An Epistemic Foundation for Authentication Logics (Extended Abstract)

Joseph Y. Halpern
Cornell University
Ithaca, NY, USA
halpern@cs.cornell.edu

Ron van der Meyden
University of New South Wales
Sydney, Australia
meyden@cse.unsw.edu.au

Riccardo Pucella
Forrester Research
Cambridge, MA, USA
riccardo@acm.org

While there have been many attempts, going back to BAN logic, to base reasoning about security protocols on epistemic notions, they have not been all that successful. Arguably, this has been due to the particular logics chosen. We present a simple logic based on the well-understood modal operators of knowledge, time, and probability, and show that it is able to handle issues that have often been swept under the rug by other approaches, while being flexible enough to capture all the higher-level security notions that appear in BAN logic. Moreover, while still assuming that the knowledge operator allows for unbounded computation, it can handle the fact that a computationally bounded agent cannot decrypt messages in a natural way, by distinguishing strings and message terms. We demonstrate that our logic can capture BAN logic notions by providing a translation of the BAN operators into our logic, capturing belief by a form of probabilistic knowledge.

1 Introduction

For over 25 years now, there has been an intuition in the world of security that formal theories of knowledge and belief should have something interesting to say about security protocols. Many logics have been designed that embody this intuition. One of the earliest and the most discussed is BAN logic [10]. While BAN logic has been the subject of many (quite legitimate!) criticisms, we believe that there are important features of the BAN approach that have been lost in more recent approaches such as model checking [32, 37], inductive-assertions methods [41], strand spaces [45], and process calculi [1]: namely, the ability to express intuitions of protocol designers regarding notions such as belief, trust, freshness, and jurisdiction. Such high-level abstractions play a significant role in informal reasoning about security protocols. It would be desirable for such intuitive ideas concerning the protocol specifications to be reflected in formal specifications, and for informal arguments concerning such notions to be reflected in formal proofs.

In this paper, we argue that a modal logic with standard notions of knowledge, probability, and time, together with atomic predicates that capture messages *sent* and *received* by an agent, and a predicate that we call extract which characterizes an agent's ability to extract information from messages, can capture most of the higher-level abstractions that we seem to need. We show that such a logic is able to handle issues that have often been swept under the rug by other approaches, and is flexible enough to capture the higher-level security notions that appear in BAN logic. We do this by providing a translation of the BAN operators into our logic, capturing belief by a form of probabilistic knowledge, and showing that the translation satisfies the BAN inference rules. The translation highlights some subtleties in the BAN framework, including some that were missed by earlier authors.

Logics in the BAN tradition have long struggled to reconcile the information-theoretic semantics of logics of knowledge and belief with the computational aspects of cryptography, which raise a version of the *logical omniscience problem*. Suppose that i sends j the message $\mathbf{m}'$, where $\mathbf{m}'$ is $\mathbf{m}$ encrypted by a

To appear in EPTCS.

shared key k. Does j know that i has sent $\mathbf{m}$ encrypted by k? If j does not have the key k, then, intuitively, the answer is no; agent j has no way of knowing that $\mathbf{m}'$ is the result encrypting $\mathbf{m}$ by k. Of course, if j were not computationally bounded, then j could figure out that $\mathbf{m}'$ is indeed $\mathbf{m}$ encrypted by k. Standard approaches to modeling knowledge treat agents as computationally unbounded; in particular, agents are assumed to know all valid formulas. Since (given a fixed encryption framework, and assuming unique encryptions) the fact that $\mathbf{m}'$ is the result of encrypting $\mathbf{m}$ by k is a valid mathematical statement, all agents will know it. This is the logical omniscience problem. There have been attempts to overcome this problem: Cohen [14], for instance, deals with it using what seems to us a rather complicated semantics for knowledge involving permutations (see Section 4 for details).

We propose a simpler and arguably far more intuitive approach that allows us to retain the standard semantics for knowledge. It has been common in the literature on authentication logics to represent the complex message that is the result of encrypting a message $\mathbf{m}$ by a key k using the *term* $\{\mathbf{m}\}_k$ in both the syntax and semantics of the logic. We depart from this approach by distinguishing two views of messages. The first views a message simply as a string of symbols; the second views the message as a term with structure. When j receives the message $\mathbf{m}'$, j knows that it received (the string) $\mathbf{m}'$. What j does not know is that it received $\mathbf{m}$ encrypted by k; j considers it possible that $\mathbf{m}'$ is $\mathbf{m}''$ encrypted by k'', or that $\mathbf{m}'$ is not the encryption of any message. To model this, we consider both strings and terms. What is sent or received are strings; we use the notation $\mathsf{s} = [\mathbf{m}]$ to denote that s is the string that represents the message (term) $\mathbf{m}$. We also allow for "impossible" runs where $\mathsf{s} = [\mathbf{m}']$; that is, we allow the agent to be uncertain as to what message is represented by the string s (even when s representing $\mathbf{m}$ is a fact of mathematics). Using such impossible runs, we can easily model the fact that i may know that s represents the encryption of some message, even though i does not know which message it is the encryption of (in all runs that i considers possible, $\mathsf{s} = [\{\mathbf{m}'\}_{k'}]$ for some message $\mathbf{m}'$ and key k') or that i knows that encryptions are unique, or that s represents the encryption of a message of length at most 20. We believe that this approach to dealing with logical omnisicience should be useful beyond the context of this paper.

2 A Logic for Security Properties

2.1 Syntax

We use a modal logic for reasoning about security protocols. We assume a finite set of principals that for simplicity we represent by integers, a set $\mathscr{K}$ of keys, a set $\mathscr{N}$ of nonces, a set $\mathscr{T}$ of plaintexts, and a set Φ of (application-specific) primitive propositions. We assume that $\mathscr{K}$ contains both symmetric keys (used in shared-key cryptography) and asymmetric keys (used in public-key cryptography), and that they can be distinguished. We also assume that keys, nonces, and plaintexts can be distinguished, so that $\mathscr{K}$, $\mathscr{N}$, and $\mathscr{T}$ are disjoint sets, and that encrypted messages can be distinguished from unencrypted messages.

Like Abadi and Tuttle [3] (AT from now on) and other BAN successors, we assume that formulas can state properties of messages, and can also be used in messages. Thus, we define formulas and messages simultaneously as follows, where we use p for a generic element of Φ, i for a generic principal (or agent), $\mathbf{m}$ for a generic message, t for a generic plaintext, k for a generic key, n for a generic nonce, α for a generic real number in $[0,1]$, $\mathbf{s}$ for a generic term of type string, s for a generic concrete string, x for a generic variable ranging over strings, and φ for a generic formula. As usual, a concrete string is a sequence of symbols from some alphabet Σ. We view messages both as strings and as terms with structure. When we write $\mathbf{m}$, we are thinking of the message as a term with structure, as is made clear in

the following grammar:

$$\begin{array}{rcl} \mathbf{s} & ::= & \mathsf{s} \mid x \\ \mathbf{m} & ::= & t \mid k \mid n \mid i \mid (\mathbf{m}_1, \mathbf{m}_2) \mid \{\mathbf{m}\}_k \mid \varphi \\ \varphi & ::= & p \mid \mathsf{sent}_i(\mathbf{s}) \mid \mathsf{recv}_i(\mathbf{s}) \mid \mathsf{extract}_i(\mathbf{m}) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid K_i\varphi \mid \bigcirc\varphi \mid \\ & & \ominus\varphi \mid \Box\varphi \mid \boxminus\varphi \mid \Pr_i(\varphi) \geq \alpha \mid \exists x\,\varphi \mid [\mathbf{m}] = \mathbf{s} \mid \mathbf{s} \sqsubseteq \mathbf{s}'. \end{array}$$

Besides the application-specific primitive propositions, we also have "built-in" primitive propositions of the form $\mathsf{sent}_i(\mathbf{s})$, $\mathsf{recv}_i(\mathbf{s})$, and $\mathsf{extract}_i(\mathbf{m})$. Note that agents send and receive strings, not message terms. The proposition $\mathsf{extract}_i(\mathbf{m})$ holds if i can "extract" the message $\mathbf{m}$ from strings it has received (and other information at its disposal). Exactly what extract means depends on the application, the capabilities of principals, and the protocol they are running. For now, we make no assumptions, viewing it as a black box. (In Section 3.2, we give a concrete implementation of extract capturing the Dolev-Yao capabilities.)

The knowledge operator $K_i\varphi$ states that agent i knows the fact φ. The temporal operator $\bigcirc\varphi$ states that φ is true at the next time step, while $\ominus\varphi$ states that φ was true at the previous time step, if any. We will use the abbreviations $\bigcirc^l\varphi$ and $\ominus^l\varphi$ (for $l \in \mathbb{N}$) for the l-fold application of $\bigcirc$ and $\ominus$, respectively, to φ. The temporal operator $\Box\varphi$ states that φ is true at the current time, and all subsequent times. Similarly, $\boxminus\varphi$ states that φ is true at the current time, and all previous times. The formula $\Pr_i(\varphi) \geq \alpha$ says that the formula φ holds with probability at least α, according to agent i. The range of quantification is strings: the formula $\exists x\,\varphi$ says that there exists a string x such that φ holds. The construction $[\mathbf{m}] = \mathbf{s}$ says that the string $\mathbf{s}$ is the encoding of the message $\mathbf{m}$. That is, we assume that every message is represented as a string. We also assume that there is a pairing function that maps pairs of strings to strings, and an encryption function that maps strings and keys to strings. We discuss this in more detail below. Finally, $\mathbf{s} \sqsubseteq \mathbf{s}'$ says that the string $\mathbf{s}'$ can be constructed from $\mathbf{s}$ and other strings using the pairing and encryption functions described in Section 2.2. We use the usual abbreviations, and write $\circledS\varphi$ for $\ominus\varphi \wedge \neg\ominus\mathbf{false}$ (φ was true at the previous step, and there was a previous step).

