

exposes assumptions about trusted synchronization paths and synchronization bounds between principals. This establishes requirements for evaluating the adequacy of the underlying clock synchronization algorithm. Also, it provides a metric for comparing protocol designs. For example, by modifying synchronization assumptions to reflect synchronization failures the designer can assess the extent of vulnerability due to a synchronization failure. Finally, exposing synchronization assumptions is a step towards engineering security architectures to minimize risk due to inconsistent knowledge. No synchronization assumptions are inherent to our logic or model. Like other security-relevant features, protocol-specific assumptions about synchronization are included in the premise set for analysis.

In this paper, we illustrated some recency policies and gave examples for reasoning about recency. We have shown how proving recency properties is fundamental to proving the correctness of practical security mechanisms such as replay detection. Authentication subject to recency properties can also prove the continued presence of an entity. Authentication based on recency of beliefs (such as beliefs about the goodness of keys) or recent-secure authentication [11] is fundamental to managing risk due to the inherent problem of obtaining concurrent knowledge in distributed systems. Recency policies place constraints on the recency of statements, such as statements about the goodness of keys and jurisdiction for issuing keys. In effect, recency policies constrain the window of inconsistent knowledge as the protocol progresses.

Revocation, or referencing and negating previously made statements about belief, jurisdiction, and keys, is an important part of practical systems. Our logic provides a general method for expressing revocation policies and for reasoning about revocation. We illustrated some revocation policies and gave an example for reasoning about the revocation of keys.

6 Acknowledgments

We thank Seema Maru for helping us to refine the logic by using it to analyze several protocols.

References

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.

[2] Martín Abadi and Mark R. Tuttle. A semantics for a logic of authentication. In *Proc. 10th Annual ACM Symposium on Principles of Distributed Computing*, pages 201–218, August 1991.

[3] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. Technical Report SRC Research Report 39, Digital Equipment Corporation, February 1989. Revised February 1990.

[4] CCITT. The directory - authentication framework. Technical Report X.509, International Telegraph and Telephone Consultative Committee, February 1993.

[5] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. Reasoning about knowledge. *The MIT Press, Cambridge, Massachusetts*, 1995.

[6] K. Gaarder and E. Sneekenes. Applying a formal analysis technique to the CCITT X.509 strong two-way authentication protocol. *Journal of Cryptology*, 3(2):81–98, 1991.

[7] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. In *Proc. 14th IEEE Symposium on Research in Security and Privacy*, pages 234–248, May 1990.

[8] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in distributed systems: Theory and practice. *ACM Trans. on Comp. Syst.*, 10(4):265–310, 1992.

[9] Y. Moses and B. Bloom. Knowledge, timed precedence and clocks. Technical Report 95-21, Weizmann Institute, July 1995.

[10] B. Clifford Neuman and Theodore Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9), 1994.

[11] Stuart G. Stubblebine. Recent-secure authentication: Enforcing revocation in distributed systems. In *Proc. 19th IEEE Symposium on Security and Privacy*, pages 224–235, May 1995.

[12] Paul F. Syverson. Adding time to a logic of authentication. In *Proc. 1st Conference on Computer and Communication Security*, pages 97–101, November 1993.

[13] Paul F. Syverson and Paul C. van Oorschot. On unifying some cryptographic protocol logics. In *Proc. 18th IEEE Symposium on Research in Security and Privacy*, pages 14–28, May 1994.

Public Signature Verification Keys Signature checking keys are good if they properly identify signatures.

$$(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$$

if and only if (1) $t_1 < \text{Time}_Q(t)$ and (2) $(r, t) \models Q$ received $_{t'}$ $[X]_{K^{-1}}$ implies $(r, t) \models P$ said $_{t'}$ X .

$$(r, t) \models (\overset{K}{\Rightarrow}_{[t_1, t_2], Q} P)$$

if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ for all $t_1 \leq t' \leq t_2$, and

$$(r, t) \models (\overset{K}{\Rightarrow}_{(t_1, t_2), Q} P)$$

if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ for some $t_1 \leq t' \leq t_2$.

Public Encryption Keys A public encryption key must yield messages that cannot be read by anyone other than the principal whose key it is.

$$(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$$

if and only if (1) $t' < \text{Time}_Q(t)$ and (2) $(r, t) \models R$ received $_{t'}$ $\{X\}_K$ and $(r, t) \models R$ received $_{t'}$ X imply either

- a. $R = P$, or
- b. $(\text{generate}(X), t'') \in \mathcal{H}_R$ for some $t'' \leq t'$.

$$(r, t) \models (\overset{K}{\Rightarrow}_{[t_1, t_2], Q} P)$$

if and only if $(r, t) \models (\overset{K}{\Rightarrow}_{t', Q} P)$ for all $t_1 \leq t' \leq t_2$, and

$$(r, t) \models (\overset{K}{\Rightarrow}_{(t_1, t_2), Q} P)$$

if and only if $(r, t) \models (\overset{Q}{\mapsto}_K Pt', Q)$ for some $t_1 \leq t' \leq t_2$.

Believes We define the possibility relation \sim_{P_i} for a principal P_i in a state (r, t) by $(r, t) \sim_i (r', t')$ if $r_i(t) = r'_i(t')$. Hence, \sim_i defines an equivalence relation for P_i among global states. We define belief relative to this equivalence class.

$$(r, t) \models P \text{ believes}_{t'} \varphi$$

if and only if $t' < \text{Time}_P(t)$ and $(r', t') \models \varphi$ at $_P$ t' for all (r', t') such that $(r', t') \sim_P (r, t)$.

$$(r, t) \models P \text{ believes}_{[t_1, t_2]} \varphi$$

if and only if $(r, t) \models P \text{ believes}_{t'} \varphi$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models P \text{ believes}_{(t_1, t_2)} \varphi$$

if and only if $(r, t) \models P \text{ believes}_{t'} \varphi$ for some $t_1 \leq t' \leq t_2$.

Synchronization Principals' clocks are not more than ϵ time units from each other. Synchronization between principals places no constraints on how far clocks drift from real time.

$$(r, t) \models \text{sync}_{t', \epsilon} (P, Q)$$

if and only if $t' < \text{Time}_P(t)$ and $|t' - t''| \leq \epsilon$ for all t'' such that $\text{Time}_Q(r, \text{Start}_P(r, t')) \leq t'' \leq \text{Time}_Q(r, \text{End}_P(r, t'))$.

$$(r, t) \models \text{sync}_{[t_1, t_2], \epsilon} (P, Q)$$

if and only if $(r, t) \models \text{sync}_{t', \epsilon} (P, Q)$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models \text{sync}_{(t_1, t_2), \epsilon} (P, Q)$$

if and only if $(r, t) \models \text{sync}_{t', \epsilon} (P, Q)$ for some $t_1 \leq t' \leq t_2$.

At

$$(r, t) \models \varphi \text{ at}_P t'$$

if and only if $t' \leq \text{Time}_P(t)$ and $(r, t'') \models \varphi$ for all t'' such that $\text{Start}_P(r, t') \leq t'' \leq \text{End}_P(r, t')$.

$$(r, t) \models \varphi \text{ at}_P [t_1, t_2]$$

if and only if $(r, t) \models \varphi \text{ at}_P t'$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models \varphi \text{ at}_P \langle t_1, t_2 \rangle$$

if and only if $(r, t) \models \varphi \text{ at}_P t'$ for some $t_1 \leq t' \leq t_2$.

4.2 Soundness

Our logic is sound, in the sense that any derivation allowed by the logic corresponds to a truth in the model. For any formula φ and set Γ of formulas, we write $\Gamma \models \varphi$ if $(r, t) \models \varphi$ for all (r, t) such that $(r, t) \models \psi$ for every $\psi \in \Gamma$.

Theorem 4.1 *Let Γ be a set of formulas and φ be a formula. If $\Gamma \vdash \varphi$, then $\Gamma \models \varphi$.*

Proof (sketch): Assume $\Gamma \vdash \varphi$. We show by induction that $\Gamma \models \varphi$. If φ is a theorem or a member of Γ , then trivially $\Gamma \models \varphi$. Otherwise, φ was derived by applying one of the inference rules to some formula ψ . We assume as an inductive hypothesis that $\Gamma \models \psi$.

