P. Syverson and S. Stubblebine, Group Principals and the Formalization of Anonymity. in *FM'99 – Formal Methods, Vol. I*, Springer-Verlag LNCS 1708, Sept. 1999, pp. 314–333.

Group Principals and the Formalization of Anonymity

Paul F. Syverson[1] and Stuart G. Stubblebine[2]

[1] Center for High Assurance Computer Systems,
Naval Research Laboratory, Washington, DC 20375, USA,
syverson@itd.nrl.navy.mil ⋆⋆⋆

[2] CertCo, 55 Broad St. - Suite 22, New York, NY 10004, USA,
stubblebine@{cs.columbia.edu, CertCo.com}
http://www.cs.columbia.edu/~ stu †

Abstract. We introduce the concept of a group principal and present a number of different classes of group principals, including threshold-group-principals. These appear to naturally useful concepts for looking at security. We provide an associated epistemic language and logic and use it to reason about anonymity protocols and anonymity services, where protection properties are formulated from the intruder's knowledge of group principals. Using our language, we give an epistemic characterization of anonymity properties. We also present a specification of a simple anonymizing system using our theory.

1 Introduction

Though principals are typically viewed as atomic, there is no reason we cannot consider the knowledge and actions taken by a group. Hence, the basic notion of a group principal. This notion appears to be a useful concept for reasoning about various properties of electronic commerce and security protocols. One such principal is a threshold-group-principal. Such a principal allows us to express properties of threshold cryptosystems [13]. Although we do not pursue this in the present paper, we believe we can give a straightforward characterization not only of threshold cryptography including signatures and confidentiality, but also (once time is introduced into our language) such things as proactive security [5] and mobile adversaries [19]. Another group principal is the or-group principal. It is useful for characterising security properties relating to anonymity.

We demonstrate the applicability of our theory by examining the issue of anonymity and privacy. Studies have shown that privacy is a great concern for users of electronic commerce. Numerous protocols have emerged for protecting the anonymity of individuals. These protocols have been in the areas of protecting general Internet communications [23], commercial transactions [25], web based communications [21, 1], email [9, 18], and electronic cash [28]. However,

⋆⋆⋆ Work by this author supported by ONR.
† Work by this author was also performed at AT&T Research.

little work has been done on formally representing or analyzing privacy in such protocols.

In this paper, we provide an epistemic language and logic and use it to reason about anonymity protocols and anonymity services. We also describe an associated model of computation. Using our language, we give an epistemic characterization of various anonymity properties. As far as we know, these basic properties have not been set out previously.

We develop the idea of looking at the environment as not a single entity for which all messages must pass, but one with individual components with different characteristics. In our model, the environment principal is no different from system principals. When you send a message you send it to an environment principal, likewise for receiving messages. All uncertainty in communication is represented in the environment principals; so, any sent message is immediately received, and all received messages have been sent by some principal. In this way, we are able specify environments particular to the threat model at hand.

We demonstrate our approach with a simple example. Typically, we have a single intruder which is a distributed group principal, composed of environment and/or (compromised) system principals. Each principal is specified by a knowledge program. Compromised principals run distinct programs from their reliable counterparts. It is also possible to have multiple intruders, each with their own separate goals, though we do not present any examples of this in this paper.

This paper does not address temporal features directly, other than to differentiate past and present (much as in [2]). Thus any timing attack on an anonymity system is beyond its scope. Temporal reasoning is expected to be added in future work, and there is no reason to expect difficulty in doing so. Indeed, the knowledge programs set out in this paper are derived from the knowledge-based programs of [11, 12], and those include temporal operators by default.

The seminal work setting out properties, goals, and mechanisms for anonymity in communication is that of Chaum (cf., e.g., [6, 7]). Our work is the first we are aware to give an epistemic characterization of anonymity properties. However, anonymity properties have been formally defined in CSP [22]. And, in [20] a formal notation was given for specifying anonymity protocols; however, that notation was not designed to specify anonymity properties or to be used in formal analysis. Also, others have defined interesting rigorous but informal notions of security properties [21].

The remainder of this paper is as follows. In Section 2 we present our model including the definitions of various types of group principals. In Section 3 we present the formal language. In Section 4 we present the logic. In Section 5 we present anonymity properties. In Section 6 we present our knowledge programs. In Section 7 we present a specification of the anonymizer protocol. In Section 8 we present our conclusions.

2 Model

Our system model is essentially built from and extends model elements described in [11] and elsewhere. We give a sketch of our model here.

2.1 Atomic Principals

There is a set of *atomic principals* $\{P_1, \ldots, P_n\}$. These are similar to the 'ordinary' principals that one associates with distributed computing. However, unlike others, we do not distinguish the environment in the model. Also, unlike others, the environment may be several (possibly disconnected) principals. Environment principals are made up of these atomic principals, just as system principals are.

2.2 Actions

Each principal can perform any one of a set of *actions* at each time. The actions that can be performed are $send(M, P_i, P_j)$, $receive(M, P_j, P_i)$, representing the sending of a message $M \in \mathcal{M}$ (the set of messages) from P_i to P_j and its receipt by P_j from P_i respectively. Principals may also perform internal operations, $int\_action(\{M_1, \ldots, M_n\}, P_i)$. This represents principal P_i performing some internal operation on the messages in the set $\{M_1, \ldots, M_n\}$, for example encryption, concatenation, decryption, etc. Principals may also do nothing at a given time. This is indicated by the performance of the null action Λ. Two particular internal actions, *record* and *purge*, will be discussed presently. We follow the example of [10] and subsequent work, that all messages are sent to the environment or received from the environment. We can thus make the simplifying assumption that all sent messages are received immediately. Any message loss, delay, modification, etc. can be represented by the actions of the environment. So, exactly one of P_i and P_j in any $send(M, P_i, P_j)$ or $receive(M, P_j, P_i)$ is always an environmental principal.

