on heuristics. In most cases we will rely on provably correct constructions of arbitrary length PRFs based on fixed length PRFs and collision free hash functions, individually or in combination; the lemmas underlying our constructions are summarized in Appendix D. The exception is the third construction given below; we'll discuss it when we get there.

NOTATION. Let g_a denote a PRF of l bits to l bits. Suppose y has length a multiple of l bits, and write it as a sequence of l bit blocks, $y = y_1 \ldots y_n$. The cipher block chaining (CBC) operator defines

$$\mathrm{CBC}_a^g(y_1 \ldots y_n) \quad = \quad \begin{cases} g_a(y_1) & \text{if } n = 1 \\ g_a(\mathrm{CBC}_a^g(y_1 \ldots y_{n-1}) \oplus y_n) & \text{otherwise} \end{cases}$$

Let H denote a collision free hash function of $\{0,1\}^*$ to $\{0,1\}^{2l}$. Let $H_1(x)$ and $H_2(x)$ denote the first l bits of $H(x)$ and the last l bits of $H(x)$, respectively. Finally $\langle x \rangle_l$ will denote some standard padding of x to string of length a multiple of l bits; for example, always add a 1 and then add enough zeroes to get to a length which is a multiple of l.

CONSTRUCTIONS. We suggest three constructions of a PRF f_a mapping long inputs to short outputs. For each construction we discuss its security and efficiency. Here $l = 64$, $H = \mathrm{MD5}$, $g = \mathrm{DES}$. The key a has length 64 bits.

(1) *The CBC PRF.* Let $f_a(x)$ be the first $l/2$ bits of $\mathrm{CBC}_a^g(\langle x \rangle_l \,.\, |\langle x \rangle_l|)$, where $|y|$ is the length of y encoded as an l-bit string. This construction is justified by Lemma D.1.[4]

[4] Lemma D.1 does not require us to drop the last $l/2$ bits of the output. We drop them for two reasons. The first is efficiency. The second is specific to DES and will not be discussed here.

(2) *The CBC/Hash PRF.* Let $f_a(x)$ be the first $l/2$ bits of $g_a(g_a(H_1(x)) \oplus H_2(x)) = \mathrm{CBC}^g_a(H(x))$. This construction is justified by Corollary D.3. In software this is significantly more efficient than the CBC construction, requiring one hash and two DES operations.

(3) *The Pure Hash PRF.* Let $f_a(x)$ be the first $l/2$ bits of $H(x \,.\, a)$. This construction was suggested in [36] as a message authentication code; we suggest the stronger assumption that it is a PRF. However no standard assumption about H of which we are aware can be used to justify the the security of this construction, and it should be viewed more as a heuristic than the two constructions suggested above.[5] In software, however, it is the most efficient of the three constructions.

All the above constructions are practical. However, since the security of the last construction is not as well justified as the first two, our overall preference is to use the CBC-Hash PRF for software applications and the CBC PRF for hardware applications.

Similar constructions can be given using other primitives; for example the SHA instead of MD5, etc.

We stress the importance, in our security considerations, of the CBC and Hash Lemmas of Appendix D; the lack of such lemmas has lead in the past to more complex assumptions about the security of CBC and other constructions (e.g., [6, Definition 2.1]).

6.2 Efficiency and Implementation Issues in our Protocols

We suggest that the random challenges be 64 bits. We suggest each identity be encoded with 64 bits. (Remember that identities only need be unique among the space of parties that share the secret key.) Even for the key exchange protocols, the parties only need share a 64-bit LL-key a; two 64-bit keys a_1 and a_2 can be derived from a by setting $a_1 = g_a(1)$ and $a_2 = g_a(2)$, for a PRF g. This method to create multiple effective keys from a single LL-key preserves all claims of provable security. The PRFs used for message authentication and message encryption can be constructed using any of the methods of the last subsection.

In implementation, redundant parts of the flows may be dropped. For example, the second flow of MAP1 specified in Figure 2 is $[A \,.\, B \,.\, R_A \,.\, R_B]_a = (A \,.\, B \,.\, R_A \,.\, R_B, f_a(A \,.\, B \,.\, R_A \,.\, R_B))$. But of course it suffices to send $(R_B, f_a(A \,.\, B \,.\, R_A \,.\, R_B))$ because A already knows the quantities A, B, R_A. Such optimizations may be made in other protocol flows. Such optimizations do not damage claims of provable security.

When $|I| = 2$ and authenticating parties are guaranteed to have distinct identities, the identity A in the second flow of MAP1 and derivative protocols may be dropped. This optimization does not damage claims of provable security.

