Secure Protocol Transformation via "Expansion": From Two-party to Groups

Alain Mayer
Bell Labs, Lucent Technologies
alain@research.bell-labs.com

Moti Yung
CertCo Corp.
moti@certco.com, moti@cs.columbia.edu

Abstract

The design of simple cryptographic protocols for elementary two-party (session oriented) tasks (such as *entity authentication* and *key transport*) has had a history (starting with [NS78]) where security has been quite evasive. Only recently we have seen protocol designs which are both *provably* secure and efficient

Currently, much attention of the designers of network systems and services is directed towards *group* operations, which will enable such important tasks as one-to-many distribution of content, group collaborative efforts, etc over the Internet and Intranets [Be98]. Rather than designing each group oriented task from scratch, we move in this work towards a more methodological approach, which derives a design of group (multicast) protocols from two-party ones. The approach, which we call *secure protocol expansion*, maintains the efficiency of the basic design and at the same time preserves provable security. It enables us to achieve efficient and secure protocols for a large variety of group tasks. We consider basic group authentication and key transport protocols, as well as functional protocol extensions like multicast perfect forward secrecy, group access-control, group announcement and termination.

Key words: protocol design, protocol transformation, secure group protocols, complexity theoretic proofs, authentication, key transport, forward secrecy.

1 Introduction

The design of efficient cryptographic protocols for the basic tasks of entity authentication and key transport had a long and troubled history of flawed and inadequate solutions. For example, fundamental "interleaving attacks" have not been recognized in their full generality until the KryptoKnight project (see e g. [B-al-91, M-al-92, TvH93, B-al-95, JTY97]). Consequently, KryptoKnight presented new approaches for authentication and key distributions taking such attacks into account and gave modular extensions to build a three-party server protocol from two-party one Bellare and Rogaway demonstrated in [BR93, BR95], using symmetric cryptosystems, a *provably secure* protocol for two-party entity authentication and authenticated key transport, respectively. Subsequently, Blake-Wilson and Menezes (see [BM97]) built on this work to design protocols for the same two tasks in the asymmetric (public key) case.

The importance of "complexity-theoretic secure solutions" is convincingly argued in the papers above However, there is no notion of robust modifications and variations of these basic protocols (of [BR93, BR95, BM97]). Further, it appears that even the smallest modifications to such protocols can invalidate their provable security, which explains the difficulty of the task at hand This observation also implies that the presented protocols should be considered "take it or leave it". There are, however, a variety of (real-life) scenarios, where the requirements for a protocol incrementally differ form the model of [BR93, BR95, BM97]. For example in multiparty communication, a group leader and a group member play different roles and thus have different capabilities A two-party interaction between the leader and a member (e.g., for key transport) is among non-symmetric partners. Thus, we might require that the leader generates the key and at the same time is the recipient of the last message of the prototol to detect unsuccessful termination. Currently, there is no alternative in such a situation to designing (and proving') a protocol from scratch, even when the end result is very similar to the solutions of [BR93, BR95, BM97]. One such example was the effort to design (and prove secure in this model) a key distribution

CCS '99 11/99 Singapore

with smart cards in [SR96], who based their design on a modified [LM93].

Our goal is to work towards a remedy for this situation by introducing instances of **secure protocol expansions**. Such transformations allow a protocol designer to systematically leverage the basic work of [BR93, BR95, BM97] to obtain a customized solution. This direction is influenced by the "modular and scalable" approach of the work on KryptoKnight which took an entity authentication protocol, extended it to a two-party key exchange of various flavors, and further extended this design to the "Needham-Schroeder" three party model.

Our transformations expand the functionality of the protocol by incremental requirements that either increase the number of parties, vary the parties' functionality, or add properties to the specified protocol. In this way, we derive increasingly more specialized protocols, customized to actual system requirements. Intuitively, our "protocol expansion approach" is to construct a new protocol P_2 via a transformation from a protocol P_1 which is already proven secure and then show that any adversary which can break P_2 can also break P_1. Indeed, the incremental expansion by small but useful steps and the proof methodology, is what typifies our transformations (rather than describing them by a formal definition, which seems hard to do). Basically, each step adds a communication step to the basic protocol, or aggregates messages based on the basic protocol. As a result, the initial structure of the basic protocol P_1 is embedded inside P_2 in certain ways (e.g., as a substructure or as an aggregated structure).

The goal of our method is primarily to achieve secure *multicasting* protocols (though it probably can be used elsewhere) The *multicast key transport problem* is stated as follows one entity (leader) wishes to select keying information and communicate it in secret to a group of other entities (members) over a distributed network If each member also desires an assurance of the leader's identity (and vice versa), this is known as *authenticated multicast key transport.* Note that even if this task is implemented as a sequence of two-party protocols (e.g., whenever a new party wants to join the multicast group), it is now executed among two unequal parties and thus new requirements may apply. A closely related problem is *multicast authentication.* Here, the leader merely desires an assurance of the members' identity (or vice versa). Based on these two basic primitives, fundamental tasks which add more functionality, such as *group access control*, *group announcement* and *termination*, and *group-key and session-key management* can be performed within the group Also, *perfect forward secrecy* (see [DOW92]) is typical additional requirement in this setting.

Our Contributions.