2.3 States

Each principal has a *local state*. Local states are assumed to be unique; although principals may not always be able to distinguish all of even their own local states. A state s_i local to principal P_i at time t is given by

$$s_i \stackrel{\triangle}{=} \langle state\_id, history, log, facts, recent \rangle$$

The *history*, is the sequence of actions that have been performed locally. The *log*, is the sequence of local actions that have been logged. Similar to the local history, the local log is complete in having an entry for each time. But, since the log reflects the local time, entries are recorded as $\langle a, t \rangle$ where $t = localtime(t')$ is the time on the local clock when t' is the actual time, and t' is the actual time that a occurred. We assume that the real clock is fine enough to reflect the occurrence of all events in the system. Thus, an advance of the real clock need not imply an advance on all local clocks, but an advance on any local clock implies an advance on the real clock. Since principals may or may not keep track of local actions, and may even 'forget' them, the log may contain a null-log-entry $\perp$ for any time t. This is not the same as a null-action Λ, which may occur in both the local history and the local log and indicates that no action was performed at that time. We also keep track of any facts that may be known by a principal, such as the public key of a local server. These are collected in a set *facts*. In a principal's initial state all of the fields except *state_id* and *facts* should be empty sequences.

We keep track of actions and of facts by means of a *record* action. *record* is defined on $\mathit{recent} \cup \mathit{knowledge}$. (The constitution of both *recent* and *knowledge* will be set out below.): For an action $a \in \mathit{recent}$, $\mathit{record}(a)$ has the effect of placing $\langle a, \mathit{localtime}(t_a))\rangle$ in the local log. For a known formula φ, $\mathit{record}(\varphi)$ has the effect of placing φ in the set of facts *facts*. We also allow sets of recent actions and sets of known formulae in the domain of *record*. The way that *record* works for these sets should be obvious from the case for individual actions and known formulae. *purge* is similarly defined on entries in *log* and facts in *facts*. $\mathit{purge}(\{\langle a_{i_1}, t_{i_1}\rangle, \ldots, \langle a_{i_m}, t_{i_m}\rangle\})$ has the effect of removing those log entries from the log and replacing them with $\perp$. $\mathit{purge}(\{\varphi_1, \ldots, \varphi_m\})$ has the effect of removing those formulae from the set of recorded facts. The recent actions *recent*, are actions that were effectively performed recently and are remembered even though they have not been logged. *recent* is always a tail segment of *history* and never includes *record* or *purge* actions.

We will introduce composite (group) principals presently. Nonetheless, each *global state* is completely determined by a tuple of the local states of all atomic principals. A *run* is a sequence of global states indexed by (actual) times, where the any local state occurring in the global state at time t is such that the relevant principal is in that local state at (actual time) t.

2.4 Knowledge

In a given local state, knowledge is entirely determined by the log, the set of facts, and the recent actions. We include in the set of formulae, *knowledge*, the closure of what is known from those three sources. More precisely we have: (1) If an action is known to a principal because it is in the log or is recent, then the principal knows that he performed that action. So, for example, if $\mathit{receive}(M, P_i, P_j) \in \mathit{recent}$ or if $\mathit{receive}(M, P_i, P_j) \in \mathit{log}$ then M **received from** P_j $P_i \in \mathit{knowledge}$. (2) If $\varphi \in \mathit{facts}$, then $\varphi \in \mathit{knowledge}$. (3) If φ can be derived from other members of *knowledge* by the axioms, then $\varphi \in \mathit{knowledge}$. That P knows φ is represented in our language by $\Box_P\ \varphi$. The dual of $\Box_P$ is $\Diamond_P$. (There are certain generic axioms for adding to the knowledge of principals, e.g., $(\Box_P\ \varphi \wedge \Box_P\ (\varphi \supset \psi)) \supset \Box_P\ \psi$. So, if $\varphi, (\varphi \supset \psi) \in \mathit{knowledge}$, then $\psi \in \mathit{knowledge}$. Axioms for knowledge will be briefly discussed below in section 4.)

2.5 Group Principals

Ordinarily, we think of principals atomically. In particular, when evaluating protocols in the Dolev-Yao framework, we view all communication as going through a single environmental principal, typically identified as the intruder. For example, anything sent between principals A and B is assumed to be known to the intruder as is anything sent between principals C and D. However, the intruder between A and B may not be able to directly communicate with the intruder between C and D. They may only be able to communicate via 'honest' principals, e.g., one intruder can signal the other by causing an honest principal between them to send certain messages to the other. (Cf. [26] for more discussion of this model of computation in a hostile environment.) This naturally engenders a view of the environment as a distributed group principal. Similarly, sets of honest principals trying to solve some threshold computation (e.g., decryption or signature) may be thought of in this way. We will find it useful to have various types of *group principals* to model these and other circumstances.

There are four kinds of group principal, *collective-group* ($\star G$), *and-group* ($\&G$), *or-group* ($\oplus G$), and *threshold-group*$(n - G)$. Each type of group principal is distinguished by how the knowledge and actions of the principal is determined by the knowledge and actions of the members of that principal. The set of group principals $\mathcal{G}$ is defined as follows: for any nonempty set of atomic principals G, $\star G$, $\&G$, and $\oplus G$ are all groups (of the indicated type). And, $n - G$ is a (threshold) group provided that $n \leq |G|$.

collective group principal: Given any set of atomic principals G, $\star G$ is a distributed group viewed collectively. What the group knows is what is known by combining the knowledge of all the group members. (This is the concept of distributed knowledge in [11].) The group actions are those taken by the group collectively. For example, if something is sent or received by any member of the group then it is sent (received) by the group. However, it may also be the case that the group performs some action, e.g., elect a leader and possible successors, that is not performed by any one of the members. In this example, each member might vote for one leader, but the succession is determined by the total number of votes received, in diminishing order.

and-group principal: written $\&G$ for an and-group of members of G, is a distributed group viewed conjunctively. We also write $(P_1 \wedge \ldots \wedge P_n)$ for the and-group of principals P_1 through P_n. What the and-group knows is what every member of the group knows. (This is the concept of everyone knowledge in [11].) The group actions are those taken by each member of the group. Thus, $\&G$ said (received) messsage M if each member of G said (received) M.

or-group principal: written $\oplus G$ for an or-group of members of G, is a distributed group viewed disjunctively. We also write $(P_1 \vee \ldots \vee P_n)$ for the or-group of principals P_1 through P_n. What the or-group knows is what at least one member of the group knows. (This does not have a correlate in [11].) The group actions are those taken by at least one member of the group. Thus, $\oplus G$ said (received) messsage M if at least one member of G said (received) M.