Protocol AKEP2 provides a particularly efficient key exchange. The computational complexity is minimally more than that of MAP1, and the communication complexity is identical. A useful instantiation in this regard is to use $f' = \mathrm{DES}$, exploiting the fact that DES is a permutation to get a uniformly distributed session key.

[5] See [4] for another viewpoint.

Acknowledgments

We thank Bob Blakley, Oded Goldreich, Amir Herzberg, Phil Janson, and the member of the CRYPTO 93 committee for all of their comments and suggestions. Especially we acknowledge Oded as suggesting that we formulate the security of fresh keys along the lines of polynomial indistinguishability (instead of the equivalent semantic security formulation we had before); and Amir for suggesting that we give Proposition 4.3, specify MAP1 in a way that does not assume authenticating entities are distinct agents from a set of cardinality two, and that to avoid the efficiency loss from padding we explicitly specify our pseudorandom functions as acting on $\{0,1\}^{\leq L(k)}$ instead of $\{0,1\}^{L(k)}$.

Most of this work was done while the first author was at the IBM T.J. Watson Research Center and second author was at IBM Austin.

References

[1] M. Abadi and M. Tuttle, "A semantics for a logic of authentication," *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing*, pp. 201–216, August 1991.

[2] M. Bellare, J. Kilian, and P. Rogaway, "The security of cipher block chaining," *Advances in Cryptology – Crypto 94 Proceedings*.

[3] M. Bellare and O. Goldreich, "On defining proofs of knowledge," *Advances in Cryptology — Proceedings of CRYPTO 92*, Springer-Verlag, 1992.

[4] M. Bellare and P. Rogaway, "Random oracles are practical: a paradigm for designing efficient protocols," *Proceedings of 1st ACM Conference on Computer and Communications Security*, November 1993.

[5] Th. Beth, M. Frisch and G. Simmons (Eds.), *Public-Key Cryptography: State of the Art and Future Directions*, E.I.S.S. Workshop, Oberwolfach, Germany, July 1991, Final Report. Lecture Notes in Computer Science, Vol. 578, Springer-Verlag, 1991.

[6] R. Bird, I. Gopal, A. Herzberg, P. Janson, S. Kutten, R. Molva and M. Yung, "Systematic design of two-party authentication protocols," *Advances in Cryptology — Proceedings of CRYPTO 91*, Springer-Verlag, 1991.

[7] M. Blum and S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits," *SIAM Journal on Computing* **13**(4), 850-864 (November 1984).

[8] M. Burrows, M. Abadi and R. Needham, "A logic for authentication," DEC Systems Research Center Technical Report 39, February 1990. Earlier versions in *Proceedings of the Second Conference on Theoretical Aspects of Reasoning about Knowledge*, 1988, and *Proceedings of the Twelfth ACM Symposium on Operating Systems Principles*, 1989.

[9] G. Brassard, C. Crépeau, S. Laplante and C. Léger, "Computationally convincing proofs of knowledge," *Proc. of the 8th STACS*, 1991.

[10] Y. Desmedt, C. Goutier, S. Bengio, "Special uses and abuses of the Fiat-Shamir passport protocol," *Advances in Cryptology — Proceedings of CRYPTO 87,* Springer-Verlag, 1988.

[11] W. Diffie, P. van Oorschot and M. Wiener, "Authentication and authenticated key exchanges," *Designs, Codes and Cryptography*, 2, 107–125 (1992).

[12] D. Dolev, C. Dwork and M. Naor, "Non-malleable cryptography," *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[13] U. Feige, A. Fiat and A. Shamir, "Zero knowledge proofs of identity," *Journal of Cryptology,* Vol. 1, pp. 77–94 (1987).

[14] U. Feige and A. Shamir, "Witness Indistinguishable and Witness Hiding Protocols," *Proceedings of the Twenty Second Annual Symposium on the Theory of Computing,* ACM, 1990.

[15] A. Fiat and A. Shamir, "How to prove yourself: practical solutions to identification and signature problems," *Advances in Cryptology – Crypto 86 Proceedings,* Lecture Notes in Computer Science Vol. 263, Springer-Verlag, A. Odlyzko, ed., 1986.

[16] O. Goldreich, "Foundations of cryptography," class notes, Technion University, Computer Science Department, Spring 1989.

[17] O. Goldreich, "A uniform complexity treatment of encryption and zero-knowledge," *Journal of Cryptology,* Vol. 6, pp. 21-53 (1993).

[18] O. Goldreich, S. Goldwasser and S. Micali, "How to construct random functions," *Journal of the ACM,* Vol. 33, No. 4, 210–217, (1986).