1 Define the security goals for multicast authentication, authenticated multicast key transport, and more complex tasks, such as group access control and forward secret group session-key distribution.
2. Construct some basic *provably secure "protocol expansions"* general enough to cover both the symmetric key and asymmetric key cases: (i) transform a two-party auth/key-transport n flow protocol into a two-party $n+1$ flow protocol (as explained later, such a transformation is needed to obtain protocols meeting additional consistency requirements); (ii) transform a two-party auth/key-transport protocol into corresponding multicast protocols; (iii) embed basic multicast protocols in group management routines to enhance the protocol functionality.
3. Construct a provably secure multicast session-key distribution protocol with forward secrecy out of basic secure multicast protocols

Multicast Background: Many secure group communication protocols have been proposed in the literature from small group collaboration (e.g., [G97]) to Internet wide IP-multicast (see [C+99] for a taxonomy of multicasting protocols). [BC95] argue that multicast communication is inherently more susceptible to security attacks than uni-cast. Multicast security is indeed one of the current research interests of the Internet community ([Be98]) The type of transformations we introduce can yield a basic "arsenal" of group-protocols which are flexible so that they can be easily matched onto different group structures and network architectures, giving solutions which are essentially as efficient as the ones proposed in the literature (e.g., [BC95, B96, M97, AMP96, JKKO94, G94, G97]) and are, in addition, provably secure Both the efficiency and flexibility are implied by the design methodology (starting from two party protocols and transform incrementally using simple expansions) We note that here, we do not consider (yet) Diffie-Hellman based approaches to group communication, for which a large body of work exists as well (see, e g., [STW96, BW98, AST98]).

Organization of the rest of the paper: In Section 2 we review and unify the two-party definitions and security models developed in [BR93, BR95, BM97]. Section 3 introduces a simple syntax to express our transformations. Our first transformation in Section 4 shows how to add a consistency requirement to key transport to ensure that a joining group member is not left in an inconsistent state with respect to the leader Sections 5 and 6 give transformations for multicast authentication and key transport. In Section 7 we introduce multicast with perfect forward secrecy Section 8 ties all the concepts together in the design of group management. Section 10 contains a few concrete protocols obtained by applying our proposed transformations All our proofs are given only as sketches of ideas rather than being formal (however they provide enough intuition to explain our claims and the essence that will guide the formalism)

2 Definitions and Model for Basic Two-Party Protocols

In the following, we present a unification and overview of the definitions and models developed for symmetric key and asymmetric key cases resp. in [BR93, BR95] and in [BM97] This unified model serves as our base for expansion This model allows the adversary to control the message schedule and interleaving of sessions among parties, it is a natural model for open networks where entities may engage in concurrent activities (e.g multiple authentication sessions). This was first pointed out in [B-al-91] and further developed in [DOW92, BR93]. The adversary controls which parties to corrupt and its actions are modeled by the queries it asks The security is formulated with respect to the non-corrupted parties Let us review the model at some formal level, for more details see the above mentioned papers

A *protocol* is implemented by a function Π:
$\Pi(1^k, A, B, K_{A,B}, c, \rho) = ((m, \sigma), \delta, \kappa)$
where k is the security parameter, A is identity of sender; B is identity of intended partner; $K_{A,B}$ is secret keying information of the sender (long-lived symmetric key with intended partner or private key). We assume that every party, including E, has access to any public keying information of all parties involved, c is the conversation so far; ρ are (possible) random coins, (m, σ) the next message sent by A to B with its signature if an asymmetric solution is used, δ is A's current decision to accept the outcome of the protocol; κ the exchanged key. A *key generator* $\mathcal{G}$ is associated with a protocol and generates the appropriate keying material This can be either a long lived symmetric key shared by two parties or public/private key portions for any given party's public key In the latter case, $\mathcal{G}$ also forms a directory *public info* containing the public keying information for each party

The *adversary* E is a probabilistic Turing machine, which takes *public info* as input (if any) and which can make *queries* to $\Pi^s_{A,B}$, which is the oracle for protocol of A attempting to talk to B in session s (the sth protocol run between A and B) The queries are summarized in Figure 1 The send-query indicates that E is sending message (m, σ) to A, claiming it is from B in session s. The reveal-query gives E the session key of s and the corrupt-query gives E the long-term secrets of A. A ***benign adversary*** faithfully executes the send-queries according to the protocol. Other restricted adversaries are possible, though not needed in the paper.

Different suggested two-party protocols use different cryptographic primitives for encryption and message authentication. Some of our transformations employ in addition *pseudo-random functions* [GGM] as a primitive, denoted $PRF_k(x)$.

A *conversation* of $\Pi^s_{A,B}$ is the sequence
$C = (t_1, \alpha_1, \beta_1)(t_2, \alpha_2, \beta_2), \ldots$,
where at time t_i the oracle was asked α_i and responded with β_i, etc

Definition 1 *(Matching Conversation, from [BR93]) Consider a protocol with R moves. Let $R = 2r - 1$. We say that C' is matching to C if C is prefixed by $(t_0, \lambda, \alpha_1)(t_2, \beta_1, \alpha_2) \ldots (t_{2r-1}, \beta_{r-1}, \alpha_r)$ and C' is prefixed by $(t_1, \alpha_1, \beta_1), (t_3, \alpha_2, \beta_2) \ldots (t_{2r-3}, \alpha_{r-1}, \beta_{r-1})$ and analogously for C matching to C'.*

Similarly, we say that C' is matching to C including signatures, if each message is signed and the signatures match as well in the conversation.