threshold group principal: written $n - G$ for a given threshold n and group G. What the n-threshold group $n - G$ knows is anything known by any collective subgroup contained in G of cardinality at least n. (This does not have a correlate in [11].) Suppose two subgroups $G', G'' \subseteq G$ s.t. $|G'| \geq n$ and $|G''| \geq n$. Ordinarily, if $\Box_{G'} \ \varphi$ and $\Box_{G''} \ (\varphi \ \supset \ \psi)$, we cannot conclude anything more specific than that $\Box_{\star(G' \cup G'')} \ \psi$. But, if G is an n-threshold group, then it follows that $\Box_{n-G} \ \psi$. Thus, it follows that $\Box_{G'} \ \psi$ (and $\Box_{G'} \ (\varphi \ \supset \ \psi)$) and $\Box_{G''} \ \psi$ (and $\Box_{G''} \ \varphi$). Another way to characterize an n-threshold group is as an or-group of collective groups, each with cardinality of at least n. Thus if $G' = \{G_1, \ldots, G_m\}$ is the set of all collective subgroups of G s.t. $|G_i| \geq n$, then

$$n - G \ \stackrel{\triangle}{=} \ \oplus G' \quad (\text{which is } (G_1 \ \vee \ \ldots \ \vee \ G_m))$$

What the n-threshold group $n - G$ does is what is done by any subgroup contained in G of cardinality at least n. Thus, $n - G$ said (received) anything said (received) by any subgroup of cardinality at least n.

3 Formal Language

Let A and B be principals, M be a message, and φ be a formula. We assume without explanation the usual logical connectives and formula building using them. Any formula is also a message, though not vice versa.

Actions. There are send and receive actions. We can record and purge both send and receive actions.

$\texttt{send}(M, \ A, \ B)$
$\texttt{receive}(M, \ B, \ A)$

Also, if `s-r-action` is a send or receive action, then we also have the purging and recording of send and receive actions.

`purge(s-r-action)`
`record(s-r-action)`

We will find it useful to have the following macro (eliminable definition):

$$\texttt{action}(X, A, B, \text{remember});$$

is a macro for

$$\texttt{action}(X, A, B); \ \texttt{record}(\texttt{action}(X, A, B)) \ ;$$

Said and Received Formula. These are formulae expressing the sending and receiving of messages, as well as of any formulae implicit in the sending or receiving of such messages.

A **said** M
A **received** M
A **said to** B M
B **received from** A M

That a formula represents the whole message sent or received is denoted by the use of quotation marks. These refer not to the bit string, but to the parsing of that string without any encryption, decryption, deconcatenation, etc[1]. So, for example, were A to send the message $\{M\}_K$ (where A knew K), then A **said** "$\{M\}_K$" and A **said** M would be true, but A **said** "M" would not.

Message Extensions. Message fields may have an origin and destination. We express this using either "to" or "from" or using both extensions.

(X **from** A **to** B)

We can further qualify certain features of a message that are common to anonymity protocols. These features include an indication of the ultimate destination of a message using "for".

(X **for** B)

Another feature common to anonymity protocols is referencing a prior message. This is common for query-response (request-response) protocols.

R **in response to** Q

Encryptions and Key Possession. Messages may also be encrypted. The encryption of M with K and A's possession of K^{-1} are expressed as follows.

$\{M\}_K$
A **has** K^{-1}

Runs Formula. A principal running a knowledge program is expressed.

A **runs** `program_name`

Knowledge. If φ is a formula in the language, and A a principal, we can express that A knows φ and that A knows possibly φ, by the following formulae respectively.

$\Box_A\, \varphi$
$\Diamond_A\, \varphi$

[1] In particular, this is not meant to be an opaque context. Thus, values may be substituted for variable names.

Did Formula. The following allow us to express any action done by a principal, as a formula. For example, suppose A, performed some action `action`. This is expressed by the following. (It might be possible to replace the following two types of formulae with one type and an appropriate temporal operator. However, we choose to keep the number of modal operator types that we use to a minimum for the present.)

A **did** (`action`)

If the action is being done now (i.e, *recently* according to our model), we can express this as:

A **does** (`action`)

4 Logic

We set out here our axioms and rules. Those for knowledge and propositional reasoning are standard. For background consult [11, 8, 16].

Propositional and Epistemic Logic. Knowledge is characterized by the **S5** axioms. Our only rules are modus ponens and necessitation (knowledge generalization):

Modus Ponens: From φ and $\varphi \supset \psi$ infer ψ.

Knowledge Generalization: From $\vdash \varphi$ infer $\vdash \Box_P\, \varphi$.

It is important to recall that knowledge generalization does not allow us to infer that P knows φ for arbitrary formulae φ. Rather, if φ is a theorem (i.e., derivable from axioms alone with no assumptions) then $\Box_P\, \varphi$ is also a theorem. In other words, all principals are expected to know all logical truths. Now for the axioms.

Ax1. All tautologies of propositional logic are axioms.

As mentioned before, $\Box_P$ and $\Diamond_P$ are duals. This means that these are interchangeable according to the definition: $\Box_P\, \varphi \leftrightarrow \neg\Box_P\, \neg\varphi$ (for any formula φ). Given formulae φ and ψ the knowledge axioms are as follows. (N.B. These axioms, together with the above rules and axiom, constitute **S5**, the most standard and well understood knowledge logic for distributed computing. The axioms may not all be ultimately necessary for intended applications. However, we begin with **S5** and leave the possible elimination of unnecessary axioms for future work.)

Ax2. Distribution Axiom, **K**: $\Box_P\, (\varphi \supset \psi) \supset (\Box_P\, \varphi \supset \Box_P\, \psi)$
Ax3. Truth Axiom, **T**: $\Box_P\, \varphi \supset \varphi$
Ax4. Positive Introspection Axiom, **4**: $\Box_P\, \varphi \supset \Box_P\, \Box_P\, \varphi$
Ax5. Negative Introspection Axiom **5**: $\neg\Box_P\, \varphi \supset \Box_P\, \neg\Box_P$

Simplifying Said and Received Formulae. These formulae may be simplified in the obvious ways. We do not list all of these, but simply give representative examples:

Ax6. A **said to** B M $\supset$ A **said** M
Ax7. B **received from** A M $\supset$ B **received** M
Ax8. A **said to** B $(M_1, \ldots, M_n)$ $\supset$ A **said to** B M_i (where $i \in \{1, \ldots, n\}$)

Assume K is an encryption key and K^{-1} the corresponding decryption key (in the symmetric case $K = K^{-1}$).