[19] O. Goldreich, S. Goldwasser and S. Micali, "On the cryptographic applications of random functions," *Advances in Cryptology — Proceedings of CRYPTO 84,* Springer-Verlag, 1984.

[20] O. Goldreich, S. Micali and A. Wigderson, "How to play any mental game," *Proceedings of the Nineteenth Annual Symposium on the Theory of Computing,* ACM, 1987, 218–229.

[21] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences* Vol. 28, 270-299 (April 1984).

[22] S. Goldwasser, S. Micali and C. Rackoff, "The knowledge complexity of interactive proof systems," *SIAM J. of Comp.*, Vol. 18, No. 1, pp. 186–208, February 1989.

[23] S. Goldwasser, S. Micali and R. Rivest, "A digital signature scheme secure against adaptive chosen-message attacks," *SIAM Journal of Computing,* Vol. 17, No. 2, 281–308, April 1988.

[24] J. Håstad, "Pseudo-random generators under uniform assumptions," *Proceedings of the Twenty Second Annual Symposium on the Theory of Computing,* ACM, 1990.

[25] R. Impagliazzo and M. Luby, "One-way functions are essential for complexity based cryptography," *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science,* IEEE, 1989.

[26] R. Impagliazzo, L. Levin and M. Luby, "Pseudo-random generation from one-way functions," *Proceedings of the Twenty First Annual Symposium on the Theory of Computing,* ACM, 1989.

[27] ISO/IEC 9798-2, "Information technology – Security techniques – Entity authentication – Part 2: Entity authentication using symmetric techniques." Draft 12, September 1992.

[28] M. Luby and C. Rackoff, "How to construct pseudorandom permutations from pseudorandom functions," *SIAM J. Computing,* Vol. 17, No. 2, April 1988.

[29] M. Luby and C. Rackoff, "A study of password security," manuscript.

[30] R. Molva, G. Tsudik, E. Van Herreweghen and S. Zatti, "*Kryptoknight* authentication and key distribution system, ESORICS 92, Toulouse, France, November 1992.

[31] R. Needham and M. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM*, Vol. 21, No. 12, 993–999, December 1978.

[32] R. Rivest, "The MD5 message-digest algorithm," IETF Network Working Group, RFC 1321, April 1992.

[33] J. Rompel, "One-way functions are necessary and sufficient for secure signatures," *Proceedings of the Twenty Second Annual Symposium on the Theory of Computing,* ACM, 1990.

[34] K. Sakurai and T. Itoh, "On the discrepancy between serial and parallel of zero-knowledge protocols," *Advances in Cryptology – Crypto 92 Proceedings,* Lecture Notes in Computer Science Vol. 740, Springer-Verlag, E. Brickell, ed., 1993.

[35] M. Tompa and H. Woll, "Random self-reducibility and zero-knowledge interactive proofs of possession of information," University of California (San Diego) Computer Science and Engineering Dept. Technical Report Number CS92-244 (June 1992). (Preliminary version in *Proceedings of the Twenty Eighth Annual Symposium on the Foundations of Computer Science,* IEEE, 1987.)

[36] G. Tsudik, "Message authentication with one-way hash functions," *Proceedings of Infocom 92.*

[37] T. Woo and S. Lam, "A semantic model for authentication protocols," *Proceedings 1993 IEEE Symposium on Research in Security and Privacy,* pp. 178–195, May 1993.

[38] A. Yao, "Protocols for secure computation," *Proceedings of the Twenty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1982.

[39] Yao, A. C., "Theory and applications of trapdoor functions," *Proceedings of the Twenty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1982.

A Proof of Theorem 4.4

We prove that MAP1 is a secure mutual authentication protocol under the assumption that f is a PRF. The first condition of Definition 4.2 is easily verified; it merely says that when the messages between A and B are faithfully relayed to one another, each party accepts. We now prove that the second condition holds.

Fix an adversary E. Recall that the domain of our PRF is $\{0,1\}^{\leq L(k)}$ and its range is $\{0,1\}^k$. In the following, Π will denote MAP1. In what follows we will be considering a variety of experiments involving the running of E with its oracles. In order to avoid confusion, we will refer to the experiment of running E with MAP1 (the experiment about which we wish to prove our theorem) as the "real" experiment.