2.2 Semantics

A *multiagent system* [19] consists of n agents and an environment, each of which is in some *local state* at a given point in time. We briefly review the relevant details here.

We assume that an agent's local state encapsulates all the information to which the agent has access. In the security setting, the local state of an agent might include some initial information regarding keys, the messages it has sent and received, and perhaps the reading of a clock. The *environment state* describes information relevant to the analysis that may not be in any agent's state. A *global state* has the form $(st_e, st_1, \ldots, st_n)$, where st_i is agent i's state, for $i = 1, \ldots, n$, and st_e is the environment state. In general, the actual form of these local states depends on the application.

We define a *run* to be a function from time to global states. A *point* is a pair (r,m) consisting of a run r and a time $m \in \mathbf{N}$. At a point (r,m), the system is in some global state $r(m)$. If $r(m) = (st_e, st_1, \ldots, st_n)$, then we take $r_i(m)$ to be st_i, agent i's local state at the point (r,m), and $r_e(m)$ to be st_e, the environment state. We formally define a *system* to consist of a set $\mathscr{R}$ of runs.

For simplicity, we restrict attention to a specific class of systems, suited to modeling security protocols. These are message-passing systems in which one (or more) of the agents is an adversary with the capacity to monitor and control message transmission. Messages have compositional structure, but are transmitted as strings. We assume that the local state of an agent at time m is a sequence of the form

$\langle e_0, e_1, \ldots, e_m \rangle$, where e_0 is the initial state (which typically contains the keys and nonces that i is initially aware of), and e_i for $i \geq 1$ is a set of events of the form $send(j, \mathsf{s})$ or $recv(\mathsf{s})$ where s is a string and j is an agent. We assume that messages (as strings) are sent or received during a *round*, where round m takes place between times $m-1$ and m. Event $send(j, \mathsf{s})$ is in $r_i(m)$ if i sends string s in round m of run r, intending that it be delivered to agent j, while event $recv(\mathsf{s})$ is in $r_i(m)$ if i receives string s in round m of run r. Note that the sender is not included in $recv(\mathsf{s})$. Intuitively, this is because the receiver may not be able to determine the sender of a message it receives. For an event x, we abuse notation and write $x \in r_i(m)$ to denote that $x \in e_k$ for some $k \leq m$. The initial state represents information such as the public and private keys to be used during the run, and other values such as nonces to be used by the agent during the run.

As we said, we distinguish between strings and message terms, and we allow agents to be "confused" about what term a string represents. We use the initial environment state of a run to encode the relationship between terms and strings. Specifically, we take $r_e(0)$ to include a collection of equations of the form $[\mathbf{m}] = \mathsf{s}$, with exactly one such equation for each message $\mathbf{m}$. We write $[\mathbf{m}]_r$ to denote the string s such that $[\mathbf{m}] = \mathsf{s}$ is in $r_e(0)$. There may be some constraints on the relationship between strings and terms. For example, in contexts where all messages are commonly known to be encoded by unique strings, we would require that there is no run r and no message terms $\mathbf{m} \neq \mathbf{m}'$ such that $[\mathbf{m}]_r = [\mathbf{m}']_r$. We now discuss some assumptions that we make for the purposes of this paper; others are discussed in Section 3.3:

- Keys k and their inverses k^{-1} are strings, and represent themselves in all runs; that is, $[k]_r = k$ and $[k^{-1}]_r = k^{-1}$ for all keys k and runs r. Similarly, principals (agents), plaintexts, nonces, and messages in the form of formulas are also represented as strings, and represent themselves.
- There is a pairing function on strings, so that if s, s' are strings, then there is another string that we denote $(\mathsf{s}, \mathsf{s}')$. Moreover, we assume that the string representing $(\mathbf{m}_1, \mathbf{m}_2)$ is the pairing of the strings representing $\mathbf{m}_1$ and $\mathbf{m}_2$; that is, $[(\mathbf{m}_1, \mathbf{m}_2)]_r = ([\mathbf{m}_1]_r, [\mathbf{m}_2]_r)$ for all runs r.
- There is a *run-dependent* encryption function on strings. That is, given a string s and a key k, there is another string that we denote $[\{\mathsf{s}\}_k]_r$ that we can think of as the result of encrypting s by k in run r. We do *not* assume that $[\{\mathsf{s}\}_k]_r = [\{\mathsf{s}\}_k]_{r'}$ for all runs r and r'. An agent may be "confused" about how s is encrypted.
- We define $[\{\mathbf{m}\}_k]_r = [\{\mathsf{s}\}_k]_r$ if $[\mathbf{m}]_r = \mathsf{s}$. That is, agents "understand" that a message is encrypted by means of an operation on the string that encodes the message.
- Encryption is unique: if $[\{\mathbf{m}\}_k]_r = [\{\mathbf{m}'\}_{k'}]_r$, then $[\mathbf{m}]_r = [\mathbf{m}']_r$ and $k = k'$ for all runs r. (This assumption is critical in the use of encryption as an authentication mechanism, and is typically assumed in the literature on authentication logics.) Moreover, $[\{\mathbf{m}\}_k]_r$ is distinct from any plaintext, nonce, key, agent name, or pairing.

Because we want to reason about probabilities, we work with *interpreted (probabilistic) systems* of the form $\mathscr{I} = (\mathscr{R}, \pi, \mathscr{C}, \{\mu_C\}_{C \in \mathscr{C}})$, where $\mathscr{R}$ is a system, π is an interpretation for the propositions in Φ that assigns truth values to the primitive propositions at the global states, $\mathscr{C}$ is a partition of the runs in $\mathscr{R}$ into cells, and for each cell $C \in \mathscr{C}$, μ_C is a probability distribution on the runs in C. The assumption is that agents are using a possibly randomized protocol, while the adversary is using a protocol that combines possibly non-probabilistic choices (such as choosing an agent to attack) with probabilistic moves. Cell C "factors out" the nonprobabilistic choices, so that, in all the runs in C, only probabilistic choices are made. This allows us to put a probability μ_C on the runs in C. We do not assume a single probability

distribution on $\mathscr{R}$, since that would require us to put a probability on the possible protocols that the adversary is using. (See [27] for further discussion of this approach.)

We restrict the possible interpretations π so as to fix a particular interpretation for the primitive propositions $\mathsf{recv}_i(\mathsf{s})$ and $\mathsf{sent}_i(\mathsf{s})$. Specifically, we require that

- $\pi(r(m))(\mathsf{recv}_i(\mathsf{s})) = \mathbf{true}$ iff $recv(\mathsf{s}) \in r_i(m)$,
- $\pi(r(m))(\mathsf{sent}_i(\mathsf{s})) = \mathbf{true}$ iff $send(j,\mathsf{s}) \in r_i(m)$, for some j.

Given our interpretation of extraction as a black box, we put no constraints here on how π interprets $\mathsf{extract}_i(\mathbf{m})$; however, we do assume that extraction is monotonic, in the sense that once an agent is able to extract a message, it will be able to extract it at all future times:

- If $\pi(r(m))(\mathsf{extract}_i(\mathbf{m}))) = \mathbf{true}$ and $m \leq n$ then $\pi(r(n))(\mathsf{extract}_i(\mathbf{m})) = \mathbf{true}$.

As we would expect,

- $\pi(r(m))([\mathbf{m}] = \mathsf{s}) = \mathbf{true}$ if $[\mathbf{m}]_r = \mathsf{s}$ (i.e., if $[\mathbf{m}] = \mathsf{s}$ is in $r_e(0)$).

Roughly speaking, the formula $\mathsf{s} \sqsubseteq \mathsf{s}'$ says that s' can be constructed from s and other strings using pairing and encryption. Since how encryption works on strings is run-dependent, we first define, for each run r, a relation $\sqsubseteq_r$ on strings as the smallest reflexive and transitive relation such that $\mathsf{s} \sqsubseteq_r (\mathsf{s},\mathsf{s}')$, $\mathsf{s}' \sqsubseteq_r (\mathsf{s}',\mathsf{s})$, and $\mathsf{s} \sqsubseteq_r [\{\mathsf{s}\}_k]_r$. We now take

- $\pi(r(m))(\mathsf{s} \sqsubseteq \mathsf{s}') = \mathbf{true}$ if $\mathsf{s} \sqsubseteq_r \mathsf{s}'$.