Ax9. (A **said** $\{M\}_K$ $\wedge$ A **has** K^{-1}) $\supset$ A **said** M
Ax10. (B **received** $\{M\}_K$ $\wedge$ B **has** K^{-1}) $\supset$ B **received** M

Note that we do *not* have any axioms reflecting authentication principles, as in [2] or its successors.

Message extensions may be removed in said and received formulae. Let *extensions*(φ) be the set of all messages that are extensions of φ. Then, for any $\psi \in$ *extensions*(φ) we have the following axioms:

Ax11. A **said** ψ $\supset$ A **said** φ
Ax12. B **received** ψ $\supset$ B **received** φ

Thus, for example, the following is a theorem of our logic:

A **said** "(X **for** C) **from** A **to** B" $\supset$
(A **said** X **from** A $\wedge$ A **said** (X **for** C) **to** B $\wedge$ A **said** X)

Sending and Receiving. Message delivery is guaranteed. Sending corresponds to saying exactly, and likewise for receiving.

Ax13. A **did** (`send`(M, A, B)) $\leftrightarrow$ B **did** (`receive`(M, B, A))
Ax14. A **did** (`send`(M, A, B)) $\leftrightarrow$ A **said to** B "M "
Ax15. B **did** (`receive`(M, B, A)) $\leftrightarrow$ B **received from** A "M"

Record Implies Did. This axiom expresses that an entity recording and action implies that it performed the action.

Ax16. A **did** (`record(s-r-action)`) $\supset$ A **did** (`s-r-action`)

Doing and Knowing What Was Done. These axioms express the conditions under which a principal knows what it has done as well as the relation between **does** and **did**.

Ax17. A **did** (`s-r-action`, remember) $\wedge$ $\neg A$ **did** (`purge(s-r-action)`) $\supset$ $\Box_A$ A **did** (`s-r-action`)
Ax18. A **does** (`s-r-action`) $\supset$ $\Box_A$ A **does** (`s-r-action`)
Ax19. A **does** (`s-r-action`) $\supset$ A **did** (`s-r-action`)

Group Axioms. These axioms relate to formulae involving group principals. We mention here only the basic ones that will be useful in the rest of the paper. Let $G = \{P_1, \ldots, P_n\}$. And, let$\varphi(G)$ be a formula with one or more (free) occurrences of G and $\varphi(P/G)$ be the formula that results from replacing every (free) occcurrence of G in φ with P.

Ax20. $\varphi(\&G) \leftrightarrow (\varphi(P_1/G) \wedge \ldots \wedge \varphi(P_n/G))$
Ax21. $\varphi(\oplus G) \leftrightarrow (\varphi(P_1/G) \vee \ldots \vee \varphi(P_n/G))$

5 Anonymity Properties

The goal of any system or protocol we will be examining is to provide some type of anonymity, that is to hide some fact about a principal or set of principals from some adversary. This can be broken into two parts. The piece of information to be protected and the nature of that protection. In this section, we will set out a characterization of both pieces.

5.1 Condenda (i.e., things to be hidden)

We might want to hide that a principal is the originator of a message or that some pair of principals are the originator and intended recipient, respectively, of a message, etc. For profile security, we may wish to hide that two messages originating from the same principal in fact originated at that principal or more strongly that they originated at any one principal.

5.2 Condens (i.e., types of hiding)

The various facts just described that are to be hidden from view may be hidden to varying degrees. We will now set out the various types of anonymity that can be achieved with respect to each of these. The principal from whom they are to be hidden is always the intruder, I. The exact nature of the intruder will vary from context to context; it may include insiders and/or outsiders to the system running active or merely passive attacks. No matter how the intruder is implemented, we are always able represent the types of anonymity with respect to an abstract intruder, I. This allows for a succinct statement of properties; however, since the following are not stated with respect to a particular principal, technically they are formula schemata rather than formulae. In practice we always specify a particular principal for the condens. In the future, we might allow actual principal variables, but we do not attempt such here. Similarly, we might consider existential quantification over principals, e.g., to reflect the hiding of arbitrary profiles, whether or not they are associated with any given principal (more generally, n-tuple of principals).

We first set out some assumptions. Our main assumption is that all condenda are of the form $\varphi(P)$. In other words, they are single formulae in which a single principal name occurs (freely). Our restriction to single principals is just for simplicity and uniformity of presentation. The generalization from hiding P **said** X to hiding P **said to** Q X is straightforward. Our restriction to single formulae is minimal. For example, if we wish to hide profiling information about P, e.g., that several facts about P are associated, this is generally expressible in a single formula, such as $\varphi(P) \supset \psi(P)$. We do not attempt to represent the hiding of arbitrary formulae of the language that do not involve principal names at all. It is unclear what role these would play in anonymity protection. However, we may explore this possibility in the future should need arise. Our next major assumption is that condenda are true of only one principal. Thus, for any formula $\varphi(P)$ for which we are considering the anonymity provided by a system or protocol, we assume

$$\Box_I\ (\varphi(P)\ \wedge\ \varphi(Q)\ \supset\ P = Q)$$

We also assume that any condendum is actually true. That is, we are not worried about trying to prevent the conclusion or even suspicion that P **said** X in the case when P did not say X.

Unknown
$\neg(\Diamond_I\ \varphi(P))$
In our current logic and language, this is basically impossible. It is logically equivalent to $\Box_I\ \neg\varphi(P)$. Thus, by axiom Ax3, this cannot be true if $\varphi(P)$ is true (which we assume). Therefore, everyone is always a suspect. The only possibility for a principal to be unknown would be if we partitioned the set of principal names so that some were meaningless to the intruder. We do not consider such an extension in this paper.

($\geq n$)-anonymizable
$\Diamond_I\ \varphi(P)\ \supset\ (\Diamond_I\ \varphi(P_1)\ \wedge\ \ldots\ \wedge\ \Diamond_I\ \varphi(P_{n-1}))$
We assume here and in the following definitions that distinct names denote distinct principals. This says that if P is a suspect wrt φ then there are $n-1$ other principals (and possibly more) who are also suspects. If there are precisely $n-1$ other principals such that $\Diamond_I\ \varphi(P_i)$ when $\Diamond_I\ \varphi(P)$, we have the more exact property of being n-anonymizable. Similarly for the properties below.