2.1 Mutual Authentication (MA) & Authenticated Key Transport (AKT)

Let *No-match*$^E(k)$ be the event in which an *non-corrupt oracle* accepts the outcome of the protocol against adversary E without the existence of another *non-corrupt* corresponding oracle that had a matching conversation Note that signatures are not included to allow E to replace a signature in a flow by a different, but also valid signature (see [BM97]).

Definition 2 *(Mutual authentication $MA_{A,B}$, from [BR93, BM97])*
(1) $\Pi^s_{A,B}$ and $\Pi^s_{B,A}$ have matching conversation (with signatures) $\Rightarrow$ both accept. (2) P(no-match$^E(k)$) is negligible.

When adversary E issues a reveal-query to $\Pi^s_{A,B}$, we say that $\Pi^s_{A,B}$ is *opened*. $\Pi^s_{A,B}$ is *fresh* if 1) it has accepted, 2) is unopened and non-corrupt, and 3) there is no opened or corrupted oracle engaging in a matching conversation with $\Pi^s_{A,B}$

Definition 3 *(Authenticated Key Transport $AKT_{A,B}$, from [BR93, BM97])*
(1) Π is secure mutual authentication protocol. (2) In presence of a benign adversary E, both $\Pi^s_{A,B}$ and $\Pi^s_{B,A}$ accepts the same key, chosen according to a predefined distribution. (3) E cannot distinguish an accepted key at a fresh oracle from a random value with non-negligible probability.

Query	Oracle reply	Oracle update
Send$(A, B, s, (m, \sigma))$	$\Pi^{(m,\sigma),\delta}(1^k, A, B, K, c^s_{A,B}\ \ (m, \sigma))$	$c^s_{A,B} \leftarrow c^s_{A,B} \cdot (m, \sigma)$
Reveal(A, B, s)	$\Pi^{\kappa}(1^k, A, B, K, c^s_{A,B} \cdot (m, \sigma))$	none
Corrupt$(A, \vec{K}')$	$\vec{K} = \{K_{A,B_i}\}$	$\vec{K} \leftarrow \vec{K}'$

Figure 1. Adversary's queries

3 Message-Format Syntax for Transformations

We introduce a message format syntax to help us formulate our protocol transformations.

For our protocol transformations, we define a message $m_{A,B}$ from $\Pi^s_{A,B}$ to $\Pi^s_{B,A}$ as being grouped into the following fields: A is the sender identification, B is the receiver identification, d^s_A is data (pay-load) created by A for session s, and d^s_B is data created by B and echoed in A's message. Let r^s_A denote a *nonce* created by A. Let e denote encryption, let a denote message authentication. Some of these fields can be empty We also subsume the nonce into the payload if no explicit transformation on the nonces is required. Depending on the basic assumption of the protocol (i e., symmetric or asymmetric cryptography), encryption and authentication in $m_{A,B}$ is done either via a shared key between A and B or with the public key of B and the private key of A respectively Hence we can describe $m_{A,B}$ in the symmetric case as $m = (B, A, r^s_A, d^s_A, d^s_B, e_{A,B}(d^s_A, d^s_B), a_{A,B}(.))$, where $a(.)$ denotes authentication applied to the preceding part of the message. If asymmetric key cryptography is used, then we have:
$m = (B, A, r^s_A, d^s_A, d^s_B, e_B(d^s_A, d^s_B), a_A(.))$.

4 Transformation to Assure Group Member Consistency

While the requirements for $\mathrm{AKT}_{A,B}$ of [BR93] are certainly necessary, they might not be sufficient for specific applications, such as multicasting. For instance, the adversary can cause any of the parties not to accept and the other to accept. In this section we show how to ensure that a joining group member is not left in an inconsistent state with respect to the group leader We present possible additional consistency requirements

Definition 4 *(Consistency for AKT)*

1. *B has accepted (holding the session-key) $\Rightarrow$ A has accepted.*
2. *A has accepted $\Rightarrow$ B has accepted (holding the session-key).*

In a multicast environment, A is typically the group-leader and B is a (prospective) member of A's group. A group-leader distributes a group key to all (authorized) parties requesting membership in the multicast group, possibly by executing a separate AKT_{A,B_i} for all requesting parties B_i A group-leader also manages a list of current group members. The first requirement above guarantees that when a member accepts, it is indeed included on the leader's group-list, while the second requirement ensures that a member being included on the leader's list implies that the member has accepted Adding both requirements implies that the AKT protocol has to solve consensus, an impossible task in the presence of our strong adversary E (see [FLP85]). Still, all correct and minimal (i.e., a party cannot accept before the receipt of the last message sent to it) protocols fulfill either the first or the second requirement

Lemma 5 *Any correct minimal AKT protocol fulfills consistency requirement 1 (2) if and only if the last message is sent by the leader A (member B).*

Proof: (idea)· Assume that the last message is sent by the prospective group member. The prospective group member must have already accepted at the time it sent the message. The leader cannot accept before the receipt of this message, since otherwise we could design a protocol with the same behavior with one less round of messages. □