MAP1 With A g Oracle. Let g be a function of $\{0,1\}^{\leq L(k)}$ to $\{0,1\}^k$. Let $[x]_g = (x, g(x))$. MAP1^g denotes the protocol in which, instead of a shared secret a, the parties share an oracle for g, and they compute $[x]_g$ wherever MAP1 asks them to compute $[x]_a$. We define the experiment of running E for MAP1^g to be the same as the experiment of running E for MAP1 except for the following difference. There is no shared secret a; instead, the oracles $\Pi^s_{i,j}$ all have access to a common g oracle and compute their flows according to MAP1^g. Note E is not given access to the g oracle. When $g = f_a$ for randomly chosen a, this experiment coincides with the real experiment. Of interest in our proof is the case of g being a truly random function; we call this the random MAP1 experiment.

The Random MAP1 Experiment. In the random MAP1 experiment we select g as a random function of $\{0,1\}^{\leq L(k)}$ to $\{0,1\}^k$, and then run the experiment of running E with MAP1^g. Recall that $\mathsf{No\text{-}Matching}^E(k)$ denotes the event that there exists an oracle $\Pi^s_{i,j}$ who accepts although no oracle $\Pi^t_{j,i}$ engaged in a matching conversation; we will refer to it also as the event that the adversary is successful. Recall that an initiator oracle is one who sends a first flow (that is, it plays the role of A in Figure 2) while a responder oracle is one who plays the opposite role (namely that of B in the same Figure). Let $T_E(k)$ denote a polynomial bound on the number of oracle calls made by E, and assume wlog that this is at least two.

Lemma A.1 *The probability that the adversary E is successful in the random* MAP1 *experiment is at most $T_E(k)^2 \cdot 2^{-k}$.*

Proof: We split the examination of acceptance into two cases.

Claim 1: Fix A, B, s. The probability that $\Pi^s_{A,B}$ accepts without a matching conversation, given that it is an initiator oracle, is at most $T_E(k) \cdot 2^{-k}$.

Proof. Suppose at time τ_0 oracle $\Pi^s_{A,B}$ sent the flow R_A. Let

$$\mathcal{R}(\tau_0) = \{\, R'_A \in \{0,1\}^k \;:\; \exists \tau, t \text{ such that } \Pi^t_{B,A} \text{ was given } R'_A \text{ as first flow at a time } \tau < \tau_0. \,\}\ .$$

If $\Pi^s_{A,B}$ is to accept, then at some time $\tau_2 > \tau_0$ it must receive $[B\,.\,A\,.\,R_A\,.\,R_B]_g$ for some R_B. If no oracle previously output this flow, the probability that the adversary can compute it correctly is at most 2^{-k}. So consider the case where some oracle did output this flow. The form of the flow implies that the oracle which output it must be a $\Pi^t_{B,A}$ oracle which received R_A as its own first flow. The probability of this event happening before time τ_0 is bounded by the probability that

$R_A \in \mathcal{R}(\tau_0)$, and this probability is at most $[T_E(k)-1]\cdot 2^{-k}$. If it happened after time τ_0 then we would have a matching conversation. We conclude that the probability that $\Pi^s_{A,B}$ accepts but there is no matching conversation is at most $T_E(k)\cdot 2^{-k}$. □

Claim 2: Fix B, A, t. The probability that $\Pi^t_{B,A}$ accepts without a matching conversation, given that it is a responder oracle, is at most $T_E(k)\cdot 2^{-k}$.

Proof. Suppose at time τ_1 oracle $\Pi^t_{B,A}$ received the flow R_A and responded with $[B\,.\,A\,.\,R_A\,.\,R_B]_g$. If $\Pi^t_{B,A}$ is to accept, then at some time $\tau_3 > \tau_1$ it must receive $[A\,.\,R_B]_g$. If no oracle previously output this flow, the probability that the adversary can compute it correctly is at most 2^{-k}. We must now consider the case where some oracle did output this flow. The form of the flow implies that the oracle which output it must be a $\Pi^s_{A,C}$ oracle.

The interaction of a $\Pi^s_{A,C}$ oracle with E has in general the form

$$(\tau_0, \lambda, R'_A), (\tau_2, [C\,.\,A\,.\,R'_A\,.\,R'_B]_g, [A\,.\,R'_B]_g)$$

for some $\tau_0 < \tau_2$. For any such interaction, except with probability 2^{-k}, there is a $\Pi^u_{C,A}$ oracle which output $[C\,.\,A\,.\,R'_A\,.\,R'_B]_g$ at some time. If $(u, C) \neq (t, B)$ then the probability that $R'_B = R_B$ is at most $[T_E(k)-2]\cdot 2^{-k}$, and thus the probability that the flow $[A\,.\,R'_B]_g$ leads $\Pi^t_{B,A}$ to accept is at most $[T_E(k)-2]\cdot 2^{-k}$. On the other hand suppose $(u, C) = (t, B)$. It follows that $\tau_0 < \tau_1 < \tau_2 < \tau_3$, $R'_A = R_A$ and $R'_B = R_B$; that is, the conversations match. We conclude that the probability that $\Pi^t_{B,A}$ accepts but there is no matching conversation is at most $T_E(k)\cdot 2^{-k}$. □