The reader may wonder why we defined the $\sqsubseteq_r$ relation on strings, rather than defining it as the subterm relation on messages. This formulation allows us to model a situation where we have $\mathsf{s} \sqsubseteq_r \mathsf{s}'$ because $\mathsf{s} = [\mathbf{m}]_r$ and $\mathsf{s}' = [\{\mathbf{m}\}_k]_r$, but the agent does not know this, since it does not realize that $\mathsf{s}' = [\{\mathbf{m}\}_k]_r$. Thus, it considers a run r' possible where $\mathsf{s} \not\sqsubseteq_{r'} \mathsf{s}'$ (because, for example, we may have $\mathsf{s}' \neq [\{\mathbf{m}\}_k]_{r'}$ even if $\mathsf{s} = [\mathbf{m}]_{r'}$).

As usual [19, 28], we say that agent i knows a fact φ at a point (r,m) if φ is true at all points i cannot distinguish from (r,m), and define i's indistinguishability relation $\sim_i$ by taking $(r,m) \sim_i (r',m')$ if $r_i(m) = r'_i(m')$. Despite making these standard choices, we do not suffer from the usual logical omniscience problems, exactly because of our distinction between strings and message terms, and the fact that we allow "impossible" runs where the string corresponding to a message is not the one that is mathematically determined.

Given our assumption that $r_i(m)$ has the form $\langle e_0,\ldots,e_m\rangle$, where e_0 is i's initial state and e_i for $i > 1$ is a set of events of the form $send(j,\mathsf{s})$ or $recv(\mathsf{s})$ describing the messages that i sent and received, it follows that $\mathscr{R}$ is a *synchronous* system where agents have *perfect recall* [19].

Considering synchronous systems with perfect recall makes it relatively straightforward to give semantics to formulas of the form $\Pr_i(\varphi) \geq \alpha$. But having a probability on runs does not suffice to give semantics to a formula such as $\Pr_i(\varphi) \geq \alpha$ at a point (r,m). To do this, we need to go from probabilities on runs to probabilities on points. Given a point (r,m), let C_r be the unique cell in $\mathscr{C}$ such that $r \in C_r$. Let $\mathscr{K}_i(r,m)$ denote the set of points that agent i cannot distinguish from (r,m), that is, the set $\{(r',m') \mid (r,m) \sim_i (r',m')\}$. Let $\mathscr{C}(r)$ be the set of points where the runs are in C_r, that is, $\mathscr{C}(r)$ consists of all the points of the form (r',m') with $r' \in C_r$. The probability μ_{C_r} on the runs of cell C_r induces a probability $\mu_{r,m,i}$ on the points in $\mathscr{K}_i(r,m) \cap \mathscr{C}(r)$ in a straightforward way. If $U \subseteq \mathscr{K}_i(r,m) \cap \mathscr{C}(r)$, define

$$\mu_{r,m,i}(U) = \frac{\mu_{C_r}(\{r' : (r',m) \in U\})}{\mu_{C_r}(\{r' : (r',m) \in \mathscr{K}_i(r,m) \cap \mathscr{C}(r)\})}.$$

The relation of satisfaction of a formula φ without free variables in an interpreted system $\mathscr{I} = (\mathscr{R},\pi)$ at point (r,m), written $(\mathscr{I},r,m) \models \varphi$, is defined inductively, as follows:

$(\mathscr{I},r,m) \models p$ iff $\pi(r(m))(p) = \mathbf{true}$

$(\mathscr{I},r,m) \models \neg\varphi$ iff $(\mathscr{I},r,m) \not\models \varphi$

$(\mathscr{I},r,m) \models \varphi_1 \wedge \varphi_2$ iff $(\mathscr{I},r,m) \models \varphi_1$ and $(\mathscr{I},r,m) \models \varphi_2$

$(\mathscr{I},r,m) \models K_i\varphi$ iff for all $(r',m') \sim_i (r,m)$, $(\mathscr{I},r',m') \models \varphi$

$(\mathscr{I},r,m) \models \bigcirc\varphi$ iff $(\mathscr{I},r,m+1) \models \varphi$

$(\mathscr{I},r,m) \models \ominus\varphi$ iff $m = 0$ or $(\mathscr{I},r,m-1) \models \varphi$

$(\mathscr{I},r,m) \models \Box\varphi$ iff for all $m' \geq m$, $(\mathscr{I},r,m') \models \varphi$

$(\mathscr{I},r,m) \models \boxminus\varphi$ iff for all $m' \leq m$, $(\mathscr{I},r,m') \models \varphi$

$(\mathscr{I},r,m) \models \Pr_i(\varphi) \geq \alpha$ iff $\mu_{r,m,i}(\{(r',m') \mid (\mathscr{I},r',m') \models \varphi\} \cap \mathscr{K}_i(r,m) \cap \mathscr{C}(r)) \geq \alpha$

$(\mathscr{I},r,m) \models \exists x\,\varphi$ if, for some concrete string s, $(\mathscr{I},r,m) \models \varphi[\mathsf{s}/x]$, where $\varphi[\mathsf{s}/x]$ is the result of replacing all free occurrences of x in φ by s.

As usual, we say that φ is *valid* in $\mathscr{I}$, written $\mathscr{I} \models \varphi$, if $(\mathscr{I},r,m) \models \varphi$ for all points (r,m) in $\mathscr{I}$.

We define the probabilistic knowledge operator $K_i^\alpha\varphi$ as an abbreviation for $K_i(\Pr_i(\varphi) \geq 1-\alpha)$. This operator simply means that, essentially, no matter which cell C the agent thinks the current point is in, the probability of φ according to i in that cell is at least $1-\alpha$.

As stated above, we do not impose any *a priori* restrictions on the interpretation of $\mathsf{extract}_i(\mathbf{m})$. Intuitively, the interpretation of extract is meant to capture the capabilities of the agents to take apart messages and construct new messages. While in principle a principal may be able to extract $\mathbf{m}_1$ from $(\mathbf{m}_1,\mathbf{m}_2)$, it may not do so at a particular point in a system, because the protocol that it is using does not break up $(\mathbf{m}_1,\mathbf{m}_2)$. Similarly, whether i can extract $\mathbf{m}$ from $\{\mathbf{m}\}_k$ depends in part on whether i "has" key k in some sense, i's protocol, and i's computational ability. We provide an interpretation of extract that captures the Dolev-Yao adversary in Section 3. However, we stress that many other interpretations of extract are possible, such as interpretations that capture guess-and-validate adversaries [33].

3 Interpreting BAN Logic

One of our claims is that the logic we introduced in Section 2 is a good foundation for security protocol logics. Burrows, Abadi, and Needham [10] developed BAN logic from similar intuitions, taking belief as a primitive rather than knowledge and probability. To provide evidence for our claim, we show how we can interpret the constructs of BAN logic by essentially rewriting them into the simpler primitives of our logic. Although we focus here on BAN logic as an example, we believe that we could similarly reconstruct other related logics.

3.1 Definition of BAN Logic

We reformulate the syntax of BAN logic, along the lines of AT. The set of formulas and set of messages are defined by mutual induction, using the grammar below. Note that messages are defined just as in Section 2.1.

$$\begin{array}{rcl} \mathbf{m} & ::= & t \mid k \mid n \mid i \mid (\mathbf{m},\mathbf{m}') \mid \{\mathbf{m}^i\}_k \mid F \\ F & ::= & i \textbf{ believes } F \mid i \textbf{ controls } F \mid i \textbf{ sees } \mathbf{m} \mid i \textbf{ said } \mathbf{m} \mid i \overset{k}{\leftrightarrow} j \mid \overset{k}{\mapsto} j \mid \textbf{fresh}(\mathbf{m}). \end{array}$$

R1. $\dfrac{i \textbf{ believes } j \overset{k}{\leftrightarrow} i \quad i \textbf{ sees } \{F^l\}_k \quad l \neq i}{i \textbf{ believes } j \textbf{ said } F}$

R2. $\dfrac{i \textbf{ believes } j \textbf{ said } (F, F')}{i \textbf{ believes } j \textbf{ said } F}$

R3. $\dfrac{i \textbf{ believes fresh}(F) \quad i \textbf{ believes } (j \textbf{ said } F)}{i \textbf{ believes } j \textbf{ believes } F}$

R4. $\dfrac{i \textbf{ believes } j \textbf{ controls } F \quad i \textbf{ believes } j \textbf{ believes } F}{i \textbf{ believes } F}$

R5. $\dfrac{i \textbf{ sees } (F, F')}{i \textbf{ sees } F}$

R6. $\dfrac{i \textbf{ believes } j \overset{k}{\leftrightarrow} i \quad i \textbf{ sees } \{F^l\}_k \quad l \neq i}{i \textbf{ sees } F}$

R7. $\dfrac{i \textbf{ believes } \overset{k}{\mapsto} i \quad i \textbf{ sees } \{F^l\}_k \quad l \neq i}{i \textbf{ sees } F}$

R8. $\dfrac{i \textbf{ believes fresh}(F)}{i \textbf{ believes fresh}((F, F'))}$

R9. $\dfrac{i \textbf{ believes } i \overset{k}{\leftrightarrow} j}{i \textbf{ believes } j \overset{k}{\leftrightarrow} i}$

Figure 1: BAN inference rules

The superscript i in $\{\mathbf{m}^i\}_k$ represents a "from"-field, intended to indicate the original sender of the message. The intuitive reading of the formulas is as follows: i **believes** F means that principal i believes formula F; i **controls** F means that principal i is an authority on or has authority or jurisdiction over F; i **sees m** means that i has received a message containing **m**; i **said m** means that principal i at some time sent a message containing **m** and, if **m** is a formula F that was sent recently, that i believes F; **fresh**(**m**) means that message **m** was sent recently; $i \overset{k}{\leftrightarrow} j$ means that principals i and j can use the shared key k to communicate (and that the key is a *good* key; we discuss what counts as a good key below); $\overset{k}{\mapsto} j$ means that key k is j's public key (and that the key is a good key) .