To show that $\Gamma \models \varphi$ if φ was obtained by modus ponens from some ψ such that $\Gamma \models \varphi$, we use a case by case analysis (omitted here) of the axioms of our logic to show that all the axioms preserve truth, in the sense that they cannot be used to derive untrue formulas from true ones. Specifically, for each axiom $\psi \supset \psi'$, we show that if $(r, t) \vdash \psi'$, then $(r, t) \vdash \psi$. It therefore follows that if φ was obtained by modus ponens from ψ , that since $\Gamma \models \psi$, $\Gamma \models \varphi$.

If φ was obtained by necessitation from some ψ such that $\Gamma \models \varphi$, then $\vdash \psi$ and $\varphi = P \text{ believes}_{t_P} \psi$ for some P and t_P . Since ψ is a theorem, $\models \psi$. Hence, for any r, t , and P , $(r, t) \models \psi$. In particular, given a point (r, t) , then for any (r', t') such that $(r, t) \sim_P (r', t')$, $(r', t') \models \psi$. It follows from the truth conditions for believes that $(r, t) \models P \text{ believes}_{t_P} \psi$ for every (r, t) . Hence $\Gamma \models P \text{ believes}_{t_P} \psi$. Since $\varphi = P \text{ believes}_{t_P} \psi$, it follows that $\Gamma \models \varphi$. ■

The full proof of Theorem 4.1 will be included in an expanded version of the paper.

5 Conclusions

We present a logic for reasoning about cryptographic protocols. Our logic supports reasoning about several important properties underlying practical cryptographic protocols: synchronization, recency, and revocation. We represent security policies as implications that are specified as assumptions during the analysis of a protocol. In the course of reasoning about properties of synchronization, recency, and revocation, the security policies advance the flow of reasoning provided the preconditions relating to the property is satisfied. In this way, the analysis yields information about the protocol together with the specified security policies. By evaluating security objectives with respect to the context of particular principals and their policies, one determines whether assumptions of the correctness proof are within the constraints of the policies of the principals.

In practice, bounds on clock synchronization can have a significant impact on the correctness of cryptographic protocols for distributed systems. We demonstrate how our logic

We define our truth conditions in such a way that for nonnegated basic formulas, only formulas about the past can be true. We do this in order to maintain a certain kind of stability of formulas that is important to the soundness of the logic. In particular, we only define statements that can be revoked, such as statements about belief, control, and keys, to be true at a given time on a principal's clock if the principal's clock has changed from that time, since otherwise it would be possible to construct runs in which the same statement is both true and false at the same time.

Primitive Propositions $(r, t) \models p$ if and only if $(r, t) \in \pi(p)$ for primitive proposition p .

Logical Connectives

$(r, t) \models \varphi \wedge \psi$ if and only if $(r, t) \models \varphi$ and $(r, t) \models \psi$.

$(r, t) \models \neg\varphi$ if and only if $(r, t) \not\models \varphi$.

$(r, t) \models (\exists t' : t_1 \leq t' \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for some t' such that $t_1 \leq t' \leq t_2$.

$(r, t) \models (\exists t_1 \geq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for some t_1 such that $t_1 \geq t_2$.

$(r, t) \models (\exists t_1 \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for some t_1 such that $t_1 \leq t_2$.

$(r, t) \models (\forall t' : t_1 \leq t' \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for every t' such that $t_1 \leq t' \leq t_2$.

$(r, t) \models (\forall t_1 \geq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for every t_1 such that $t_1 \geq t_2$.

$(r, t) \models (\forall t_1 \leq t_2)\varphi$ if and only if $(r, t) \models \varphi$ for every t_1 such that $t_1 \leq t_2$.

Relations

$(r, t) \models t_1 \leq t_2$ if and only if $t_1 \leq t_2$.

$(r, t) \models t_1 \geq t_2$ if and only if $t_1 \geq t_2$.

Receives P derives X from the set of received messages and P 's current key set.

$$(r, t) \models P \text{ received}_{t'} X$$

if and only if $t' \leq \text{Time}_P(t)$ and $X \in \text{submsgs}_{\text{Keyset}_P(r, t)}(\text{Msgs}(r, t))$.

$$(r, t) \models P \text{ received}_{[t_1, t_2]} X$$

if and only if $(r, t) \models P \text{ received}_{t'} X$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models P \text{ received}_{\langle t_1, t_2 \rangle} X$$

if and only if $(r, t) \models P \text{ received}_{t'} X$ for some $t_1 \leq t' \leq t_2$.

Has

$$(r, t) \models P \text{ has}_{t'} K$$

if and only if $t' \leq \text{Time}_P(t)$ and $K \in \text{Keyset}_P(r, t')$.

$$(r, t) \models P \text{ has}_{[t_1, t_2]} K$$

if and only if $(r, t) \models P \text{ has}_{t'} K$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models P \text{ has}_{\langle t_1, t_2 \rangle} K$$

if and only if $(r, t) \models P \text{ has}_{t'} K$ for some $t_1 \leq t' \leq t_2$.

Says

$$(r, t) \models P \text{ says}_{t'} X$$

if and only if $t' \leq \text{Time}_P(t)$ and there is some message M and principal $Q \in \mathcal{P}$ such that $(\text{send}(M, Q), t') \in \text{History}_P(r, t)$ and $X \in \text{submsgs}_{\text{Keyset}_P(r, t')}(M)$.

$$(r, t) \models P \text{ says}_{[t_1, t_2]} X$$

if and only if $(r, t) \models P \text{ says}_{t'} X$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models P \text{ says}_{\langle t_1, t_2 \rangle} X$$

if and only if $(r, t) \models P \text{ says}_{t'} X$ for some $t_1 \leq t' \leq t_2$.

Said

$$(r, t) \models P \text{ said}_{t'} X$$

if and only if $t' \leq \text{Time}_P(t)$ and there is some $t'' \leq t'$ such that $P \text{ says}_{t''} X$.

$$(r, t) \models P \text{ said}_{[t_1, t_2]} X$$

if and only if $(r, t) \models P \text{ said}_{t'} X$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models P \text{ said}_{\langle t_1, t_2 \rangle} X$$

if and only if $(r, t) \models P \text{ said}_{t'} X$ for some $t_1 \leq t' \leq t_2$.

Controls Principals neither lie about formulas they control, nor make contradictory statements about formulas they control using the same timestamp.

$$(r, t) \models P \text{ controls}_{t'} \varphi$$

if and only if (1) $t' < \text{Time}_P(t)$ and (2) $(r, t) \models P \text{ says}_{t'} \varphi$ implies $(r, t) \models \varphi$ at P t' .

$$(r, t) \models P \text{ controls}_{[t_1, t_2]} \varphi$$

if and only if $(r, t) \models P \text{ controls}_{t'} \varphi$ for every $t_1 \leq t' \leq t_2$, and

$$(r, t) \models P \text{ controls}_{\langle t_1, t_2 \rangle} \varphi$$

if and only if $(r, t) \models P \text{ controls}_{t'} \varphi$ for some $t_1 \leq t' \leq t_2$.

Fresh A message is fresh if it has not been said before in the run.

$$(r, t) \models \text{fresh}_{t', P} X$$

if and only if $t' < \text{Time}_P(r, t)$ and $(r, t) \not\models Q \text{ said}_{t', P} X$ for all principals Q .

Shared Keys Shared keys provide both confidentiality and originator identification.

$$(r, t) \models P \xleftrightarrow{K}_{t', R} Q$$

if and only if (1) $t_1 < \text{Time}_R(t)$, (2) $(r, t) \models S \text{ received}_{t'} [X]_K$ implies $(r, t) \models P \text{ said}_{t'} X$ or $(r, t) \models Q \text{ said}_{t'} X$, and (3) $(r, t) \models S \text{ received}_{t'} \{X\}_K$ and $(r, t) \models S \text{ received}_{t'} X$ imply either

- $S \in \{P, Q\}$
- $(\text{generate}(X), t'') \in \text{History}_S$ for some $t'' \leq t'$.

$$(r, t) \models P \xleftrightarrow{K}_{[t_1, t_2], R} Q$$

if and only if $(r, t) \models P \xleftrightarrow{K}_{t', R} Q$ for all $t_1 \leq t' \leq t_2$, and

$$(r, t) \models P \xleftrightarrow{K}_{\langle t_1, t_2 \rangle, R} Q$$

if and only if $(r, t) \models P \xleftrightarrow{K}_{t', R} Q$ for some $t_1 \leq t' \leq t_2$.