Now, let us consider the case where E manages to terminate AKT in a state where either a member B or the leader A has not accepted. If the member has not accepted, it can simply restart the AKT protocol. If a prospective member B has accepted, but not the leader, then we must decide whose responsibility it is to restart the AKT. It is likely that at that point, the leader does not have proof of B's authenticity Hence, if we put the responsibility onto the leader, we further increase the load and open up vulnerabilities to denial of service attacks against the leader. The only other option in this case is to have B time-out after it accepted and has not received any group-messages. Given these considerations, it is advantageous using an AKT protocol guaranteeing Consistency Requirement 1 Indeed, many of the actual protocols (e g , [G97]) are designed this way. We now show how to transform any secure protocol with Consistency Requirement 2 (such as the [BR93] protocol) into a secure protocol with Consistency Requirement 1. N denotes the number of messages exchanged in AKT An example of a concrete application of this transformation can be found in Section 10. Let the resulting protocol be denoted by AKT*

- Construct $\mathrm{AKT}^s_{A,B}$ and $\mathrm{AKT}^{s'}_{A,B}$ ($s' \neq s$, e.g., $s' = s+1$) $\mathrm{AKT}^{s'}$ transports a dummy key, e.g., a null string
- For all messages in rounds #i ($3 \leq i \leq N$) and session s. $m^i_{A,B} = (B, A, d^s_A, d^s_B, a(.))^i$ is replaced by $m = (B, A, (d^s_A, d^s_B)^i, (d^{s'}_A, d^{s'}_B)^{i-2}, a(\,))$ and similarly for $m_{B,A}$
- A new message #$(N+1)$ from A to B is created, which is taken as the message #$(N-1)$ from A to B in $\mathrm{AKT}^{s'}_{A,B}$. (Data which does not influence B's decision can be omitted)

Theorem 6 *AKT* is a secure authenticated key transport protocol.*

Proof: (idea): We first show that AKT* is a secure MA. We assume that E break MA against $\mathrm{AKT}^*_{A,B}$, and show that in this case E is also successful against $\mathrm{AKT}_{A,B}$: Consider the first flow accepted by the receiver, in a conversation of $\mathrm{AKT}^*_{A,B}$ which distinguishes it from a matching conversation Since $\mathrm{AKT}^s_{A,B}$ and $\mathrm{AKT}^{s'}_{A,B}$ are both secure MA when executed alone, and since both $\Pi^{s^*}_{A,B}$ and $\Pi^{s^*}_{B,A}$ are fresh, E must have combined some information of the flows of these two sessions in order to compute the distinguishing flow. But in this case, E can simply observe the flows of $AKT^s_{A,B}$ to break MA against $AKT^{s'}_{A,B}$.

It is straightforward that in the presence of a benign adversary, both parties always accept the same key, transported in the $\mathrm{AKT}^s_{A,B}$-part of AKT* We are left to show that E cannot guess the resulting key at a fresh party· Since $AKT^s_{A,B}$ is secure and since $AKT^{s'}_{A,B}$ does not use any data related to the resulting key, E obtaining non-negligilbe information on the key is equivalent to breaking $\mathrm{AKT}^s_{A,B}$. □

5 From Authentication to Multicast Authentication

In this section, we show a secure protocol expansion, transforming a two-party MA into a multicast authentication protocol: A single party A wants to authenticate n distinct parties $\vec{B} = \{B_1, B_2, \dots B_n\}$ at the same time. A might communicate via a single multicast message or by many one-to-one messages. Let *No-match*$^E_i(k)$ denote the previously defined event for oracles $\Pi^s_{A,\vec{B}}, \Pi^s_{B_i,A}$.

Definition 7 *(Multicast Authentication)* *(1) $\forall i$: $(\Pi^s_{A,\vec{B}}, \Pi^s_{B_i,A})$ have matching conversation (with signatures) $\Rightarrow$ both accept wrt to i. (2) $\forall i$: $P(\textit{no-match}^E_i(k))$ is negligible.*

Note that the above definition requires that if A has some non-matching conversations, then still the parties involved in matching ones have to accept. We now show a transformation of a mutual authentication protocol $\mathrm{MA}_{A,B}$ into a multicast authentication protocol $\mathrm{MA}_{A,\vec{B}}$, with the simplifying assumption that no encryption is used in $\mathrm{MA}_{A,B}$ (if this is not the case, we can use the transformation of the key transport protocol in the subsequent section):

- Consider n instances MA_{A,B_i} for $1 \leq i \leq n$, where the n messages sent by A in each round are of the form $m_{A,B_i} = (B_i, A, r^i_A, d^i_{B_i}, a(.))$, assuming wlog that the session number (s) for MA_{A,B_i} is i.

  If MA is symmetric-key based:
  in each round, replace the n messages m_{A,B_i}
  by a single multicast message $m =$
  $((B_1, \dots B_n), A,$
  $(PRF_{r^1_A}(A, B_1), \dots, PRF_{r^1_A}(A, B_n)),$
  $(d^1_{B_1} \dots d^n_{B_n}), (a_{A,B_1}(\,), \dots, a_{A,B_n}(.)))$

  If MA is asymmetric-key based
  in each round, replace the n message m_{A,B_i}
  by a single multicast message $m =$
  $((B_1, \quad, B_n), A,$
  $(PRF_{r^1_A}(A, B_1), \dots, PRF_{r^1_A}(A, B_n),$
  $(d^1_{B_1}, \dots, d^n_{B_n}), a_A(\,))$
- A accepts or rejects the outcome for each B_i separately.