BAN logic uses inference rules to derive new formulas from others. These capture the intended meaning of the primitives. The most significant rules appear in Figure 1.

3.2 A Probabilistic Interpretation

We now define a translation from BAN formulas to formulas in our logic. We write F^T to denote the result of translating the BAN logic formula F to a formula in our logic. Since formulas include messages and are messages, we also need to translate messages; $\mathbf{m}^M$ denotes the translation of a message in the BAN framework to a message in our framework. Note that F^T (the translation of F viewed as a formula) is slightly different from F^M (the translation of F viewed as a message), in that the former is of type formula, whereas the latter is of type message.

The translation of messages that are not formulas is defined inductively in the obvious way: for a primitive message $\mathbf{m}$, $\mathbf{m}^M = \mathbf{m}$, and $(\mathbf{m}_1, \mathbf{m}_2)^M = (\mathbf{m}_1^M, \mathbf{m}_2^M)$. We translate encryptions $\{\mathbf{m}^i\}_k$ by treating the "from"-field as concatenated to the end of the encrypted message; thus, $\{\mathbf{m}^i\}_k^M = \{(\mathbf{m}^M, i)\}_k$. The translation F^M of a formula F viewed as a message is F^T, where F^T is the translation of F viewed as a formula.

Our translation for **believes** is based on a definition of belief due to Moses and Shoham [38]. They assume that an agent operates with a set of default assumptions, expressed as a formula A. An agent's belief in φ, relative to assumptions A, can then be captured by the formula $K_i(A \Rightarrow \varphi)$. That is, the agent believes φ relative to assumptions A if it knows that φ holds under assumptions A. We use a probabilistic version of this idea. Like AT, we use a set of good runs for the assumptions relative to which the agent reasons. Intuitively, these are the runs in which undesirable events such as the adversary guessing a nonce do not occur. We differ from AT in the way that the set of good runs is obtained. AT define the good

runs by a complicated fixed point construction based on the original set of beliefs ascribed to the agents by the BAN logic analysis. We allow any set of runs to be taken as the good runs, but typically, the prior probability of the set of good runs would be high (a fact that can be expressed in the logic) so that agents have reasonable grounds to trust the conclusions they can draw from the assumption that a run is good. Moreover, for the soundness of one axiom, we need agents to always assign positive probability to a run being good.

The particular choice of good runs used in proving that a protocol satisfies a BAN logic specification will depend on the details of the protocol and the system used to model the behaviour of the adversary. Let *good* be a primitive proposition that expresses "the run is good". We take the translation of i **believes** F to be

$$\neg K_i^0 \neg good \wedge K_i^0(good \Rightarrow F^T) .$$

The second clause says that i believes F if it knows with probability 1 that when a run is a good, F^T is true. The interpretation of belief as knowing with probability 1 is standard in the economics literature. We have modified this so as to make belief depend only on what happens in the good runs. The first clause, $\neg K_i^0 \neg good$, requires that the set of good runs has positive probability in at least one cell. This prevents an agent from vacuously believing a fact just because it knows that the probability of a run being good is 0.

The translation of $(i$ **sees** $\mathbf{m})^T$ is $K_i(\mathsf{extract}_i(\mathbf{m}^M))$. Thus, agent i "sees" $\mathbf{m}$ if it knows that it has extracted it. We work with this translation here because it helps to satisfy R1, but it is not the only candidate. Another reasonable translation, corresponding to the statement that i has received a string that he knows contains the encoding of $\mathbf{m}$, is $\exists x, y([\mathbf{m}^M] = x \wedge \mathsf{recv}_i(y) \wedge K_i(x \sqsubseteq y))$. The difference lies in whether i **sees** $\mathbf{m}$ is meant to imply that i knows how $\mathbf{m}$ is composed.

Roughly speaking, BAN interpret i **said** $\mathbf{m}$ as "$\mathbf{m}$ was a submessage of a message that i sent at some point in the past". The BAN reading of **said** also involves claims about belief; BAN assumes that all formulas said recently by i are believed by i. We do not make this assumption in our translation, because we do not view it as an intrinsic part of **said**. Rather, we capture it in the systems for which we prove the translation sound. Given this, we translate i **said** $\mathbf{m}$ as

$$\exists x, y([\mathbf{m}^M] = x \wedge \Diamond\!\!\!\!\circledS(\neg \mathsf{sent}_i(y) \wedge \bigcirc \mathsf{sent}_i(y) \wedge K_i(x \sqsubseteq y)).$$

Thus, roughly speaking, i said $\mathbf{m}$ if at some point strictly in the past i sent a string $y = \mathsf{s}'$, and i knew at the beginning of the round in which s' was sent that $x = [\mathbf{m}^M]_r$ was a substring of s'. (There are some significant subtleties in this translation that relate to a known error in AT identified in [44]; we expand on this in Section 3.4.)

Capturing that k is a good key between i and j depends on what we mean by "good key". There are at least two possible interpretations. One is that no one other than possibly i and j has extracted the key. Accordingly, we take $(i \overset{k}{\leftrightarrow} j)^T$ to be $\mathsf{extract}_i(k) \wedge \mathsf{extract}_j(k) \wedge \bigwedge_{i' \neq i,j} \neg \mathsf{extract}_{i'}(k)$. This translation would not hold in protocols where the key k is provided to i and j by a key server. To cover this, AT propose the interpretation "no one but i and j sends messages encrypted with k" for the length of the protocol interaction. We could encode this also, as well as other ways to make explicit the beliefs of the agents about the behaviour of the key server. (See Section 3.4 for more discussion of good keys.)

Formula $\overset{k}{\mapsto} j$ says that k is j's public key, and that the key is a good key. The formula is intended to mean that only j knows the key k^{-1}. Thus, its translation is similar in spirit to that of the formula for shared keys, and the same comments apply; we take $(\overset{k}{\mapsto} j)^T$ to be $\mathsf{extract}_j(k^{-1}) \wedge \bigwedge_{\{i:i \neq j\}} \neg \mathsf{extract}_i(k^{-1})$. Of course, situations involving key escrow would require a different translation. Again, the strength of our approach is that it allows us to easily express such variants.

A message is fresh if it could not have been sent, except possibly recently, where "recently" means "in the last l steps". We leave it to the user to decide what counts as "recently", by choosing a suitable l. Thus, the translation of **fresh**(**m**) is

$$\exists x([\mathbf{m}^M] = x \wedge \ominus^l \bigwedge_i \boxminus(\neg\exists y(\neg \mathsf{sent}_i(y) \wedge \bigcirc \mathsf{sent}_i(y) \wedge x \sqsubset y))).$$

While it may capture a notion relevant to reasoning about replay attacks, this notion of freshness does not capture what we believe should be meant by a nonce being "good". Intuitively, this is due to the requirement for unpredictability of the nonce; we return to this issue in Section 4.

We interpret i **controls** F as "i believes F if and only if F is true". Thus, the translation of i **controls** F is $K_i^0(good \Rightarrow F^T) \Leftrightarrow F^T$. This captures, to some extent, the intuition that i is an authority on F. Roughly speaking, there is no way for F to change without agent i knowing it, so F is in some sense "local" to agent i.

This completes the translation. For the language that does not include the **controls** operator, the translation is linear. A BAN F formula is translated to a modal formula whose length is linear in $|F|$. With **controls** in the language, the translation becomes exponential. It is not clear that formulas with nested occurrences of **controls** arise naturally. For the language with no nested occurrences of **controls**, the translation is again linear.

One other comment: although we have called our translation a "probabilistic translation", we in fact use probability in only a limited way. The only probabilistic statements that we make are "with probability 1" (K_i^0) and "with probability greater than 0" ($\neg K_i^0 \neg$). To capture this, we could have simply used a standard belief operator (that satisfies the axioms of the modal logic KD45), and avoided dealing with probability altogether. We have used probability here because in the full paper we consider a more general probabilistic translation, which is also sound, where we take believing to be knowing with some probability α. The main consequence of this more general interpretation is that the translation of the BAN inference rules is more constrained. For example, if a BAN inference rule involves beliefs in the antecedent and in the conclusion of the rule, the probability associated with those beliefs must be related. We leave further details to the full paper.

3.3 Evaluating the Interpretation

To what extent does the translation above capture BAN logic? The minimum we can ask for is that the translation validates the inference rules of BAN logic. This is what we argue in this section.

In order to validate the BAN inference rules, we need to restrict to systems that satisfy certain properties. Intuitively, these restrictions are made implicitly by BAN logic, and must be made explicit in order to prove the soundness of the translation.