The probability that there exists an oracle which accepts without a matching conversation is at most $T_E(k)$ times the bound obtained in the claims, which is $T_E(k)^2\cdot 2^{-k}$ as desired. ■

The Proof Concluded. To conclude the proof, we argue by contradiction. Suppose that the probability that adversary E is successful in the real experiment is not negligible. Then there is an infinite set K and a constant c such that for all $k \in K$ the probability of $\mathsf{No\text{-}Matching}(k)$ in the real experiment is at least k^{-c}. We will now construct a polynomial time test T which distinguishes random functions from pseudo-random functions. T receives as an oracle a function g: $\{0,1\}^{\leq L(k)} \to \{0,1\}^k$ which is chosen according to the following experiment: flip a coin C, and if $C = 1$ let g be a random function, else pick a at random and let $g = f_a$. T's job is to predict C with some advantage. T's strategy is to run the experiment of running E for MAP1^g. In this experiment, T itself simulates all oracles $\Pi^s_{i,j}$, answering E by running the protocol MAP1^g in the manner of these oracles. If E is successful (note that T can tell whether or not E succeeded) then T predicts 0 (g is pseudo-random) else T predicts 1 (g is random). Lemma A.1 and our assumption about the success of E when $g = f_a$ (the real experiment) imply that T has advantage k^{-d}, for some $d > 0$ and all $k \in K$, contradicting the pseudo-randomness of f.

B Proof of Theorem 5.2

We prove that AKEP1 is a secure authenticated key exchange protocol under the assumption that f, f' are PRFs.

The proof that AKEP1 is a secure mutual authentication protocol is analogous to the proof of Theorem 4.4 given in Appendix A and is omitted. Condition (1) of Definition 5.1 is easily verified:

the session key α is chosen in AKEP1 according to S_k and so in the presence of a benign adversary the oracles certainly accept, and with this same key. We concentrate on the proof that condition (2) of Definition 5.1 is satisfied.

Fix an adversary E. Recall that we are using two PRFs: f_{a_1}: $\{0,1\}^{\leq L(k)} \rightarrow \{0,1\}^k$ and f'_{a_2}: $\{0,1\}^k \rightarrow \{0,1\}^{\sigma(k)}$. The first is for the authentication and the second is to encrypt the session key. In what follows Π will denote AKEP1, and the "real" experiment will denote the experiment of running E for AKEP1.

AKEP1 With A g' Oracle. Let g' be a function mapping : $\{0,1\}^k$ to $\{0,1\}^{\sigma(k)}$. Let $\mathcal{E}_{g'}(\alpha, r) = (r, g'(r) \oplus \alpha)$. Let $\{\alpha\}_{g'}$ be the random variable resulting from picking $r \in \{0,1\}^k$ at random and outputting $\mathcal{E}_{g'}(\alpha, r)$. AKEP1$^{g'}$ denotes the protocol in which the parties share a secret a_1 and an oracle for g'. Whenever AKEP1 asks them to compute $\{\alpha\}_{a_2}$ they compute $\{\alpha\}_{g'}$. The experiment of running E for AKEP1$^{g'}$ is the same as the experiment of running E for AKEP1 except that the second part of the shared key, namely a_2, is absent, and instead the oracles $\Pi^s_{i,j}$ all have access to a common g' oracle and compute their flows according to AKEP1$^{g'}$. E does not have access to g'. When $g' = f'_{a_2}$ for randomly chosen a_2, this experiment coincides with the real experiment.

The Random AKEP1 Experiment. In the random AKEP1 experiment we select g' as a random function of $\{0,1\}^k$ to $\{0,1\}^{\sigma(k)}$, and then run the experiment of running E with AKEP1$^{g'}$. As before, let $T_E(k)$ denote a polynomial bound on the number of oracle calls made by E.

Lemma B.1 *In the random AKEP1 experiment, advantage$^E(k)$ is negligible.*

Proof: Let $c > 0$ be a constant. We will show that *advantage*$^E(k) \leq k^{-c}$ for all sufficiently large k.