27. P still-believes $_{t_3} (\overset{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q \text{ at}_P t_3)$

since

28. P believes $_{t_2} (\neg \overset{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q \text{ at}_P t^*)$

where $t_3 \geq t^* \geq t_P$. Therefore, we are prevented from using an extended belief like 27 to derive the originator of the received message in statement 21 at t_3 .

The above example illustrates the basic procedure for reasoning about revocation. Using our logic, one can specify and reason about more complex revocation lists and more complex revocation policies.

4 Model of Computation

In this section, we sketch our model of computation. An important feature of our model is the description of relevant timing of events that can play a central role in authentication and security policies. Our model is quite general since we make no assumptions about the synchronization of principals' clocks. Our model builds upon the works of [2, 5, 9, 13].

We consider a setting in which a finite set of *principals* $\mathcal{P} = \{P_1, \dots, P_n\}$ communicate by sending messages to each other. In addition, there is a special *environment principal* P_e that represents relevant aspects of the external environment. Each principal has a clock that represents his local time. P_e 's clock represents the global time. (In general, the global time is not known to the principals). All times are real numbers. The behavior of the principals will be described by a run, and the set of possible runs will constitute a system.

Formulas, messages, and other basic values are defined as in Section 2.1. Given a message M and a set \mathcal{K} of keys, we define $\text{submsgs}_{\mathcal{K}}(M)$ to be the union of $\{M\}$ and

- a. $\text{submsgs}_{\mathcal{K}}(X_1) \cup \dots \cup \text{submsgs}_{\mathcal{K}}(X_n)$ if $M = (X_1, \dots, X_n)$,
- b. $\text{submsgs}_{\mathcal{K}}(X)$ if $M = \{X\}_K$ and $K \in \mathcal{K}$,
- c. $\text{submsgs}_{\mathcal{K}}(X)$ if $M = [X]_{K^{-1}}$,
- d. $\text{submsgs}_{\mathcal{K}}(X)$ if $M = X \text{ at}_t P$.

Thus $\text{submsgs}_{\mathcal{K}}(M)$ is the set of messages derivable from M by reading submessages and using keys in \mathcal{K} . We extend $\text{submsgs}_{\mathcal{K}}$ to sets of messages in the obvious way.

We define several *basic events* that may take place for principals.

- $\text{send}(X, P_j)$: send message X to P_j ,
- $\text{receive}(X)$: receive message X , and
- $\text{generate}(X)$: generate message X .

A *timestamped event* is a pair (v, t) where v is a basic event and t is a time. A *history* is a (possibly infinite) sequence of timestamped events. If (v, t) is a timestamped event in the history \mathcal{H} , we write $(v, t) \in \mathcal{H}$. \mathcal{H} is a *sequential history* at t_i if (1) $t_1 < t_2$ for every t_1, t_2 such that $(v_1, t_1), (v_2, t_2) \in \mathcal{H}$ and (v_1, t_1) appears earlier in the sequence than (v_2, t_2) , and (2) $t \leq t_i$ for every $(v, t) \in \mathcal{H}$.

Associated with each principal P_i ($1 \leq i \leq n$) is a local state that contains the all the information P_i has access to, namely, P_i 's identity, the current local time, keys P_i can use, and a history of events that have taken place for P_i . Formally, a *local state* is a tuple $s_i = (i, t_i, \mathcal{K}_i, \mathcal{H}_i)$, where t_i is a time, \mathcal{K}_i is a set of keys, and \mathcal{H}_i is a sequential history at t_i .

As mentioned before, the *environment* P_e captures relevant aspects of the system not captured in the local states of the principals, including message modification, replay, loss, and delay. The environment state contains

- a. a time t_e , called the *real time*,
- b. a sequential history \mathcal{H}_e at t_e , and
- c. for $1 \leq i \leq n$, a *message buffer* b_i for P_i containing the set of messages sent to P_i but not yet delivered.

The *global state* of a system with n principals is a $(n+1)$ -tuple of the form (s_e, s_1, \dots, s_n) , where s_e is the environment state and each s_i is the local state of principal P_i . A *run* is a function $r : \mathbb{R} \rightarrow \mathcal{G}$, associating a global state with every point of real time. Given a run r and a time t , we denote the local state of P_i in the global state $r(t)$ by $r_i(t)$. We sometimes write (r, t) to denote $r(t)$, and call (r, t) a *point*.

Given a run r , a time t , and a principal P_i , $\text{Time}_{P_i}(r, t)$ is defined to be P_i 's local time, i.e. the time in P_i 's local state $r_i(t)$. Similarly, $\text{History}_{P_i}(r, t)$ is defined to be the history in the local state $r_i(t)$, and $\text{Keyset}_{P_i}(r, t)$ is defined to be the key set in the local state $r_i(t)$. We also define $\text{Msgs}_{P_i}(r, t) = \{M : (\text{receive}(M), t') \in \text{History}_{P_i}(r, t) \text{ for some } t'\}$, so $\text{Msgs}_{P_i}(r, t)$ is the set of messages received by P_i at or before time t in run r . Since P_i 's local time t_i may be the same for a period of real time in the run r , the set $\{t : \text{Time}_{P_i}(r, t) = t_i\}$ generally contains more than a single element. We write $\text{Start}_{P_i}(r, t_i)$ and $\text{End}_{P_i}(r, t_i)$ to denote the minimum and maximum element, respectively, of this set.

A run r is legal if certain monotonicity and consistency conditions hold. Namely, r is *legal* if for every principal P ,

- a. if $t \leq t'$, then $\text{Time}_P(r, t) \leq t'$,
- b. if $t \leq t'$, then $\text{Keyset}_P(r, t) \subseteq \text{Keyset}_P(r, t')$,
- c. if $K \in \text{Keyset}_P(r, t)$ then either
 - (a) $(\text{generate}(K), t') \in \text{History}_P(r, t)$ for some $t' \leq t$, or
 - (b) there exists a sequence $\mathcal{K}_0, \mathcal{K}_1, \mathcal{K}_2, \dots, \mathcal{K}_m$ such that $\mathcal{K}_m = \text{Keyset}_P(r, t)$, $\mathcal{K}_0 = \text{Keyset}_P(r, t')$ for some $t' \leq t$, and if $K \in \mathcal{K}_\ell$, then either $K \in \mathcal{K}_{\ell-1}$ or $K \in \text{submsgs}_{\mathcal{K}_{\ell-1}}(\text{Msgs}_P(r, t))$.
- d. if $(\text{receive}(X), t') \in \text{History}_P(r, t)$, then $(\text{send}(X, P), t'') \in \text{History}_Q(r, t)$ for some Q and t'' such that $\text{End}_Q(r, t'') \leq \text{Start}_P(r, t')$.

A *system* is a set of legal runs. Typically, we will be interested in a set of runs consisting of relevant events occurring before the start of a protocol followed by possible executions of that protocol.

4.1 Truth Conditions

In this section, we inductively define the truth of formulas in our model. An *interpretation* π is a function $\pi : \mathcal{G} \times \varphi \rightarrow \{\text{T}, \text{F}\}$ assigning truth values to primitive propositions at each global state. A pair $\mathcal{I} = (\mathcal{R}, \pi)$ is an *interpreted system*. The truth of formulas is evaluated with respect to a given point (r, t) in an interpreted system \mathcal{I} . We write $(\mathcal{I}, r, t) \models \varphi$ to indicate that φ is true at the point (r, t) of \mathcal{I} , and $(\mathcal{I}, r, t) \not\models \varphi$ to indicate that φ is not true at the point (r, t) of \mathcal{I} . Where \mathcal{I} is clear from context, we write simply $(r, t) \models \varphi$ and $(r, t) \not\models \varphi$.

Reasoning about Synchronization Our logic enables one to deduce trusted synchronization paths and to derive synchronization bounds between principals from initial assumptions and messages. Deducing trusted synchronization paths and deriving the synchronization bounds is a necessary step when protocol correctness depends on synchronization.