We say that agents *have no additional prior information beyond guesses* in an interpreted system $\mathscr{I}$ if the initial states of all agents includes all public keys, their own private keys, the nonces required by their protocol (in the case of nonadversary agents), a finite set of other keys or nonces they have guessed, and nothing else. We also need to make precise the intuition that agents tell the truth, since BAN logic assumes that when a (nonadversary) agent sends a formula, it believes the formula. Without this requirement, we cannot ensure the validity of R3. Implicit in the notion of honesty is the idea that an agent does not forge "from"-fields in messages. Furthermore, BAN logic assumes that agents' capabilities of creating and decomposing messages are those characterized by the Dolev-Yao model. We capture these capabilities, together with the assumption that agents not forge "from"-fields, by providing a suitable interpretation of extract. The idea is to define a set of strings $can\_generate_i(r,m)$, which should

be thought of as the set of strings that i can generate given the information it has in state $r_i(m)$. There are two ways that i can generate a string. It can pair strings that it can generate to form a more complicated string, or it can "pick apart" a pair into its components.

Suppose that we have a function $init(st)$ that, given a local state $st = \langle e_0, e_1, \ldots, e_m \rangle$, returns the set of strings contained in the initial state e_0 (roughly speaking, these are the keys and nonces that i is initially aware of). Given a point (r,m), define $can\_generate_i(r,m)$ to be the smallest set S of strings satisfying the following conditions:

(1) $\{\mathsf{s} : recv(\mathsf{s}) \in r_i(m)\} \cup \{\mathsf{s} : \mathsf{s} \in init(r_i(m))\} \cup \{j : j \text{ is an agent}\} \cup \{\varphi : \varphi \text{ is a formula}\} \subseteq S$;

(2) $(\mathsf{s}, \mathsf{s}') \in S$ iff $\mathsf{s}, \mathsf{s}' \in S$;

(3) if $[\{\mathsf{s}\}_k]_r \in S$ and $k^{-1} \in S$, then $\mathsf{s} \in S$ (recall that if k is symmetric key, then we identify k and k^{-1});

(4) if $\mathsf{s}, k \in S$, then $[\{\mathsf{s}\}_k]_r \in S$.

We now want to connect $can\_generate_i(r,m)$ with what i knows about message terms at the point (r,m). We make some additional assumptions regarding what agents know. Specifically, if $(r,m) \sim_i (r',m')$, then we assume that (a) if $[\{\mathsf{s}\}_k]_r$ and k^{-1} are both in $can\_generate_i(r,m)$, then $[\{\mathsf{s}\}_k]_r = [\{\mathsf{s}\}_k]_{r'}$, and (b) if $\mathsf{s}, k \in can\_generate_i(r,m)$, then $[\{\mathsf{s}\}_k]_r = [\{\mathsf{s}\}_k]_{r'}$. Roughly speaking, (a) says that if i "has" the decryption key k^{-1} and "has" the string $[\{\mathsf{s}\}_k]_r$, which is the encryption of s under k in r, then i knows that $[\{\mathsf{s}\}_k]_r$ is the encryption of *string* s under k, while (b) says that if i "has" s and k, then i knows that $[\{s\}_k]_r$ is the encryption of s under k.

With these assumptions, as the following result shows, $can\_generate_i(r,m)$ depends only on i's local state at (r,m).

Proposition 1. *If $r_i(m) = r'_i(m')$, then $can\_generate_i(r,m) = can\_generate_i(r',m)$.*

We say interpreted system $\mathscr{I}$ *models agent i as a Dolev-Yao agent* if for all $m \geq 0$ and all messages $\mathbf{m}$,

(1) $\pi(r(m))(\mathsf{extract}_i(\mathbf{m})) = \mathbf{true}$ if and only if $[\mathbf{m}]_r \in can\_generate_i(r,m)$, and

(2) if $send(i,\mathsf{s}) \in r_i(m+1)$ and $send(i,\mathsf{s}) \notin r_i(m)$, then $\mathsf{s} \in can\_generate_i(r,m)$.

Note that the second clause restricts agents to sending messages that they can generate. We place a further restriction on what are called "nonforging" agents. Although a nonforging agent i can generate all the messages in $can\_generate_i(r,m)$, we assume that, when sending a message, i does not forge signatures; that is, i will not send a message with a "from"-field that states that the message is from some other agent. Define $can\_generate_i^{\mathrm{NF}}(r,m)$ just as $can\_generate_i(r,m)$, except that condition (4) is replaced by the following variant:

(4′) If $\mathsf{s}, k \in S$, then $[\{\mathsf{s}, i\}_k]_r \in S$.

Rule (4′) ensures that when the agent constructs an encrypted message, it includes a "from"-field set to its own name. The definition of a *nonforging Dolev-Yao agent* is just like that of a Dolev-Yao agent, except for the use of $can\_generate_i^{\mathrm{NF}}(r,m)$ instead of $can\_generate_i(r,m)$ in the second clause.

Finally, we say that an agent i is honest if, whenever i says something, then i believes that it will be true when the message is received. (The honesty assumption is how we capture BAN's requirement on **said** that agents believe that a formula they are sending is true.) Formally, agent i is an *honest Dolev-Yao agent* in an interpreted system $\mathscr{I}$ if

(1) i is a nonforging Dolev-Yao agent in $\mathscr{I}$, and

(2) for all BAN formulas F,

$$\mathscr{I} \models \exists \mathsf{s}, \mathsf{s}'([F^M] = \mathsf{s} \wedge \neg \mathsf{sent}_i(\mathsf{s}') \wedge \bigcirc \mathsf{sent}_i(\mathsf{s}') \wedge K_i(\mathsf{s} \sqsubseteq \mathsf{s}')) \Rightarrow K_i^0(\bigwedge_{0 \le l' \le l} \bigcirc^{l'} F^T).$$

The intuition for the last condition is that an agent says only things that it believes will still be true some time in the near future when its message is received. Again, this is parameterized by a time l, which should be taken as the same time parameter used to interpret freshness.

Observe that while the restriction to Dolev-Yao agents is hardwired into the definitions of **said** and **sees** by AT, we model it using extract instead. This means that our logic can be used to deal with adversaries other than those that satisfy the Dolev-Yao properties, without changing the underlying syntax and semantics. Similarly, rather than hardwiring honesty into the definition of **said**, we model it as an assumption on the class of systems. We can therefore model the kind of operators BAN advocate without being tied to the particular choices made by BAN and their successors.

A further assumption we need to make for the soundness of R3 is regarding our notion of *good* runs. It says that, in a good run, agents always consider it possible that runs are good (or, more precisely, assign that event positive probability). We say that a system $\mathscr{I}$ *maintains goodness* if, for all agents i and all points (r,m), we have

$$(\mathscr{I}, r, m) \models good \Rightarrow \neg K_i^0 \neg good.$$

Note that in any system where all finite prefixes of runs have positive probability, this axiom will be sound.

We would now like to show that the translation of Section 3.2 preserves the validity of the BAN inference rules. Note that an instance of a BAN inference rule has the form "from F_1 [and F_2] infer F_3". We translate this instance into a formula of the form $F_1^T [\wedge F_2^T] \Rightarrow F_3^T$. Thus, for example, an instance of rule R3 translates to the formula

$$\begin{array}{c}(\neg K_i^0 \neg good \wedge K_i^0(good \Rightarrow (\mathbf{fresh}(F))^T) \wedge K_i^0(good \Rightarrow (j \ \mathbf{said} \ F)^T)) \\ \Rightarrow (\neg K_i^0 \neg good \wedge K_i^0(good \Rightarrow (\neg K_j^0 \neg good \wedge K_j^0(good \Rightarrow F^T)))).\end{array}$$

Note that the translation $(j \ \mathbf{said} \ F)^T$ involves the message translation F^M. This is why, for instance, even if F and F' are equivalent, $(j \ \mathbf{said} \ F)^T$ and $(j \ \mathbf{said} \ F')^T$ may not be, while $(j \ \mathbf{believes} \ F)^T$ and $(j \ \mathbf{believes} \ F')^T$ are.

The following theorem, whose proof is in the full paper, assures us that the translation preserves soundness. In the theorem statement, we use the notation r_{ij}^T to emphasize that the formulas in the translation of r refer to agents i and j,

Theorem 2. *The translation r_{ij}^T of an instance r_{ij} of the BAN inference rule* R1 *is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses and where agent i is a nonforging Dolev-Yao agent. The translation r_{ij}^T of an instance r_{ij} of the BAN inference rule* R3 *is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses, maintain goodness, and where agent j is honest. Finally, the translation r^T of an instance r of* R2 *and* Rn *for $n \ge 4$ is valid in systems that model Dolev-Yao agents that have no additional prior information beyond guesses.*

Soundness tells us that our translated constructs satisfy properties analogous to those satisfied by the original BAN constructs. Of course, there are many translations with this property, some less interesting than others. For example, the translation that sends every BAN formula to the formula *true* also validates the BAN inference rules. We hope that the reader agrees that our translation captures the spirit of the BAN rules.