A *view* of E consists of all the oracle queries made by E, the responses to them, and E's own coin tosses; that is precisely what E sees. We denote by *view*(k) the random variable whose value is the view of the interaction of E with its oracles. A particular view will usually be denoted ξ. We will be interested in two properties ξ may possess. If for any accepting oracle there exists an oracle with a matching conversation then we say ξ is *authentic*. If $(r_1, y_1), \ldots, (r_n, y_n)$ denote the encryptions output by oracles in the transcript and $r_1, \ldots, r_n$ are distinct then we say ξ is *non-colliding*. Recall that b denotes the bit flipped in our answer to a test query in the definition of measuring *advantage*$^E(k)$.

Now fix a particular authentic and non-colliding view ξ. Suppose E is pointing to (fresh) oracle $\Pi^s_{A,B}$. Since $\Pi^s_{A,B}$ has accepted and ξ is authentic, there is an oracle $\Pi^t_{B,A}$ which engaged in a matching conversation. This means the encryption for this conversation was selected by one of the oracles (specifically, the one who played the role of the responder). The oracle's being fresh means that any matching partner is unopened. Since ξ is non-colliding it follows that conditioned on *view*$(k) = \xi$, the key $\alpha^s_{A,B}$ is uniformly distributed over S_k, and E's advantage in predicting the bit b is 0.

Let N_k denote the set of non-authentic views and C_k the set of colliding views. We claim that AKEP1$^{g'}$, with g' chosen at random, still remains a secure mutual authentication; the proof of this is analogous to the proof of Theorem 4.4 and hence is omitted. Based on this claim, we know that the probability of N_k is at most $k^{-c}/2$ for large enough k. On the other hand the probability of C_k is at most $T_E(k)^2 \cdot 2^{-k}$ which is at most $k^{-c}/2$ for large enough k. Combined with the above we conclude that E's advantage is at most k^{-c}. ■

THE PROOF CONCLUDED. We argue by contradiction. Suppose that the probability that $advantage^E(k)$ is not negligible in the real experiment. Then there is an infinite set K and a constant c such that for all $k \in K$ the advantage of E in the real experiment is at least k^{-c}. We will now construct a polynomial time test T which distinguishes random functions from pseudo-random functions. T receives as an oracle a function g': $\{0,1\}^k \to \{0,1\}^{\sigma(k)}$ which is chosen according to the following experiment: flip a coin C, and if $C = 1$ let g' be a random function, else pick a_2 at random and let $g' = f'_{a_2}$. T's job is to predict C with some advantage. T's strategy is to run the experiment of running E for AKEP1$^{g'}$. In this experiment, T itself simulates all oracles $\Pi^s_{i,j}$, answering E by running the protocol AKEP1$^{g'}$ in the manner of these oracles. In this process T will itself select all the session keys for the oracles. T can answer the reveal queries for oracles of which it selected they key; the probability that T must answer a reveal query for an oracle whose key T didn't select is negligible. Finally E outputs (i, j, s, test). If T had not herself chosen $\alpha^s_{i,j}$ in the execution then T outputs 0; this happens with negligible probability. Else T now flips a fair coin b. If $b = 0$ she returns $\alpha^s_{i,j}$, else she returns a random sample from S_k. Now E outputs $b' = \mathsf{Guess}^E(k)$. T outputs 0 (g' is pseudo-random) if $b' = b$ and 1 (g' is random) otherwise. Lemma B.1 and our assumption about the advantage of E when $g' = f'_{a_2}$ (the real experiment) imply that T has advantage k^{-d}, for some $d > 0$ and all $k \in K$, contradicting the pseudo-randomness of f'.

C Proof of Proposition 4.3

The proof is by contradiction. Let E be a polynomial time adversary such that the probability of $\mathsf{Multiple\text{-}Match}^E(k)$ is not negligible. We will show that Π is not a secure MA protocol. In what follows $\mathsf{Multiple\text{-}Match}^E_1(k)$ denotes the event that a responder oracle $\Pi^t_{j,i}$ accepts and there exist distinct (initiator) oracles $\Pi^s_{i,j}, \Pi^{s'}_{i,j}$ which have had a matching conversation with $\Pi^t_{j,i}$. $\mathsf{Multiple\text{-}Match}^E_2(k)$ denotes the event that an initiator oracle $\Pi^s_{i,j}$ accepts and there exist distinct (responder) oracles $\Pi^t_{j,i}, \Pi^{t'}_{j,i}$ which have had a matching conversation with $\Pi^s_{i,j}$. We split the proof into two cases.