Possible Anonymity
$\Diamond_I\ \varphi(P)\ \wedge\ \Diamond_I\ \neg\varphi(P)$

The intruder cannot rule out $\varphi(P)$ but cannot rule out $\neg\varphi(P)$. Basically, he has no knowledge about this condendum.

($\leq n$)-suspected
$\Box_I\ (\varphi(P)\ \vee\ \varphi(P_1)\ \vee\ \ldots\ \vee\ \varphi(P_{n-1}))$
The intruder has narrowed things down to no more than n suspects, one of which is P.

($\geq n$)-anonymous
$\Diamond_I\ \varphi(P)\ \wedge\ \Diamond_I\ \varphi(P_1)\ \wedge\ \ldots\ \wedge\ \Diamond_I\ \varphi(P_{n-1})$
The intruder has narrowed things down to no fewer than n suspects, one of which is P.

($\leq m$)-suspected implies ($\geq n$)-anonymous
$\Box_I\ (\varphi(P)\ \vee\ \varphi(P_1)\ \vee\ \ldots\ \vee\ \varphi(P_{m-1}))\ \supset\ (\Diamond_I\ \varphi(P)\ \wedge\ \Diamond_I\ \varphi(P_1)\ \wedge\ \ldots\ \wedge\ \Diamond_I\ \varphi(P_{n-1}))$
Here $n \leq m$. The idea is that even if the intruder has narrowed things down to m or fewer suspects, he cannot narrow down who is φ to fewer than n. In the case where $n = m$, proving this property is like saying "OK, let's assume for the sake of argument that the intruder has narrowed it down to m suspects. By this property, he cannot do any better than that." This is stronger than a simple bound on intruder knowledge: it is a bound even when we assume a given degree of knowledge for the intruder.

Exposed
$\Box_I\ \varphi(P)$
We say a formula is exposed if the intruder knows the truth of the formula, i.e., he knows exactly who it is that φ.

5.3 Other Characterizations of Anonymity

Ours is by no means the first attempt to characterize anonymity. Reiter and Rubin present a range of "degrees of anonymity" in [21] from "absolute privacy" to "provably exposed". There are two important differences between their approach and ours. First, their definitions are not given in a formal language and are not designed to have a formal specification or analysis. Second, their approach is probabilistic while ours is possibilistic. We will return to this point presently.

A formal characterization of anonymity has been given in terms of CSP in [22]. The basic idea there is to describe a system by means of a process P and a renaming function f and to consider a system anonymous if mapping the process to the image of f and back yields the same process. Space precludes a clear setting out of their characterization. Put no doubt too succinctly, with respect to our characterization above, the parameters allow one to vary the principal P and the formula φ and perhaps the intruder doing the observation. Thus, one can capture many different condenda and different intruders. However, it appears that there is only one condens that they consider. On the other hand, they have the advantage of expressing things entirely in terms of CSP, which is a well understood formalism. The logic in this paper is meant as an alternative, not a replacement for the CSP approach. As different people have different tastes regarding the approach with which they are comfortable, it is good to have alternatives. One approach seems to have a more succinct and intuitive expression of properties while the other has an existing framework and analysis tool. In any case, they are not necessarily mutually exclusive. It is conceivable that one could have a process algebra semantics for a logic such as in this paper. We might thereby take a step towards combining the advantages of theorem provers and model checker, such as in the NRL Protocol Analyzer.

Like Schneider and Sidiropoulos, our characterizations of anonymity are possibilistic rather than probabilistic. And, like them we would hope to bring in probabilistic language at some point. However, there is reason to think that most of the contributions will occur on the possibilistic front.

First, it is often difficult to assign probabilities. In our case this is both because we are concerned with the nonprobabilistic behaviour of users at the system interface and because any assignment of probabilities based on expected behavior may be altered by an active intruder. Assigning probabilities can also be misleading if not done correctly. For example, if 99 out of 100 remailers only forward messages from one client to a second remailer, we might be tempted to think that messages coming through the second remailer have a 99 percent chance of being from that client. But, a moments reflection will show this to be incorrect. Second, even when probabilities can be assigned, adding probabilistic expressiveness to a formal language usually greatly adds to the complexity of specification and analysis.

Both of these points are well illustrated in the information flow security literature. The basic concept of noninterference as introduced in [15] is possibilistic, and most of the analysis, system design, and development of related properties that has gone on since then has been of a possibilistic nature. In fact, the only substantial systems built to date that have been been designed to be noninterfering in any sense have taken a possibilistic approach. Nonetheless, it is possible to give a probabilistic characterization of noninterference [4, 17]. And,a system satisfying these probabilistic properties is clearly more secure. Nonetheless, virtually no significant design or analysis has been done in this area, no doubt due to the complexity. (Some recent encouraging advances have been made by Volpano and Smith [27].) This state of affairs has been mirrored on the formal level as well. There have been possibilistic characterizations of many possibilistic noninterference properties in a variety of formalisms, including notably epistemic logic [3, 14]. And, there have even been some epistemic characterizations of probabilistic noninterference [24]. But, again, most of the development as well as discussion of more complex systems has been in terms of possibilistic properties. Our expectation is that the situation is likely to be analogous when formally analyzing anonymity. Probabilistic characterizations may still be applied to substantial systems, for example *Crowds*, but it is unclear if these will prove both general and amenable to formal specification or analysis.

6 Knowledge Programs

Systems and environments that we discuss will be specified via *knowledge programs* following the approach of Fagin et al. [11, 12]. All our knowledge programs have the following form:

case of
  if [*knowledge test* #1]
   do [*action* #1]
  if [*knowledge test* #2]
   do [*action* #2]
  ⋮
end case

Knowledge tests are conjunctions of formulae where each conjunct is preceded by $\Box_P$ or $\neg\Box_P$ for some principal P. Actions are performed by the principal running the program. Each action given in the consequent of a clause may be a series of actions to be performed by the principal. The knowledge test and action given in any one clause are considered atomic. At any one time, at most one clause of the program will fire. Also, in a properly specified knowledge program, knowledge tests should be mutually exclusive. Thus, at any one time, only one clause of a properly specified program can fire. In the execution of a knowledge program, recent actions, defined in the model in section 2, are ones taken during the execution of the current clause.[2]

As noted above in section 2, the system environment can be viewed as a group principal made up of many smaller environments. We will now examine this point in more detail. Our reasons are at least twofold: (1) The environment programs we will set out presently are very simple. Thus, they serve as an accessible introduction to knowledge programs. (2) The environment programs we will set out are generic and will be used to describe the environment for subsequently presented examples.