Note that an asymmetric-key based MA yields a more efficient transformation, since A has to compute authentication only once per message We do not assume anything on the underlying transport mechanism, in particular we allow for the possibility that A's message is broken into pieces (smaller messages) according to the different recipients either at A or in transit (e.g., there is a B_i joining at a later point in time or IP multicast)

Theorem 8 *$MA_{A,\vec{B}}$ is a secure multicast authentication protocol.*

Proof: (idea): The first condition of definition 7 is easily verified. We now show that if the adversary E is successful against $\mathrm{MA}_{A,\vec{B}}$ (i.e., p(no-match) is not negligible), it is also successful against MA_{A,B_i}, for any i:

We assume that E is successful. Hence there must a first flow, accepted by the receiver, in a conversation of $\mathrm{MA}_{A,\vec{B}}$, which distinguishes it from a matching conversation. Assume this flow is in the conversation among $(\Pi^s_{B_i,A}, \Pi^s_{A,\vec{B}})$. Since MA_{A,B_i} executed by itself is secure and since the use of *PRF* maintains the unpredictability of the message content relative among possible messages pieces of the big message of A, E must have used some information obtained by a flow of some $\Pi^s_{B_j,A}$ or by the fields added to a flow by $\Pi^s_{A,\vec{B}}$ when compared to Π^s_{A,B_i} We now show that in this case, E can also break MA_{A,B_i} executed by itself: we distinguish two cases. Case 1: the "helpful" information originates in a message by $\Pi^s_{B_j,A}$. In this case, E can do the following when when MA_{A,B_i} executes by itself: E executes a corrupt-query for B_j, observes the flows of MA_{A,B_i} and computes the helpful information on its own at B_j. Case 2: the "helpful" information originates in a message by $\Pi^s_{A,\vec{B}}$ (wlog in a field destinated for $\Pi^s_{B_j,A}$ in the first *authenticated* flow by $\Pi^s_{A,\vec{B}}$). In this case E can do the following when MA_{A,B_j} executes by itself it uses this helpful field to compute a corresponding flow in MA_{A,B_i}, which will be accepted by $\Pi^{s'}_{B_i,A}$ We are assured of this acceptance, since $\Pi^{s'}_{B_i,A}$ does not interact with $\Pi^s_{B_j,A}$ and thus is not aware that this flow shares some fields (e.g , nonces) a flow received by $\Pi^s_{B_j,A}$ Thus we have reached a contradiction $\square$

Since the *PRF*'s "task" is to hold the message pieces together in the same message:

Corollary 9 *If a one-to-many transport mechanism is used (e.g., simultaneous broadcast medium), then the use of PRF is unnecessary.*

6 From Auth. Key Transport to Multicast Key Transport

In this section, we show a secure protocol expansion, transforming a two-party AKT into a multicast key transport protocol: A single party A wants to transport the same key to a group of other parties $\vec{B} = \{B_1, B_2, \ldots B_n\}$. An oracle $\Pi^s_{A,\vec{B}}$ (or $\Pi^s_{B_j,A}$) is fresh if 1) it has accepted, 2) it is unopened, and noncorrupt, and 3) there is no opened or corrupted oracle engaging in a matching conversation with any other oracle in session s, where s defines the group as the set of oracles B_i engaging in a matching conversation with the group leader A.

Definition 10 *(Multicast Key Transport)*
(1) Π is a secure multicast authentication protocol. (2) In presence of a benign adversary E, $\Pi^s_{A,\vec{B}}$ and each $\Pi^s_{B_i,A}$ accept the same key, chosen according to a predefined distribution. (3) E cannot distinguish an accepted key at a fresh oracle from a random value with non-negligible probability.

We now show a transformation of a secure key transport protocol $\mathrm{AKT}_{A,B}$, in which the key is transmitted from A to B into a multicast key transport protocol $MKT_{A,\vec{B}}$, where we assume that AKT does not transmit any unencrypted data Otherwise, we can simply combine this transformation with the one in the previous section, as demonstrated for a concrete application in Section 10

- Consider n instances AKT_{A,B_i} for $1 \le i \le n$, where the n messages sent by A in each round are of the form $m_{A,B_i} = (B_i, A, r^i_A, e_(d^i_A, d^i_{B_i}), a(.))$, assuming wlog that the session number is i.

  If AKT is symmetric Always replace the n messages M_{A,B_i} by a single multicast message $m =$
  $(B_1, \ldots B_n), A,$
  $(PRF_{r^1_A}(A, B_1), \ldots, PRF_{r^1_A}(A, B_n)),$
  $(e_{A,B_1}(d^1_A, d^1_{B_1}), \ldots, e_{A,B_n}(d^1_A, d^n_{B_n})),$
  $(a_{A,B_1}(\,), \ldots, a_{A,B_n}(\,))$

  If AKT is asymmetric: Always replace the n message m_{A,B_i} by a single multicast message $m =$
  $(B_1, \ldots B_n), A,$
  $(PRF_{r^1_A}(A, B_1), \ldots, PRF_{r^1_A}(A, B_n)),$
  $e_{B_1}(d^1_A, d_{B_1}), \ldots, e_{B_n}(d^1_A, d_{B_n}), a_A(\,)$
- A accepts or rejects the outcome for each B_i separately

Theorem 11 *$MKT_{A,\vec{B}}$ is a secure multicast key transport protocol.*

Proof: (idea): Same proof as for Theorem 8 shows that MKT is a secure multicast authentication protocol. Since Π_{A,B_i} is only fresh, if all member-oracles are fresh, and each key transport is secure separately, a possible success of E implies that E knowing some flow containing the (encrypted) key helps in guessing the key within the same session (i.e , without any additional chosen message attacks). But this gives E the capability of breaking the underlying AKT. $\square$

7 Multicast Perfect Forward Secrecy

In this section, we show an expansion, which achieves the notion of forward secrecy for Multicast Key Transport This notion is important and has been an issue in the context of two parties. We extend the notion of a session as follows

Definition 12 *(Session)*
Collection of data messages sent within the same group. The collection can be defined, e.g., by the leader.