3.4 Subtleties in the BAN translation

There is an important subtlety in the translation of i **said** $\mathbf{m}$. It is actually *not* quite the case that i must know at time $m'-1$ that $[\mathbf{m}^M]$ is a substring of s'; what i must know at time $m'-1$ is that the string s that represents $\mathbf{m}^M$ in the current run is a substring of s'. There is a big difference between $K_i\exists x([\mathbf{m}]=x\wedge x\sqsubseteq \mathsf{s}')$ and $\exists x([\mathbf{m}]=x\wedge K_i(x\sqsubseteq \mathsf{s}'))$. A few examples might help to explain the distinction. First, suppose that, in run r, j sends i an unencrypted string $\mathsf{s}=[\mathbf{m}]_r$. This means that i can extract $\mathbf{m}$ in run r. Then j sends i the string $\mathsf{s}'=[\{\mathbf{m}\}_k]_r$. Finally, i forwards s' to some other player j' in round m' of r. Since i does not have the key k at time $m'-1$, the beginning of the round when it sends s' to j', i does not realize that s' represents the encryption of $\mathbf{m}$. That is, although $\mathsf{s}\sqsubseteq_r \mathsf{s}'$, there may be a run r' such that $r'_i(m'-1)=r_i(m'-1)$ but $[\{\mathsf{s}\}_k]_{r'}\neq \mathsf{s}'$. Thus, $K_i(\mathsf{s}\sqsubseteq \mathsf{s}')$ does *not* hold at $(r,m'-1)$ (although $\mathsf{s}\sqsubseteq \mathsf{s}'$ does). As a consequence, i **said** $\mathbf{m}$ does not hold at (r,m).

Now suppose that $\mathbf{m}_2=\{\mathbf{m}_1\}_{k'}$, and that in run r, $[\mathbf{m}_2]_r=\mathsf{s}$, where k' is a key that i does not have, and j sends i the string $\mathsf{s}'=[\{\mathbf{m}_2\}_k]_r$, where k is a shared key between i and j, and i then forwards s' to j' in round m'. Under natural asssumptions, since i has key k, i "understands" that s' is the encryption of s by k, so in all runs r' that i considers possible at $(r,m'-1)$, $\mathsf{s}\sqsubseteq \mathsf{s}'$. Thus, $(\mathscr{I},r,m'-1)\models [\mathbf{m}_2]=\mathsf{s}\wedge K_i(\mathsf{s}\sqsubseteq \mathsf{s}')$, so $(\mathscr{I},r,m'-1)\models \exists x([\mathbf{m}_2]=x\wedge K_i(x\sqsubseteq \mathsf{s}'))$. On the other hand, $(\mathscr{I},r,m'-1)\models \neg K_i\exists x([\mathbf{m}_2]=x\wedge x\sqsubseteq \mathsf{s}')$. Since i cannot decrypt $\mathbf{m}_2$, it may well be that $[\mathbf{m}_2]_{r'}\neq \mathsf{s}$ in some run r' that i considers possible.

As we show, our translation of **said** makes R1 sound; the alternative translation

$$\exists y(\diamondsuit\circledS(\neg \mathsf{sent}_i(y)\wedge \bigcirc \mathsf{sent}_i(y)\wedge K_i\exists x([\mathbf{m}^M]=x\wedge x\sqsubseteq y))$$

does not. It is not clear exactly which translation most closely captures what BAN had in mind for **said**. We suspect that they were not aware of these subtleties. One piece of evidence in support of this suspicion is that, as previously noted by Syverson and van Oorschot [44] the AT translation of **said** does not make R1 sound (despite AT's claims to the contrary). They run into trouble precisely on examples like our second example, with nested encryptions. Note that we are not claiming that our translation is the "right" translation of **said**, although something like our translation seems to be needed to make R1 sound. We may instead want to consider a different translation, and interpret **said** differently. What is important is that our logic helps us clarify the relevant issues.

The meaning of "good key" in our gloss of the formula $i \overset{k}{\leftrightarrow} j$ is also not as simple as we have made out. Many protocols studied in the literature assume the existence of a key server in charge of distributing session keys to principals. In such a context, a good key is not only known to the principals exchanging messages, but also of course to the server that initially distributed the key. In some sense, the interpretation of "good key" depends on details of the protocol being executed, such as whether it requires a key server. This to us suggests that "good key" is not an appropriate primitive; it is a complex notion that is too protocol dependent. (Moreover, we would argue that it is better to make explicit in the server specification the allowed server behavior with respect to the keys that it generates, rather than hide this in a primitive of the logic.) In any case, we can easily accommodate such a definition of "good key" with trusted servers by assuming that the server, as well as i and j, can extract the key. For simplicity, however, we consider only the interpretation of "good key" given above. Note that both BAN and AT interpret k being a good key as a statement that also talks about the future; in essence, if k is a good key, it remains so throughout a protocol interaction. We can capture such an interpretation by prefixing the translated formula by a $\Box$ operator. Of course, this interpretation precludes the analysis of protocols that leak the key value. (See Nessett [39] and Burrows, Abadi and Needham [11] for discussion.) We would like the analysis to reveal such leaks, rather than presupposing that they do not happen.

There is yet another subtlety. It is consistent with our translation that k is a good key between i and j, but neither i nor j knows this. One obvious reason is that i and j may consider it possible that the key has leaked. We might then hope that i and j know that, if the run is *good* (so that there is no leakage), then k is a good key. But even this may not be the case under our definition. For example, i may not know that j can extract k. Although we do not need it to show that the BAN axioms are sound, we might consider requiring that, on runs that are *good*, i and j know that a key shared between them is good. More precisely, let $E_G^{good}\varphi$ be an abbrevation $\wedge_{i\in G}K_i(good \Rightarrow \varphi)$; that is, all agents in G know that, if the run is good, then φ holds. We might want to require that $(i \overset{k}{\leftrightarrow} j)^T$ implies $E_{\{i,j\}}^{good}\varphi$ (where φ is the translation we give above).

We could go even further. Rather than just requiring i and j to know that if the run is good, the key is shared, we might want this fact to be common knowledge among i and j. To make this precise, let $(E_G^{good})^{h+1}\varphi$ be an abbreviation for $E_G^{good}((E_G^{good})^h\varphi)$. Define $C_G^{good}\varphi$ so that it holds exactly if $(E_G^{good})^h\varphi$ holds for all $h \geq 1$. We could then consider taking $(i \overset{k}{\leftrightarrow} j)^T$ to be $C_{\{i,j\}}^{good}\varphi$, where φ is the translation we used above. We did not do this, in part because it is not clear that it is really desirable to make such strong requirements for shared keys. For example, suppose that in an environment where message transmission is not completely reliable (so messages may be lost), i and j already have a shared key k, and i decides that it should be refreshed. So i tells j that k should be replaced by k' (using a message encrypted by k). While j can acknowledge receipt of the message and i can acknowledge the acknowledgment, as is well known, no amount of back and forth will make it common knowledge that k' is a shared key, even if all runs are good [25]. Nevertheless, this should not prevent i and j from starting to use key k' as a shared key. Again, the main point we want to make here is not that our translation is the "right" translation (that will typically be application-dependent), but that our logic lets us clarify the issues.

4 Related Work

The goal of understanding the foundations of authentication logic is not new, and goes back to early attempts at providing a semantics for the original BAN logic. As we mentioned, BAN logic was originally defined through a set of inference rules without a semantics tied to the actual protocols that the logic was meant to analyze. The work of Abadi and Tuttle[3] and others [23, 44, 43, 48] sought to provide a direct semantics for both BAN logic and its subsequent generalizations meant to make it more widely applicable. This is not the place to trace the history of BAN logic, but a big point of contention has always been the idealization needed to be performed on the protocols [34]. Idealization can be understood as a way to ascribe a "meaning" to the various steps of a protocol, and some work has gone towards understanding such idealization [34, 30]. In contrast to providing a direct semantics and an account of idealization, we instead supply a semantics to BAN logic operators by decomposing them into more primitive and well-understood logical operators; our semantics makes it possible to avoid idealization altogether by analyzing the multiagent systems generated by the protocol under consideration, subject to an abstraction of cryptography that captures that agents have uncertainty about how cryptography works.

Several logics for reasoning about security protocols based on knowledge and not subject to idealization have been proposed, going as far back as CKT5 [7]. Van der Meyden and Su have used epistemic logic to model check the dining cryptographers protocols [35]. In many such logics (e.g., [4, 46]) knowledge or belief is not interpreted as truth at all possible worlds, but rather as a form of algorithmic knowledge [26], much as with our extract primitive, but with a fixed semantics. Surveys of the application of epistemic logic to reason about security protocols include [18, 42]. In general, work in this area does not use probabilistic operators, as we have done: we think that ultimately, security needs to be analysed

probabilistically.