6.1 Generic Environment Programs

The following programs describe environments between the various principals. Recall that we assume message delivery is guaranteed; all uncertainty, delay, etc. is reflected in the behavior of the environment. Note also that the clauses for environment principals are often simpler than for system principals. This is because the environments we set out here are not doing anything based on message content other than the **to** or **from** fields; they simply forward any message they receive or block it, possibly recording the events. More sophisticated environments, e.g., doing selective forwarding based on message content or timing, are possible. We will not describe them here.

We typically assume a single environment between any two system principals. This we call a *pairwise*[3] environment. In some sense, the communication graph for the system is fully connected, but with an environment principal between any two system principals (much as in [26], although our environments need not be hostile). However, in practice many of these environment principals will

[2] Unlike the "knowledge-based programs" of [11, 12], our knowledge programs do not have "standard tests" (those not involving epistemic operators) because we have yet to see a need for these tests in any of the examples we have looked at; although, there is no reason they could not be added in if needed. There are also more important differences. We have placed all uncertainty in the principals (including explicitly represented environment principals). Thus, e.g., all sent messages are received, and all received messages were sent by someone, albeit possibly an environment principal.

[3] It is might seem natural to call these 'atomic environments'. However, a complex environment that, e.g., forwards messages between two principals based on the traffic it sees between two others could not be reduced to such atomic principals. Hence, this would be a misnomer. Detailed discussion of such environments is beyond the scope of this paper.

simply block any transmission they receive. And, we will not bother to specify these in cases where there is obviously no direct communication between the two principals or we do not care if there is (and so can assume that there is not). Also, we are often more interested in an environment principal that is a distributed group of the pairwise environment principals just mentioned, for example, when we consider several distinct clients sending queries through the Anonymizer. We now give some examples of basic environments, from which more complex environments can be built.

A reliable environment between principals is one that simply passes any messages sent between them without any alteration, delay, recording, etc.

Reliable_Environment_Program :
  if $[\Box_E\ E$ **received from** P_i "M **to** P_j" $\wedge\ \neg\Box_E\ E$ **said to** P_j "M **to** P_j "$]$
  do $[$`send`$(M$ **to** $P_j,\ E,\ P_j)\ ]$

A remembering environment between principals is just like the reliable environment except that it keeps track of all messages it passes.

Remembering_Environment_Program :
  if $[\Box_E\ E$ **received from** P_i "M **to** P_j" $\wedge\ \neg\Box_E\ E$ **said to** P_j "M **to** P_j "$]$
  do $[$`record(receive(`M **to** $P_j,\ E,\ P_i)\ )$;
    `send`$(M$ **to** $P_j,\ E,\ P_j,$remember$)\ ]$

A simple blocking environment is one that simply blocks (drops) all messages that pass through it. It thus does no action, i.e., Λ. But, to explicitly contrast it with the next environment, we give it the following redundant description.

Simple_Blocking_Environment_Program :
  if $[\Box_E\ E$ **received from** P_i "M **to** P_j" $\wedge\ \neg\Box_E\ E$ **said to** P_j "M **to** P_j "$]$
  do $[\Lambda]$

A remembering blocking environment is one that blocks (drops) all messages that pass through it, but records the message receptions.

Remembering_Blocking_Environment_Program :
  if $[\Box_E\ E$ **received from** P_i "M **to** P_j" $\wedge\ \neg\Box_E\ E$ **said to** P_j "M **to** P_j "$]$
  do $[$`record(receive(`M **to** $P_j,\ E,\ P_i)\ )\ ]$

An environment may forward only messages sent from or to a selected principal (possibly a group principal). By selecting which traffic it forwards, the environment may reveal traffic information to other parts of the intruder elsewhere, e.g., in a system employing chained remailers or other forwarding mechanisms. A pairwise environment that selects based on sender or receiver would be trivial. It would simply block (or forward) all messages in one direction and block or forward all messages in the other direction. This is thus the first presented example of an environment that will typically only be used to describe an environment that is a group principal. We set out an example of an environment that selectively forwards only messages from a particular principal, P_0. The program itself is virtually identical to that of the remembering environment, except that it only forwards if the sending principal is the particular principal specified.

Sender_Selecting_Environment_Program(P_0) :
if [$\Box_E$ (E **received from** P_i "M **to** P_j" $\wedge$ $P_i = P_0$) $\wedge$
$\neg\Box_E$ E **said to** P_j "M **to** P_j "]
do [`record(receive(`M **to** P_j, E, P_i)) ;
`send(`M **to** P_j, E, P_j, remember)]

Despite the fact that some environments are not reducible to pairwise environments, pairwise environments will serve as basic building blocks in many cases. We therefore find it useful to refer to them succinctly. Let '$E_{P_iP_j}$' denote the environment between system principals P_i andP_j. Thus, $E_{P_iP_j}$ **runs** *Program* means that messages between P_i and P_j are delivered according to Program. Note also that this is meant to cover messages in both directions. Thus, we assume $E_{P_jP_i} = E_{P_iP_j}$.

6.2 Theorems for Environment Programs

In the course of our analysis, we will have to assess what various principals have or have not done and what they know or don't know. This information comes primarily from the program specifications, the assumptions about who is running what program, and what initial messages are sent and facts known. A main way we are able to formally derive things based on the knowledge programs is by means of program theorems. These have the general form:

$$(P \textbf{ runs } \textit{Program} \ \wedge \textit{precondition}) \ \supset \ \textit{postcondition}$$

However, for the purposes of the analysis we do in this paper, we can more specifically assume that the only way for the postcondition to obtain is for the principal to run the program and the precondition to hold. This allows us to strengthen the form to:

$$(P \textbf{ runs } \textit{Program} \ \wedge \textit{precondition}) \ \leftrightarrow \ \textit{postcondition}$$

We present examples of these program theorems below. They can be generated automatically from the corresponding knowledge programs. This will ultimately be useful for automated analysis. For now we must be content to set them out by hand.