Suppose we wish to deduce a trusted synchronization path and derive the degree of synchronization between P and S to interpret the time t_s on S 's clock in terms of P 's clock for the belief in statement 16. As with other formulas, principals may come to believe synchronization formulas from the extension of initial assumptions or from receipt of messages containing synchronization formulas. For purposes, of illustrating the synchronization axiom, assume we derive the following beliefs, where I is some intermediate principal that has known synchronization bounds with both P and S .

$$17. P \text{ believes}_{t_1} \text{sync}_{[t_s - \max(\epsilon_1, \epsilon_2), t_s + \max(\epsilon_1, \epsilon_2)], \epsilon_1} (I, P)$$

$$18. P \text{ believes}_{t_1} \text{sync}_{[t_s - \max(\epsilon_1, \epsilon_2), t_s + \max(\epsilon_1, \epsilon_2)], \epsilon_2} (I, S)$$

We apply the clock synchronization axiom A6c to statements 17 and 18.

$$19. P \text{ believes}_{t_1} \text{sync}_{t_s, \epsilon} (S, P)$$

where $\epsilon = \epsilon_1 + \epsilon_2$. Hence, we have deduced a synchronization bound between S and P .

Converting the Time of Reference Converting the time of reference from one principal's clock to another's is relatively straight forward given a belief about synchronization bounds for the time in question. We apply the time conversion axiom A6a to statements 16 and 19.

$$20. P \text{ believes}_{t_1} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q \text{ at}_P \langle t_s - \epsilon, t_s + \epsilon \rangle$$

The time of reference for the goodness of Q 's verification key is now in terms of P 's clock.

Reasoning about Recency In practice, it is sometimes important that principals believe statements that are recent. For example, in [11] it is noted that Kerberos [10] requires certain recency constraints for authentication. The time that clients authenticate principals is carried in a ticket that is presented later to application servers when requesting service. Application servers can choose to reject tickets if the time of authentication does not meet the application server's policy for recency.

We now give an example for reasoning about recency. Suppose we have

$$21. P \text{ received}_{t_3} [X]_{K_Q^{-1}}$$

where, if authentic, the statement $Q \text{ says}_t X$ has great value. To reduce the chances that K^{-1} is no longer a good key for Q , we may require tight constraints on the recency of the goodness of Q 's key used in deducing $Q \text{ says}_t X$ from the received message in statement 21. Suppose our initial assumptions include the recency policy pertaining to Q 's keys for all values of K'_Q , t'_b , and t'_e :

$$22. P \text{ believes}_t \stackrel{K'_Q}{\Rightarrow}_{[t'_b, t'_e], S} Q \text{ at}_P t' \wedge (t' \leq t \leq \hat{t} \leq \delta + t') \supset \\ P \text{ still-believes}_t \stackrel{K'_Q}{\Rightarrow}_{[t'_b, t'_e], S} Q \text{ at}_P \hat{t}$$

Finally, we apply the particular instance of the policy of statement 22 to statement 20.

$$23. P \text{ still-believes}_{t_3} ((\stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q) \text{ at}_P t_3)$$

provided that there exists t^* such that $t_S - \epsilon \leq t^* \leq t_S + \epsilon$ and $t_S + \epsilon \leq t_1 \leq t_3 \leq t^* + \delta$. Therefore, we have updated our belief about Q 's key to t_3 and we can proceed to analyze our received message in statement 21 using K_Q provided $t_b \leq t_3 \leq t_e$.

Recency Policy for Enforcing Replay Detection Determining the recency of statements is important for replay detection mechanisms. An approach used in practice by the Kerberos network authentication protocol [10] requires synchronization between a client and server be sufficiently small such that the server can store and check for replayed messages during this window. Messages falling outside the window timeout since the server can not check the storage to determine if the message is a replay. Given that we can deduce a synchronization path between principals originating and authenticating the message, analysis consists of deducing message authentication assuming a "believe if recent" policy where the recency interval δ is chosen to reflect the length of message storage on the server. However, one can not overlook the dependency of recency intervals on sufficiently small synchronization bounds. Protocol analysis using our logic makes this dependency explicit.

Reasoning about Revocation Our logic allows us to reason about revocation of formulas. We give an example for reasoning about revocation of keys however the techniques apply to all formulas (e.g., jurisdiction, synchronization, etc.). Our example assumes a certificate revocation list similar to that of the X.509 standard [4]. The certificate revocation list contains a timestamp t , and zero or more serial numbers unambiguously identifying revoked keys. For simplicity, we assume the following revocation message containing one serial number.

$$\text{Message 2 } S \rightarrow P : [t, s_Q]_{K_S^{-1}}$$

We idealize the certificate revocation list as follows.

$$\text{Message 2 } S \rightarrow_{t_2} P : [S \text{ says}_t \neg \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q]_{K_S^{-1}}$$

where K_Q is the key uniquely identified by the serial number s_Q . Of course, to idealize particular statements about keys from the revocation list we need to have the corresponding key information.

Suppose we are able to deduce the following from revocation lists and other formulas

$$24. P \text{ believes}_{t_2} (\stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q \text{ at}_P t_P)$$

$$25. P \text{ believes}_{t_2} (\neg \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q \text{ at}_P t^*)$$

where $t^* \geq t_P$. Suppose our goal is to interpret the received message in statement 21 subject to a "believe until revoked" concerning the goodness of Q 's keys. That is, our policy assumption is for all values of K'_Q , t'_b , and t'_e ,

$$26. P \text{ believes}_t \stackrel{K'_Q}{\Rightarrow}_{[t'_b, t'_e], S} Q \text{ at}_P t' \wedge (t' \leq t \leq t'') \wedge (\forall \hat{t} : \\ t' \leq \hat{t} \leq t'') \neg (P \text{ believes}_{t''} (\neg \stackrel{K'_Q}{\Rightarrow}_{[t'_b, t'_e], S} Q \text{ at}_P \hat{t})) \supset \\ P \text{ still-believes}_{t''} (\stackrel{K'_Q}{\Rightarrow}_{[t'_b, t'_e], S} Q \text{ at}_P t'')$$

We are unable to obtain the extended belief for $t_3 \geq t_2$

2. $P \text{ believes}_{t'} (\forall t) \stackrel{K_S}{\Rightarrow}_{t,P} S \wedge t' \leq t \supset$
 $P \text{ believes}_{t''} \stackrel{K_S}{\Rightarrow}_{t,P} S$

This statement says that beliefs about the goodness of S 's signature verification key are stable as time progresses. We caution the reader that stable belief assumptions should be used sparingly in protocols that must tolerate security failures such as breaches of private keys. However, in some designs, stable assumptions, although strong, may be justified, for example if keys are embedded in hardware.

Applying this policy to statement 1 we deduce the updated belief at t_1 .

3. $P \text{ believes}_{t_1} (\forall t) \stackrel{K_S}{\Rightarrow}_{t,P} S$

In practice, protocols may be designed to recover from the compromise of a key or the loss of jurisdiction. Unlike other BAN-like logics we can reason about these protocols using policies for belief extension such as “believe until revoked” or “believe if recent”. Analysis assuming these policies is performed in a manner similar to the analysis of this protocol fragment and is illustrated later in this section.

Identifying the Originator Originator identification axioms help deduce the originator of a message. From the receipt of Message 1 we have the following.

4. $P \text{ received}_{t_1} [S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q]_{K_S^{-1}}$

We apply the originator identification axiom A8 using the updated belief for the signature key of statement 3 and the received message statement 4.

5. $P \text{ believes}_{t_1} S \text{ said}_{t_1, P} [S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q]_{K_S^{-1}}$

This statement illustrates the technique for qualifying subscript times with the identifier of a principal when the time of the reference is different from the principal making the statement. The subscript t_1, P on *said* indicates that P believes this certificate was sent at or before t_1 on P 's local clock.

Fresh Uses of Key Our logic clarifies ambiguity about fresh keys and fresh messages created using keys. A key is fresh if it has never been sent in a message. For example, a principal may generate a random session key in an authentication exchange. However, some keys are never sent in messages. It is useful to express the notion that a key has not been used to encrypt or sign a message. For example, consider the following statement.

6. $P \text{ believes}_{t_0} (\forall P', K'_{P'}, t'_b, t'_e, t'_s)$
 $\text{fresh}_{t^*, P} ([S \text{ says}_{t'_s} \stackrel{K'_{P'}}{\Rightarrow}_{[t'_b, t'_e], S} P']_{K_S^{-1}})$

This says that P believes that certificates signed using K_S^{-1} were fresh at time t^* , for example, because S was not given K_S^{-1} before time t^* . If we assume an appropriate belief extension policy we can extend this belief to t_1 . We instantiate the particular belief relating to the certificate in the received message, as follows.