Claim C.1 *Suppose the probability of* $\mathsf{Multiple\text{-}Match}^E_1(k)$ *is not negligible. Then* Π *is not a secure MA protocol.*

Proof: We begin with the following observation. If responder oracle $\Pi^t_{j,i}$ accepts and distinct (initiator) oracles $\Pi^s_{i,j}, \Pi^{s'}_{i,j}$ have had matching conversations with $\Pi^t_{j,i}$, then the first flow output by the oracles $\Pi^s_{i,j}, \Pi^{s'}_{i,j}$ is the same. Thus our assumption implies that there is an infinite set K and a constant c such that for all $k \in K$ the following is true: there exist $i, j \in I$ such that if initiator oracles $\Pi^1_{i,j}, \Pi^2_{i,j}$ are asked queries $(i, j, 1, \lambda)$ and $(i, j, 2, \lambda)$ respectively, then the probability that the responses are the same is at least k^{-c}. Based on this observation, we construct an adversary E' such that for all $k \in K$ the probability of $\mathsf{No\text{-}Matching}^{E'}(k)$ is at least $k^{-c}/|I|^2$, as follows.

E' picks $i, j \in I$ at random. She makes query $(i, j, 1, \lambda)$ and lets α^1_1 denote the response. Next she makes the query $(j, i, 1, \alpha^1_1)$ and lets β^1_1 denote the response; let τ_1 denote the time at which this happens. Next she makes the query $(i, j, 2, \lambda)$ and lets α^2_1 denote the response; let $\tau_2 > \tau_1$ denote the time at which this happens. If $\alpha^2_1 \neq \alpha^1_1$ then E' halts. Else E' makes the query $(i, j, 2, \beta^1_1)$. Beginning with the response to this query, E' faithfully relays messages between $\Pi^2_{i,j}$ and $\Pi^1_{j,i}$. One

can check that oracles $\Pi^2_{i,j}$ and $\Pi^1_{j,i}$ accept (while no other oracles do). However, they have *not* had matching conversations. The reason is that the query $(i, j, 2, \lambda)$ was made at time τ_2, but the string β^1_1 used as the next flow to $\Pi^2_{i,j}$ was obtained (albeit from $\Pi^1_{j,i}$) at an *earlier* time $\tau_1 < \tau_2$. Of course the reason it could serve as response is that $\alpha^1_1 = \alpha^2_1$. Our conclusion now follows from the fact that the probability that $\alpha^1_1 = \alpha^2_1$ is at least $k^{-c}/|I|^2$ for all $k \in K$, which concludes the proof. ■

Claim C.2 *Suppose the probability of* $\mathsf{Multiple\text{-}Match}^E_2(k)$ *is not negligible. Then* Π *is not a secure MA protocol.*

Proof: We begin with the following observation. If initiator oracle $\Pi^s_{j,i}$ accepts and distinct (responder) oracles $\Pi^t_{j,i}, \Pi^{t'}_{j,i}$ have had matching conversations with $\Pi^s_{i,j}$, then the first flow provided to the oracles $\Pi^s_{j,i}, \Pi^{s'}_{j,i}$ is the same, and the flows output by these oracles in response are also the same. This observation will be refined before we build the adversary E' defeating the mutual authentication.

Let $i, j \in I$ and consider the following experiment. Make query $(i, j, 1, \lambda)$ and let α^1_1 denote the response. Now make queries $(j, i, 1, \alpha^1_1)$ and $(j, i, 2, \alpha^1_1)$ and let β^1_1, β^2_1 denote the responses respectively. Let $p_{i,j}(k)$ be the probability that $\beta^1_1 = \beta^2_1$. Then our assumption implies that there is an infinite set K and a constant c such that for all $k \in K$ there exist $i, j \in I$ such that $p_{i,j}(k) \geq k^{-c}$.

E' picks $i, j \in I$ at random. She then makes query $(i, j, 1, \lambda)$ and lets α^1_1 denote the response. Next she makes query $(j, i, 1, \alpha^1_1)$ and lets β^1_1 denote the response. Next she makes query $(i, j, 1, \beta^1_1)$ and lets α^1_2 denote the response; let τ_1 denote the time at which this happens. Now she makes query $(j, i, 2, \alpha^1_1)$ and lets β^2_1 denote the response; let τ_2 denote the time at which this happens. If $\beta^2_1 \neq \beta^1_1$ then E' halts. Else E' makes the query $(j, i, 2, \alpha^1_2)$. Beginning with the response to this query, E' faithfully relays messages between $\Pi^1_{i,j}$ and $\Pi^2_{j,i}$. One can check that oracles $\Pi^1_{i,j}$ and $\Pi^2_{j,i}$ accept (while no other oracles do). However, they have *not* had matching conversations. The reason is that the query $(j, i, 2, \alpha^1_1)$ was made at time τ_2, but the string α^2_1 used as the next flow to $\Pi^2_{j,i}$ was obtained (albeit from $\Pi^1_{i,j}$) at an *earlier* time $\tau_1 < \tau_2$. Of course the reason it could serve as response is that $\beta^1_1 = \beta^2_1$. Our conclusion now follows from the fact that the probability that $\beta^1_1 = \beta^2_1$ is at least $p_{i,j}(k)/|I|^2 \geq k^{-c}/|I|^2$ for all $k \in K$, which concludes the proof. ■