Definition 13 *(Forward-Secret Session-Key Trans-*

port)
(1) Π is a secure multicast authentication protocol. (2) In presence of a benign adversary E, $\Pi^s_{A,\vec{B}}$ and each $\Pi^s_{B_i,A}$ accept the same session-key, according to a predefined distribution. (3) E cannot distinguish an accepted key for a session σ at an oracle, which was fresh until the end of σ (but E is allowed to corrupt it at any point thereafter), from a random string.

We now show a reduction from an asymmetric and a symmetric MKT to a forward-secret session-key transport protocol FSSKT.

- We assume a group which already shares a group-key.
- Each group-member chooses a short-lived public and private key (via some standard key generating function G).
- Each member executes a symmetric 2-party AKT protocol *s-AKT* with the leader which uses the group-key to securely transport the member's short-lived public key to the leader.
- The group leader distributes a session group-key by executing an asymmetric MKT protocol *a-MKT*, which uses the short-lived keying material.
- Each member uses the session group-key to transmit data to any other member during the session
- At the end of the session, each member removes all short-lived keying material

Theorem 14 *FSSKT is a secure forward secret key transport protocol.*

Proof: (idea): Since *a-MKT* is a secure multicast authentication protocol, FSSKT is as well. In the presence of a benign adversary, *s-MKT* delivers the short-lived keying material of every group-member to every other group-member and *a-MKT* delivers the session key from the leader. We now show that the session-key is protected We assume that E is successful against FSSKT. (1) *s-AKT* is a secure authenticated key transport protocol (2) *a-MKT* is a secure multicast key transport protocol. (3) the private keys used in *a-MKT* and the session-key are removed at the end of s. (4) the private keys used in *a-MKT* are never transmitted. (1) and the fact the E cannot execute a corrupt-query until s has terminated imply that every member receives every other members public key. This together with (2) implies that E cannot guess the session-key before s has terminated. (3) and (4) imply that E cannot guess the session-key after a corrupt-query, which was executed after the termination of s. □

8 Group Management: Access Control & Group-Key Transport

We first discuss the requirements of *access control and group-key distribution (transport)*. We then show how the protocols developed so far can be used as building blocks to obtain secure and robust solutions.

We assume that there is a group-leader A, which manages a group of parties B_i. A distributes a group-key κ_A to a party B_i, if B_i requests membership in the group and is in the group's access list, which is maintained by A. A also maintains a list of current members. Access control and group-key distribution essentially requires that each party obtains the group-key upon request if and only if it is on the access control list and that the group leader maintains an accurate list of parties which are currently members of the group. Let AC_A be the access control list, let L_A be the list of current members, both located on A, let req-member$_{B_i}$ be true if B_i has requested membership, and let member$_{B_i}$ be true if B_i has obtained membership.

Definition 15 *(Access Control and Group-Key Transport $AC\text{-}GKT_{A,G}$)*

1. $B_i \in L_A \Rightarrow (B_i \in AC_A$ *and* $\textit{req-member}_{B_i})$
2. $\textit{member}_{B_i} \Rightarrow (B_i$ *holds group key* κ_A *and* $B_i \in L_A)$
3. *In the presence of a benign adversary.* $(\textit{req-member}_{B_i}$ *and* $B_i \in AC_A) \Rightarrow \textit{member}_{B_i})$
4. *Group key is secure in the sense of Definition 10*

   A session key for the group assures prefect forward secrecy.

The first requirement ensures that if the leader accepts a new group member, that this party is on the access control list and has indeed requested to join. The second requirement guarantees that if a party accepted the membership, it indeed obtained the group-key and is included on the leader's group-list The third requirement ensures that if a party on the access control list requests to join, the leader honors the request

8.1 Embedding MKT and FSSKT into AC-GKT

We would like to build a secure AC-GKT protocol out of secure MKT- and FSSKT-protocols Those protocols are in turn built out of AKT-protocols, such as presented in [BR93, BM97]. For an AKT, typically not much thought is given as to which side is generating the resulting session key. In our case, we clearly want the leader to chose the session key. Thus we embed an MKT protocol accordingly. Depending on the MKT (in fact, the underlying AKT), either the leader or the members initiate the protocol In the first case, we embed the protocol such that the leader uses the first message to announce the group formation. In the second case, the members use the first message to indicate a join-request. In this case, we need to have a separate group announcement mechanism. This might be via a shared whiteboard ("pull") or another multicast ("push"). If authentication is needed, this can be implemented via a leader initiated $\mathrm{MA}_{A,\vec{B}}$, where the group announcement is included in the last message of the leader.