7. $P \text{ believes}_{t_1} \text{fresh}_{t^*, P} ([S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q]_{K_S^{-1}})$

Deducing the Time a Message was Sent We can deduce a message was sent within a time interval if we believe the message was fresh at the beginning of the interval. We use the nonce-verification axiom A19, belief axiom A1, and statements 5 and 7 to deduce that S sent the certificate sometime between t^* and t_1 :

8. $P \text{ believes}_{t_1} S \text{ says}_{(t^*, t_1), P}$
 $[S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q]_{K_S^{-1}}$

We remove the signature from statement 8 using axiom A15 for saying.

9. $P \text{ believes}_{t_1} S \text{ says}_{(t^*, t_1), P} S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q$

Deducing Beliefs and Presence from Jurisdiction Suppose as an initial assumption we believe that S has jurisdiction over the time that time-stamped certificates are believed accurate for all times after t^* .

10. $P \text{ believes}_{t_0} (\forall t \geq t^*) S \text{ controls}_{[t^*, t], P}$
 $(\forall P', K'_{P'}, t'_b, t'_e, t'_s) S \text{ says}_{t'_s} \stackrel{K'_{P'}}{\Rightarrow}_{[t'_b, t'_e], S} P'$

Depending on the design details, it may have been appropriate to make the stronger policy assumption that P believes that S controls the time (i.e., timestamps) that S sends any message. Using a belief extension policy assumption, we extend this belief to t_1 where $t_1 \geq t_0$ and instantiate it for the particular certificate and interval in question:

11. $P \text{ believes}_{t_1} S \text{ controls}_{[t^*, t_1], P} S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q$

We apply the jurisdiction axiom A20 and A1 to statements 9 and 11.

12. $P \text{ believes}_{t_1} (S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q \text{ at}_P \langle t^*, t_1 \rangle)$

To remove $\text{at}_P \langle t^*, t_1 \rangle$ from the *says* in statement 12 we first append $\text{at}_P t_1$ to statement 12 using A3. Next we apply the reduction axiom on stable formulas A6d using A1. Finally, we remove $\text{at}_P t_1$ using A3.

13. $P \text{ believes}_{t_1} (S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q)$

Statement 13 proves the *presence* of principal S at time t_s using S 's clock as a reference. Simply proving the presence of a principal at a particular time is a design goal of some protocols. Although other BAN-like logics are capable of proving presence, they are limited in reasoning about the time of presence. Later, we translate t_s into a time on P 's clock using synchronization assumptions.

In a similar manner as before, we assume general beliefs about jurisdiction of key certificates for Q :

14. $P \text{ believes}_{t_0} (\forall t) S \text{ controls}_t$
 $(\forall K'_Q, t'_b, t'_e) \stackrel{K'_Q}{\Rightarrow}_{[t'_b, t'_e], S} Q$

and we instantiate the jurisdiction assumptions for the particular certificate in question.

15. $P \text{ believes}_{t_1} S \text{ controls}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q$

Next we apply the jurisdiction axiom A20 to statements 13 and 15.

16. $P \text{ believes}_{t_1} (\stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q \text{ at}_S t_s)$

We deduce that P believes that K_Q is a good key at t_s on S 's clock. Where protocol correctness relies on clock synchronization, we need to derive a trusted clock synchronization path between S and P to determine what t_s on S 's clock means in reference to P 's clock.

2.3 Extension of Beliefs

As defined above, our logic is somewhat restrictive because it is not possible to extend beliefs from one time to a later time. This is done for two reasons. First, it allows for the possibility of beliefs to change over time. Second, it supports the use of different policies for extension of beliefs. Some example policies follow. Note that these are not axioms, but must be explicitly assumed as initial assumptions if they are to be used in a proof. The simplest policy allows arbitrary extension of belief, essentially treating all beliefs as stable. More restrictive policies require a new construct, P *still-believes* _{t} φ , that acts like P *believes* _{t} φ , except that the belief extension policy does not apply to it. Assuming this were defined, some example policies can be formulated as follows.

- P1. **Stable beliefs:** P *believes* _{t} $\varphi \wedge (t \leq t') \supset$
 P *believes* _{t'} φ
- P2. **Believe if recent:** P *believes* _{t} φ *at* _{P} $t' \wedge$
 $(t' \leq t \leq \hat{t} \leq \delta + t') \supset P$ *still-believes* _{t} φ *at* _{P} \hat{t}
- P3. **Believe until revoked:**
 P *believes* _{t} φ *at* _{P} $t' \wedge (t' \leq t \leq t'') \wedge$
 $(\forall \hat{t} : t' \leq \hat{t} \leq t'') \neg (P$ *believes* _{t''} $((\neg \varphi)$ *at* _{P} $\hat{t})) \supset$
 P *still-believes* _{t''} φ *at* _{P} t''
- P4. **Believe if recent and not revoked:**
 P *believes* _{t} φ *at* _{P} $t' \wedge (t' \leq t \leq t'' \leq \delta + t') \wedge$
 $(\forall \hat{t} : t' \leq \hat{t} \leq t'') \neg (P$ *believes* _{t''} $((\neg \varphi)$ *at* _{P} $\hat{t})) \supset$
 P *still-believes* _{t''} φ *at* _{P} t''

These policies are simple, but useful. Policy P1 results in a setting in which beliefs are stable, since any belief at a given time can be extended to a later time. Policy P2 says that formulas are believed for a window of time of length δ . These policies should not be assumed if the protocol allows for revocation of the particular formula specified in the policy. If the protocol allows for revocation, then other policies such as P3 may be more appropriate. Policy P3 says that principals believe formulas until they receive notification to revoke the belief. Policy P4 says that principals believe formulas if they are recent and not believed to be revoked.

It would also be possible to define more complicated policies. For example, one might wish to define a policy in which the extension of beliefs depends on the formula in question. There are several reasons for representing policies as inferences to be assumed in a proof. It allows more flexibility, as the policy can be tailored to an individual situation. In addition, it allows for consideration of a particular protocol with respect to multiple policies. The ability to reason about the conjunction of policies and protocols will be especially important as public and private key infrastructures are deployed and new and unanticipated policies are put into use.

2.4 Using the Logic

In this section, we sketch the use of our logic to perform syntactic analysis of protocols. As is done in previous logics [2, 3, 13], the first step is to *idealize* the protocol. In this step, the messages of the protocol are written in the language of messages defined in Section 2.1. In the idealization, each message is labeled with a time that represents the time that the message is received. Once the protocol is idealized, we can attempt to prove protocol goals, such as participants agreeing on keys. The premise set of our proof consists of

any initial assumptions, such as shared keys between participants. In addition, for each message

$$P \rightarrow_t Q : X$$

in the idealization, Q *received* _{t} X is included in the premise set where t is the time on Q 's local clock when the message is received.

We recommend that the time of a principal's beliefs in any initial assumptions is no later than when the principal starts to operate. If this restriction is followed, one is less likely to overlook policy assumptions used to make formulas and beliefs current at the time of message receipt.

A *proof* consists of a sequence of formulas. Each formula must either be in the premise set, be an axiom, or be derivable from formulas earlier in the sequence by modus ponens or by necessitation.

3 Example Techniques

In this section, we analyze a protocol fragment to illustrate techniques for specifying and reasoning about cryptographic protocols using our logic. We show techniques for reasoning about synchronization, recency, and revocation.

Idealizing Timestamps using “says” Our logic is unambiguous in the notion of the time when a message was sent using the *says* construct and the notion that the t on X *at* _{P} t is the time the formula is true using P 's clock as a reference. Furthermore, since principals are assumed to say formulas that are true we can drop the *at* _{P} t and merely use *says* if the time the message sent is the same time that the sender believes the formula is true. If a timestamp indicates the time a message is sent and a second timestamp indicates the time the referenced formula is true, then both the *says* and *at* _{P} t constructs are used for idealization. The examples that follow illustrate how idealizing both notions of timestamps is important for reasoning about the correctness of cryptographic protocols.

Consider a public key identity certificate where Q is a principal, K_Q is Q 's public signature verification key, s_Q is the certificate serial number, t_b and t_e are the begin and end times of the key validity period, t_s is a timestamp indicating the time the certificate information was deemed accurate, and K_S^{-1} is the private signature key of the certifying authority principal S , as shown by the following message.

$$\text{Message 1 } S \rightarrow P : [Q, K_Q, s_Q, t_b, t_e, t_s]_{K_S^{-1}}$$

We idealize this timestamped certificate as

$$\text{Message 1 } S \rightarrow_{t_1} P : [S \text{ says}_{t_s} \stackrel{K_Q}{\Rightarrow}_{[t_b, t_e], S} Q]_{K_S^{-1}}$$

where the timestamp t_s is a subscript parameter of *says* and the goodness of the key is being asserted at that time.