A1. (E **runs** Reliable_Environment_Program $\wedge$
E **did** (`receive(`M **to** P_j, E, P_i))) $\leftrightarrow$ E **did** (`send(`M **to** P_j, E, P_j))
A2. (E **runs** Remembering_Environment_Program $\wedge$
E **did** (`receive(`M **to** P_j, E, P_i))) $\leftrightarrow$
`record(receive(`M **to** P_j, E, P_i)) ; `send(`M **to** P_j, E, P_j, remember)
A3. (E **runs** Simple_Blocking_Environment_Program $\wedge$
E **did** (`receive(`M **to** P_j, E, P_i))) $\leftrightarrow$ E **did** (Λ)

A4. (E **runs** Remembering_Blocking_Environment_Program $\wedge$
E **did** (`receive`(M **to** P_j, E, P_i))) $\leftrightarrow$
E **did** (`record`(`receive`(M **to** P_j, E, P_i)))
A5. (E **runs** Sender_Selecting_Environment_Program $\wedge$
`receive`(M **to** P_j, E, P_i) **did** ($\wedge$)$P_i = P_0$) $\leftrightarrow$
E **did** (`record`(`receive`(M **to** P_j, E, P_i)) ;
`send`(M **to** P_j, E, P_j, remember))

7 Anonymizer Example

Space precludes presenting more than the knowledge programs for our example. We here describe the standard analysis procedure to be followed if space permitted. We would begin by setting out the knowledge programs that characterize the system principals when operating properly (uncompromised). We would then proceed to the analysis. This consists of (1) setting out the condenda, (2) giving the contexts, i.e., setting out the specific system and environment principals and the programs they are running, and specifying the intruder (here is where we would specify compromised principals if necessary), (3) giving the program theorems (relating pre- and postconditions to the programs being run), and (4) assessing the anonymity protections afforded by the given programs under the given conditions.

7.1 Anonymizer Knowledge Programs

The Anonymizer [1] is a Web proxy service that receives queries submitted by a client, strips off any identifying information, and forwards the query to the relevant server. When replies are received from the server, it forwards these back to the client. We will now give knowledge programs that specify an anonymizer, a client, and a server. For our purposes, we assume multiple clients and possibly multiple anonymizers; however, it is only necessary to assume one server.

Variables for principal names should be fairly intuitive. We assume that there is one environment $E_{\oplus \mathcal{C} A_j}$ between an anonymizer A_j and the set of clients $\mathcal{C}$, that use it and one environment $E_{A_j S}$ between an anonymizer A_j and a server S. The variable Q represents a query and R represents a response to a query. We also assume that communication between a client C_i and the corresponding user U_i occurs without any intervening environment. For contexts where this is not true, it should be clear how to add in the relevant environment principal.
Client_Program$_{C_i}$:
case of
if [$\Box_{C_i}$ C_i **received** "(Q **for** S) **from** U_i" $\wedge$
$\neg\Box_{C_i}$ C_i **said to** $E_{\oplus \mathcal{C} A_j}$ "(Q **for** S) **from** C_i **to** A_j "]
do [`send`((Q **for** S) **from** C_i **to** A_j, C_i, $E_{\oplus \mathcal{C} A_j}$, remember)]
if [$\Box_{C_i}$ (C_i **received from** $E_{\oplus \mathcal{C} A_j}$ "R **in response to** Q **from** A_j" $\wedge$
C_i **said to** $E_{\oplus \mathcal{C} A_j}$ "(Q **for** S) **from** C_i **to** A_j ") $\wedge$
$\neg\Box_{C_i}$ C_i **said to** U_i "R **in response to** Q **from** S "]

do [send(R **in response to** Q **from** S, C_i, U_i) ;
purge(send((Q **for** S) **from** C_i **to** A_j, C_i, $E_{\oplus \mathcal{C} A_j}$))]
end case

Anonymizer_Program$_{A_j}$:
case of
if [$\Box_{A_j}$ A_j **received from** $E_{\oplus \mathcal{C} A_j}$ "(Q **for** S) **from** C_i **to** A_j" $\wedge$
$\neg\Box_{A_j}$ A_j **said to** $E_{A_j S}$ "Q **from** A_j **to** S "]
do [send(Q **from** A_j **to** S, A_j, $E_{A_j S}$, remember) ;
purge(receive((Q **for** S) **from** C_i **to** A_j, A_j, $E_{\oplus \mathcal{C} A_j}$)) ;
record(receive((Q **for** S) **from** C_i **to** A_j, A_j, $E_{\oplus \mathcal{C} A_j}$))]
if [$\Box_{A_j}$ (A_j **received from** $E_{A_j S}$ "R **in response to** Q **from** S" $\wedge$
A_j **received from** $E_{\oplus \mathcal{C} A_j}$ "(Q **for** S) **from** C_i **to** A_j") $\wedge$
$\neg\Box_{A_j}$ A_j **said to** $E_{\oplus \mathcal{C} A_j}$ "(R **in response to** Q **from** S) **from** A_j **to** C_i "]
do [send((R **in response to** Q **from** S) **from** A_j **to** C_i, A_j, $E_{A_j C_i}$) ;
purge(receive((Q **for** S) **from** C_i **to** A_j, A_j, $E_{A_j C_i}$)) ;
purge(send(Q **from** A_j **to** S, A_j, $E_{A_j S}$))]
end case

Server_Program$_S$:
case of
if [$\Box_S$ S **received from** $E_{A_j S}$ "Q **from** A_j **to** S" $\wedge$
$\neg\Box_S$ S **said to** $E_{A_j S}$ "R **in response to** Q **from** S **to** A_j "]
do [send(R **in response to** Q **from** S **to** A_j, S, $E_{A_j S}$) ;
record(receive(Q **from** A_j **to** S, S, $E_{A_j S}$))]
end case
The above assumes the server logs queries (but not responses).

7.2 Anonymizer Condenda

As noted above, we have no space to set out our analysis. Nonetheless, we at least state the condenda that the Anonymizer is expected to hide. The following are examples of formulae that should be hidden from the intruder. The operating environment and the nature of the intruder will be set out below, in addition to demonstrations of the level of condendum hiding afforded against specified intruders in specified environments.