We now show a solution, assuming that the underlying AKT implies that the prospective members initiate the protocol. Let s-MKT* (a-MKT*), denote an Authenticated Multicast Key Transport Protocol with Consistency 1 using symmetric (asymmetric crypto) and let FSSKT be a forward secret session-key protocol. The protocol P^* used below either denotes s-MKT* or a-MKT*, depending whether initially the leader shares a key with each prospective member or whether some public-key infrastructure is available

- Upon setting req-member$_{B_i}$ to true, B_i initiates its part of $P^*_{A,\vec{B}}$.
- A only participates with B_i in $P^*_{A,\vec{B}}$ if $B_i \in AC_A$
- A waits to collect all (or a number above some threshold) initial messages from B_i and then engages in $P^*_{A,\vec{B}}$ to distribute κ_A
- After accepting with respect to B_i, A includes B_i in L_A.
- After accepting, B_i sets member$_{B_i}$.
- To implement a forward-secret sub-session, FSSKT (using s-MKT* and a-MKT* as subroutines) is invoked.

Theorem 16 *Above is a secure AC-GKT.*

Proof: (idea): Condition 1: $B_i \in L_A$ holds only if A accepts AC-GKT and thus (1) A had verified that B_i is in AC and (2) A had a matching conversation with B_i, which implies that B_i had sent the first message. Condition 2: member$_{B_i}$ holds only if B_i accepts AC-GKT and thus executed an MKT with Consistency 1 Condition 3 and 4 follow directly from MKT correctness. □

9 Group Policies

In a broadcast network, group-announcement and termination can be implemented via a leader initiated $\mathrm{MA}_{A,\vec{B}}$ protocol. A member joining or (smoothly) leaving an existing group can execute a two-party authentication protocol (incremental two-party addition to the existing multicast protocol which maintains security and efficiency).

According to the *group policy*, a change in the composition of the group, might necessitate to redistribute a group key Further discussions on group policies are beyond the scope of this paper.

10 Actual Protocols via Transformations

In this Section we apply our method on concrete examples. In the following, $(x)_{PEK_B}$ denotes encryption of x under B's public key and $(m)_{SSK_B}$ denotes the message m together with B's signature on m Let $(x)_{K^1_{A,B}}$ denote the encryption of x under a shared key and let $(m)_{K^2_{A,B}}$ denote the message m together with its message authentication code (MAC) under a shared key. R_B and r_B denote nonces of B. We assume an underlying synchronous broadcast mechanism to simplify the exposition of the resulting protocols

In Figures 2 and 3, we recap the s-AKT protocol presented by Bellare/Rogaway ([BR93]) and the a-AKT protocol by Blake-Wilson/Menezes ([BM97]), respectively. Figure 4 shows the Blake-Wilson Menezes AKT protocol, after being transformed to fulfill Consistency 1 and subsequently transformed into a multicast key transport protocol. Finally, Figure 5 shows the Bellare/Rogaway key transport protocol, after it has been transformed into a multicast key transport protocol with Consistency 2. The results in this paper immediately imply that both these protocols are proven secure, given that their respective precondition of each member having a public/private key pair or the leader sharing a secret key with each member, is satisfied.

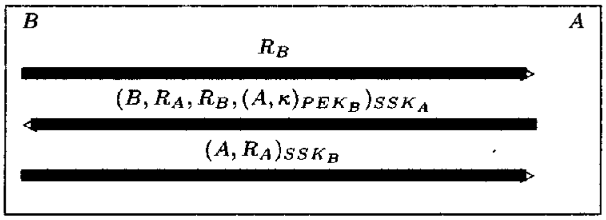

Figure 2. Blake-Wilson/Menezes asymmetric auth. key transport (a-AKT) protocol

11 Conclusion and Outlook

We have designed a basic "arsenal" of secure group management protocols. Rather than starting from scratch, we have systematically transformed basic, previously proven secure two-party protocols into the

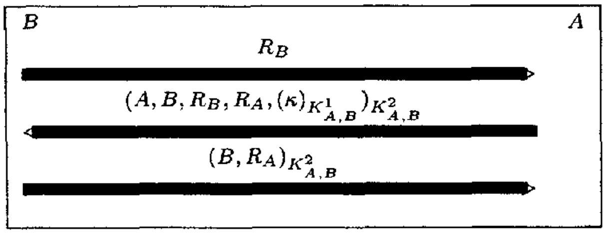

Figure 3: Bellare/Rogaway symmetric auth key transport (s-AKT) protocol

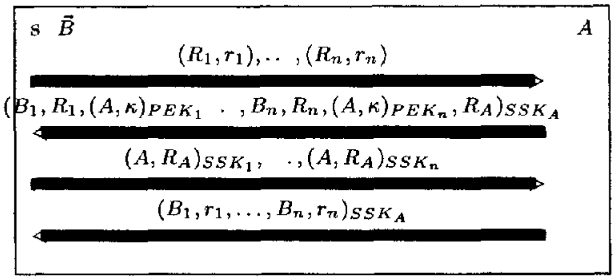

Figure 4 a-MKT with Consistency 1; Blake-Wilson/Menezes as starting point

desired protocols By doing so, we have obtained intuitive and simple protocols whose proof of security is more straightforward than if we had started from scratch.