Stable Beliefs We extend the time of beliefs using belief extension policies. Suppose we are given as assumption the initial belief at t_0 that K_S is a good key for S for all time t .

$$1. P \text{ believes}_{t_0} (\forall t) \stackrel{K_S}{\Rightarrow}_{t, P} S$$

To extend statement 1 to time t_1 where $t_1 > t_0$ we assume a variant of stable belief policy P1 particular to principal S 's key.

- b. $(\forall t \geq t')\varphi$ and $(\forall t \leq t')\varphi$ are formulas if t' is a time.

In addition, any time t that appears in a formula can be replaced by t, P to yield another formula. Similarly, $[t_1, t_2]$ can be replaced by $[t_1, t_2], P$ and $\langle t_1, t_2 \rangle$ can be replaced by $\langle t_1, t_2 \rangle, P$.

The semantics of all formulas are made precise by the definitions given in Section 4.

2.2 Axioms

Our axiom system includes two inference rules. For any formulas φ and ψ , and time t ,

- R1. Modus Ponens: From φ and $\varphi \supset \psi$ infer ψ .
R2. Necessitation: If $\vdash \varphi$, then from φ infer $P \text{ believes}_t \varphi$.

$\vdash \varphi$ means that φ is a *theorem*, i.e. it is derivable from the axioms alone. Since the \vdash symbol is a metalinguistic symbol, it does not actually appear in any proofs. We write $\Gamma \vdash \varphi$ if from the set Γ of formulas, it is possible to derive φ .

We take all tautologies of propositional logic as axioms. In addition, we take as axioms all instances of the following axiom schemas.

Belief

- A1. $P \text{ believes}_t \varphi \wedge P \text{ believes}_t (\varphi \supset \psi) \supset P \text{ believes}_t \psi$
A2. $P \text{ believes}_t \varphi \equiv P \text{ believes}_t P \text{ believes}_t \varphi$
A3. $P \text{ believes}_t \varphi \equiv P \text{ believes}_t (\varphi \text{ at}_P t)$

Multiple levels of belief are equivalent to a single level. Since principals can be mistaken, $P \text{ believes}_t \varphi \supset \varphi \text{ at}_P t$ does not hold.

Time and Synchronization

- A4. Time interval axioms
a. $P \text{ believes}_{[t_1, t_2]} \varphi \equiv (\forall t : t_1 \leq t \leq t_2) (P \text{ believes}_t \varphi)$
b. $P \text{ believes}_{\langle t_1, t_2 \rangle} \varphi \equiv (\exists t : t_1 \leq t \leq t_2) (P \text{ believes}_t \varphi)$

We also include analogous axioms for *controls*, *received*, *says*, *said*, \leftrightarrow , \mapsto , \Rightarrow , *has*, *fresh*, and *sync*.

A5. Monotonicity axioms

- a. $P \text{ received}_t X \wedge t' \geq t \supset P \text{ received}_{t'} X$
b. $P \text{ said}_t X \wedge t' \geq t \supset P \text{ said}_{t'} X$
c. $X \text{ has}_t K \wedge t' \leq t \supset X \text{ has}_{t'} K$
d. $\text{fresh}_{t, P} X \wedge t' \leq t \supset \text{fresh}_{t', P} X$
e. $(\varphi \text{ at}_P t_1) \text{ at}_P t_2 \wedge t_1 \leq t_2 \supset \varphi \text{ at}_P t_1$
f. $\text{sync}_{t, \epsilon} (P, Q) \wedge \epsilon \leq \epsilon' \supset \text{sync}_{t, \epsilon'} (P, Q)$

A6. Synchronization and time conversion axioms

- a. $\varphi \text{ at}_P t \wedge \text{sync}_{t, \epsilon} (P, Q) \supset \varphi \text{ at}_Q \langle t - \epsilon, t + \epsilon \rangle$
b. $\text{sync}_{t, \epsilon} (P, Q) \supset \text{sync}_{\langle t - \epsilon, t + \epsilon \rangle, \epsilon} (Q, P)$
c. $\text{sync}_{[t - \max(\epsilon_1, \epsilon_2), t + \max(\epsilon_1, \epsilon_2)], \epsilon_1} (Q, P) \wedge \text{sync}_{[t - \max(\epsilon_1, \epsilon_2), t + \max(\epsilon_1, \epsilon_2)], \epsilon_2} (Q, R) \supset \text{sync}_{t, (\epsilon_1 + \epsilon_2)} (P, R)$
d. $(\varphi \text{ at}_P t_1) \text{ at}_P t_2 \wedge t_2 \geq t_1 \supset \varphi \text{ at}_P t_2$, where φ is $\psi \text{ at}_P t$ for some ψ , P , and t , or φ is $Q \text{ says}_t X$, $Q \text{ said}_t X$, $Q \text{ has}_t X$, or $Q \text{ received}_t X$, for some Q , X , and t .

Originator Identification

- A7. $P \xleftrightarrow{K}_{t, P} Q \wedge P \text{ received}_t [X]_{K^{-1}} \supset Q \text{ said}_{t, P} X \wedge Q \text{ said}_{t, P} [X]_{K^{-1}} \wedge Q \text{ has}_{t, P} K^{-1}$
A8. $\xleftrightarrow{K}_{t, P} Q \wedge P \text{ received}_t [X]_{K^{-1}} \supset Q \text{ said}_{t, P} X \wedge Q \text{ said}_{t, P} [X]_{K^{-1}} \wedge Q \text{ has}_{t, P} K^{-1}$

Principals use signatures to deduce the originator of messages.

Receiving

- A9. $P \text{ received}_t (X_1, \dots, X_n) \supset P \text{ received}_t X_i$
A10. $P \text{ received}_t \{X\}_{K^{-1}} \wedge P \text{ has}_t K^{-1} \supset P \text{ received}_t X$
A11. $P \text{ received}_t [X]_{K^{-1}} \supset P \text{ received}_t X$

Principals can decrypt encrypted messages if they have decryption keys, and principals can read signed messages with or without the signature verification key. (However, in order to actually verify the originator of a signed message, it is necessary to have the key.)

Saying

- A12. $P \text{ said}_t (X_1, \dots, X_n) \supset P \text{ said}_t X_i$
A13. $P \text{ says}_t (X_1, \dots, X_n) \supset P \text{ says}_t X_i$
A14. $P \text{ said}_t [X]_{K^{-1}} \supset P \text{ said}_t X$
A15. $P \text{ says}_t [X]_{K^{-1}} \supset P \text{ says}_t X$
A16. $P \text{ said}_t X \supset (\exists t' \leq t) P \text{ says}_{t'} X$
A17. $P \text{ says}_t X \supset P \text{ said}_t X$

A14 and A15 state that principals are responsible for the contents of signed messages that they send, even if they did not sign them. If instead, P wants to forward a signed message $[X]_{K^{-1}}$ from Q without being responsible for the contents, P would send the message “ $Q \text{ said}_t [X]_{K^{-1}}$ ”.

Freshness

- A18. $\text{fresh}_t X \supset \text{fresh}_t F(X, Y)$

It is assumed that the function F in A18 actually depends on the argument X .

Nonce-Verification

- A19. $\text{fresh}_{t_1, P} X \wedge P \text{ said}_{t_2} X \supset P \text{ says}_{\langle t_1, t_2 \rangle} X$

Jurisdiction

- A20. $P \text{ controls}_t \varphi \wedge P \text{ says}_t \varphi \supset \varphi \text{ at}_P t$

Symmetry of Shared Keys

- A21. $P \xleftrightarrow{K} Q \equiv Q \xleftrightarrow{K} P$

such as the correct use of cryptographic functions. Principals are trusted to communicate honestly, hence, senders believe the truth of message contents. Also, principals can be trusted to keep secrets. In particular, although their keys may become compromised and thereby allow others to decrypt encrypted messages, principals are assumed not to pass on the decrypted contents themselves unless specified by the protocol. Keys are assumed to be adequate for their purposes and cryptosystems are assumed to be secure and properly used. For example, we assume that it is only possible for a principal to create the value $\{X\}_K$ that is the encryption of X with key K if the principal has both X and K , and conversely, if a principal attempts to decrypt $\{X\}_K$ with the wrong key, the result will not be a message. In practice, this is easily achievable by formatting messages according to a specified format before encrypting them. Principals can recognize their own encrypted messages, and will not mistakenly attribute them to another principal.