We note that this proof exploited the fact that the number of moves in a mutual authentication protocol is at least 3.

D Two Lemmas for Pseudo-Random Function Construction

We summarize here the lemmas justifying the constructions of PRFs given in Section 6. These lemmas are from [2].

We recall the problem. We are given a PRF g_a of l-bits to l-bits. We want a PRF which takes longer inputs to l-bit outputs. In what follows m will denote an upper bound on the desired input length. Formally $l = l(k)$ and $m = m(k)$ are both polynomially bounded functions of the security parameter k, and the PRFs are members of parameterized families of functions etc. Discussions here, however, will be informal. We will state things in terms of specific functions and omit mention

of k; the formal statements of the lemmas are easily reconstructed. We refer the reader to Section 6 for the definition of CBC_a^g and $\langle x \rangle_l$.

Lemma D.1 (CBC Lemma, [2]) *Suppose* $g_a\colon \{0,1\}^l \to \{0,1\}^l$ *is a PRF. Then, for any polynomially bounded* m, $f_a(x) = \mathrm{CBC}_a^g(\langle x \rangle_l)$ *is a PRF from* $\{0,1\}^m$ *to* $\{0,1\}^l$. *Also,* $f_a^*(x) = \mathrm{CBC}_a^g(\langle x \rangle_l \,.\, |\langle x \rangle_l|)$ *is a PRF from* $\{0,1\}^{\leq m}$ *to* $\{0,1\}^l$.

Although cipher block chaining is used in many places, Lemma D.1 provides the first justification of its use as a PRF (or MAC) which depends *only* on the security of the underlying function.

A hash function H specifies for each k a map $H_k\colon \{0,1\}^* \to \{0,1\}^{h(k)}$. A collision seeking algorithm is a polynomial time algorithm which on input 1^k outputs a pair of distinct strings $x, y \in \{0,1\}^*$; we say the algorithm is successful if $H_k(x) = H_k(y)$. We say that H is collision free if every collision seeking algorithm has negligible success probability. In what follows we will as usual shove the asymptotics under the rug and regard a collision free hash function as $H\colon \{0,1\}^* \to \{0,1\}^h$. Under the assumption that g_a is a PRF and H is collision free, the CBC/Hash PRF is provably secure. The following lemma is the first step.

Lemma D.2 (Hash Lemma, [2]) *Suppose* $H\colon \{0,1\}^* \to \{0,1\}^h$ *is a collision free hash function and* $g_a\colon \{0,1\}^h \to \{0,1\}^l$ *is a PRF. Then* $f_a\colon \{0,1\}^{\leq m} \to \{0,1\}^l$ *defined by* $f_a(x) = g_a(H(x))$ *is a PRF.*

Since $l = 64$ (DES) and $h = 128$ (MD5) in our applications, the following simple corollary of the above lemmas is worth stating. For completeness we give the simple proof. Assuming $h = 2l$, let $H_1(x), H_2(x)$ denote the first l bits and the last l bits of $H(x)$, respectively.

Corollary D.3 ([2]) *Let* $H\colon \{0,1\}^* \to \{0,1\}^{2l}$ *be a collision free hash function, and let* $g_a\colon \{0,1\}^l \to \{0,1\}^l$ *be a PRF. Then* $f_a\colon \{0,1\}^{\leq m} \to \{0,1\}^l$ *defined by* $f_a(x) = g_a(\, g_a(H_1(x)) \oplus H_2(x)\,)$ *is a PRF.*

Proof: Regarding a $2l$ bit input y as a pair of l bit blocks, $y = y_1 y_2$, let $g'_a(y) = \mathrm{CBC}_a^g(y) = g_a(g_a(y_1) \oplus y_2)$. Lemma D.1 says that $g'_a\colon \{0,1\}^{2l} \to \{0,1\}^l$ is a PRF. But $f_a(x) = g'_a(H(x))$. So f_a is a PRF by Lemma D.2. ■