Cohen and Dam [14, 15, 16] identified some of the same subtleties we identified in the interpretation of the BAN logic operators, but they address those issues differently. They develop two different semantics that use ideas from counterpart theory and the *de dicto/de re* distinction in modal logic to address the logical omniscience problem. Similar to AT, what is sent and received in the semantics are *message terms*, rather than *strings*, as in our work. Both semantics work with permutations ρ on the set of message terms, which also extend to transformations on local states. The first semantics [15] defines knowledge at a global state s as $s \models K_i\varphi$ if for all global states s' and permutations ρ such that $s'_i = \rho(s_i)$, we have $s' \models \rho(\varphi)$. This semantics validates the BAN style axiom $\mathsf{recv}_i(m) \Rightarrow K_i\mathsf{recv}_i(m)$ for all messages m, including messages m such as $\{M\}_k$ in situations where i cannot extract k. The second semantics [16], which, judging from [14], they appear to consider to be the more satisfactory, makes a distinction between semantic message variables x and syntactic message variables m, and allows quantification over both. Here the semantics adds an assignment V from variables to equivalence classes of ground message terms with respect to equations capturing the behaviour of cryptography, such as $\{\{M\}_k\}_k = M$. They define

- $s, V \models K_i\varphi$ if for all global states s' and permutations ρ such that $s'_i = \rho(s_i)$, we have $s', \rho \circ V \models \varphi$.
- $s, V \models \forall x(\varphi)$ if for all equivalence classes e of ground message terms, we have $s, V[x \mapsto e] \models \varphi$.
- $s, V \models \forall m(\varphi[m/x])$ if for all ground message terms M, we have $s, V \models \varphi[M/x]$.

This semantics validates the formula $\forall x(\mathsf{recv}_i(x) \Rightarrow K_i\mathsf{recv}_i(x))$ but not the formula $\forall m(\mathsf{recv}_i(m) \Rightarrow K_i\mathsf{recv}_i(m))$. The latter is equivalent to the conjunction of $\mathsf{recv}_i(M) \Rightarrow K_i\mathsf{recv}_i(M)$ for all messages M. In effect, this semantics treats semantic message variables x somewhat similarly to our *string* interpretation of messages, but whereas our logic treats message terms and strings as having distinct types, they allow formulas such as $x = M$ where a semantic variable x is equated to a message term M. The semantics of our logic more concretely matches an operational semantics, and maintains a type distinction between messages and strings, so we would express the same equivalence as $x = [M]$. It would be interesting to investigate whether there is any formal correspondence between the two approaches. (One apparent obstacle to this is that our semantics allows a situation where an agent considers it possible that a string it has received is the encoding of *no* term (e.g., it is a random string injected in a guessing attack by the adversary) whereas in the Cohen and Dam semantics, everything that is received is semantically an equivalence class of message terms.)

Our approach to logical omniscience is similar to models used in the cryptographic literature in which encryption is modelled as a randomly chosen function, not known to the agents [22], who discover its behaviour by making calls to the function on particular values. Such models can be captured in our framework by appropriate construction of the interpreted system. (It is more common in cryptography, however, to use the *random oracle* model [12], in which the random function represents a hash function rather than an encryption function, and to use this as the basis for the construction of ciphers.) Van Ditmarsch et al [47] use a similar idea, but their approach is purely epistemic and they do not have probability in their models. The connections that we draw to BAN logic are not developed in this work.

We are careful in our interpretation to distinguish between the freshness of a nonce—that it has not been used before—from its unpredictability—how likely it is to be guessed. Freshness is captured by the simple statement that no message containing the nonce was recently sent, while unpredictability is captured by a probability distribution over the choice of nonces during a run of the protocol. Thus, these two aspects of nonces are kept separate. There has been little discussion in the literature about this distinction, and nonces are often implicitly taken to be unpredictable, even though the framework of analysis used technically captures only freshness (for instance, the spi calculus [1], or MSR [13]). When

not required to be unpredictable, nonces can be taken to be sequence numbers, or timestamps. There has been some work discussing the use of timestamps for nonces (e.g., Neuman and Stubblebine [40]). Somewhat related are the notions of authentication tests described by Guttman [24], where a distinction is made between using nonces that, from the perspective of a given agent, should be secret when sent (and therefore should be unpredictable) from those that need to be secret only when received.

The probabilistic interpretation of BAN logic tells us that the symbolic reasoning done in BAN logic can be understood in terms of more realistic probabilistic beliefs, accounting for the probabilities associated with the choice of nonces and the choices of keys. From that perspective, our work resembles a general class of work that seeks to obtain results about more realistic models of cryptography, either by using symbolic reasoning and showing that results of such an analysis can be interpreted quantitatively, or by using a logic that is explicitly more quantitative.

Abadi and Rogaway [2], building on previous work by Bellare and Rogaway [6], compare the results obtained by a symbolic analysis with those obtained by a more computational view of cryptography. They show that, under various conditions, the former is sound with respect to the latter, that is, terms that are assumed indistinguishable in a symbolic analysis remain indistinguishable under a concrete encryption scheme. This work has been extended in various ways (e.g., [36]).

Other formal approaches to security protocol analysis have tried to apply techniques from symbolic analysis directly to more realistic models of cryptography (such as computational cryptography [21]) by viewing messages as strings of bits and adversaries as randomized polynomial-time algorithms. The resulting logics are fundamentally probabilistic, in a way similar to our probability-based interpretation of BAN logic. Examples of such approaches include those of Backes, Pfitzmann, and Waidner [5] and Datta et al. [17], as well as automated tools such as CryptoVerif [8]. In general, it is difficult in such models to express local information, that is, the fact that at a given point in the protocol, a particular agent has certain information. These approaches are geared instead towards more global properties, which say there is some probability of a particular formula holding at all points in the execution (or at the end of the interaction).

5 Conclusion

We have introduced in this paper a simple modal propositional logic to reason about security protocols, based on well-understood modal operators for knowledge, probability, and time. We have shown how those primitive notions can be used to capture the more high-level operators of BAN logic, helping us to understand the intuitions underlying BAN logic and, more importantly, to capture important aspects of reasoning about security protocols that we claim cannot be adequately expressed in a non-epistemic way. A further advantage of the translation is that it allows us to apply well developed model-checking techniques [9, 20, 29, 31, 35] to verifying the correctness of security protocols whose specifications are expressed using BAN logic. (Here it becomes useful that the translation is linear, at least, if we restrict the use of the **controls** operator.)

Our ultimate goal is to design a logic that will be useful for reasoning about all aspects of security, based on the logic we introduced here. In this paper, we have focused on high-level issues concerning the expressibility of high-level operators using well-understood primitive concepts. Other aspects of reasoning about security protocols that we have hinted at need to be further investigated. A particularly significant aspect is how to deal with the issue of what an adversary can compute; in this paper, we have sidestepped the problem using the proposition `extract`. We are currently investigating more general approaches to deal with this issue.

Acknowledgements

We thank the TARK reviewers for their insightful comments. Halpern was supported in part by NSF grants IIS-0911036 and CCF-1214844, by ARO grant W911NF-14-1-0017, and by the Multidisciplinary University Research Initiative (MURI) program administered by the AFOSR under grant FA9550-12-1-0040. Van der Meyden was supported by ARC grant DP120102489.

References

[1] M. Abadi & A. D. Gordon (1999): *A Calculus for Cryptographic Protocols: The Spi Calculus*. *Information and Computation* 148(1), pp. 1–70, doi:10.1006/inco.1998.2740.

[2] M. Abadi & P. Rogaway (2002): *Reconciling Two Views of Cryptography (The Computational Soundness of Formal Encryption)*. *Journal of Cryptology* 15(2), pp. 103–127, doi:10.1007/s00145-007-0203-0.

[3] M. Abadi & M. R. Tuttle (1991): *A Semantics for a Logic of Authentication*. In: *Proc. 10th ACM Symposium on Principles of Distributed Computing*, pp. 201–216, doi:10.1145/112600.112618.

[4] R. Accorsi, D. Basin & L. Viganò (2001): *Towards an awareness-based semantics for security protocol analysis*. In Jean Goubault-Larrecq, editor: *Proc. Workshop on Logical Aspects of Cryptographic Protocol Verification*, *Electronic Notes in Theoretical Computer Science* 55.1, Elsevier Science Publishers, pp. 5–24, doi:10.1016/S1571-0661(04)00242-7.

[5] M. Backes, B. Pfitzmann & M. Waidner (2003): *A Composable Cryptographic Library with Nested Operations*. In: *Proc. 10th ACM Conference on Computer and Communications Security (CCS'03)*, ACM Press, pp. 220–230, doi:10.1145/948109.948140.

[6] M. Bellare & P. Rogaway (1993): *Entity Authentication and Key Distribution*. In: *Proc. 13th Annual International Cryptology Conference (CRYPTO'93)*, *Lecture Notes in Computer Science* 773, Springer-Verlag, pp. 232–249, doi:10.1007/3-540-48329-2_21.

[7] P. Bieber (1990): *A Logic of Communication in Hostile Environment*. In: *Proc. 3rd IEEE Computer Security Foundations Workshop (CSFW'90)*, IEEE Computer Society Press, pp. 14–22, doi:10.1109/CSFW.1990.128181.

[8] B. Blanchet (2008): *A Computationally Sound Mechanized Prover for Security Protocols*. *IEEE Transactions on Dependable and Secure Computing* 5(4), pp. 193–207, doi:10.1109/TDSC.2007.1005.

[9] I. Boureanu, M. Cohen & A. Lomuscio (2009): *Automatic verification of temporal-epistemic properties of cryptographic protocols*. *Journal of Applied Non-Classical Logics* 19(4), pp. 463–487, doi:10.3166/jancl.19.463-487.

[10] M. Burrows, M. Abadi & R. Needham (1990): *A Logic of Authentication*. *ACM Transactions on Computer Systems* 8(1), pp. 18–36, doi:10.1145/77648.77649.