We also make certain assumptions about the use of our logic. In particular, we assume that users of our logic properly idealize protocols and make “reasonable” assumptions when carrying out protocol analysis. For example, assumptions should not contradict each other, and should reflect the real world in which the correctness of the protocol is being claimed.

In Section 2.1, we formally define the set of messages and formulas in the logic. In Section 2.2, we define the axioms that allow us to derive one formula from another.

2.1 Messages and formulas

\mathcal{T} is the set of primitive *terms*. We assume \mathcal{T} contains the real numbers and several distinguished subsets of constant symbols called: *principals*, *public keys*, *private keys*, *shared keys*, *times*, *data constants* (such as nonces), and *primitive propositions*. Together, the set of public keys, private keys, and shared keys, are called *keys*. We assume that the set of times is ordered. Each key has an *inverse*. Given a public or shared key K , we write K^{-1} to denote its inverse. If K is a public key, then K^{-1} is a private key. If K is a shared key, then K^{-1} is a shared key and $K^{-1} = K$. As a notational convention, we always denote a private key as the inverse of the corresponding public key.

$\mathbf{F}_{\mathcal{T}}$ is the set of functions on primitive terms. In particular, there are distinguished functions in $\mathbf{F}_{\mathcal{T}}$ that represent encryption ($\{X\}_K$), digital signatures ($[X]_{K^{-1}}$), and concatenation ((X_1, \dots, X_n)).

Some messages have truth values while others, such as times, principal’s names and nonces, do not. We identify a subset $\mathcal{F}_{\mathcal{T}}$ of $\mathcal{M}_{\mathcal{T}}$ that is the set of *formulas*. All formulas have truth values. Messages and formulas in the logic are defined by mutual induction. Given the set of terms \mathcal{T} , we define the set of messages, $\mathcal{M}_{\mathcal{T}}$, to be the smallest set satisfying the following conditions:

- M1. φ is a message if φ is a formula.
- M2. X is a message if $X \in \mathcal{T}$.
- M3. $F(X_1, \dots, X_n)$ is a message if X_1, \dots, X_n are messages and F is any n -ary function in $\mathbf{F}_{\mathcal{T}}$.

In particular, it follows from M3 that $\{X\}_K$, $[X]_{K^{-1}}$, and (X_1, \dots, X_k) are messages if X is a message, K is a public or shared key, and X_1, \dots, X_k are messages.

The set of formulas, $\mathcal{F}_{\mathcal{T}}$, is the smallest set satisfying the following conditions.

- F1. p is a formula if p is a primitive proposition,
- F2. $\neg\varphi$ and $\varphi \wedge \psi$ are formulas if φ and ψ are formulas.
- F3. $t_1 \leq t_2$ and $t_1 \geq t_2$ are formulas if t_1 and t_2 are times or real numbers.
- F4. If P is a principal and φ is a formula, then
 - a. $P \text{ believes}_t \varphi$ and $P \text{ controls}_t \varphi$ are formulas if t is a time.
 - b. $P \text{ believes}_{[t_1, t_2]} \varphi$ and $P \text{ controls}_{[t_1, t_2]} \varphi$ are formulas if $t_1 \leq t_2$ are times.
 - c. $P \text{ believes}_{\langle t_1, t_2 \rangle} \varphi$ and $P \text{ controls}_{\langle t_1, t_2 \rangle} \varphi$ are formulas if $t_1 \leq t_2$ are times.
- F5. If P is a principal and X is a message, then
 - a. $P \text{ received}_t X$, $P \text{ said}_t X$, and $P \text{ says}_t X$ are formulas if t is a time.
 - b. $P \text{ received}_{[t_1, t_2]} X$, $P \text{ said}_{[t_1, t_2]} X$, and $P \text{ says}_{[t_1, t_2]} X$ are formulas if $t_1 \leq t_2$ are times.
 - c. $P \text{ received}_{\langle t_1, t_2 \rangle} X$, $P \text{ said}_{\langle t_1, t_2 \rangle} X$, and $P \text{ says}_{\langle t_1, t_2 \rangle} X$ are formulas if $t_1 \leq t_2$ are times.
- F6. If P and Q are principals, K_1 is a shared key, and K_2 is a public key, then
 - a. $P \xrightarrow{K_1}_t Q$, $\xrightarrow{K_2}_t P$, $\xrightarrow{K_2}_t P$ are formulas if t is a time.
 - b. $P \xrightarrow{K_1}_{[t_1, t_2]} Q$, $\xrightarrow{K_2}_{[t_1, t_2]} P$ and $\xrightarrow{K_2}_{[t_1, t_2]} P$ are formulas if $t_1 \leq t_2$ are times.
 - c. $P \xrightarrow{K_1}_{\langle t_1, t_2 \rangle} Q$, $\xrightarrow{K_2}_{\langle t_1, t_2 \rangle} P$ and $\xrightarrow{K_2}_{\langle t_1, t_2 \rangle} P$ are formulas if $t_1 \leq t_2$ are times.
- F7. If P is a principal and K is a key, then
 - a. $P \text{ has}_t K$ is a formula if t is a time.
 - b. $P \text{ has}_{[t_1, t_2]} K$ is a formula if $t_1 \leq t_2$ are times.
 - c. $P \text{ has}_{\langle t_1, t_2 \rangle} K$ is a formula if $t_1 \leq t_2$ are times.
- F8. If X is a message and P is a principal, then
 - a. $\text{fresh}_{t, P} X$ is a formula if t is a time.
 - b. $\text{fresh}_{[t_1, t_2], P} X$ is a formula if $t_1 \leq t_2$ are times.
 - c. $\text{fresh}_{\langle t_1, t_2 \rangle, P} X$ is a formula if $t_1 \leq t_2$ are times.
- F9. If P and Q are principals and ϵ is a real number, then
 - a. $\text{sync}_{t, \epsilon}(P, Q)$ is a formula if t is a time.
 - b. $\text{sync}_{[t_1, t_2], \epsilon}(P, Q)$ is a formula if $t_1 \leq t_2$ are times.
 - c. $\text{sync}_{\langle t_1, t_2 \rangle, \epsilon}(P, Q)$ is a formula if $t_1 \leq t_2$ are times.
- F10. If P is a principal, t is a time, and φ is a formula, then $\varphi \text{ at}_P t$ is a formula.
- F11. If φ is a formula and t is a time, then
 - a. $(\exists t : t_1 \leq t \leq t_2)\varphi$ is a formula if t_1 and t_2 are times.
 - b. $(\exists t \geq t')\varphi$ and $(\exists t \leq t')\varphi$ are formulas if t' is a time.
- F12. If φ is a formula and t is a time, then
 - a. $(\forall t : t_1 \leq t \leq t_2)\varphi$ is a formula if t_1 and t_2 are times.

reformulated by Abadi and Tuttle [2], and later by Syverson and van Oorschot [13]. Unlike these logics, we qualify all statements with time to allow arbitrary revocation of keys, beliefs, and jurisdiction. We use a notation and formalism that is similar to the BAN-related logics to highlight the parallels between our logic and the BAN-related logics.

The work of Syverson on adding time to a logic of authentication [12], although related, has a different focus than our work. His work incorporates temporal formalism into the semantics model of BAN logic [2] using temporal notions of “all points in the run prior to the current one,” and “at some point in the run prior to the current one.” Our logic, on the other hand, allows for arbitrary granularity of time.

Recent work of Stubblebine [11] is more closely related to our work. It introduces the notion of *recent-secure authentication*. Recent-secure authentication characterizes an authentication policy based on the recency of security information (such as keys) used in authenticating channel or message. In particular, it requires that at the time of authentication, verifiers must have received a recent account attesting to the accuracy of security-relevant information used for authentication. Authentication according to recent-secure policies ensures that revocation is fail-safe. That is, if security information is revoked at time t then no verifier will authenticate a message using revoked credentials after time $t + \delta$ where δ is the recency interval. Our work formalizes the notions of recency and of individual authentication policies based on recency presented by Stubblebine. Unlike this previous work, our method does not assume clock synchronization and allows for reasoning about trusted synchronization paths and synchronization bounds between multiple protocol participants.