G1 C_i **said** (Q **for** S)
G2 C_i **received** R **in response to** Q **from** S
G3 C_i **received** R **in response to** Q
G4 S **said** R **in response to** Q $\supset$ C_i **said** Q

8 Conclusion

We have introduced the basic notion of a group principal and an associated model, language, and logic. We have demonstrated the utility of these by defining anonymity properties and specifying an anonymity protocol. Space limitations preclude presenting the analysis of that protocol with respect to anonymity.

Even if we had space to set it out, the assessment by hand of anonymity in the example we have specified is tedious and complex. In fact it would be infeasible to provide the quantitative by-hand assessments of anonymity we envision for complex systems involving many principals. However, with the theory established in this paper, we have a starting point for investigating suitable automated analysis techniques such as incorporating the use of model checkers.

Another direction for future work is the analysis of other types of security properties using our characterization of group principals. In particular, we believe we can ultimately give a characterization of such things as threshold cryptography and proactive security.

Acknowledgements: We thank Cathy Meadows and the anonymous referees for helpful comments and suggestions.

References

1. The Anonymizer. `http://www.anonymizer.com`
2. Michael Burrows, Martín Abadi, and Roger Needham. *A Logic of Authentication*, Research Report 39, Digital Systems Research Center, February 1989.
3. P. Bieber and F. Cuppens. A logical view of secure dependencies. *Journal of Computer Security*, 1(1):99–129, 1992.
4. Randy Browne. *Stochastic Non-Interference: Temporal Stochastic Processes Without Covert Channels.* Odyssey Research Associates, Ithaca, NY, November 1989. unpublished manuscript.
5. R. Canetti, R. Gennaro, A. Herzberg, and D. Naor. "Proactive Security: Long-term Protection Against Break-ins", *CryptoBytes*, vol. 3, no. 1, Spring 1997, pp. 1–ff. (Available at `http://www.rsa.com/rsalabs/pubs/cryptobytes/`)
6. D. Chaum. "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", *Communications of the ACM*, v. 24, n. 2, Feb. 1981, pp. 84-88.
7. D. Chaum, "Security without Identification: Transaction Systems to Make Big Brother Obsolete", CACM (28,10), October 1985, pp. 1030–1044.
8. Brian F. Chellas. *Modal Logic: An Introduction*, Cambridge University Press, Cambridge, 1980.
9. L. Cottrell. *Mixmaster and Remailer Attacks,* `http://obscura.obscura.com/~ loki/remailer/remailer-essay.html`
10. D. Dolev and A. Yao, "On the Security of Public Key Protocols", *IEEE Transactions on Information Theory*, 29(2): 198–208, March 1983.
11. R. Fagin, J. Halpern, Y. Moses, and M. Vardi, *Reasoning About Knowledge*, The MIT Press, 1995.
12. Ronald Fagin, Joseph Y. Halpern, Yoram Moses, and Moshe Y. Vardi, "Knowledge-Based Programs", *Distributed Computing* 10(4): 199–225, 1997.
13. P. Gemmell. "An Introduction to Threshold Cryptography", *CryptoBytes*, vol. 2, no. 3, Winter 1997, pp. 7–12. (Available at `http://www.rsa.com/rsalabs/pubs/cryptobytes/`)

14. Janice Glasgow, Glenn MacEwen, and Prakash Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, August 1992.
15. J. A. Goguen and J. Meseguer. Security policies and security models. In *Proceedings of the 1982 IEEE Computer Society Symposium on Security and Privacy*, Oakland, CA, 1982.
16. Robert Goldblatt. *Logics of Time and Computation,* 2^{nd} *edition*, volume 7 of *CSLI Lecture Notes*, CSLI Publications, Stanford, 1992.
17. James W. Gray, III. Toward a mathematical foundation for information flow security. *Journal of Computer Security*, 1(3):255–294, 1992. A preliminary version appears in Proc. 1991 IEEE Symposium on Research in Security and Privacy, Oakland, CA, May, 1991.
18. C. Gülcü and G. Tsudik. "Mixing Email with *Babel*", *1996 Symposium on Network and Distributed System Security*, San Diego, February 1996.
19. R. Ostrovsky and M. Yung. "How to Withstand Mobile Virus Attacks", *Proc.* 10^{th} *ACM Symposium on Principles of Distributed Computation (PODC91)*, ACM Press, 1991, pp. 51–59.
20. M. Reed, P. Syverson, and D. Goldschlag. "Protocols using Anonymous Connections: Mobile Applications", *Security Protocols:* 5^{th} *International Workshop, Paris, France, April 1997, Proceedings*, B. Christianson, B. Crispo, M. Lomas, and M. Roe, eds., LNCS vol. 1361, Springer-Verlag, 1998, pp. 13–23.
21. M. Reiter and A. Rubin. *Crowds: Anonymity for Web Transactions*, DIMACS Technical Reports 97-15, April 1997 (revised August 1997).
22. S. Schneider and A. Sidiropoulos. "CSP and Anonymity", *ESORICS 96*, E. Bertino, H. Kurth, G. Martella, and E. Montolivio, eds., LNCS vol. 1146, Springer-Verlag, 1996, pp. 198–218.
23. P. Syverson, D. Goldschlag, and M. Reed. "Anonymous Connections and Onion Routing", *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, Oakland, CA, IEEE CS Press, May 1997, pp. 44–54.
24. Paul F. Syverson and James W. Gray, III. The epistemic representation of information flow security in probabilistic systems. In *Proc.* 8^{th} *IEEE Computer Security Foundations Workshop*, pages 152–166, County Kerry, Ireland, June 1995.
25. Stuart G. Stubblebine, Paul F. Syverson, and David M. Goldschlag, "Unlinkable Serial Transactions: Protocols and Applications", to appear in *ACM Transactions on Information Systems and Security.*
26. Paul F. Syverson, "A Different Look at Secure Distributed Computation", in 10^{th} *IEEE Computer Security Foundations Workshop (CSFW10)*, IEEE CS Press, pp. 109–115, June 1997.
27. Dennis Volpano and Geoffrey Smith, "Probabilistic Noninterference in a Concurrent Language", in 11^{th} *IEEE Computer Security Foundations Workshop (CSFW11)*, IEEE CS Press, pp. 34–43, June 1998.
28. Peter Wayner. *Digital Cash: Commerce on the Net*, AP Professional, Chestnut Hill, Mass., 1996.