[11] M. Burrows, M. Abadi & R. M. Needham (1990): *Rejoinder to Nessett*. *ACM Operating Systems Review* 24(2), pp. 39–40, doi:10.1145/382258.382790.

[12] R. Canetti, O. Goldreich & S. Halevi (2004): *The random oracle methodology, revisited*. *J. ACM* 51(4), pp. 557–594, doi:10.1145/1008731.1008734.

[13] I. Cervesato, N. Durgin, P. Lincoln, J. Mitchell & A. Scedrov (1999): *A Meta-Notation for Protocol Analysis*. In: *Proc. 12th IEEE Computer Security Foundations Workshop (CSFW'99)*, IEEE Computer Society Press, pp. 55–69, doi:10.1109/CSFW.1999.779762.

[14] M. Cohen (2007): *Logics of Knowledge and Cryptography: Completeness and Expressiveness*. Ph.D. thesis, KTH.

[15] M. Cohen & M. Dam (2005): *Logical Omniscience in the Semantics of BAN Logic*. In: *Proc. Workshop on Foundations of Computer Security (FCS'05)*.

[16] M. Cohen & M. Dam (2007): *A complete axiomatization of knowledge and cryptography*. In: *Proc. 22nd Annual IEEE Symposium on Logic in Computer Science (LICS'07)*, doi:10.1109/LICS.2007.4.

[17] A. Datta, A. Derek, J. C. Mitchell, V. Shmatikov & M. Turuani (2005): *Probabilistic Polynomial-Time Semantics for a Protocol Security Logic*. In: *Proc. 32nd Colloquium on Automata, Languages, and Programming (ICALP'05)*, pp. 16–29, doi:10.1007/11523468_2.

[18] F. Dechesne & Y. Wang (2010): *To know or not to know: epistemic approaches to security protocol verification*. *Synthese* 177(1), pp. 51–76, doi:10.1007/s11229-010-9765-8.

[19] R. Fagin, J. Y. Halpern, Y. Moses & M. Y. Vardi (1995): *Reasoning about Knowledge*. MIT Press.

[20] P. Gammie & R. van der Meyden (2004): *MCK: Model Checking the Logic of Knowledge*. In: *Proc. 16th International Conference on Computer Aided Verification (CAV'04)*, Lecture Notes in Computer Science, Springer-Verlag, pp. 479–483, doi:10.1007/978-3-540-27813-9_41.

[21] O. Goldreich (2001): *Foundations of Cryptography: Volume 1, Basic Techniques*. Cambridge University Press, doi:10.1017/CBO9780511546891.

[22] O. Goldreich, S. Goldwasser & Silvio Micali (1986): *How to construct random functions*. *J. ACM* 33(4), pp. 792–807, doi:10.1145/6490.6503.

[23] L. Gong, R. Needham & R. Yahalom (1990): *Reasoning about Belief in Cryptographic Protocols*. In: *Proc. 1990 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp. 234–248, doi:10.1109/RISP.1990.63854.

[24] J. D. Guttman (2002): *Security protocol design via authentication tests*. In: *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW'02)*, IEEE Computer Society Press, pp. 92–103, doi:10.1109/RISP.1990.63854.

[25] J. Y. Halpern & Y. Moses (1990): *Knowledge and common knowledge in a distributed environment*. *Journal of the ACM* 37(3), pp. 549–587, doi:10.1145/79147.79161.

[26] J. Y. Halpern & R. Pucella (2002): *Modeling adversaries in a logic for reasoning about security protocols*. In: *Proc. Workshop on Formal Aspects of Security (FASec'02)*, *Lecture Notes in Computer Science* 2629, pp. 115–132, doi:10.1145/79147.79161.

[27] J. Y. Halpern & M. R. Tuttle (1993): *Knowledge, probability, and adversaries*. *Journal of the ACM* 40(4), pp. 917–962, doi:10.1145/153724.153770.

[28] J. Hintikka (1962): *Knowledge and Belief*. Cornell University Press.

[29] X. Huang, C. Luo & R. van der Meyden (2011): *Symbolic model checking of probabilistic knowledge*. In: *Proc. Conf. on Theoretical Aspects of Rationality and Knowledge (TARK-2011)*, pp. 177–186, doi:10.1145/2000378.2000399.

[30] D. Kindred & J. M. Wing (1997): *Closing the idealization gap with theory generation*. In: *DIMACS Workshop on Design and Formal Verification of Security Protocols*.

[31] A. Lomuscio, H. Qu & F. Raimondi (2017): *MCMAS: an open-source model checker for the verification of multi-agent systems*. *STTT* 19(1), pp. 9–30, doi:10.1007/s10009-015-0378-x.

[32] G. Lowe (1998): *Casper: A Compiler for the Analysis of Security Protocols*. *Journal of Computer Security* 6, pp. 53–84, doi:10.3233/JCS-1998-61-204.

[33] G. Lowe (2004): *Analysing protocols subject to guessing attacks*. *Journal of Computer Security* 12(1), pp. 83–97, doi:10.3233/JCS-2004-12104.

[34] W. Mao (1995): *An augmentation of BAN-like logics*. In: *Proc. 8th IEEE Computer Security Foundations Workshop (CSFW'95)*, IEEE Computer Society Press, pp. 44–56, doi:10.1109/CSFW.1995.518552.

[35] R. van der Meyden & K. Su (2004): *Symbolic Model Checking the Knowledge of the Dining Cryptographers*. In: *Proc. 17th IEEE Computer Security Foundations Workshop (CSFW'04)*, IEEE Computer Society Press, pp. 280–291, doi:10.1109/CSFW.2004.1310747.

[36] D. Micciancio & B. Warinschi (2004): *Soundness of Formal Encryption in the Presence of Active Adversaries*. In: *Proc. Theory of Cryptography Conference (TCC'04)*, *Lecture Notes in Computer Science* 2951, Springer-Verlag, pp. 133–151, doi:10.1007/978-3-540-24638-1_8.

[37] J. Mitchell, M. Mitchell & U. Stern (1997): *Automated analysis of cryptographic protocols using Murφ*. In: *Proc. 1997 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp. 141–151, doi:10.1109/SECPRI.1997.601329.

[38] Y. Moses & Y. Shoham (1993): *Belief as defeasible knowledge*. *Artificial Intelligence* 64(2), pp. 299–322, doi:10.1016/0004-3702(93)90107-M.

[39] D. M. Nessett (1990): *A Critique of the Burrows, Abadi and Needham Logic*. *ACM Operating Systems Review* 24(2), pp. 35–38, doi:10.1145/382258.382789.

[40] B. C. Neuman & S. Stubblebine (1993): *A Note on the Use of Timestamps as Nonces*. *ACM Operating Systems Review* 27(2), pp. 10–14, doi:10.1145/155848.155852.

[41] L. C. Paulson (1998): *The Inductive Approach to Verifying Cryptographic Protocols*. *Journal of Computer Security* 6(1/2), pp. 85–128, doi:10.3233/JCS-1998-61-205.

[42] R. Pucella (2015): *Knowledge and Security*. In H. van Ditmarsch, J. Y. Halpern, W. van der Hoek & B. Kooi, editors: *Handbook of Epistemic Logic*, College Publications, pp. 591–655.

[43] S. Stubblebine & R. Wright (1996): *An authentication logic supporting synchronization, revocation, and recency*. In: *Proc. 3rd ACM Conference on Computer and Communications Security (CCS'96)*, ACM Press, pp. 95–105, doi:10.1145/238168.238195.

[44] P. F. Syverson & P. C. van Oorschot (1994): *On Unifying Some Cryptographic Protocol Logics*. In: *Proc. 1994 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, pp. 14–28, doi:10.1109/RISP.1994.296595.

[45] F. J. Thayer, J. C. Herzog & J. D. Guttman (1999): *Strand Spaces: Proving Security Protocols Correct*. *Journal of Computer Security* 7(2/3), pp. 191–230, doi:10.3233/JCS-1999-72-304.

[46] B. Toninho & L. Caires (2010): *A Spatial-Epistemic Logic for Reasoning about Security Protocols*. In: *Proc. 8th International Workshop on Security Issues in Concurrency*, pp. 1–15, doi:10.4204/EPTCS.51.1.

[47] H. Van Ditmarsch, J. Van Eijck, I. Hernndez-Antn, F. Sietsma, S. Simon & F. Soler-Toscano (2012): *Modelling cryptographic keys in dynamic epistemic logic with demo*. In J. Bajo Pérez, M. A. Sánchez, P. Mathieu, J. M. Corchado Rodríguez, E. Adam, A. Ortega, M. N. Moreno García, E. Navarro, B. Hirsch, H. L. Cardoso & V. Julián, editors: *Highlights on Practical Applications of Agents and Multi-Agent Systems*, Springer, pp. 155–162, doi:10.1007/978-3-642-28762-6_19.

[48] G. Wedel & V. Kessler (1996): *Formal Semantics for Authentication Logics*. In: *Proc. 4th European Symposium on Research in Computer Security (ESORICS'96)*, *Lecture Notes in Computer Science* 1146, Springer-Verlag, pp. 219–241, doi:10.1007/3-540-61770-1_39.