The work of Gaarder and Snekenes [6] extends BAN logics to handle issues within the scope of our logic. They introduce the notion that a participant claims a formula is, was, or will be good in a time interval and the notation that the local clock of a participant has a time within an interval. The work makes the strong assumption that the duration of a protocol run must be short, and that participants commit themselves to believing that messages are correct for the duration of the protocol. The logic has the limiting characteristic of other BAN-like logics that beliefs may not be revoked at an arbitrary time. Also, the assumptions used are too restrictive for reasoning about long-lived security associations. Furthermore, reasoning about synchronization and synchronization bounds is beyond the scope of their work. Our work removes these restrictions by allowing arbitrarily long protocol durations in which participants beliefs may change over time.

1.2 Our Work

The main difference between our logic and the BAN logic and its successors is that we allow for the possibility of revocation of keys, jurisdictions, and more generally, of arbitrary beliefs. That is, we allow for a principal to hold a belief at one point in the run of a protocol, and not to hold that belief later in the run. We are also much more strict about the extension of beliefs from a given point in time to a later point. In particular, a belief at a given time can only be extended to a later time according to the policies that are specified as part of the assumptions of the protocol.

Another significant difference between our logic and the BAN logic is our use of timestamps. BAN treats all timestamps as nonces. As such, the important property of a timestamp is its freshness. However, there are cases where it is the

recentness, rather than the freshness, of a timestamp that is important. This is the case, for example, with a Kerberos ticket [10] that may have been sent before, so is not fresh, but is still valid because it was created recently enough. Our logic explicitly captures the time of the timestamp and uses this time to reason about recentness.

In the BAN-like logics, the response to a nonce-based challenge is idealized as the same nonce as in the challenge. We feel that this is not the proper way to handle challenge-response protocols in our logic, since in particular, it is the freshness of both the challenge and the response that allows the recipient to determine the presence of the responder. In using our logic, the response should be idealized as it is said, allowing the differentiation between the freshness of the challenge and the freshness of the response. As an example of the usefulness of this approach, consider a protocol in which P issues a challenge X to Q at time t_1 , Q responds with $X + 1$ encrypted with a shared key K at time t_2 , and then Q sends the same response again at t_3 . Our logic allows P to determine Q 's presence between t_1 and t_2 because of the freshness of Q 's response at time t_2 , but does not allow P to determine Q 's presence between t_2 and t_3 , since the message at t_3 may have been a replay. This example illustrates how our logic allows for different beliefs to be concluded at different times.

In addition, we have added a synchronization construct to allow reasoning about trusted synchronization paths and synchronization bounds between protocol participants. Consequently, analysis makes explicit the impact that derivation of trustworthy synchronization paths and synchronization bounds have on achieving protocol goals.

Although the addition of time into our logic yields a slightly more complicated logic, we feel the additional complexity is justified due to the considerable gains in being able to specify and analyze important properties of systems and policies found in practice. Also, we feel that the analysis technique can easily be reduced to implementation, thereby removing much of the complexity of analysis.

We define our logic in Section 2. In Section 3, we analyze a protocol fragment to highlight several interesting aspects of the use of our logic. In Section 4, we define our model of computation and sketch a proof of the soundness of the logic with respect to the model of computation.

2 The Logic

Before proceeding to a formal definition of our logic, we first give an intuitive description. Our logic consists of formulas that make statements about the knowledge and beliefs of the principals involved.

We make statements such as P *believes_t* X (P believes X holds at time t) and P *received_t* X (P received X at time t), about messages such as $\{X\}_K$ (X encrypted by K) and $[X]_{K^{-1}}$ (X signed by K^{-1}). We assume each principal has a local concept of time, and that different principals' times may not agree. In our basic formulas, a time refers to the local time of the principal in the formula. Where there is no principal, or more than one principal, involved, we explicitly state the principal whose time is being referenced. We use $\langle t_1, t_2 \rangle$ to indicate that a formula holds at all times between t_1 and t_2 , and we use $\langle t_1, t_2 \rangle$ to indicate that a formula holds at some time between t_1 and t_2 .

Our logic makes some simplifying assumptions similar to those made by related logics (cf. [3]). We do not attempt to answer every important question about cryptographic protocols. Our goal is not to reason about implementation details

An Authentication Logic Supporting Synchronization, Revocation, and Recency

Stuart G. Stubblebine*

Rebecca N. Wright†

Abstract

Distributed systems inherently involve dynamic changes to the value of security attributes such as the goodness of encryption keys. Since concurrent knowledge is usually infeasible or impractical, it is often necessary for the participants of distributed protocols to determine and act on beliefs that may not be supported by the current state of the system. Policies for determining beliefs in such situations can range from extremely conservative, such as only believing statements if they are very recent, to extremely optimistic, such as believing all statements that are not yet known to be revoked. Such security policies often are heavily dependent on timing of received messages and on synchronization between principals.

We present a logic for analyzing cryptographic protocols that has the capability to specify time and synchronization details. This capability considerably advances the scope of known techniques for both expressing practical authentication policies of protocol participants as constraints, and for reasoning about protocol goals subject to these constraints. In the course of reasoning about protocol goals, one is able to deduce requirements for trust between protocol participants, synchronization between protocol participants, and timeliness of message contents.

Our logic is flexible, and can support a wide range of security policies. The ability to reason about the conjunction of individual participant policies and protocols will be especially important as public and private key infrastructures are deployed and new and unanticipated policies are put into use.

1 Introduction

In distributed systems, the lack of physical protection on a communication path means that communication is susceptible to attacks involving eavesdropping, modification, and impersonation. Protocols using cryptographic mechanisms such as encryption and digital signatures form the basis of security in distributed systems by providing protection against such attacks. Unfortunately, cryptographic protocol design has historically been error prone as evidenced by numerous vulnerabilities cited in the literature. However, analysis and design techniques have proven useful in detecting protocol vulnerabilities.

*AT&T Bell Laboratories, 600 Mountain Avenue, Room 2A-345A, Murray Hill, NJ, USA, 07974. E-mail: stubblebine@research.att.com.

†AT&T Bell Laboratories, 600 Mountain Avenue, Room 2T-314, Murray Hill, NJ, USA, 07974. E-mail: rwright@research.att.com.

We present a logic for analyzing cryptographic protocols with a capability to specify time and synchronization details. This capability considerably advances the scope of known techniques for both expressing practical authentication policies of protocol participants as constraints, and for reasoning about protocol goals subject to these constraints. Since we can reason about properties of synchronization, recency, and revocation, one can analyze protocol goals that are beyond the scope of existing analysis techniques. Using our logic one can derive protocol goals such as the presence of participants at particular times, the goodness of keys at particular times, and participants beliefs at particular times. More specifically, goals might involve showing the long-term evolution of beliefs in practical systems characterized by dynamic changes to the goodness of keys, jurisdiction, authorization, and quality of synchronization, showing the correctness of replay detection mechanisms using timestamps, showing beliefs about the goodness of keys or authorization “tickets” subject to policies requiring that these statements are recent and aren’t known to be revoked, or showing the time of presence and continued presence of remote principals.

In the course of reasoning about protocol goals, one is able to deduce requirements such as trust between protocol participants, synchronization bounds between participants, and recency or timeliness of message contents. We separate the protocol itself from security policies that specify, for example, the methods principals use for determining beliefs from previously received messages, and handling of revocation. Policies can range from conservative, such as only believing statements if they are very recent, to optimistic, such as believing all statements that are not yet known to be revoked.

1.1 Related Work

Considerable effort has been spent on understanding, extending, and improving the use of logics for cryptographic protocols [2, 6, 7, 12, 13]. Additional research has been performed in calculi for access control and authentication in distributed systems [1, 8, 11]. However, these approaches fall short of our objectives for reasoning about synchronization, recency, and revocation.

In their seminal paper [3], Burrows, Abadi, and Needham introduced a logic for reasoning about authentication protocols. In the BAN logic and its successors (for example, [2, 7, 13]), time is divided into two epochs, the past and the present. The past is all time before the current run of the protocol being considered; the present begins at the start of the current run. Since all beliefs held in the present are assumed to be stable during the current run of the protocol, principals are not allowed to change their beliefs during the run. In particular, in their logic, if a principal P believes that K is a good key at some point in the run of a protocol, then P must believe that K is a good key for the duration of the protocol.

Our underlying logic is closely based on the BAN logic, as

Also appears in *Proceedings of the 3rd ACM Conference on Computer and Communications Security*, March 14-16, 1996, New Delhi, India.