We believe that this approach will prove useful for a number of other applications, which are based on authentication and key transport One such direction, is embedding multicast protocols within a network topology in the presence of misbehaving nodes inside and outside the multicast group Another future direction is incorporating the work on the multiparty Diffie-Hellman based protocols (e.g , [STW96, BW98, AST98]) into the protocol expansion framework

References

[AMP96] A Aziz, T Markson, H Prafullchandra, SKIP Extensions for IP Multicast *Internet Draft*, 1996

[AST98] G Atienese, M Steiner, G Tsudik, Authenticated Group Key Agreement and Friends Proc *ACM CCS*, 1998

[B96] A Ballardie, Scalable Multicast Key Distribution *RFC-1949*, 1996

[BC95] A Ballardie and J Crowcroft, Multicast Specific Security Threats and Counter-Measures *ISOC Symposium on Network and Distributed System Security*, 1995

[BW98] K Becker and U Wille, Communication Complexity of Group Key Distribution Proc *ACM CCS*, 1998.

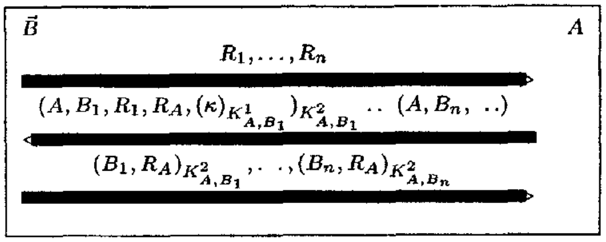

Figure 5 s-MKT with Consistency 2; Bellare/Rogaway as starting point

[BR93] M Bellare and P. Rogaway, Entity Authentication and Key Distribution *Crypto'93* D Stinson (Ed.)

[BR95] M Bellare and P Rogaway, Provably Secure Session Key Distribution – The Three Party Case *27th ACM STOC*, 1995.

[Be98] S M Bellovin, Cryptography and the Internet (invited talk) *Advances in Cryptology – Crypto 98, Springer-Verlag LNCS 1462*, H Krawzcyk (Ed), 1998, pp. 46–55

[B-al-91] R Bird, I Gopal, A. Herzberg, P Janson, S Kutten, R Molva, M. Yung, Systematic design of 2-party authentication protocols, *Advances in Cryptology – Crypto 91*, J Feigenbaum (Ed) LNCS 576, Springer Verlag (1991)

[B-al-95] R Bird, A Herzberg, P Jansen, R Molva, I Gopal, S Kutten, and M Yung, The KryptoKnight Light-Weight Family of Authentication and Key-Distribution Protocols *ACM/IEEE Trans. on Networking*, March 1995

[BM97] S Blake-Wilson and A Menezes, Entity Authentication and Authenticated Key Transport Protocols Employing Asymmetric Techniques *Security Protocols Workshop*, 1997

[C+99] R Canetti, J Garay, G Itkis, D Micciancio, M Naor and B Pinkas, Multicast Security A Taxonomy and Some Efficient Constructions Proc *IEEE INFOCOM*, 1999

[DOW92] W Diffie, P van Oorschot, and M Wiener, Authentication and Authenticated Key Exchanges *Designs, Codes, and Cryptography*, **2**, 1992

[FLP85] M J Fischer, N. Lynch, and M Paterson, Impossibility of Distributed Consensus with One Faulty Processor *Journal of the ACM*, **32**(2) 374 – 382, 1985

[GGM] O Goldreich S Goldwasser and S Micali, How to Construct Random Functions. *J of the ACM 33*, (1986), pp 792-807

[G97] L Gong, Enclaves Enabling Secure Collaboration over the Internet. *IEEE J. Selected Ares in Comm.*, **15**, 1997

[G94] L Gong, New Protocols for Third Party-Based Authentication and Secure Broadcast *2nd ACM Conf on Computer and Communications Security*, 1994.

[JKKO94] M JUST, E KRANAKIS, D KRIZANC, P VAN OORSCHOT, On Key Distribution via True Broadcasting *2nd ACM Conf. on Computer and Communications Security*, 1994

[JTY97] P JANSON, G TSUDIK AND M YUNG, Scalability and Flexibility in Authentication Services The KryptoKnight Approach. Proc *IEEE INFOCOM*, 1997.

[LM93] T LEIGHTON AND S MICALI, Secret-key agreement without public key cryptography, *Advances in Cryptology– Crypto 93, Springer-Verlag LNCS 773*, D. Stinson (Ed.), 1994, pp 456–479

[M97] S MITTRA, Iolus A Framework for Scalable Secure Multicasting *ACM SIGCOMM'97*

[M-al-92] R MOLVA, G TSUDIK, E VAN HERREWEGHEN, S ZATTI, "KryptoKnight authentication and key distribution system," Proc. ESORICS 92, Toulouse, France (23-25 Nov 92)

[NS78] R M NEEDHAM, AND M D SCHROEDER, Using Encryption for Authentication in Large Networks of Computers *CACM 21 (12)*, (1978) 993-998

[SR96] V SHOUP AND A RUBIN, Session Key Distribution using Smart Cards *Advances in Cryptology – Eurocrypt 96*, U Maurer (Ed) LNCS 1070, Springer Verlag (1996)

[STW96] M STEINER, G TSUDIK, M WAIDNER, Diffie-Hellman Key Distribution Extended to Groups. Proc *ACM CSS*, 1996

[TvH93] G TSUDIK, E VAN HERREWEGHEN, On Simple and Secure Key Distribution, The 1-st ACM Conf on Compu and Comm. Security, 1993, 49-57